

***“ALGORITMOS DE SEGMENTACIÓN DE OBJETOS
DINÁMICOS EN SECUENCIAS DE VIDEO
MEDIANTE APRENDIZAJE PROFUNDO”***

TESIS

QUE PARA OBTENER EL GRADO DE

***DOCTOR EN CIENCIAS EN INGENIERÍA
ELECTRÓNICA***

PRESENTA:

M.C. ABIMAEEL GUZMÁN PANDO

DIRECTOR DE LA TESIS:

DR. MARIO IGNACIO CHACÓN MURGUÍA



SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE
MÉXICO



CHIHUAHUA, CHIH., FEBRERO 2021



Chihuahua, Chih. 15 de febrero de 2021

ROGELIO ENRIQUE BARAY ARANA

JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Doctor en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno **ABIMAEEL GUZMÁN PANDO** titulado "*Algoritmos de segmentación de objetos dinámicos en secuencias de video mediante aprendizaje profundo*", dirigido por el Dr. Mario Ignacio Chacón Murguía, ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

ATENTAMENTE

*Excelencia en Educación Tecnológica®
"La técnica por el Engrandecimiento de México"*

DR. MARIO IGNACIO CHACÓN MURGUÍA
DIRECTOR DE TESIS

DR. JAVIER VEGA PINEDA
MIEMBRO DEL COMITÉ TUTORIAL

DR. JUAN ALBERTO RAMÍREZ QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL

DR. LUIS FRANCISCO CORRAL MARTÍNEZ
MIEMBRO DEL COMITÉ TUTORIAL

DRA. GRACIELA MARÍA DE JESÚS RAMÍREZ ALONSO
MIEMBRO DEL COMITÉ TUTORIAL





Chihuahua, Chih. **15 de febrero de 2021**

ABIMAEEL GUZMÁN PANDO
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Doctor en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada *"Algoritmos de segmentación de objetos dinámicos en secuencias de video mediante aprendizaje profundo"*, dirigida por el Dr. Mario Ignacio Chacón Murguía con el siguiente contenido de capítulos:

- I Antecedentes generales en detección de movimiento, aprendizaje profundo y percepción de movimiento
- II Algoritmos de aprendizaje profundo enfocados a detección de movimiento
- III Algoritmos de detección de movimiento inspirados en la visión biológica
- IV Red neuronal convolucional para detección de movimiento con información temporal (2FraCNN)
- V Algoritmo de detección de movimiento basado en la visión biológica de las aves rapaces (DeepFoveaNet)
- VI Resultados y conclusiones generales

ATENTAMENTE

Excelencia en Educación Tecnológica®
"La técnica por el Engrandecimiento de México"

ROGELIO ENRIQUE BARAY ARANA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN





CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 15 de febrero de 2021, el que suscribe, ABIMAEEL GUZMÁN PANDO con número de control D18060574, del Doctorado en Ciencias en Ingeniería Electrónica adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección de el Dr. Mario Ignacio Chacón Murguía y cede los derechos del trabajo titulado *"Algoritmos de segmentación de objetos dinámicos en secuencias de video mediante aprendizaje profundo"*, al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

Abimael G

ABIMAEEL GUZMÁN PANDO



DECLARACIÓN DE ORIGINALIDAD

En la ciudad de Chihuahua el día 15 de febrero de 2021, el que suscribe, ABIMAEEL GUZMÁN PANDO con número de control D18060574, del Doctorado en Ciencias en Ingeniería Electrónica adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta ser autor intelectual del presente trabajo de Tesis titulado *"Algoritmos de segmentación de objetos dinámicos en secuencias de video mediante aprendizaje profundo"*, bajo la dirección de el Dr. Mario Ignacio Chacón Murguía; y manifiesta que el contenido es original y que las fuentes de información consultadas para su fundamentación están debidamente citadas y referenciadas.

Abimael G

ABIMAEEL GUZMÁN PANDO

*A mi esposa Marisol.
A mis padres Rosa y Efrén.
A mis hermanos Aldahír y Esmeralda.*

*Por acompañarme en el camino,
formarme, amarme y ser
un ejemplo de lucha
y esfuerzo.*

De parvis grandis acervus erit-
De las cosas pequeñas se nutren las cosas grandes

AGRADECIMIENTOS

A Dios y mi familia. A Dios por darme el entendimiento y la sabiduría necesaria para realizar este trabajo y por mostrarme su infinito amor en todo momento. Agradezco a mis padres Efrén y Rosa, por el apoyo y formación que me brindaron cada día con su amor incondicional, por ser un ejemplo y una inspiración para lograr cada objetivo que me he propuesto. A mis hermanos Esmeralda y Aldahír, quienes siempre me han mostrado su solidaridad y cariño. A mi esposa Marisol, quien a través de su amor y bondad, me ha dado fortaleza e inspiración para superarme y ser una mejor persona día con día.

Al Instituto Tecnológico de Chihuahua que fue pilar en mi formación académica y profesional. Al Fondo Mixto de Fomento a la Investigación Científica y Tecnológica CONACYT por el apoyo brindado para la realización de este proyecto.

A mi director de tesis el Dr. Mario Chacón por la formación y enseñanza que me transmitió a lo largo de los años que trabajamos juntos. Especialmente agradezco sus consejos, paciencia, apoyo y compromiso para instruir en mí sus conocimientos y experiencia. Mi más sincero respeto y admiración hacia su persona y trabajo.

A mis miembros de comité: Dra. Graciela Ramírez, Dr. Juan Ramírez, Dr. Francisco Corral, Dr. Javier Vega, y Dr. Sandoval Rodríguez. Les agradezco por ofrecerme su apoyo al ser parte de mi formación académica, brindarme asesorías y los conocimientos necesarios para concluir en tiempo y forma con el plan de estudios del DCIE. Además, agradezco por el tiempo que invirtieron en revisar esta tesis, por sus comentarios y aportaciones para hacer de este un trabajo de calidad. Un especial agradecimiento a la Dra. Graciela Ramírez por su solidaridad y amistad.

Al personal académico del DEPI, la M.C. Alma Corral por ser una persona amable, franca y solidaria, por todo su apoyo y consejos. Reitero mi más sincero respeto y admiración para todos los miembros de comité y al personal académico del DEPI.

A todos mis amigos y compañeros del PVR y DEPI, Luis, David, Manuel, Eduardo, Miguel, José, Jorge, Carlos, Moisés, Misael, Nadia, Brenda, Oscar, Andrea, Daniel, Raúl, Iván, Alonso, Javier, entre otros. Ellos hicieron que mi estancia en el DEPI fuera más llevadera y amena brindándome su apoyo, amistad y consejos.

RESUMEN

ALGORITMOS DE SEGMENTACIÓN DE OBJETOS DINÁMICOS EN SECUENCIAS DE VIDEO MEDIANTE APRENDIZAJE PROFUNDO

M.C. Abimael Guzmán Pando

Doctorado en Ciencias en Ingeniería Electrónica
División de Estudios de Posgrado e Investigación del
Tecnológico Nacional de México/I.T. Chihuahua
Chihuahua, Chih., 2020

Director de Tesis: Dr. Mario Ignacio Chacón Murguía

En este trabajo, se presentan dos propuestas de algoritmos enfocados a detección de objetos dinámicos en video mediante técnicas de aprendizaje profundo. La primera propuesta es llamada 2FraCNN (*Two Frame Convolutional Neural Network*) y consiste en una red convolucional que considera información temporal del video en una arquitectura tipo *Encoder-Decoder*. La arquitectura de 2FraCNN se basa en el algoritmo *FlowNet*, pionero en estimar flujo óptico mediante un esquema supervisado. La segunda propuesta plantea una idea novedosa y original inspirada en la visión biológica de las aves rapaces. El algoritmo es llamado DeepFoveaNet y su esquema modular emula la visión monocular de las aves rapaces mediante dos módulos de redes convolucionales *Encoder-Decoder*, de periferia y de enfoque. La red de periferia se encarga de detectar el movimiento general de la escena y la red de enfoque emula el efecto de magnificación producido en la fovea profunda para captar características de los objetos con mayor detalle.

La evaluación de 2FraCNN realizada en la base de datos *Change Detection (CDnet14)* demostró la robustez del algoritmo propuesto para manejar diferentes situaciones críticas de video con altos índices de desempeño, tales como: fondos dinámicos, ruido en la escena debido al mal clima, sombras, objetos intermitentes, camuflaje, entre otras. 2FraCNN se ubicó dentro los primeros lugares del ranquin de evaluación de *CDnet14*, superó ampliamente a los algoritmos DeepBS, SuBSENSE y PAWCS, y obtuvo resultados competitivos respecto a los primeros cuatro métodos de la base de datos, que son FgSegNet_v2, FgSegNet_S, FgSegNet y CascadeCNN. Además, 2FraCNN se ubicó en el 12° lugar en el instrumento de evaluación *Performance, Velocity, Autoadaptability, Documentation and Novelty (PVADN)*, demostrando ser un algoritmo competente respecto a una evaluación que considera aspectos tanto prácticos como teóricos.

Por otro lado, los resultados cuantitativos y cualitativos de DeepFoveaNet mostraron que el uso de la red de fovea profunda ayudó en la correcta detección de objetos dinámicos, incluso los más pequeños y evitó que estos se perdieran durante los cuadros del video. DeepFoveaNet logró situarse en la séptima posición en el ranking de desempeños sobre todas las categorías de video de la base de datos *CDnet14*. Los mejores resultados fueron en las categorías de video; sombras, básicos, cámaras térmicas y movimiento de cámaras, con valores de F-Measure mayores a 0.98. Además, demostró consistencia en sus resultados al ser evaluado mediante los videos de la base de datos LASIESTA, aun y cuando en la gran mayoría de videos solo se usaron 100 cuadros para su entrenamiento.

A diferencia de algunos de los algoritmos del estado del arte que ocupan los primeros lugares de la base de datos Change Detection, DeepFoveaNet y 2FraCNN no dependen de modelos de red previamente entrenadas, algoritmos generadores de fondo y una extensa cantidad de imágenes para entrenamiento. Además, su arquitectura permite que la red aprenda información espaciotemporal del video e incluyen un esquema de balanceo de clases novedoso para proveer una correcta convergencia.

CONTENIDO

LISTA DE FIGURAS	<i>x</i>
LISTA DE TABLAS.....	<i>xvi</i>
GLOSARIO.....	<i>xvii</i>
INTRODUCCIÓN	1
CAPÍTULO I. ANTECEDENTES GENERALES EN DETECCIÓN DE MOVIMIENTO, APRENDIZAJE PROFUNDO Y PERCEPCION DE MOVIMIENTO.....	3
1.1. TÉCNICAS ENFOCADAS A DETECCIÓN DE MOVIMIENTO	3
1.1.1. Modelos básicos	3
1.1.2. Modelos combinados.....	5
1.1.3. Modelos estadísticos	6
1.1.4. Modelos estadísticos avanzados.....	6
1.1.5. Modelos difusos	9
1.1.6. Modelos de subespacios.....	10
1.1.7. Diccionarios de aprendizaje	11
1.1.8. Modelos de transformación de dominio.....	12
1.1.9. Modelos tensores robustos	13
1.1.10. Modelos basados en redes neuronales artificiales.....	14
1.1.11 Otros.	15
1.1.12. Análisis de tendencias	16
1.2. VALIDACIÓN DE ALGORITMOS DE DETECCIÓN DE MOVIMIENTO.....	18
1.2.1. Métricas para evaluación de desempeño	20
1.2.2. Bases de datos	24
1.3. APRENDIZAJE DE MÁQUINA	27
1.3.1. Aprendizaje profundo.....	28
1.3.1.1. Redes neuronales convolucionales.....	30
1.3.1.1.1. Formalización.....	32
1.3.1.1.2. Capas de agrupamiento o <i>pooling</i>	34
1.3.1.1.3. Capas totalmente conectadas (<i>fully conected</i>).....	36

1.3.1.1.4. Funciones de activación	37
1.3.1.1.5. Funciones de pérdida.....	39
1.3.1.1.6. Técnicas de regularización.....	42
1.3.1.1.7. Técnicas de optimización.....	48
1.4 PERCEPCIÓN DE MOVIMIENTO MEDIANTE LA VISIÓN BIOLÓGICA.....	52
1.4.1. Aspectos neurofisiológicos de la percepción de movimiento	52
1.4.1.1. Detectores de contraste.....	54
1.4.1.2. Orientación	55
1.4.1.3. Movimiento	56
1.4.2. Modelo de una neurona simple mediante filtros de Gabor	58
1.4.3. Modelo de una neurona compleja mediante filtros de Gabor 3D	60
CAPÍTULO II. ALGORITMOS DE APRENDIZAJE PROFUNDO ENFOCADOS A DETECCIÓN DE MOVIMIENTO.....	64
2.1. RED AUTOORGANIZADAS MULTICAPA EN MODELADO DE FONDO (SMSOM-BM).	64
2.2. MODELO NEURONAL DE RESPUESTA MIXTA (NeRM)	66
2.3. SUSTRACCIÓN DE FONDO MEDIANTE RED NEURONAL CONVOLUCIONAL (CONVNET)	67
2.4. ARQUITECTURA PROFUNDA DE APARIENCIA Y MOVIMIENTO (A-Net/M-Net)	68
2.5. FOREGROUND SEGMENTATION NETWORK (FGSEGNET).....	69
2.6. RED CONVOLUCIONAL EN CASCADA (CASCADECNN).....	70
2.7. RED CONVOLUCIONAL PARA SUSTRACCIÓN DE FONDO (DEEPBS)	71
2.8. MODELADO DE FONDO CON REDES CONVOLUCIONALES DILATADAS (ATROUSCNN)	72
2.9. REDES CONVOLUCIONALES 3D PARA DETECCIÓN DE MOVIMIENTO (3DCNN).....	74
2.10. FLUJO ÓPTICO MEDIANTE REDES CONVOLUCIONALES (<i>FLOWNET</i>).....	76
2.11. RESUMEN DEL CAPÍTULO	79
CAPÍTULO III. ALGORITMOS DE DETECCIÓN DE MOVIMIENTO INSPIRADOS EN LA VISIÓN BIOLÓGICA	81
3.1. VECINDARIO RETINOTÓPICO MEDIANTE RED SOM	81
3.2. RED AUTOORGANIZADA CON ADAPTACIÓN DE RETRASO	82
3.3. RED NEURONAL DE PERCEPCIÓN DE MOVIMIENTO ROTACIONAL	83
3.4. PREVENCIÓN DE COLISIONES BASADO EN EL DETECTOR DE MOVIMIENTO LGMD	84

3.5. DETECCIÓN DE MOVIMIENTO INSPIRADA EN LA ESTRUCTURA VISUAL DE LA MOSCA	85
3.6. ALGORITMO BASADO EN NEURONA STDM SENSIBLE A PEQUEÑOS MOVIMIENTOS	86
3.7. RED RETINOTOPICA AUTOORGANIZADA (RESOM)	87
3.8. RED DE SUMATORIA BINOCULAR PARA DETECCIÓN DINÁMICA (X-NET)	89
3.9. RESUMEN DEL CAPÍTULO	90
 CAPÍTULO IV. RED NEURONAL CONVOLUCIONAL PARA DETECCIÓN DE MOVIMIENTO CON INFORMACIÓN TEMPORAL (2FRACNN).....	92
4.1. ARQUITECTURA INSPIRADA EN <i>FLOWNET</i> PARA DETECCIÓN DE MOVIMIENTO	93
4.2. ESQUEMA DE ENTRENAMIENTO	100
4.3. IMPLEMENTACIÓN	104
4.4. EVALUACIÓN DEL MODELO PROPUESTO Y ANÁLISIS DE LOS RESULTADOS	105
4.4.1 Evaluación en CDnet14.....	105
4.4.2 Evaluación en PVADN	109
4.5. CONCLUSIONES DEL MÉTODO 2FRACNN	110
 CAPÍTULO V. ALGORITMO DE DETECCIÓN DE MOVIMIENTO BASADO EN LA VISIÓN BIOLOGICA DE LAS AVES RAPACES (DEEPFOVEANET).....	112
5.1. VISIÓN BIOLÓGICA DE LAS AVES RAPACES.....	113
5.2. ALGORITMO BIOINSPIRADO EN LA VISIÓN BIOLÓGICA DE LAS AVES RAPACES.....	116
5.2.1. Red convolucional de periferia/Arquitectura (<i>Peripheral-CNN</i>).....	117
5.2.2. Red convolucional de fovea profunda/Arquitectura (<i>Deepfovea-CNN</i>)	118
5.2.3. Esquema de entrenamiento.....	121
5.2.4. Etapa de agregación	123
5.2.5. Reubicación del cuadrante de interés	123
5.2.6. Implementación del algoritmo propuesto.....	124
5.3. EVALUACIÓN DEL MODELO PROPUESTO Y ANÁLISIS DE LOS RESULTADOS	126
5.3.1. Evaluación en <i>CDnet14</i>	126
5.3.2. Evaluación en LASIESTA	131
5.4. ANÁLISIS DE LAS CARACTERÍSTICAS APRENDIDAS POR LOS FILTROS DE LAS REDES PERIPHERAL-CNN Y DEEPFOVEA-CNN QUE COMPONEN AL ALGORITMO DEEPFOVEANET	134

5.4.1. Discusión y conclusiones acerca de los filtros aprendidos de las redes Peripheral-CNN y Deepfovea-CNN del algoritmo DeepFoveaNet	165
5.5. CONCLUSIONES DEL MÉTODO DEEPFOVEANET	168
CAPÍTULO VI. RESULTADOS Y CONCLUSIONES GENERALES	169
REFERENCIAS	172

LISTA DE FIGURAS

Figura 1.1. Porcentaje de uso de los diferentes tipos de técnicas dentro de los 79 algoritmos analizados.	17
Figura 1.2. a) GT, b) resultado obtenido de algún algoritmo de detección de movimiento, c) comparación de la imagen a) con b), VP (zonas rojas), FP (zonas azules), FN (zonas amarillas) y VN (zonas negras).....	19
Figura 1.3. Técnicas tradicionales de aprendizaje de máquina en comparación con técnicas de <i>Deep learning</i>	29
Figura 1.4. Características aprendidas por los filtros a través del entrenamiento [129].	32
Figura 1.5. Modelos de neuronas: a) neurona biológica, b) neurona artificial.	33
Figura 1.6. Capas de <i>Average pooling</i> y <i>Max pooling</i>	35
Figura 1.7. Invarianza básica a la translación al aplicar capas de <i>pooling</i> . Los pixeles con valor de “1” muestran el patrón y las casillas sin valor numérico en la imagen poseen valor de cero. Los cambios de color muestran las posiciones sobre las que se desplaza la ventana de <i>max pooling</i> de tamaño 3x3.....	35
Figura 1.8. Invarianza básica a la rotación al aplicar capas de <i>max pooling</i> . Los pixeles con valor de “1” muestran el patrón y las casillas sin valor numérico en la imagen tienen valor de cero. Los cambios de color muestran las posiciones sobre las que se desplaza la ventana de <i>max pooling</i> de tamaño 3x3.	36
Figura 1.9. Ejemplo de una capa totalmente conectada.	36
Figura 1.10. Gráficas función de pérdida Hinge.	40
Figura 1.11. Gráficas función de pérdida <i>BinaryCrossEntropy</i>	42
Figura 1.12. Situaciones de <i>overfitting</i> , correcta generalización y <i>underfitting</i>	43
Figura 1.13. Gráfica de la ecuación 33 con $Ct=1$; a) caso dos dimensiones We , b) caso tres dimensiones We	44
Figura 1.14. Efecto de regularización L1 en función de pérdida.	45
Figura 1.15. Efecto de regularización L2 en función de pérdida.	46
Figura 1.16. Comparación de una red estándar a) y de una red con <i>dropout</i> b).	47
Figura 1.17. Red estándar y red con <i>Dropout</i>	47
Figura 1.18. Importancia de la selección correcta del parámetro η	49
Figura 1.19. Comportamiento del algoritmo SGD con momento.	50
Figura 1.20. Patrones de movimiento.....	52
Figura 1.21. Experimentos de Hubel y Wiesel [139], 1959, para entender como el cerebro toma la información de los ojos y la transforma en la percepción visual compleja de nuestro mundo.	53
Figura 1.23. a) Representación esquemática de la estructura de la retina. La luz penetra por la izquierda y atraviesa diferentes capas celulares hasta alcanzar las células fotorreceptoras (conos y bastones). b) respuestas de las células ganglionares a un determinado estímulo de luz.....	54
Figura 1.24. Neuronas simples preferentes a distintas orientaciones: a) 0°, b) 45° y c) 90°.	55
Figura 1.25. Disparos de una neurona simple (con preferencia a la orientación horizontal, 0°) al ser excitada con un patrón de luz en diferentes orientaciones.....	56
Figura 1.26. Neurona compleja sensible al movimiento hacia la izquierda.....	56

Figura 1.27. Integración de respuestas individuales de neuronas CN en el área MT para la estimación del movimiento.	57
Figura 1.28. Funciones de Gabor 2D modelando los campos receptivos de las neuronas simples.....	59
Figura 1.29. Extracción de características mediante filtros de Gabor. a) Imagen de entrada, b) Banco de filtros de Gabor 2D, c) Características extraídas. Imagen modificada de [154].	60
Figura 1.30. Secuencia de movimiento representada en un patrón espacio temporal que corresponde a una velocidad u orientación específica. a) barra vertical en punto inicial antes de moverse, b) barra vertical en su punto final después del movimiento, c) muestreo del movimiento de la barra vertical a través del tiempo, d) patrón espacio temporal generado cuya orientación corresponde a la velocidad de desplazamiento horizontal de la barra vertical. Las imágenes e), f), g) y h) muestran una barra horizontal desplazándose hacia arriba, ejemplificando el caso en desplazamientos verticales.	61
Figura 1.31. Filtro de Gabor espaciotemporal aplicado en detección de movimiento.....	62
Figura 1.32. Modelo espaciotemporal para la extracción de la energía del movimiento.....	63
Figura 2.1. Detección de movimiento por SMSOM-BM [84].	65
Figura 2.2. Esquema general del NeRM [85]......	66
Figura 2.3. Esquema del proceso de clasificación de un pixel centrado en un parche de tamaño $T_a \times T_a$ [86].	67
Figura 2.4. Estructura del modelo A-Net y M-Net [87]......	68
Figura 2.5. Estructura de FgSegNet [10]......	69
Figura 2.6. Estructura de CascadeCNN [8]......	70
Figura 2.7. Arquitectura de red CNN [8].	70
Figura 2.8. Sustracción de fondo con DeepBS [88]......	71
Figura 2.9. Convolución <i>atrous</i> con <i>kernel</i> de tamaño 3×3 y diferentes tasas de dilatación <i>dl</i>	73
Figura 2.10. Arquitectura de la red con capas tipo <i>atrous</i> para detección de movimiento.	74
Figura 2.11. Comparación de una convolución 2D y una en 3D.	74
Figura 2.12. Arquitectura de la red 3D para detección de movimiento.	75
Figura 2.13. Representaciones distintas de vectores de movimiento.	76
Figura 2.14. Problemas de apertura de área y brillo constante en flujo óptico.	77
Figura 2.15. problema de apertura de área en flujo óptico.....	77
Figura 2.16. Etapa de refinamiento en <i>FlowNet</i>	78
Figura 2.17. Arquitectura FlowNet 2.0.	79
Figura 3.1. Estimulo moviéndose de forma horizontal en el tiempo a) activa la célula <i>a</i> que proyecta conexiones excitatorias a un grupo cercano de células en la capa L2. En el tiempo b) el estímulo cambia de posición y activa la célula <i>e</i> que propaga su excitación a las células <i>h</i> , <i>g</i> y <i>f</i> . Solo <i>f</i> es fuertemente estimulada por las células <i>b</i> y <i>e</i> indicando la dirección del movimiento horizontal. Imagen tomada de [161].	82
Figura 3.2. Ajuste de posición de retraso, figura tomada de [162]. Cada figura muestra retrasos de tiempo en las conexiones de cada locación de la capa de retina con una neurona particular en la capa de corteza visual.	83

Figura 3.3. Mapeo de selectividad direccional, figura tomada de [162]. El eje x y y indican locación en la capa de corteza visual. En cada locación, se representa la respuesta de una neurona cortical. La longitud de la flecha representa la fuerza de la respuesta de la neurona al estímulo que se mueve en la dirección de esta. Las neuronas sin flechas no obtuvieron respuesta al estímulo. a) mapa inicial, un poco desorganizado. b) Mapa después del entrenamiento, el mapa se vuelve más organizado con diferentes áreas que responden a distintos estímulos de entrada y con transiciones suaves entre áreas.	83
Figura 3.4. Ilustración esquemática del algoritmo [163] con arreglo de 16 neuronas selectivas a la dirección del movimiento.	84
Figura 3.5. Red neuronal artificial basada en la estructura visual de la mosca [165].	85
Figura 3.6. Red neuronal artificial STDM [166].	86
Figura 3.7. Ilustración esquemática del mapeo de los pixeles o fotoreceptores. Cada cuadrícula denota un pixel que corresponde a un fotoreceptor, los rectángulos punteados en rojo representan el campo receptivo de un omatidio [166].	87
Figura 3.8. Arquitectura RESOM [33], cada vector W_m^z contiene 4 neuronas	87
Figura 3.9. Detección de movimiento mediante RESOM [33].	89
Figura 3.10. Forma de equis en el nervio óptico del sistema de visión binocular humana, [167].	90
Figura 3.11. Arquitectura X-Net, fusión de capas Encoder-Decoder emulando la estructura binocular del nervio óptico [167].	90
Figura 4.1. Esquema general del algoritmo propuesto 2FraCNN; a) etapa Encoder, b) etapa Decoder.	94
Figura 4.2. Propagación hacia adelante etapa <i>Encoder</i>	97
Figura 4.3. Elementos involucrados en la parte <i>Decoder</i> de la arquitectura.	97
Figura 4.4. Proceso de obtención de la capa CONV1.	100
Figura 4.5. Desempeño en F-Measure contra la proporción promedio de número de pixeles de fondo por cada pixel de objeto dinámico en cada video.	103
Figura 4.6. Aumento de datos realizando copias del objeto de interés. a) cuadro anterior, b) cuadro actual y c) GT original. d), e) y f) imágenes modificadas con copias.	104
Figura 4.7. Mejoras en los resultados después de aumento de datos y descarte de cuadros. RAAD: resultados antes de aumento de datos y descarte de cuadros, RDAD: resultados después de aumento de datos y descarte de cuadros, videos: a) port_0_17fps, categoría lowFramerate, b) turbulence1 y c) turbulence2, categoría turbulence.	107
Figura 4.8. Resultados de segmentación de 2FraCNN en los videos, a) skating, b) twoPositionPTZCam, c) office, d) badminton, e) canoe, f) sofa, g) turnpike_0_5fps, h) winterStreet, i) copyMachine, j) library y k) Turbulence3.	108
Figura 4.9. Resultados de 2FraCNN en PVADN.	110
Figura 5.1. Comparación de la estructura de un ojo de águila contra un ojo humano.	113
Figura 5.2. Estructura de doble fovea en el ojo del águila. (a) águila calva. (b) vista superior de la cabeza seccionada del águila para mostrar ubicación de ambas foveas.	114

Figura 5.3. Imágenes histológicas de foveas seccionadas. Izquierda: fovea profunda de un Sand martin europeo. Derecha: fovea poco profunda de un águila dorada. Imagen de [182].....	115
Figura 5.4. Vista superior del sistema de visión del águila calva haciendo énfasis en la estructura del ojo y la magnificación de imagen producida en la fovea profunda...	115
Figura 5.5. Comparación de los lentes de una telefoto contra la estructura de la fovea profunda de un ave rapaz. a) lentes de una telefoto. b) El lente negativo es remplazado por la superficie convexa de la fovea profunda [187].	115
Figura 5.6. Esquema general del algoritmo propuesto basado en la visión biológica de las aves rapaces.	117
Figura 5.7. Ubicaciones posibles de atención visual. a) división de los cuadros de entrada en cuadrantes, b) acercamiento aplicado a la imagen de entrada en el cuadrante C_2 seleccionado, emulando el efecto producido por la fovea profunda de las aves rapaces.	119
Figura 5.8. Esquema de arquitectura de la red de fovea profunda.	120
Figura 5.9. Cuadros de entrenamiento utilizados en la red de periferia.	121
Figura 5.10. Proceso de extracción de cuadros de entrenamiento utilizados en la fovea profunda.	122
Figura 5.11. Diagrama funcional del algoritmo propuesto basado en la visión biológica de las aves rapaces.	125
Figura 5.12. Comparación de detección de los resultados del modelo <i>Peripheral-CNN</i> (periferia) y <i>DeepFoveaNet</i> usando fovea profunda.	128
Figura 5.13. Comparación de los resultados de detección de <i>DeepFoveaNet</i> contra algoritmos del estado del arte en un video por cada categoría de la base de datos <i>CDnet14</i> . Videos, BW: Badweather, BL: Baseline, CJ: CameraJitter, DB, DynamicBackground, IO: Intermitent Object Motion, LF: LowFramerate, NV: Night Videos, PT: PTZ, SH: Shadow, TH: Thermal, TU: Turbulence.	130
Figura 5.14. Arquitecturas de las redes <i>Peripheral-CNN</i> y <i>Deepfovea-CNN</i>	134
Figura 5.15. Mapas de características primera capa convolucional CONV1 red <i>Peripheral-CNN</i>	135
Figura 5.16. Mapas de características primera capa convolucional CONV1 red <i>Deepfovea-CNN</i>	136
Figura 5.17. Ampliación: mapas de características generados por filtros sensibles a bordes primera capa convolucional CONV1 red <i>Peripheral-CNN</i>	137
Figura 5.18. Ampliación: mapas de características generados por filtros sensibles a bordes primera capa convolucional CONV1 red <i>Deepfovea-CNN</i>	137
Figura 5.19. Ampliación: mapas de características generados por filtros sensibles al color o textura en la capa convolucional CONV1 red <i>Peripheral-CNN</i>	138
Figura 5.20. Ampliación: mapas de características generados por filtros sensibles al color o textura en la capa convolucional CONV1 red <i>Deepfovea-CNN</i>	138
Figura 5.21. Ampliación: mapas de características generados por filtros sensibles a rasgos de los objetos de interés en la capa convolucional CONV1 red <i>Peripheral-CNN</i>	139

Figura 5.22. Ampliación: mapas de características generados por filtros sensibles a rasgos de los objetos de interés en la capa convolucional CONV1 red <i>Deepfovea-CNN</i>	139
Figura 5.23. Ampliación: mapas de características generados por filtros sensibles al movimiento en la capa convolucional CONV1 red <i>Peripheral-CNN</i>	140
Figura 5.24. Ampliación: mapas de características generados por filtros sensibles al movimiento en la capa convolucional CONV1 red <i>Deepfovea-CNN</i>	140
Figura 5.25. Mapas de características generados por filtros ETG con preferencia al movimiento hacia la izquierda a diferentes velocidades: $v=1$, $v=2$, $v=3$ y $v=4$	141
Figura 5.26. Comparación de mapas de características; a) mapa generado por el filtro 27 aprendido por la red <i>Peripheral-CNN</i> en la capa CONV1, b) mapa generado con un filtro ETG para extracción de movimiento.....	142
Figura 5.27. Estructura del filtro 27 aprendido por la red en la capa CONV1; a) plano XYZ, b) plano XZ.	143
Figura 5.28. Mapas de características capa convolucional CONV2 red <i>Peripheral-CNN</i>	143
Figura 5.29. Mapas de características capa convolucional CONV2 red <i>Deepfovea-CNN</i>	144
Figura 5.30. Mapas de características capa convolucional CONV3 red <i>Peripheral-CNN</i>	144
Figura 5.31. Mapas de características capa convolucional CONV3 red <i>Deepfovea-CNN</i>	145
Figura 5.32. Mapas de características capa convolucional CONV4 red <i>Peripheral-CNN</i>	145
Figura 5.33. Mapas de características capa convolucional CONV4 red <i>Deepfovea-CNN</i>	146
Figura 5.34. Mapas de características capa convolucional CONV5 red <i>Peripheral-CNN</i>	146
Figura 5.35. Mapas de características capa convolucional CONV5 red <i>Deepfovea-CNN</i>	147
Figura 5.36. Mapas de características capa convolucional CONV6 red <i>Peripheral-CNN</i>	147
Figura 5.37. Mapas de características capa convolucional CONV6 red <i>Deepfovea-CNN</i>	148
Figura 5.38. Mapas de características capa convolucional CONV7 red <i>Peripheral-CNN</i>	149
Figura 5.39. Mapas de características capa convolucional CONV7 red <i>Deepfovea-CNN</i>	150
Figura 5.40. Mapas de características capa convolucional CONV8 red <i>Peripheral-CNN</i>	150
Figura 5.41. Mapas de características capa convolucional CONV8 red <i>Deepfovea-CNN</i>	151
Figura 5.42. Mapa de características capa ESTIMATION1. Redes: a) <i>Peripheral-CNN</i> , b) <i>Deepfovea-CNN</i>	152
Figura 5.43. Capa CONCAT1 de las redes <i>Peripheral-CNN</i> y <i>Deepfovea-CNN</i>	153
Figura 5.44. Mapas de características capa TCONV1 red <i>Peripheral-CNN</i>	153
Figura 5.45. Mapas de características capa UPSAMPLE1 red <i>Peripheral-CNN</i>	154
Figura 5.46. Concatenación CONCAT1 de los mapas de características TCONV1, CONV4 y UPSAMPLE1.....	154
Figura 5.47. Mapas de características capa TCONV1 red <i>Deepfovea-CNN</i>	155
Figura 5.48. Mapas de características capa UPSAMPLE1 red <i>Deepfovea-CNN</i>	156
Figura 5.49. Mapas de características capa CONCAT1 red <i>Deepfovea-CNN</i>	156
Figura 5.50. Mapas de características, capa ESTIMATION2: a) <i>Peripheral-CNN</i> , b) <i>Deepfovea-CNN</i>	157
Figura 5.51. Mapas de características capa TCONV2 red <i>Peripheral-CNN</i>	157
Figura 5.52. Mapas de características capa TCONV2 red <i>Deepfovea-CNN</i>	158
Figura 5.53. Mapas de características, UPSAMPLE2: a) red <i>Peripheral-CNN</i> , b) <i>Deepfovea-CNN</i>	158

Figura 5.54. Información de bajo y alto nivel utilizada en la capa de concatenación, CONCAT2 de los mapas de características TCONV2, CONV3 y UPSAMPLE2 en la red <i>Peripheral-CNN</i>	159
Figura 5.55. Información de bajo y alto nivel utilizada en la capa de concatenación CONCAT2 en la red <i>Deepfovea-CNN</i>	160
Figura 5.56. Mapas de características, capa ESTIMATION3: a) red <i>Peripheral-CNN</i> , b) red <i>Deepfovea-CNN</i>	161
Figura 5.57. Mapas de características capa TCONV3 red <i>Peripheral-CNN</i>	161
Figura 5.58. Mapas de características capa UPSAMPLE3 red <i>Peripheral-CNN</i> y capa final, red <i>Deepfovea-CNN</i>	162
Figura 5.59. Información de bajo y alto nivel utilizada en la capa de concatenación CONCAT3 en la red <i>Peripheral-CNN</i>	162
Figura 5.60. Mapas de características capa DETECTION.....	163
Figura 5.61. Mapas de características capa DETECTION después de la función sigmoideal en red <i>Peripheral-CNN</i>	163
Figura 5.62. Combinación de las estimaciones de ambas redes (<i>Peripheral-CNN</i> y <i>Deepfovea-CNN</i>) para obtener la detección final del movimiento.	164
Figura 5.63. Interpretación de las características aprendidas por la red AlexNet, imagen tomada de [205]......	166

LISTA DE TABLAS

Tabla 1.1. Clasificación por tipo de técnica de los 79 algoritmos analizados.....	4
Tabla 1.2. Soluciones y retos por tipo de técnica.....	18
Tabla 1.3. Matriz de confusión.....	20
Tabla 1.4. Bases de datos encontradas en la literatura.	26
Tabla 1.5. URL de bases de datos encontradas en la literatura.	26
Tabla 1.6. Características y tipos de funciones de activación.....	38
Tabla 2.1. Resumen de las técnicas analizadas de aprendizaje profundo para detección de movimiento.....	80
Tabla 4.1. Configuración de Hiperparámetros parte Encoder.....	99
Tabla 4.2. Configuración de Hiperparámetros parte Decoder.	100
Tabla 4.3. Resultados del algoritmo propuesto 2FraCNN para todos los videos de la base de datos <i>CDnet14</i>	106
Tabla 4.4. Comparación de los desempeños de 2FraCNN en la métrica F-Measure.....	106
Tabla 4.5. Resultados de 2FraCNN en el instrumento de evaluación PVADN. Los valores en corchetes son el intervalo de valores de la métrica.....	110
Tabla 5.1. Configuración de Hiperparámetros de la red de fovea profunda en la parte Encoder.	120
Tabla 5.2. Configuración de Hiperparámetros de la red de fovea profunda en la parte Decoder.	120
Tabla 5.3. Comparación de resultados de los modelos propuestos, evaluando la red de periferia <i>Peripheral CNN</i> y el modelo completo <i>DeepFoveaNet</i> (con ambas redes, de periferia y de fovea profunda) para todos los videos de la evaluación parcial de <i>CDnet14</i>	126
Tabla 5.4. Resultados del modelo propuesto sobre todos los videos de <i>CDnet14</i>	128
Tabla 5.5. Resultados del modelo propuesto sobre todos los videos de <i>LASIESTA</i>	132
Tabla 5.6. Resultados de algoritmos evaluados en la base de datos <i>LASIESTA</i>	132
Tabla 5.7. Resultados de DeepFoveaNet en la métrica F-Measure sobre todos los videos de <i>LASIESTA</i>	133

GLOSARIO

<i>Símbolo</i>	<i>Descripción</i>
a	Tasa que especifica elipticidad de la envolvente Gaussiana en el dominio espacial.
Act	Tipo de función de activación
b	Sesgo o bias
bk	El modelo del fondo bk se obtiene con el promedio de los valores de los píxeles en el diccionario de fondo bl sobre una cierta longitud de memoria bm .
bl	Diccionario de aprendizaje en algoritmo DeepBS.
bm	Longitud de memoria adaptiva en DeepBS.
C	Cuadrante de video
$CEPs$	Número de píxeles erróneos cuyos vecinos con conectividad cuatro también son píxeles de error, en métrica pCEPs.
Ci_1, Ci_2	Variables incluidas en la métrica SSIM para evitar la inestabilidad.
Cm	Capas de convolución para la parte decoder
Con_m	Capas de concatenación para la parte decoder
Ct	Constante.
$D(x,y,t)$	Detección final binarizada
$Det_{Fo}(x,y,t)$	Salida de la red de fovea profunda
$Det_M(x,y,z)$	Integración de las salidas de las redes mediante el valor máximo de $Det_{Pe}(x,y,t)$ y $Det_{Fo}(x,y,t)$
$Det_{Pe}(x,y,t)$	Salida de la red de periferia
d_{iq}	Distancia de cada nodo q respecto al píxel i .
dl	Tasa de dilatación en redes de convolución Atrous.
DT	Distancia mínima entre ip e i para la métrica D-Score.
E	Función de pérdida del error cuadrado medio.
E	Función de pérdida asociada al error de entrenamiento
E_{Hinge}	Función de pérdida <i>Hinge loss</i> .
Eps	Píxeles de error en métrica pEPs
f	Frecuencia de modulación de la onda senoidal
$F(x,y,z)$	Concatenación de dos imágenes sucesivas respecto al eje z
Fi	Tamaño del filtro de convolución o deconvolución
$fm_{i,j}$	Mapa de características obtenido al realizar la operación de convolución de cada capa i con el j -ésimo kernel K .
Fmp	Mapa de características generado en una capa de <i>pooling</i>

$fmT(x,y,z)$	Mapa de características generado al realizar la operación de deconvolución
FN	Falso negativo
f_o	Frecuencia de modulación inicial.
FP	Falso positivo
Fs	Porcentaje de píxeles en movimiento en DeepBS.
G	La imagen original o el mapa de características de capas de convolución previas.
g	Valor de los gradientes cuadrados.
$Gabor_{1D}$	Función de Gabor en una dimensión.
$Gabor_{2D}$	Función de Gabor en dos dimensiones.
$Gabor_{3D}$	Función de Gabor en tres dimensiones.
GT	Ground truth
H_{Ent}	Entropía de la de señal.
i	Pixel de la imagen I
$I(t)$	Imagen en el tiempo t .
Ib	Promedio de $V^{m,t}_{x,y,z}$
Id	Imagen en RESOM usada para inhibir información aprendida de objetos dinámicos
I_E	Imagen del modelo de fondo estimado usado en la métrica AGE
I_e	Imagen de detección de objetos dinámicos en RESOM
I_{GT}	Imagen de GT usada en la métrica AGE
ip	Pixel de imagen binaria en la métrica D-Score.
iw, ih	Posiciones que se recorren dentro de la ventana con tamaño Wk y Hk en la operación de <i>pooling</i>
J	Función de costo.
∇J	Gradiente del error
j	Índice iterador
$J_{BinaryCrossEntropy}$	Función de costo <i>binary cross entropy</i> .
J_{Hinge}	Función de costo <i>Hinge loss</i> .
k	Índice iterador
Kd	kernel de deconvolucion
Ke	Kernel, ventana o filtro de convolución
km_1, km_2	Constantes pequeñas con valor menor a 1 en Ci_1 y Ci_2 .
L	Función de costo regularizada.
$\tilde{Y}_l$	Es la salida “disminuida” o “adelgazada” generada al multiplicar los valores del vector r_l con la salida Y_l
l	Índice
L_l	Función de costo <i>binary crossentropy</i>
Lm	Rango dinámico de valor de gris en Ci_1 y Ci_2 .
m	Parámetro adaptivo de momento.
M_{PSNR}	Máximo número de nivel de gris en métrica PSNR.

MSE	Error cuadrático medio en métrica PSNR.
n	Evento discreto.
nc	Clases {1: objeto en movimiento, 2: fondo}
N_{clases}	Número total de clases jc presentes
N_{Datos}	Número de datos
N_f	Número de iteraciones que define la capacidad de aprendizaje por cuadro en algoritmo RESOM
N_f	Número de filtros Ke en el proceso de convolución
Ng	Número total de cuadros de GT
ni	Número de iteraciones que la red procesa en RESOM
Np	Número total de píxeles en la imagen.
n_p	Posición del píxel i en ConvNet.
Np	Número total de píxeles en la imagen
n_r	Índice refractivo de la superficie de la fovea profunda.
Ns	Número de píxeles de objetos en movimiento en la imagen de acuerdo con DeepBS.
N_{Total}	Número total de píxeles en una imagen en la métrica pEPs
n_v	Índice refractivo del humor vítreo
Nw	Número total de pesos o parámetros We
Obj	Conjunto de objetos u
O_M	Objeto con área menor
$P(i)$	La probabilidad estimada por la red para que el píxel q pertenezca a un objeto dinámico.
P, Q	Distribuciones de probabilidad.
pad	Agregación de ceros al mapa de características
P_{cnc}	Factor de ponderación de la clase jc para balanceo de clases
$posx$	Parámetro para ubicar el objeto O_M en el eje X
$posy$	Parámetro para ubicar el objeto O_M en el eje Y
q	Nodo en el aprendizaje del modelo SMSOM
$Q(i)$	La probabilidad del píxel q en la imagen del GT de pertenecer a un objeto dinámico
q^*	Nodo ganador en el aprendizaje del modelo SMSOM
r	Término de regularización que se agrega a la función $LPérdida$
Ra	Radio de la curvatura de la superficie
$re_{v,\theta,\psi}(x,y,t)$	Respuesta espaciotemporal de un filtro de Gabor 3D.
r_l	Es un vector de variables aleatorias independientes
$R_{v,\theta}$	Energía del movimiento obtenida mediante filtros de Gabor 3D.

S	Cantidad de pasos o <i>strides</i> de convolución
sd	Distancia de la cúspide a la curvatura de la fóvea profunda
Sp	Suma de p_1 y p_2 para localizar el cuadrante C del objeto O_M
t	Variable temporal.
T	Desviación estándar de función Gabor 3D.
T_a	Longitud en pixel del tamaño de un parche en ConvNet.
th	Parámetro de umbral de binarización
Tm	Mapa de características transpuesto
Tm	Capas de deconvolución para la parte decoder
u	Objeto o blob en $D(x,y,t)$ con área entre 10 y 100 píxeles
Um	Capas de muestro hacia arriba parte decoder
u, v	coordenadas espaciales de ancho y alto del kernel K .
v	Velocidad de movimiento o fase del factor cosenoidal
$V(t)$	Parámetro del paso de actualización de pesos, en el algoritmo gradiente descendente.
$V^{m,t}_{x,y,z}$	Imagen obtenida por las neuronas R, G, y B en RESOM
VN	Verdadero negativo
VP	Verdadero positivo
W, H, Z	Tamaño en ancho, alto y profundidad.
We	Parámetro asociado a los filtros de la red
Wf, Hf	Tamaño en ancho y alto del filtro Ke
Wn, Hn	Tamaño en ancho y alto del mapa de características generado en el proceso de convolución
W^z_m	Neuronas en red RESOM
x_{in}	Muestra i -ésima de entrada en entrenamiento.
x, y, z	coordenadas espaciales de ancho, alto y profundidad.
y_i	Muestra i -ésima de salida en entrenamiento.
Y_l	Salida l -ésima de la red convolucional
y_p, y_t	y_p es la predicción realizada por la red para un dato q específico y y_t es el <i>target</i> o valor deseado asociado al pixel q .
$Y_{PSNR}, U_{PSNR}, V_{PSNR}$	valores calculados de PSNR en cada una de las bandas Y, U y V respectivamente, en la métrica CQM.
$ZF(x,y,z)$	concatenación $ZF(x,y,z)$ de las dos imágenes con acercamiento $ZI(x,y,z,t)$ y $ZI(x,y,z,t-1)$ en el eje de profundidad z

$ZI(x,y,z,t)$	acercamiento de la imagen actual en la posición del cuadrante seleccionado C
α_P	Parámetro definido experimentalmente en la métrica D-Score.
α_{train}	Parámetro de aprendizaje que se requiere ajustar manualmente para regular la velocidad de actualización.
β_1	Tasa de decaimiento para el parámetro adaptivo de momento m .
β_2	Parámetro de decaimiento para la suma de los gradientes cuadrados g .
β_m^z	Parámetro que caracteriza el comportamiento del vecindario retinotópico en RESOM
γ	Parámetro de momento.
ε	Parámetro que asegura la no división entre cero en algoritmos optimizadores.
η	Parámetro de aprendizaje.
η_m^z	Neuronas con la respuesta más baja en el algoritmo RESOM
λ	Parámetro que controla la contribución de la regularización r
μ_t	Media de la función Gaussiana
ρ	Tasa de decaimiento en algoritmo RMSprop.
ρ_q	Valor obtenido al usar un kernel gaussiano aplicado en la distancia del del nodo q respecto al nodo ganador.
σ	Desviación estándar de la función Gaussiana.
τ	Parámetro de umbral en SMSOM.
Ψ	Offset de fase (en grados).
θ	Es la orientación del eje mayor de la Gaussiana elíptica (en grados).
$\emptyset_t$	Factor de promediado exponencial de los gradientes cuadrados
ϑ	Ancho de señal Gaussiana.
ω_{piq}^l	Peso en el vecindario de nodos de pixeles i en la red SMSOM-BM.
$ z$	Operación de concatenación en el eje z

INTRODUCCIÓN

Los sistemas de visión por computadora enfocados a detectar movimiento en video, han sido extensamente utilizados en aplicaciones de vigilancia, identificación de objetos, análisis de marcha, navegación autónoma, monitoreo, entre otras [1]–[5]. La detección de movimiento consiste en separar para cada uno de los cuadros del video, los píxeles pertenecientes a objetos dinámicos (*foreground*) de las partes de fondo (*background*), lo que deriva en una imagen binaria donde los píxeles de objetos dinámicos suelen tener valor de “1” y “0” los de fondo.

La detección de movimiento no es una tarea sencilla debido a la variedad de cambios que se pueden presentar en un ambiente real no controlado. Kentaro Toyama [6] identificó 10 problemas en el campo de video vigilancia y Thierry Bouwmans [7] extendió la lista a 13 situaciones críticas, entre ellas; ruido en la imagen, vibraciones o movimientos en cámara, uso de cámaras con ajustes automáticos, cambios de iluminación repentina, camuflaje, efecto fantasma, sombras, fondos dinámicos, entre otras.

Recientemente, las técnicas de aprendizaje profundo (*Deep learning*), más específicamente las redes neuronales convolucionales, han demostrado su robustez para afrontar la mayor parte de los problemas de video con altos índices de desempeño [8]–[10]. Aunado a esto, las redes convolucionales destacaron considerablemente en la metodología de evaluación del trabajo de tesis “Análisis y comparación de algoritmos de modelado de fondo” [11]. Esta metodología contempló 119 algoritmos evaluados bajo los criterios desempeño, velocidad, autoadaptabilidad, documentación y actualidad, y actualmente se encuentra publicada en [12].

Por lo anterior, en este trabajo surge el interés por el uso de técnicas de aprendizaje profundo para el desarrollo de un algoritmo de segmentación de objetos dinámicos, además se estudia la detección de movimiento biológica y algoritmos inspirados en el sistema de visión para lograr una propuesta de algoritmo original, robusta y con desempeños competentes respecto al estado del arte para resolver las diferentes situaciones críticas presentes en videos. Esto permitirá explotar los potenciales que ofrece utilizar este tipo de técnica y generar conocimiento que impacte y contribuya de manera significativa en el avance científico.

Para dar a conocer la contribución de este trabajo, el desarrollo de esta tesis se organiza de la siguiente manera.

El capítulo I presenta el estado del arte actual en algoritmos enfocados a detección de movimiento y se introducen los conceptos de aprendizaje de maquina y aprendizaje profundo. Además, se describe el proceso de percepción de movimiento mediante la visión biológica y algunos modelos de energía y espaciotemporales derivados de su estudio.

El capítulo II presenta una descripción y análisis de algoritmos de aprendizaje profundo enfocados a detección de movimiento y se resumen sus características: ventajas, desventajas, modo de trabajo y arquitecturas, con el fin de dar un panorama general al lector sobre el estado del arte actual del tema de interés.

El capítulo III describe y analiza los algoritmos de detección de movimiento basados en la visión biológica. Algunos de estos trabajos se inspiran en el comportamiento de las células complejas que se encuentran en la corteza visual, otros en neuronas en el cerebro de la langosta que responde con mayor excitación a movimientos de objetos que se aproximan e incluso trabajos inspirados en la visión de la mosca.

El capítulo IV presenta la propuesta de un algoritmo de aprendizaje profundo mediante redes convolucionales basado en la arquitectura de *FlowNet* [13]. El modelo propuesto es nombrado 2FraCNN y emplea la información temporal de dos cuadros sucesivos en una arquitectura tipo *Encoder-Decoder* para la estimación de movimiento con altos índices de desempeño.

El capítulo V presenta la propuesta original de un algoritmo de aprendizaje profundo mediante redes convolucionales inspirado en el funcionamiento de la visión biológica de las aves rapaces. El modelo propuesto es llamado DeepFoveaNet y emula la visión monocular de las aves rapaces mediante dos módulos de redes convolucionales *Encoder-Decoder*, que combinan la capacidad de magnificación producida por la fovea profunda en este tipo de aves y la información del contexto contenida en la periferia. Gracias a esto, DeepFoveaNet permite detectar incluso los objetos dinámicos más pequeños.

Finalmente, en el capítulo VI se presentan los resultados y conclusiones generales de este trabajo de tesis.

CAPÍTULO I. ANTECEDENTES GENERALES EN DETECCIÓN DE MOVIMIENTO, APRENDIZAJE PROFUNDO Y PERCEPCIÓN DE MOVIMIENTO

1.1. Técnicas enfocadas a detección de movimiento

Para obtener un panorama amplio sobre los tipos de técnicas que existen actualmente, se llevó a cabo un análisis de trabajos previos enfocados en detección de movimiento del año 2013 a la actualidad. En total se encontraron 79 algoritmos, los cuales se clasificaron y resumieron por tipo de técnica en doce categorías, modelos: básicos, combinados, estadísticos, basados en redes neuronales, estadísticos avanzados, difusos, de subespacios, de transformación de dominio, tensores robustos, aprendizaje profundo y otros. De estas categorías, los algoritmos de aprendizaje profundo son un tema de principal interés en este trabajo, por lo que serán descritos a detalle en el capítulo II. En la Tabla 1.1 se muestra la clasificación de algoritmos propuesta que fue basada en [7], la columna categorías engloba de una manera general el método utilizado para la detección de movimiento y las subcategorías muestran de una manera más específica la técnica utilizada.

A continuación, se da una descripción de las categorías y se resume la información más relevante para cada uno de los diferentes enfoques.

1.1.1. Modelos básicos

Son algoritmos que usan técnicas elementales, entre ellas se encuentran el filtro de promedio, el filtro de mediana y diferencia de cuadro [14]–[16]. Estas técnicas continúan siendo muy utilizadas debido a que son fáciles de implementar, tienen una baja cantidad de parámetros y consumen pocos recursos. No obstante, últimamente se han desarrollado nuevos algoritmos que pueden afrontar una mayor cantidad de situaciones críticas en video. Una de estas técnicas es el filtro de promedio recursivo también llamado aproximación por filtro de promedio o *running average filter*, RAF [14]. Este algoritmo es relativamente acertado para modelar el fondo cuando la iluminación es constante. Sin embargo, en existencia de cambios repentinos de iluminación o fondos dinámicos, resulta en un alto número de falsos positivos. El problema principal radica en que se asume un parámetro de aprendizaje igual para todos los píxeles de la imagen. Por lo que en [17] se propone una adaptación, en la cual el parámetro de aprendizaje es diferente para cada píxel. Este parámetro de aprendizaje también es usado para variar el umbral de binarización en el proceso de detección de movimiento. El método propuesto por Mahmoudpour y Kim [18], usa dos modelos

Tabla 1.1. Clasificación por tipo de técnica de los 79 algoritmos analizados.

CATERGORÍAS	TÉCNICAS	ARTÍCULOS
Básicos (8)	Modelos basados en promedio	[17],[18],[19]
	Modelos basados en mediana	[20],[21],[22],[23]
	Diferencia de cuadro	[24]
Combinados (4)	Combina modelos Gaussianos, las redes SOM y un método que mide cambios de iluminación	[25]
	Combina <i>Multi-view learning</i> , <i>Markov Random Field</i> (MRF) y filtro de mediana temporal	[26]
	Combina KDE, <i>Optical flow</i> y <i>temporal difference</i>	[27]
	Combina <i>Optical flow</i> , GMM y TBMOD	[28]
Estadísticos (2)	Modelo de mezclas Gaussianas autoadaptivas (GMM)	[29]
	<i>Kernel Density Estimation</i> y <i>textons</i>	[30]
Redes Neuronales (8)	<i>Discrete Time Cellular Neural Network</i> (DTCNN)	[31]
	<i>Self-Organizing Maps</i> (SOM)	[32],[33],[34]
	<i>Restricted Boltzmann Machine</i>	[35]
	<i>Weightless Neural Networks</i>	[36]
	<i>Hopfield-type neural networks</i>	[37]
	<i>Radial Basis Function Based</i>	[38]
Modelos Estadísticos Avanzados (22)	Modelo no paramétrico mediante LBP (<i>Local Binary Patterns</i>)	[39]
	Modelos no paramétricos mediante ViBe (<i>Visual Background Extractor</i>)	[40],[41]
	Modelo no paramétrico basando en entropía por bloques	[42]
	Modelo SoAF (<i>Stability of Adaptive Feature</i>)	[43]
	GMM y <i>color space module</i> (CSM)	[44]
	Modelos SBM y STBM (Aquí se proponen dos métodos y son evaluados individualmente)	[45]
	Modelo WeSamBE (<i>Weight-Sample-Based Method for Background Subtraction</i>)	[46]
	Múltiples modelos de fondo MBSV 0	[47]
	Modelo MBS- basado en superpíxeles DBSCAN	[48]
	Métodos basados en AMBER (<i>Adapting Multi-resolution Background Extractor</i>)	[49],[50]
	Mejora al método PBAS	[51]
	Modelo basado en características LRGB	[52]
	Mejora al modelo no paramétrico SURF (<i>Speeded Up Robust Features</i>)	[53]
	Modelo no paramétrico basado en Sift flow	[54]
	Aproximación estocástica	[55]
Modelos Difusos (2)	Modelo de aprendizaje compartido basado en GMM	[56]
	Modelo de mezcla de Gaussianas globalmente espaciadas e incertidumbres (SGGMM)	[57]
Modelos de subespacios (8)	SuBSENSE (<i>Self-Balanced SENSitivity SEGmenter</i>)	[58]
	Modelo basado en <i>Graph cut</i> y GMM	[59]
	Correlograma de <i>Kernel</i> difuso (KFC)	[60]
	Diferencia de histograma de color difuso (FCDH)	[61]
	<i>Block-based RPCA</i> (<i>Robust Principal Component Analysis</i>)	[62]
	RPCA y IRLS (<i>Iteratively Reweighted Least Squares</i>)	[63]
	Mejora al método OR-PCA (<i>Online Robust- Principal Component Analysis</i>)	[64]
	Online Robust PCA via Stochastic Optimization	[65]
	COROLA (<i>Contiguous Outliers Representation via Online Low Rank Approximation</i>)	[66]
	DECOLOR (<i>Detecting Contiguous Outliers in the Low-rank Representation</i>)	[67]
Diccionarios de aprendizaje (6)	Detección de movimiento mediante TVRPCA (<i>Total Variation Regularized RPCA</i>)	[68]
	Descomposición basada en Superpíxeles	[69]
	PAWCS (<i>Pixel-based- Adaptive Word Consensus</i>)	[70]
	Local saliency analysis	[71]
	<i>Superpixel-based codebook</i>	[72]
Modelos de transformación de dominio (3)	BMTDL (<i>Background Modeling Through Dictionary Learning</i>)	[73]
	BREW-DLHPM (<i>Background Representation-Dictionary Learning and Historical Pixel Maintenance</i>)	[74]
	Modelo basado en codificación esparcida K-SVD	[75]
Modelos tensores robustos (2)	Modelo BWM (<i>Background Wavelet Model</i>)	[76]
	Memoria de diferencia de cuadros MFD (<i>Memory-based Frame Differencing</i>) y Wavelet	[77]
	Enfoque basado en Wavelet y LBP(<i>Local Binary Pattern</i>)	[78]
Otros (3)	Tensor de flujo con mezclas Gaussianas particionadas FTSG (<i>Flux Tensor with Split Gaussian models</i>)	[79]
	Tensor bayesiano robusto por medio de factorización BRTF (<i>Bayesian Robust Tensor Factorization</i>)	[80]
Aprendizaje profundo (<i>Deep learning</i>) (11)	Modelo basado en programación genética	[81]
	Modelo basado en controlador PID	[82]
	Modelo basado en CTU (<i>Coding Tree Unit</i>)	[83]
	<i>Stacked Multilayer Self-Organizing Map Background Model</i> (SMSOM-BM)	[84]
	<i>Neural Response Mixture</i> (NeRM)	[85]
79		[86],[87],[10],[8],[88] [89],[9],[13],[90]
Algoritmos en total		

de fondo generados mediante aproximación por filtro de promedio con velocidades de adaptación distinta; baja y alta respectivamente. Además, se cuenta con un parámetro de ganancia de compensación que se ajusta dependiendo del grado de iluminación. En [19], se combinan operaciones morfológicas con el algoritmo recursivo RAF para mejorar la detección de los objetos en movimiento.

Por su parte, los modelos recursivos basados en mediana, utilizan la información actual y posterior de los cuadros de video para generar la estimación del fondo [15]. El principal problema de esta técnica es la lenta recuperación a los cambios repentinos en el escenario de fondo. Como una mejora a este método, en [16] se agregó una variable $1/\alpha$ como una constante de rapidez de cambio. Al variar el valor de alfa, el modelo de fondo se actualiza, más rápido, o más lento. Algunos modelos basados en filtro de mediana recursivo encontrados en la literatura son [20]–[23]. Otro algoritmo básico propuesto en [24], se basa en los métodos *Frame Difference Model* (FDM) y *Background Difference Model* (BDM). En FDM se calcula la diferencia de los cuadros previos y el actual y mediante un umbral se decide si los pixeles pertenecen a objetos dinámicos o fondo. Por otro lado, BDM cuenta para cada pixel de la imagen actual, el número de veces que han sido considerados como fondo en n previas máscaras binarias almacenadas, con el fin de lograr una actualización más rápida del modelo de fondo al sobrepasar un cierto umbral.

1.1.2. Modelos combinados

Estos modelos son una combinación de dos o más técnicas de detección de movimiento y debido a que cada técnica se le da un peso similar, resulta difícil dar una clasificación específica respecto a su técnica. Estos modelos son presentados a continuación.

El método [25] resuelve los cambios repentinos de fuentes de iluminación ya sea naturales o artificiales combinando las características de los algoritmos; mezcla de gaussianas y redes autoorganizadas (SOM por sus siglas en inglés). Una ventaja significativa de esta propuesta es el hecho que no requiere inicialización o entrenamiento complejo. Sin embargo, no alcanza un procesamiento en tiempo real para videos con resolución de 320x240. En [26], se combina las características del filtro de mediana, la técnica *Markov Random Field* (MRF) y aprendizaje por múltiples vistas. El método maneja sombras, cambios de iluminación y ruido suave. Sin embargo, en situaciones como movimientos repentinos de cámara y fondos dinámicos, se obtiene un bajo desempeño. Otro método propuesto en [27] se basa en la combinación de flujo óptico LK y *Kernel Density Estimation*. Cuando la densidad del *Kernel* es más pequeña que un valor de umbral es considerado para ser parte del fondo, mientras que una densidad de *Kernel* grande es considerada para ser parte del modelo de objetos en movimiento. El algoritmo de flujo óptico LK es usado para disminuir ruido en la segmentación final. En [28] se describe el método *monitoring-ensemble-learning detection* (MELD) que integra las características de GMM, flujo óptico LK y *texture-*

based moving object detection (TBMOD), mantiene un bajo costo computacional y es capaz de adaptar los pesos de cada método automáticamente. MELD, permite flexibilidad ya que es posible combinar diferentes algoritmos o enfoques dependiendo de la situación.

1.1.3. Modelos estadísticos

Los modelos estadísticos son una forma matemáticamente formalizada de realizar aproximaciones y hacer predicciones de un fenómeno en específico (en este caso de estudio, decidir si un pixel pertenece o no al objeto en movimiento) considerando el comportamiento aleatorio de las observaciones (píxeles). Los modelos estadísticos se dividen en paramétricos y no paramétricos. Los paramétricos, están basados en distribuciones de probabilidad conocidas (normal, gaussiana, uniforme, exponencial, etc.), mientras que los no paramétricos son usados cuando se asume que los datos no se ajustan a ninguna distribución de probabilidad conocida. Por lo que, su distribución no puede ser definida a priori, pues son los datos observados los que la determinan. A continuación, se describen algunas de las técnicas que caen dentro de esta categoría.

El algoritmo autoadaptativo presentado en [29] ajusta sus parámetros de acuerdo con las características del escenario. El método es llamado SAG (*self-adaptive GMM*) y combina dos procesos: a nivel cuadro y a nivel pixel. SAG se basa en el modelo de mezclas de gaussianas propuesto por Zivkovic y Van Der Heijden [91], y el método *multi-dimensional Gaussian kernel density transform* (MDGKT) [92]. Por su parte C. Spampinato *et al.* [30] presentan un método no paramétrico que modela el fondo y los objetos dinámicos por medio de *textons* y el algoritmo *Kernel Density Estimation*. Los *textons* son elementos usados para describir texturas y pueden ser definidos como un conjunto de patrones compartidos sobre una región de la imagen. Como trabajo futuro los autores planean reducir el costo computacional del algoritmo.

1.1.4. Modelos estadísticos avanzados

Estos métodos son derivados de los modelos estadísticos convencionales como las mezclas de Gaussianas (GMM) o el método *Kernel Density Estimation* (KDE). No obstante, los modelos estadísticos avanzados realizan combinaciones entre modelos estadísticos convencionales o proponen el uso de otras distribuciones paramétricas y no paramétricas, con el propósito de crear algoritmos más robustos y con mejores características.

En [39] se presenta un método robusto orientado a videos de sistemas de vigilancia en alta resolución. El cual, utiliza el cálculo de vectores de orientación de movimiento mediante el método de orientación de movimiento de histograma (MOH), el uso de cuadro de diferencia y patrones binarios locales (LBP) [93]. Los objetos en movimiento se segmentan por medio de un algoritmo de bloques adaptivos particionados. El método logra buenos resultados en situaciones de inicialización del fondo con camuflaje y *bootstrapping*. Por otro lado, para mejorar al método no paramétrico *Visual Background Extractor* (ViBe) se añaden algunas características en [40]; una estructura que indica el nivel de intermitencia (parpadeo) de cada pixel, una operación morfológica de cierre sobre la máscara binaria de la detección de movimiento y un módulo para evitar insertar objetos de interés en el modelo de fondo. El algoritmo propuesto tiene la ventaja de poder realizar procesamiento en tiempo real a velocidades mayores a 30 FPS. Sin embargo, cuenta con dificultades en situaciones de camuflaje. Los algoritmos ViBe y *Graph cut* son utilizados para realizar la detección de movimiento en [41] en el método GC STViBe (*Graph Cut S/T ViBe*). El *Graph* (grafo) usado en este método, es conformado como un conjunto de nodos definidos por los pixeles de la imagen de entrada y un conjunto indirecto de bordes que están conectados a los nodos. Por su parte, en [42] se propone un método que utiliza la probabilidad y entropía de bloques de dimensión $W \times W$ para tomar la decisión si el bloque pertenece a objeto en movimiento o a fondo. Dan Yan *et al.* presentan un algoritmo llamado *Stability of Adaptive Feature* (SoAF) [43], el cual utiliza características de color RGB, escala de gris y textura. Como descriptor de textura se realiza una mejora al método SILTP para hacerlo invariante a la rotación llamado *Rotated Scale Invariant Local Ternary Pattern* (R-SILTP). El método [44] selecciona el mejor espacio de color entre RGB y YCbCr mediante un módulo llamado *color space module* (CSM) que basa su criterio de decisión de color optimo en el error cuadrático medio (MSE), luego cada pixel se modela como una mezcla de gaussianas. En [45] se proponen los métodos *Spatially-consistent Background Modeling* (SBM) y *Spatiotemporal Background Modeling* (STBM). SBM utiliza el método GMM, como modelo inicial de fondo. Luego, la similitud entre dos pixeles es obtenida mediante *minimax path* siendo extraído con el uso de *minimum spanning tree* (MST). El algoritmo STBM es una extensión de SBM en el dominio temporal usando flujo óptico como base para obtener vectores de movimiento entre los pixeles actuales y los pasados. Para esta propuesta los autores seleccionaron el algoritmo de flujo óptico más rápido y simple. Por lo que, una mejora al método STBM podría ser probar con otros algoritmos de flujo óptico relativamente más lentos, pero con mejores desempeños. Otro

enfoque es presentado por S. Jiang and X. Lu en [46], llamado *Weight-Sample-Based Method for Background Estimation* (WeSamBE) y asume que para cada pixel, el modelo de fondo puede ser representado a través de una colección de N muestras de fondo previas. Cada muestra de fondo se liga a un peso. El peso actúa como una forma de caracterizar la importancia de cada muestra y se refuerzan o penalizan automáticamente de acuerdo con su grado de actividad, lo que permite al algoritmo adaptarse rápidamente a los cambios de escenario. En [47] se presenta un algoritmo para manejar cambios repentinos de iluminación, es llamado *Multimode Background Subtraction Version 0* (MBS V0). MBS V0 primero es entrenado a manera de generar un banco de múltiples modelos de fondo gaussianos (BMB). Luego, se selecciona el mejor modelo BMB para producir una máscara binaria (BM) para cada uno de los canales del espacio de color utilizado. Las BM son fusionadas para obtener una máscara de detección final. En [48] se presenta una adaptación de MBS V0 que añade un módulo de segmentación basado en superpíxeles. El objetivo es la eliminación de ruido espacial considerando las estimaciones de probabilidad iniciales y la información de color de un vecindario de píxeles bajo un cuadro de superpíxeles. Otro método que entra en la categoría de modelo estadístico avanzado es [49], el cual usa multimodelos por pixel basados en el algoritmo no paramétrico AMBER [50] considerando un mayor número de *templates*. Por su parte, el método PBAS se adapta bien a cambios de iluminación, pero el procedimiento que distingue los cambios en objetos es deficiente, por lo que, en [51] se propone un método adaptivo de actualización de fondo que funciona tanto en nivel pixel como objeto. El enfoque propuesto por J. D. Romero *et al.* [52] resuelve los cambios de iluminación repentinos y graduales a través de un método llamado *lightness-red-green-blue* (LRGB). Este modelo compara dos imágenes y encuentra píxeles con luminosidad escalada. En [53] se presenta un enfoque que realiza una mejora al método original SURF [94], el algoritmo mejoró la velocidad y precisión del original SURF, al reducir la complejidad de cálculo. J. Dou y J. Li en [54] proponen un método basado en *Sift flow* (*scale-invariant feature transform*) [95] para modelar el fondo y detectar objetos en movimiento. Cada pixel es modelado con un grupo de descriptores adaptivos *Sift flow* que son calculados sobre una región rectangular alrededor del pixel. En [55] se presenta un enfoque basado en aprendizaje por medio de aproximación estocástica. El método fue ideado para trabajar con cámaras PTZ. Sin embargo, posee la desventaja de necesitar sobreajustes a sus parámetros para obtener mejores resultados en las diferentes secuencias de video. H. Lu *et al.* [56] proponen un novedoso método de modelos GMM compartidos para modelar fondo y objetos en primer plano. El mecanismo

compartido explota la relación espaciotemporal entre los píxeles, ya que dinámicamente para cada pixel se busca un modelo óptimo de fondo u objeto en primer plano en una región $N \times N$. El enfoque, llamado *Spatially Global Gaussian Mixture Models and uncertain* (SGGMM) propuesto en [57] es un modelo de incertidumbre que se enfoca en mejorar y aumentar el color de fondo para obtener una detección robusta. Para la realización del método SGGMM se utilizan dos modelos gaussianos: de fondo y primer plano. El método afronta problemas relativos a sombras, vibraciones y fondos dinámicos. Sin embargo, el usuario debe especificar el número de gaussianas. Otra técnica es el modelo estadístico no paramétrico SuBSENSE [58], método de segmentación a nivel pixel que depende de las características espacio temporales binarias y de información de color para detectar movimiento. La detección de movimiento se basa en un método de búsqueda de patrones *Local Binary Similarity Patterns* (LBSP), donde los píxeles vecinos semejantes al valor central tendrán un valor de 1 y aquellos que su valor difiera serán etiquetados con un 0. El método es extremadamente flexible y puede adaptarse incluso a los escenarios más difíciles. La velocidad de procesamiento de este algoritmo lo hace ideal para aplicaciones en tiempo real. En [59] se propone un método basado en grafos y una función de energía N^2cut para evaluar las relaciones espaciales en la segmentación y los cortes normalizados. El algoritmo trabaja a 20 FPS en videos de alta definición tomados por una cámara GoPro.

1.1.5. Modelos difusos

Son modelos que permiten hacer inferencias con base en reglas heurísticas. Los modelos basados en lógica difusa se basan en una adaptación matemática al lenguaje que utilizamos comúnmente, el cual, es un lenguaje ambiguo e impreciso.

En [60] se propone el uso de un correlograma multicanal difuso para la detección de movimiento. Es basado en el modelo de correlograma difuso en [96]. El método no requiere de cuadros de entrenamiento para el modelo de fondo inicial y resuelve *bootstrapping*. Además, mantiene todos sus parámetros constantes excepto un umbral th . D. K. Panda y S. Meher en [61] proponen una técnica para el modelado de fondo a través de la combinación de la diferencia del histograma de color (CDH) [97] y lógica difusa. El método se compone de cinco etapas principales: 1) determinar la diferencia de color de histograma (CDH), 2) emplear *Fuzzy c-means* para formar un histograma de color difuso o *fuzzy color difference histogram* (FCDH), 3) inicialización del modelo de fondo, 4) detección de objetos en primer plano usando concordancia de similitudes entre

histogramas y 5) finalmente el mantenimiento del fondo es logrado al comparar la concordancia de similitud del FCDH del fondo y del cuadro actual, contra un umbral específico. El método ofrece una solución a problemas debidos a la variación de iluminación y fondo dinámico. No obstante, falla en suprimir sombras y *bootstrapping*.

1.1.6. Modelos de subespacios

Son modelos basados en una descomposición derivada de un espacio específico, algunos de los métodos que entran en esta categoría son: análisis de componentes principales, descomposición ortogonal, descomposiciones basadas en súperpixeles, matrices basadas en bajo rango, descomposición esparcida, entre otros.

RPCA (*Robust Principal Component Analysis*) es un modelo que asume que los pixeles del fondo entre cuadros adyacentes están relacionados linealmente, mientras que los pixeles pertenecientes al objeto en movimiento están esparcidos. Por lo que, una matriz de observación puede ser descompuesta en una matriz de bajo rango para el fondo y una esparcida para los objetos dinámicos. Una adaptación al modelo RPCA es presentada en [62]. El método logra eliminar la influencia de sombras y presenta buenos resultados en fondos dinámicos y cambios de iluminación. Otros métodos basados en RPCA, se presentan en [63] y [64]. El primero propone un esquema iterativo IRLS (*Iteratively Reweighted Least Squares*) en la factorización de la matriz de bajo rango y se incluyen restricciones espaciales al proceso de minimización. El segundo realiza una mejora al método OR-PCA (*Online Robust PCA*) a través de inicialización por lotes. OR-PCA resuelve los inconvenientes de los cálculos complejos y el bajo tiempo de procesamiento del tradicional RPCA a través de un método recursivo. La mejora propuesta a OR-PCA fue inspirada en el método de diccionarios de aprendizaje en línea [65], por lo que, algunos de los primeros cuadros son usados para el procesamiento por lotes. El método posee buenos resultados para eliminación de sombras y se observa una mejor detección que los métodos FPCP, RPCA, SSGoDec, DECOLOR, GRASTA y OR-PCA. En [66] se presenta el algoritmo COROLA (*Contiguous Outliers Representation via Online Low Rank Approximation*). COROLA adopta una aproximación de matriz de descomposición de bajo rango en línea para poder resolver situaciones con escenarios complejos con fondos dinámicos o ruidosos. El algoritmo se basa en los enfoques DECOLOR [67] y OR-PCA [65]. A diferencia del método DECOLOR, el costo computacional de COROLA es independiente del número de imágenes procesadas. Por lo que, solo se incrementa el tiempo al incrementar el

tamaño de resolución espacial de la imagen. En [68] se propone un algoritmo de detección de movimiento llamado *Total Variation Regularized RPCA* (TVRPCA), en el cual, se descompone el video analizado en tres partes: fondo estático de bajo nivel, fondo dinámico esparcido y objetos en primer plano suavizados y esparcidos. El video es tratado como un tensor 3D que incluye información temporal. TVRPCA se comparó con algoritmos como RPCA y DECOLOR, mostrando que posee una clara ventaja, especialmente en videos con la presencia de mal clima o fondos complejos. Algunos problemas que presenta son camuflaje y pérdidas cuando existen objetos pequeños que se mueven a gran velocidad, como automóviles que están lejanos a la cámara. En [69] se presenta una matriz de descomposición basada en superpíxeles, junto con una norma de regularización de máximos (Max-Norm) y una estructura de restricciones esparcidas. En general el método cuenta con un buen desempeño en el manejo de fondos dinámicos y cambios de iluminación. Pero tiene un bajo desempeño en objetos intermitentes o vibraciones de cámara.

1.1.7. Dicionarios de aprendizaje

Los diccionarios de aprendizaje son un tipo de modelo esparcido o disperso donde la suposición básica de la representación dispersa es que la imagen actual, se puede representar como una combinación lineal de vectores que forman un diccionario. A continuación, se presentan algunos métodos que caen dentro de esta categoría.

P. L. St-Charles *et al.* [70] proponen un método mediante el uso de un nuevo tipo de enfoque basado en *Word Consensus* llamado PAWCS (*Pixel-based Adaptive Word Consensus*). PAWCS es basado en la caracterización y monitoreo de las diferentes representaciones de fondo mediante el uso de diccionarios locales usando características de color y textura. Además, se integra el uso de características LBSP (*Local Binary Similarity Pattern*) las cuales, ayudan en situaciones de variaciones de iluminación. Otro enfoque que hace uso de codificación esparcida se presenta en [71] que fue pensando para trabajar con videos del espectro infrarrojo. El método utiliza la información de velocidad y aceleración en vectores propios para construir un modelo *local saliency* que ayuda a extraer rápidamente el área del objeto de interés y un modelo de representación esparcida es usado para separar los objetos dinámicos del fondo. Los experimentos mostraron que el método puede detectar efectivamente el contorno completo de los objetos en movimiento sin conocimientos previos del escenario. Además, los resultados de detección fueron robustos para manejar situaciones bajo fondos dinámicos o cambios de iluminación, pero falla en detectar objetos

pequeños. En [72], se presenta un método que mejora la eficiencia de detección de objetos dinámicos mediante un algoritmo esparcido de *codebook* basado en superpixel. Los píxeles vecinos se agrupan en superpíxeles de acuerdo con sus similitudes en brillo y textura. Los resultados cualitativos y cuantitativos para escenarios externos e internos muestran que el método logra reducir significativamente los recursos de memoria y mejorar el tiempo de procesamiento sin perder precisión en la detección de movimiento. En [73] se propone un enfoque para modelar el fondo llamado *Background Modeling Through Dictionary Learning* (BMTDL). BMTDL construye una representación de los datos a través de una combinación lineal de elementos básicos llamados *atoms* que forman un diccionario. El método aprovecha la correlación espacial entre píxeles y muestra estabilidad a cambios de iluminación, vibraciones de cámara y fondos dinámicos. Sin embargo, el desempeño del algoritmo en cuanto a costo computacional está relacionado con el parámetro de frecuencia de *update* y el tamaño de los parches. En [74] se propone una representación de fondo basada en diccionarios de aprendizaje y mantenimiento histórico de los píxeles, el método es abreviado BREW-DLHPM y posee un mecanismo de mantenimiento de fondo a nivel pixel. Los resultados experimentales muestran que BREW-DLHPM puede trabajar en tiempo real en resoluciones de 320x240 a 61 FPS. Sin embargo, a más resolución espacial, mayor es el costo computacional. C. David and V. Gui [75] proponen un método de modelado de fondo compuesto de una combinación lineal esparcida de elementos de un diccionario de aprendizaje K-SVD [98] y un conjunto de coeficientes asociados a cada posición de la imagen. A manera de reducir los problemas introducidos por fondos dinámicos es añadido un modelo Gaussiano en los coeficientes. Se usan dos algoritmos de umbral. El primero elimina los errores debido a fondos dinámicos y posible ruido, y el segundo finaliza la segmentación y es calculado mediante *K-means*.

1.1.8. Modelos de transformación de dominio

El principio de estos modelos se basa en separar el fondo y los objetos dinámicos mediante la ayuda de una transformación de dominio. Para esto, diferentes transformaciones pueden ser usadas, tal como ondulas *Wavelet*, Transformada rápida de Fourier, *Hadamard*, entre otras.

A manera de reducir el efecto de las variaciones de luz mediante *Wavelet* en [76] se plantea un algoritmo llamado *Background Wavelet Model* (BWM), en él se usa una norma H_2 a manera de extraer los coeficientes de energía entre el cuadro actual y el cuadro de fondo. La norma H_2 es

usada para medir la covarianza en estado estable o la energía de la respuesta de salida. En [77] se presenta un método que utiliza el análisis de onduletas a manera de mejorar el desempeño de un algoritmo de diferencia de cuadros basado en memoria MFD (*Memory-based Frame Differencing*). La idea básica es filtrar el ruido a través del uso de una memoria, permitiendo que el detector de movimiento ignore los cambios causados por el ruido. Los resultados experimentales muestran que el método logra detectar objetos dinámicos pequeños y lentos, también objetos con bajo contraste y en ambientes con cambios de iluminación. Otro método mediante transformación de dominio es llamado TGMV (*texture guided weighted voting*) [78]. El método propuesto emplea la transformación de *Wavelet* para descomponer la imagen en bandas de frecuencia, y así lograr diferenciar los objetos dinámicos pequeños y difícilmente notorios del escenario de fondo. Además, se utiliza un esquema de pesos basado en intensidad y texturas extraídas mediante LBP (*Local Binary Pattern*). El método logra una robusta segmentación de objetos dinámicos en situaciones de camuflaje en comparación a los métodos con los que se evalúa.

1.1.9. Modelos tensores robustos

Los modelos tensores se basan en la asunción de que no todas las relaciones en la naturaleza son lineales, pero la mayoría son diferenciables y de este modo se pueden aproximar localmente con sumas de funciones multilineales. Así la mayoría de las magnitudes en física se pueden expresar como tensores. Matemáticamente, un tensor es una entidad algebraica representada por un cierto número de componentes que generaliza los conceptos escalar, vector y matriz de cualquier sistema de coordenadas elegido. A continuación, se presentan los enfoques que caen dentro de esta categoría.

En [79] se presenta un sistema de detección llamado *Flux Tensor with Split Gaussian models* (FTSG) que aprovecha los beneficios de fusionar un método basado en un tensor espaciotemporal y un modelo gaussiano para producir mejores resultados, reduciendo los errores debido a cambios de iluminación, ruido y efecto halo. Ofrece tres clasificaciones en la máscara de detección: objetos en movimiento, objetos estáticos y sombras o iluminación. Una desventaja es su tiempo de procesamiento no apto para tiempo real. Q. Zhao *et al.* [80] proponen un modelo generado por medio de la factorización de un tensor bayesiano robusto (BRTF). Se basa en el uso del tensor de bajo rango CANDECOMP/PARAFAC (CP) [99], que utiliza la información global, mientras que un tensor esparcido captura la información local, obteniendo una predicción robusta de la

distribución. El método tiene la característica de que todos los parámetros del modelo pueden ser aprendidos automáticamente de los datos observados.

1.1.10. Modelos basados en redes neuronales artificiales

Son modelos matemáticos inspirados en el comportamiento de las redes neuronales biológicas. Son capaces de aprender de la experiencia, de generalizar y de extraer características a partir de los datos de entrada.

En [31] se presenta un modelo de segmentación para objetos dinámicos y estáticos mediante *Discrete Time Cellular Neural Network* (DTCNN). El método propuesto involucra tres principales bloques para la realización de la segmentación dinámica: registro de fondo dinámico, detección de objetos dinámicos y mejoras a la segmentación de objetos. Una ventaja notoria del método es que no requiere ajuste manual de sus parámetros para procesar diferentes secuencias de video. El enfoque por G. Ramírez-Alonso *et al.* [32] APPSA, presenta un nuevo sistema de modelado de fondo basado en dos mapas autoorganizados (SOM por sus siglas en inglés) que se adaptan en forma paralela a diferentes velocidades. El algoritmo tiene una arquitectura paralela autoadaptiva. Este puede identificar automáticamente el problema principal que afecta al escenario y a la detección de los objetos en movimiento. La identificación es lograda mediante cuatro módulos clasificados en: escenas con fondos estáticos, escenas normales, escenas dinámicas y muy dinámicas. Otro enfoque Retinotopic SOM (RESOM) [33] se inspira en el mecanismo de la corteza visual humana. RESOM puede adaptar sus parámetros de aprendizaje basándose en el comportamiento del escenario. Sin embargo, el método aún presenta problemas debido al camuflaje y su tiempo de procesamiento es lento. Otro método llamado SOM-CNN es propuesto en [34], se basa en los métodos *Self-Organized Maps* (SOM) y *Cellular Neural Networks* (CNN). El método es una mejora a la arquitectura RESOM que propone un esquema novedoso de umbralización llamado *Neighbor Threshold CNN* (NTCNN). Por su parte, en [35] se propone un método de modelado de fondo a través del uso de una máquina de Boltzmann restringida o RBM. La RBM es una red neuronal, RN, de dos capas, donde la primera capa es definida como la capa visible, la cual se alimenta de la intensidad del pixel de la imagen, y la segunda capa es llamada oculta y funciona como un detector de características. Una de las principales desventajas de RBM es que su desempeño es proporcional al número de capas ocultas, debido a que cada capa oculta representa una característica prototipo. El método BGWiS [36] puede ser visto como un histograma del

número de ocurrencias de todos los subpatrones binarios ocurridos durante el entrenamiento. En este caso de estudio se asume que el modelo de fondo de un pixel está formado por la combinación de los más frecuentes subpatrones contenidos en las RAMs. Este método fue probado en la base de datos SBI y previamente en la base de datos *Change Detection* [100], demostrando ser uno de los métodos que utiliza los mismos parámetros iniciales en todos los videos. En [37] se presenta *Gibbs-Markov random field* (GMRF) y una RN tipo *Hopfield* (HTNN). En los resultados experimentales, el método mostró buenos resultados resolviendo situaciones de baja iluminación y cambios de iluminación repentinos. El método propuesto en [38] se basa en una RN de función radial (RBF) que se compone de 3 capas; de entrada, oculta y de salida. Un módulo de generación de multifondos (MBG) determina el número de capas ocultas en la red RBF. El método es adecuado para manejar escenarios dinámicos o estáticos y puede alcanzar una velocidad de 34 FPS con una resolución de 320x240.

1.1.11 Otros

En esta clasificación, se contemplan algoritmos nuevos, que no se basan en ningún modelo de los descritos anteriormente o algoritmos basados en modelos poco estudiados, a manera que resulta complicado incorporarlos en alguna clasificación.

En [81] se usa programación genética (GP) para combinar los resultados de los mejores algoritmos de detección de movimiento en *Change Detection* y obtener máscaras binarias con mejores desempeños. Se realizaron diferentes combinaciones y se observó que los algoritmos llamados IUTIS-3 y IUTIS-5 tuvieron los mejores resultados. Por otro lado, Philipp Tiefenbacher *et al.* [82] presentan un modelo basado en un controlador proporcional integral y derivativo (PID). El controlador PID regula la decisión del umbral de objetos dinámicos y la tasa de actualización del modelo de fondo. El método obtuvo buenos resultados para manejar fondos dinámicos. En [83] se propone un enfoque para modelado de fondo mediante el uso de bloques de características espaciotemporales correlacionadas y extraídas directamente de videos comprimidos en HEVC. Un video HEVC consiste en una secuencia de cuadros, los cuales están divididos en bloques que no se traslapan llamados *Coding Tree Unit* (CTU). Los resultados experimentales del método propuesto muestran un buen desempeño para situaciones de fondos dinámicos. El método también posee la ventaja de que es relativamente rápido, el promedio de procesamiento oscila entre 25-30 FPS en resoluciones de video de 720x420. Por otro lado, existe una trade-off entre el parámetro del tamaño

de bloque CTU y el desempeño en *F-Measure* y velocidad; a mayor tamaño, mejor velocidad, pero menor *F-Measure* y a menor tamaño, menor velocidad, pero mejor *F-Measure*, por lo que, este parámetro deber ser elegido cuidadosamente.

1.1.12. Análisis de tendencias

De los 79 algoritmos analizados enfocados a detección de movimiento en video, se encontró que las tres técnicas más utilizadas por los autores son: modelos estadísticos avanzados 28%, aprendizaje profundo 14% y un empate entre redes neuronales 10%, modelos de subespacio 10% y básicos 10%. Lo anterior se observa en la Figura 1.1. Se cree que los modelos estadísticos avanzados han tomado popularidad en los últimos 5 años debido a que la gran mayoría posee parámetros autoadaptivos. Sin embargo, uno de los inconvenientes de este tipo de técnica es que entre más robustos son los algoritmos, requieren mayor cantidad de parámetros, lo que conlleva a algoritmos más complejos y a un mayor tiempo de procesamiento para realizar los ajustes necesarios.

De acuerdo con la Figura 1.1, las técnicas supervisadas que son redes neuronales y de aprendizaje profundo representan un porcentaje de uso del 24%, estas técnicas supervisadas en especial las basadas en aprendizaje profundo (posicionadas en segundo lugar) han ganado popularidad en el área de detección de movimiento debido a los altos índices de desempeños en bases de datos públicas. Además se posicionan dentro de los primeros lugares de acuerdo con el instrumentos de evaluación [12] donde se evaluaron 119 algoritmos enfocados a detección de movimiento respecto a los criterios documentación, autoadaptibilidad, velocidad, novedad y desempeño. Incluso, de acuerdo con [8] las técnicas de aprendizaje profundo han obtenido mejores desempeños que los de un humano en realizar GT para segmentación del movimiento. Además, gracias a los avances respecto a bases de datos (cada vez más y más grandes), hardware dedicados, métodos de optimización y regularización, se obtiene una mayor facilidad para realizar nuevos modelos. Por lo tanto, se espera que estas técnicas sean aún más utilizadas en los próximos años y se obtengan algoritmos cada vez más robustos. Por otro lado, de la Figura 1.1 se puede observar que los modelos de subespacios y básicos se posicionan en tercer lugar. Los de subespacios involucran en su mayoría métodos basados en descomposiciones en matrices de bajo rango (correspondiente al fondo) y una matriz de esparcimiento (correspondiente a objetos dinámicos), estos métodos han mostrado ser muy eficientes para el manejo de fondos dinámicos y variaciones

de luminosidad. Pero, la mayoría funcionan por lotes de cuadros y cuando el tamaño de los datos crece, aumenta la complejidad computacional y los vuelve lentos para aplicaciones en tiempo real. Por lo que, actualmente se ha optado por desarrollar métodos de aproximaciones de descomposición de matrices en línea. En lo que respecta a los modelos básicos continúan siendo aplicados al desarrollo de algoritmos aún y cuando existen métodos nuevos con mejores características, posiblemente esto se deba a que su funcionamiento es simple, en comparación de las técnicas nuevas que involucran mayor complejidad.

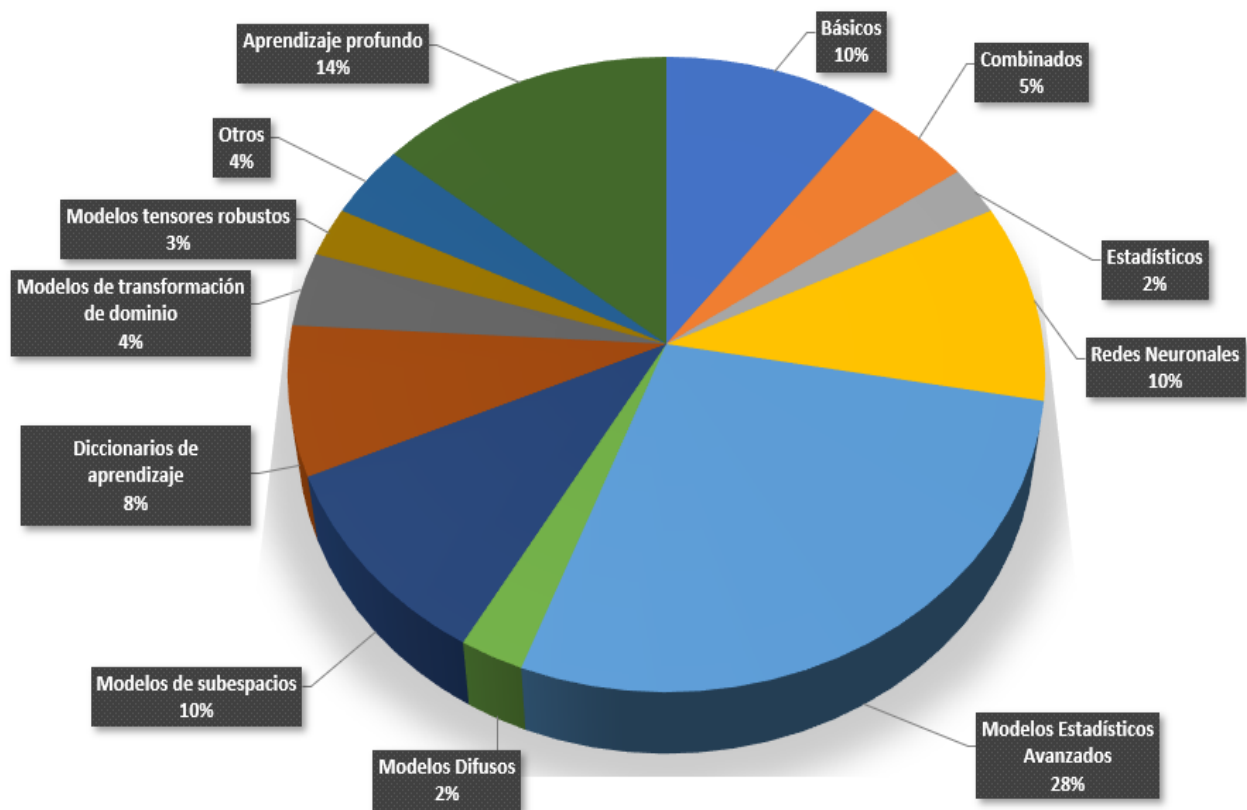


Figura 1.1. Porcentaje de uso de los diferentes tipos de técnicas dentro de los 79 algoritmos analizados.

A continuación, en la Tabla 1.2 se resumen los problemas abiertos y las soluciones que ofrecen los algoritmos analizados de acuerdo con su tipo de técnica. Esto con el fin de identificar áreas de mejora para el desarrollo de futuros algoritmos.

Tabla 1.2. Soluciones y retos por tipo de técnica.

Técnica	Soluciones y problemas abiertos
Modelos básicos	Son relativamente acertados para modelar el fondo cuando la iluminación es constante. Sin embargo, en existencia de cambios repentinos de iluminación o fondos dinámicos, resulta en un alto número de falsos positivos. La recuperación a los cambios repentinos en el escenario de fondo es lenta.
Modelos combinados	Alcanzan un procesamiento en tiempo real para videos con resolución de 320x240. Manejan sombras, cambios de iluminación y ruido suave. Sin embargo, en situaciones como movimientos repentinos de cámara y fondos dinámicos, se obtiene un bajo desempeño.
Modelos supervisados (redes neuronales y aprendizaje profundo)	Altamente eficientes en el manejo de fondos dinámicos y una gran variedad de problemas de video. Algunos algoritmos pueden adaptar sus parámetros de aprendizaje basándose en el comportamiento del escenario. Sin embargo, aún presentan problemas debido al camuflaje y su tiempo de procesamiento es elevado. La mayoría requieren uso de GPU para alcanzar procesamiento en tiempo real y requieren un extenso conjunto de imágenes de entrenamiento.
Modelos estadísticos y estadísticos avanzados	Afrontan problemas relativos a sombras, vibraciones y fondos dinámicos. Pero entre más robustos son los algoritmos, requieren mayor cantidad de parámetros y en ocasiones se necesita realizar ajustes manuales en reiteradas ocasiones para obtener mejores resultados.
Modelos difusos	Resuelven fondos dinámicos y cambios de iluminación, pero tienen problemas en suprimir sombras y poseen un costo computacional elevado.
Modelos de subespacios	Manejan cambios de iluminación y fondos dinámicos, pero requieren de cálculos complejos y alto tiempo de procesamiento. Además, existen problemas por camuflaje, detección de objetos dinámicos pequeños, objetos intermitentes y vibraciones de cámara.
Diccionarios de aprendizaje	Resuelven fondos dinámicos, sombras y cambios de iluminación. Sin embargo, fallan en detectar objetos pequeños. Costo computacional elevado depende de la cantidad de datos en el diccionario y el tamaño de la imagen.
Modelos de transformación de dominio	Solucionan cambios de iluminación y camuflaje, pero presentan problemas en situaciones de fondos dinámicos.
Modelos tensores robustos	Manejan cambios de iluminación, ruido y objetos que se vuelven estáticos. Poseen tiempos de procesamiento no aptos para tiempo real y requieren de cálculos complejos.
Otros	Algunos requieren cuadros de entrenamiento y resultados de segmentación de otros algoritmos.

1.2. Validación de algoritmos de detección de movimiento

Para validar los algoritmos de detección de movimiento se utiliza un *Ground truth* o GT que representa el resultado deseado o esperado y contiene información que nos ayuda a conocer que tan cercano está el resultado de un algoritmo, del valor o característica ideal. En algoritmos de modelado de fondo el *Ground truth* puede hacer referencia a la imagen que contiene el modelo de fondo deseado sin objetos dinámicos. Por otro lado, para algoritmos enfocados en detección de movimiento generalmente se usa una imagen binaria, que es elaborada manualmente, en esta imagen se identifican los pixeles que pertenecen a los objetos en movimiento con valor de 1 y el fondo con 0. El *Ground truth* al ser comparado con el resultado de detección del algoritmo (máscara binaria) genera las siguientes clasificaciones de pixeles:

- **Falsos Positivos (FP):** se producen cuando existen pixeles clasificados por el algoritmo con valor 1, pero en el GT son 0. Es decir, un pixel fue erróneamente detectado como parte de un objeto en movimiento cuando en realidad era parte del fondo.
- **Falsos Negativos (FN):** se obtienen cuando existen pixeles clasificados por el algoritmo con valor 0, pero en el GT tienen valor de 1. Es decir, un pixel fue erróneamente detectado como parte del fondo cuando debió ser detectado como objeto en movimiento.
- **Verdadero Positivo (VP):** se producen cuando existen pixeles con valor 1 correctamente asignados. Los pixeles del objeto en movimiento son asignados correctamente.
- **Verdadero Negativo (VN):** se producen cuando existen pixeles con valor 0 correctamente asignados. Los pixeles del fondo son asignados correctamente

En la Figura 1.2 se muestran las clasificaciones de pixeles previamente descritas, al realizar la comparación del GT contra el resultado obtenido por un algoritmo. En la Figura 1.2c, las zonas amarillas pertenecen a los falsos negativos, las azules a los falsos positivos, la roja a los verdaderos positivos y la negra a los verdaderos negativos.

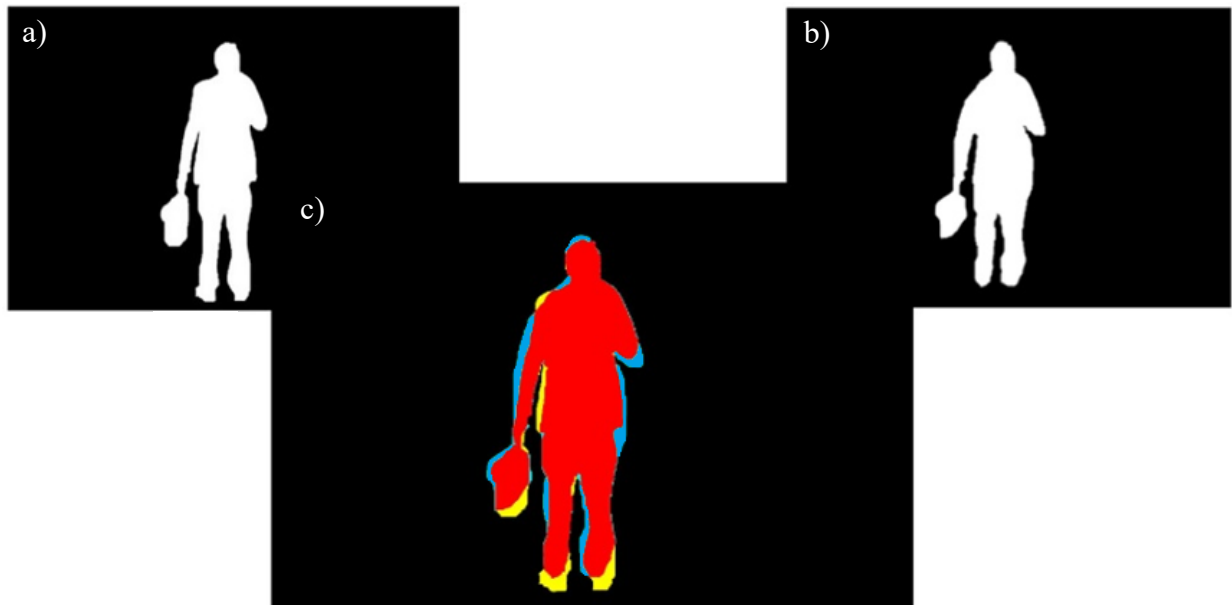


Figura 1.2. a) GT, b) resultado obtenido de algún algoritmo de detección de movimiento, c) comparación de la imagen a) con b), VP (zonas rojas), FP (zonas azules), FN (zonas amarillas) y VN (zonas negras).

Normalmente, para dar una mejor visualización a las cuatro clasificaciones mostradas se utiliza una matriz de confusión; una herramienta de evaluación que compara los valores reales contra los valores de predicción. Para el caso de un problema enfocado a detección de movimiento, la matriz de confusión se compone de dos filas y dos columnas que reportan de manera cuantitativa el

número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Esta matriz resume los resultados de las pruebas realizadas y sirve como una forma rápida de verificar el desempeño del algoritmo.

Lo anterior se observa en la Tabla 1.3, por un lado, se tendrá la condición verdadera que en este caso es representada mediante el GT y por el otro lado se tendrá la condición de predicción, la cual será el resultado obtenido por el algoritmo de detección de movimiento.

Tabla 1.3. Matriz de confusión.

		<i>Condición de predicción</i>	
		<i>Objeto en movimiento</i>	<i>Fondo</i>
<i>Condición verdadera</i>	<i>Objeto en movimiento</i>	<i>Verdadero Positivo (VP)</i>	<i>Falso Negativo (FN)</i>
	<i>Fondo</i>	<i>Falso Positivo (FP)</i>	<i>Verdadero Negativo (VN)</i>

Las cuatro clasificaciones mostradas FP, FN, VP y VN son la base de algunas de las métricas enfocadas a medir el desempeño del algoritmo, las cuales son mostradas a continuación.

1.2.1. Métricas para evaluación de desempeño

Una forma cuantitativa de conocer que tan cercano es el resultado de un algoritmo con respecto al GT, es a través de métricas que permiten medir el desempeño del algoritmo. A continuación, se muestran algunas métricas reportadas en [101]:

Especificidad (*Specificity*): cantidad o proporción de resultados negativos clasificados correctamente. La especificidad es lo contrario a la sensibilidad.

$$Specificity = \frac{VN}{VN + FP} \quad (1)$$

Sensibilidad (*Sensitivity* o *Recall*): mide la capacidad de detectar resultados positivos correctamente y está definida por la siguiente relación:

$$Recall = \frac{VP}{VP + FN} \quad (2)$$

Precisión (*Precision*): definida como la capacidad de un sistema de dar el mismo resultado en mediciones diferentes realizadas en las mismas condiciones. La precisión refleja la repetibilidad de la medida y dice que cantidad de píxeles son relevantes:

$$Precision = \frac{VP}{VP + FP} \quad (3)$$

Medida-F (*F-Measure*): es una medida de exactitud empleada en análisis estadístico de clasificación binaria, considera la precisión y la sensibilidad para dar un promedio ponderado de estas dos medidas. Mientras más cercano a 1 se encuentre el resultado de esta medida, mejor será el desempeño. Se define por:

$$F-Measure = \frac{2 * (Presicion * Recall)}{Presicion + Recall} \quad (4)$$

FPR (*False Positive Rate*): tasa de falsos positivos, es una métrica muy utilizada al realizar comparaciones, es conocida también como tasa de falsas alarmas. Se calcula a través de la relación del número de falsos positivos entre el número total de eventos negativos y se expresa por la siguiente relación:

$$FPR = \frac{FP}{FP + TN} \quad (5)$$

FNR (*False Negative Rate*): tasa de falsos negativos; métrica complementaria a la sensibilidad. Refleja la frecuencia con la que se producen clasificaciones erróneas. Se expresa mediante la siguiente ecuación:

$$FNR = \frac{FN}{VP + FN} \quad (6)$$

PWC (*Percentage of Wrong Classifications*): porcentaje de clasificación errónea. Es una métrica que mide la porción o número de falsos positivos y falsos negativos del total de una clasificación. Es muy usada para reportar comparaciones entre algoritmos debido a que toma en consideración todas las muestras mal clasificadas. A menor valor en esta métrica, mejor será el desempeño del algoritmo. Se calcula mediante:

$$PWC = \frac{100 * (FN + FP)}{(VP + FN + FP + VN)} \quad (7)$$

Las siguientes métricas son usadas cuando el GT representa el modelo de fondo ideal para algoritmos enfocados a modelado de fondo y son reportadas en [102]:

PSNR (*Peak signal-to-noise ratio*): relación señal a ruido de pico. Es una métrica que define la relación entre la máxima energía posible de una señal y el ruido que afecta su representación fidedigna. Su uso más habitual es como medida cuantitativa de la calidad de compresión en imágenes. Usualmente se expresa en decibeles, donde a mayor valor de PSNR, mejor será la estimación de la imagen respecto al GT. Se define mediante:

$$PSNR = 10 \log_{10} ((M_{PSNR} - 1) / MSE) \quad (8)$$

Donde M_{PSNR} es el máximo número de nivel de gris y MSE es el error cuadrático medio entre el GT y la imagen.

pCEPs (*Percentage of Clustered Error Pixels*): relación de CEPs (*Cluster Error Pixels*) con respecto al número total de píxeles N en una imagen. Su valor varía en el rango de 0 a 1. A valores más bajos, mejor será la estimación del fondo respecto al GT. Se calcula mediante:

$$pCEPs = \frac{CEPs}{N} \quad (9)$$

Siendo, CEPs el número de píxeles erróneos cuyos vecinos con conectividad cuatro también son píxeles de error.

pEPs (*Percentage of Error Pixels*): relación de EPs (*Error Pixels*) con respecto al número total de píxeles N_{Total} en una imagen. Los píxeles de error o EPs son definidos como aquellos cuyo valor en el modelo de fondo calculado difiere del modelo de GT. La métrica pEPs toma valores en el rango de $[0, 1]$, a un valor más bajo, mejor será el fondo estimado. Se define mediante:

$$pEPs = \frac{EPs}{N_{Total}} \quad (10)$$

AGE (*Average Gray-level Error*): promedio de error de nivel de gris. Es el promedio de la diferencia absoluta de los niveles de gris entre la imagen de referencia I_{GT} y el modelo de fondo estimado I_E . Sus valores se encuentran en el rango de $[0, L-1]$, siendo L el máximo valor de nivel de gris.

$$AGE = \text{mean}(|I_{GT} - I_E|) \quad (11)$$

CQM (*Color Image Quality Measure*): Medida de calidad en imágenes a color. Esta métrica se basa en el cálculo de PSNR en las bandas únicas del espacio de color YUV, asume valores en decibelios y cuanto mayor es el valor de CQM, mejor es la estimación de fondo. Se define mediante:

$$CQM = (Y_{PSNR} * 0.9449) + \frac{U_{PSNR} + V_{PSNR}}{2 * 0.0551} \quad (12)$$

Donde, Y_{PSNR} , U_{PSNR} , V_{PSNR} , son los valores calculados de PSNR en cada una de las bandas Y, U y V respectivamente.

Otras:

MCC (*Matthews correlation coefficient*): métrica usada para medir la calidad de dos clases binarias. Devuelve un valor en el rango $[-1, 1]$, siendo +1 la predicción ideal, 0 una mala predicción y -1 la segmentación inversa ideal [103]. Se calcula mediante:

$$MMC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (13)$$

SSIM (*Structural similarity*): métrica que compara los patrones locales de intensidad de los píxeles que han sido normalizados en luminiscencia y contraste [104]. Medida de calidad de la imagen desde la perspectiva de la formación o estructura de la imagen, se calcula mediante:

$$SSIM(I_X, I_Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)} \quad (14)$$

Donde, I_X e I_Y son las imágenes por comparar, siendo μ_X y μ_Y las medias y σ_X^2 y σ_Y^2 las varianzas de I_X e I_Y respectivamente, C_{i1} y C_{i2} son dos variables incluidas para evitar la inestabilidad, definidas por:

$$Ci_1 = (km_1 Lm)^2, \quad Ci_2 = (km_2 Lm)^2 \quad (15)$$

Siendo, Lm el rango dinámico de valores de gris (255 para imágenes de 8 bits) y km_1 , km_2 constantes pequeñas con valor menor a 1, comúnmente definidas con valores $k_1=0.01$ y $k_2=0.03$.

D-Score: métrica enfocada a medir la calidad de la imagen, considera la localización de los errores en relación a la posición del GT [105]. Es una medida de similitud de imágenes binarias basada en una transformación de distancia, definida mediante:

$$D-Score = \exp(-(\ln(2 \cdot DT(i)) - \alpha_p)^2) \quad (16)$$

Donde, α_p es un parámetro, el cual, entre más bajo, más cercano a cero será el resultado de la métrica, por lo que experimentalmente fue definido $\alpha_p = 5/2$. $DT(i)$ es la transformación de distancia, la cual es dada por:

$$DT(i) = \min(d(ip, i), \forall i \in I_{GT}) \quad (17)$$

Siendo, I_{GT} el conjunto de pixeles que pertenecen al GT y $d(ip, i)$ la distancia entre el pixel ip de la imagen binaria estimada contra el pixel i del GT. Por lo que, $DT(i)$ es la distancia mínima entre ip e i , para cada pixel $i \in I_{GT}$.

FSD: métrica reportada en [106] cuyo nombre proviene del acrónimo de las tres métricas *F-Measure*, *SSIM* y *D-Score*. La métrica FSD combina las tres últimas en un valor de peso promediado, mediante:

$$FSD = \frac{F-Measure + SSIM + (1 - D-Score)}{3} \quad (18)$$

1.2.2. Bases de datos

Actualmente, se encuentran disponibles un número considerable de bases de datos usadas para evaluar el desempeño de los algoritmos de detección de movimiento. Estas bases de datos cuentan con una serie de videos con diferentes situaciones críticas que muy a menudo se presentan en aplicaciones reales de detección de movimiento, tales como: vibraciones, lluvia, nieve, sombras,

fondo dinámico, entre otras. Además, algunas bases de datos cuentan con una serie de *Ground truth*, los cuales, aunados a los videos, sirven para evaluar el desempeño del algoritmo.

Debido a la naturaleza de esta tesis, las bases de datos aquí presentadas se enfocan en detección de movimiento y modelado de fondo. Las bases de datos enfocadas a detección de movimiento utilizan un *Ground truth* que representa la imagen de detección ideal, donde se separan los objetos dinámicos del fondo, por lo general estas bases de datos reportan videos de larga duración con una gran cantidad de cuadros (aproximadamente de 500 a 10000) y el número de *Ground truth* varía desde uno por video, hasta uno cada cuadro de video. Por otro lado, las bases de datos enfocadas en modelado de fondo reportan un *Ground truth* que representa el modelo de fondo deseado, en éstas no se evalúa la detección de movimiento sino el modelo de fondo obtenido por el algoritmo, normalmente los videos contenidos en este tipo de base de datos son de duración corta (aproximadamente de 6 a 2000 cuadros) y se reporta un *Ground truth* por video. Estas bases de datos son utilizadas ocasionalmente en detección de movimiento ya que, si un algoritmo es capaz de modelar el fondo, en una etapa posterior se podría realizar la detección de movimiento al comparar el fondo generado contra el cuadro actual del video y umbralizar la imagen obtenida.

A continuación, en la

Tabla 1.4 se hace un listado de algunas de las bases de datos encontradas en la literatura con su respectiva descripción. Las bases de datos marcadas con una estrella pertenecen a bases de datos enfocadas a modelado de fondo.

Algunos de los enlaces que vinculaban a las bases de datos fueron cambiados, por lo que en la Tabla 1.5 se presenta la ubicación de cada una de las bases de datos al momento de redactar este documento.

De las bases de datos mostradas en la Tabla 1.5, solo algunas proporcionan en el mismo sitio, un *benchmark* para evaluar algoritmos con base en métricas de evaluación de desempeño y ofrecen una tabla de posiciones para ubicar los algoritmos que mejor pueden enfrentar un número de situaciones críticas. Tres de ellas, enfocadas en detección de movimiento son *Change detection*, *LASIESTA* y *BMC*. Mientras que para modelado de fondo dos de las bases de datos que ofrecen *benchmark* son *SBI* y *SBMnet*.

Tabla 1.4. Bases de datos encontradas en la literatura.

Base de datos	Descripción	Ground truth
Wallflower	7 videos cortos con diferentes situaciones críticas. Objetos en movimiento, el cambio de iluminación, fondos dinámicos, camuflaje, etc.	1 GT por cada video.
ATON	11 videos: 4 con cámaras rectilíneas, 2 con cámaras multidireccionales y 5 de detección de sombra.	Únicamente 113 GT en un video de la categoría de detección de sombra.
LIMU	8 videos con categorías como cambio de iluminación en una habitación, la parada de autobús en el día y en la tarde, tráfico y otros videos más prestados de PETS.	1 GT cada 15 cuadros por cada video.
PETS 2016	29 videos en diferentes escenarios, enfocados en detección y reconocimiento de patrones de comportamiento humano.	Un paquete de GT ¹ por cada video.
UCSD	18 videos con diferentes temáticas (<i>Birds, Boats, Freeway</i> , etc.).	Un paquete de GT ¹ por cada video.
CITIC RGB-D	4 videos (<i>Suitcase, Crossing, LCDScreen y LabDoor</i>).	Un paquete de GT ¹ por cada video.
Change detection	53 videos divididos en 11 categorías con 4 o 6 videos cada una.	Un paquete de GT ¹ por cada video.
SBMnet	79 videos divididos en 8 categorías con 5 o 6 videos en cada una.	1 GT ² por video.
SBI	14 videos con diferentes temáticas.	1 GT ² por video.
Fish4knowledge	14 videos bajo el agua con 7 categorías diferentes representando diferentes retos complejos.	Un paquete de GT ¹ por cada video.
MAR	20 videos con tema marítimo para detección, clasificación y tracking.	2 GT por video.
SZTAKI	5 videos: de los cuales 2 son tomados de ATON	Un paquete de GT ¹ por cada video.
BMC	29 videos divididos en categorías: real, sintéticos y modo de aprendizaje.	Un paquete de GT ¹ por cada video.
OTCBVS	12 videos con diferentes temáticas, mostrando en su mayoría el espectro no visible infrarrojo.	Un paquete de GT ¹ solo en algunos videos.
LASIESTA	26 secuencias reales de interior y 22 de exterior organizadas en diferentes categorías.	1 GT por cada cuadro de video.
AVSS	18 videos: 3 videos en la categoría múltiples rostros, 5 para rostro simple, 3 para gente en audiovisual y 7 para detección de vehículos.	Un paquete de GT ¹ solo en algunos videos.

¹ No cubierto en su totalidad (no todos los cuadros del video cuentan con GT o no se encuentran disponibles al público).

² El GT corresponde al modelo de fondo deseado.

Tabla 1.5. URL de bases de datos encontradas en la literatura.

Base de datos	Localización
Wallflower	https://www.microsoft.com/en-us/research/project/test-images-for-wallflower-paper/
ATON	http://cvrr.ucsd.edu/aton/shadow/
LIMU	http://limu.aist.kyushu-u.ac.jp/dataset/en/
PETS	http://www.cvg.reading.ac.uk/PETS2016/
UCSD	http://www.svcl.ucsd.edu/projects/background_subtraction/
CITIC RGB-D	http://atcproyectos.ugr.es/mvision/index.php?option=com_content&view=article&id=45&Itemid=57
Change detection	http://www.changedetection.net
SBMnet	http://www.scenebackgroundmodeling.net
SBI	http://sbmi2015.na.icar.cnr.it/SBIdataset.html
Fish4knowledge	http://f4k.dieei.unict.it/datasets/bkg_modeling/
MAR	http://labrococo.dis.uniroma1.it/MAR/
SZTAKI	http://web.eee.sztaki.hu/~bcsaba/FgShBenchmark.htm
BMC	http://bmc.iut-auvergne.com
OTCBVS	http://vcipl-okstate.org/pbvs/bench/
LASIESTA	https://www.gti.ssr.upm.es/data/lasiesta_database.html
AVSS	http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html

1.3. Aprendizaje de máquina

El aprendizaje de máquina o *machine learning* (ML) es una rama de la inteligencia artificial (IA) producto de las ciencias de la computación y la neurociencia, en la que se construyen modelos o algoritmos que son capaces de aprender y hacer predicciones a través del análisis de los datos. El concepto “aprender” bajo este contexto, se define como el proceso en el que el desempeño de un algoritmo (respecto a una tarea específica) mejora a través de la experiencia [107]. Actualmente, existen modelos de ML aplicados en casi todas las tareas que requieren una gran cantidad de datos, tales como: motores de búsqueda, filtros antispam en mensajes de correo electrónico, software que aprenden a detectar fraudes en transacciones, detección de rostros para aplicaciones de vigilancia, asistentes personales, comandos de reconocimiento de voz, automóviles con sistemas de prevención de accidentes, reconocimiento de objetos, detección de movimiento, entre otras [108]. La importancia del ML radica en permitir a las computadoras aprender de una forma intuitiva o automática y desempeñar tareas que son difíciles de describir formalmente. A continuación se presentan algunas de las tareas que pueden ser resueltas con ML [109]:

- Clasificación: El algoritmo aprende a decidir a qué categoría de salida pertenece algún dato de entrada específico. Esta tarea puede ser realizada de manera no supervisada o supervisada.
- Regresión: se utiliza cuando se desea predecir un valor numérico dada una sucesión de datos de entrada. Por lo tanto, se pueden hacer predicciones de los datos respecto a su comportamiento en muestras previas. Por ejemplo, predecir el consumo de algún producto en un año, teniendo la información de años previos.
- Transcripción: se le otorga al algoritmo un conjunto de datos no estructurados y este debe ser capaz de transcribir la información a una forma textual. Por ejemplo, el reconocimiento de voz, donde se recibe una señal de audio y se genera una secuencia de caracteres o palabras.
- Traducción: la entrada de esta tarea de aprendizaje de máquina consiste en una secuencia de símbolos que el algoritmo debe codificar y decodificar para su interpretación en algún otro idioma.
- Agrupación (*clustering*): se basa en encontrar los grupos naturales de los datos y asociar cada dato a uno de estos grupos. Uno de sus usos es segmentación de imágenes.

- Filtrado de ruido: en esta tarea se le otorga una entrada corrompida al algoritmo y a su salida se debe predecir una salida limpia de su versión corrupta.
- Estimación de densidad: nos permite capturar la distribución de los datos observados y mediante esta se puede inferir la probabilidad de pertenencia de una muestra nueva.

Las tareas anteriores son principalmente agrupadas en cuatro categorías de aprendizaje: supervisado, no supervisado, autosupervisado y reforzado [110]. El aprendizaje supervisado es el más frecuentemente utilizado y consiste en mapear los datos de entrada a una salida conocida o *target* mediante un conjunto de ejemplos de entrenamiento. El aprendizaje no supervisado consiste en mapear los datos de entrada a una salida esperada sin la necesidad o ayuda de un *target*, en este caso el modelo deduce la estructura de los datos de entrada. Este tipo de aprendizaje es usado por ejemplo en compresión de datos, filtrado de ruido y *clustering*. Por su parte el aprendizaje autosupervisado es un caso semejante al aprendizaje supervisado, pero sin *targets* etiquetados manualmente, se puede pensar en este tipo de aprendizaje como un caso sin intervención humana en el lazo de retroalimentación. Por último, el aprendizaje reforzado consiste en un mecanismo que recibe la información acerca de su entorno y aprende a escoger acciones que maximicen alguna recompensa. Su uso ha sido extendido en aplicaciones como video juegos a manera de que el jugador reciba una puntuación mayor de acuerdo con sus acciones a través del juego.

1.3.1. Aprendizaje profundo

Aprendizaje profundo (*deep learning*) es un subcampo de *Machine learning* que está inspirado en la forma particular de detectar características del cerebro humano. *Deep learning* permite trabajar con jerarquía anidada de conceptos, donde cada concepto se encuentra definido en relación a conceptos más simples y representaciones más abstractas [110]–[112]. En contraste con las técnicas tradicionales de aprendizaje de máquina (SVM, *K-Nearest Neighbors*, modelos bayesianos, etc.), en las cuales es necesario una etapa donde un experto extrae las características a manera de reducir la complejidad de los datos y obtener separabilidad entre clases; las técnicas de *deep learning* aprenden características a través de su arquitectura de capas ocultas, eliminando la necesidad de *handcrafted features*. Lo anterior se ilustra en la Figura 1.3, en ella se resalta la diferencia entre las técnicas de aprendizaje de máquina tradicionales y *deep learning* aplicadas en un problema de detección de automóviles.

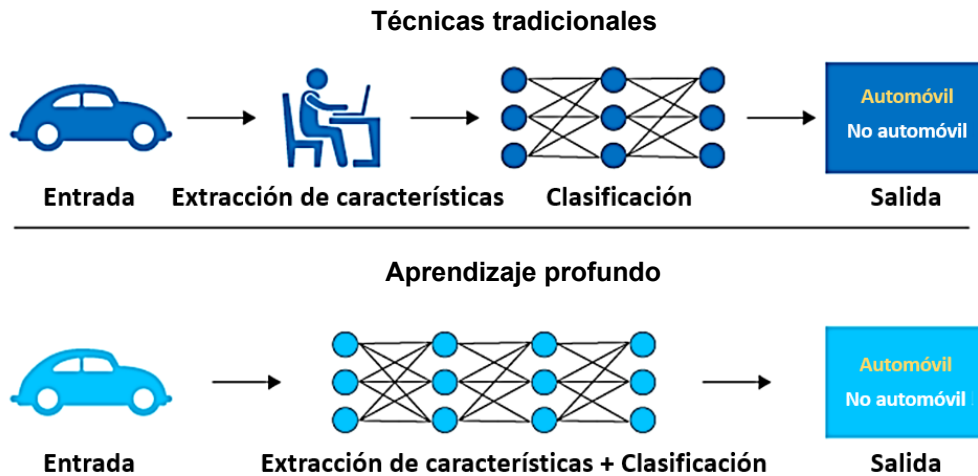


Figura 1.3. Técnicas tradicionales de aprendizaje de máquina en comparación con técnicas de *Deep learning*.

Las técnicas de *deep learning* no son un tema novedoso, tal con lo describen en el libro *Deep Learning with Python* [110], términos como *backpropagation* y *convolutional neural network* ya eran conocidos en 1989. Sin embargo, gracias a los avances en los últimos años en hardware, bases de datos y algoritmos, estas técnicas fueron adquiriendo importancia hasta actualmente ser *deep learning* una de las áreas en un mayor grado de desarrollo. En lo que respecta al hardware entre los años 1990 y 2010 las CPU se volvieron más rápidas. Como resultado, hoy en día es posible correr pequeños modelos de *deep learning* en una laptop, lo cual resultaba imposible 25 años atrás. Los modelos de *deep learning* más sofisticados por lo general requieren un poder computacional mayor que el de una laptop. Por lo que, alrededor de los años 2000, compañías como NVIDIA y AMD invirtieron en el desarrollo de unidades de procesamiento gráfico o GPU que trabajan con gran cantidad de datos de una manera paralela y veloz. Esta inversión benefició a la comunidad científica cuando en 2007, NVIDIA lanzó CUDA, una interfaz de programación de propósito general para GPU. Luego, alrededor del año 2011 se realizaron las primeras implementaciones de redes neuronales artificiales en CUDA. Hoy en día, existen GPU dedicadas a aplicaciones de *deep learning* y productos como las unidades de procesamiento de tensores o TPU fabricadas por Google, las cuales son diez veces más rápidas y eficientes que las GPU modernas.

Además del progreso exponencial en el hardware de almacenamiento de datos, en los últimos 20 años, otro de los puntos que ha contribuido al desarrollo del *deep learning* son las bases de datos. Actualmente existen compañías dedicadas a realizar bases de datos de miles o incluso millones de imágenes (SBMnet, ImageNet, BMC, SBI, Change Detection, etc.). Estas bases de

datos proporcionan una vía de fácil acceso para entrenar algoritmos y demostrar sus desempeños en competencias públicas, motivando constantemente a los investigadores a mejorar sus desarrollos. Finalmente, otra contribución de progreso en algoritmos de *deep learning* fueron algunas mejoras que ocurrieron alrededor de los años 2009 y 2010, respecto a funciones de activación, esquemas de penalización de pesos y algoritmos de optimización (RMSProp [113], Adam [114], Adagrad [115], Adadelta [116], etc.). Estas permitieron entrenar modelos de red con diez o más capas de una manera más eficiente y para el 2014 al 2016 aún más avances fueron descubiertos como *batch normalization*, *residual connections*, *depthwise separable convolutions*, entre otras. Por lo que, actualmente se pueden entrenar modelos de miles de capas de profundidad en un menor tiempo.

1.3.1.1. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN), han jugado un rol importante en la historia del *Deep Learning*. En el año 1990 fueron empleadas para la lectura de cheques, con lo que se consiguió leer cerca del 10% de todos cheques en los bancos de Estados Unidos [109]. Veintidós años después de este hecho, Alex Krizhevsky presentó una red neuronal convolucional para clasificar 15 millones de imágenes catalogadas en 1000 categorías diferentes. Esta red fue nombrada *AlexNet* [117] y ganó la competencia denominada *ImageNet Large-Scale Visual Recognition Challenge* del año 2012 con un error en las predicciones de 15.4%, mientras que el segundo lugar obtuvo un error de 26.2%. Ésta marcada diferencia fue la que detonó el interés de la comunidad científica en el uso de las redes neuronales convolucionales para la clasificación de imágenes.

Luego, en 2014 Karen Simonyan y Andrew Zisserman [118] presentaron la red llamada VGG, la cual nace de la necesidad de reducir el número de parámetros en las capas convolucionales y mejorar el tiempo de entrenamiento. En el mismo año, se desarrolla *GoogleNet*, la cual tuvo como principales aportes el desarrollo de un módulo denominado *Inception* [119]. El módulo *inception* se compone de múltiples filtros de convolución que se aplican a la misma entrada y cuyos resultados son concatenados. Esto permite que el modelo de red aproveche la extracción de características de múltiples niveles. Este tipo de arquitectura además planteó una expansión a lo ancho, contrario a cómo comúnmente eran diseñadas las arquitecturas donde se incrementaba únicamente el número de capas en profundidad. Luego, *Microsoft* desarrolla *ResNet* [120] y gana

la misma competencia que en el año 2012 lo hiciera *AlexNet*, con un porcentaje de error de 3.6%, bastante superior al 15.4% de *AlexNet* 3 años atrás.

No solo en área de clasificación de imágenes se lograron grandes resultados. En tareas de reconocimiento de objetos, Ross Girshick en conjunto con otros investigadores desarrollaron una serie de algoritmos denominados *Region Based CNNs (R-CNN)*, *Fast R-CNN* y *Faster R-CNN* [121] [122] [123], los cuales son una evolución uno del otro a manera de mejorar la velocidad. Estos algoritmos, usan un método de búsqueda selectiva para extraer 2000 regiones de una sola imagen, para luego clasificar las regiones mediante CNN y detectar los objetos de interés.

Por otro lado, a inicios de 2017, Andrew G. Howard [124] diseñó una red denominada *Mobilenet* para el uso de redes neuronales convolucionales en teléfonos celulares, para ello desarrolló una nueva forma de realizar la convolución en una imagen, en la que se reducen la cantidad de operaciones a realizar por parte del procesador, sacrificando un poco la precisión en el reconocimiento de imágenes.

A pesar de los excelentes resultados generados mediante el uso de redes CNN, una de las principales desventajas de estas se da en la construcción de la arquitectura de la red, ya que, a pesar de existir métodos [125] y principios para su diseño [126][127], comúnmente la práctica es la que ayuda a conseguir la mejor arquitectura para un objetivo en particular. En palabras de Francois Chollet creador de keras, librería desarrollada para la implementación de algoritmos de *Deep Learning*, “crear la arquitectura adecuada de una red, es más un arte que una ciencia” [3]. Por lo que, a pesar del avance tan significativo que se ha presentado en los últimos años en el desarrollo de arquitecturas, aún hace falta el entender los mecanismos internos del funcionamiento de las redes neuronales convolucionales. Matthew D. Zeiler y Rob Fergus [129] realizaron un esfuerzo para visualizar e interpretar las características aprendidas por los filtros de una red convolucional entrenada, relacionado las características aprendidas de la red con los patrones de aprendizaje jerárquico en la corteza visual. Tal como se muestra en la Figura 1.4, en las primeras capas de la red se observan filtros de bordes o color y conforme se visualizan capas más profundas se observan patrones más definidos como formas o figuras.

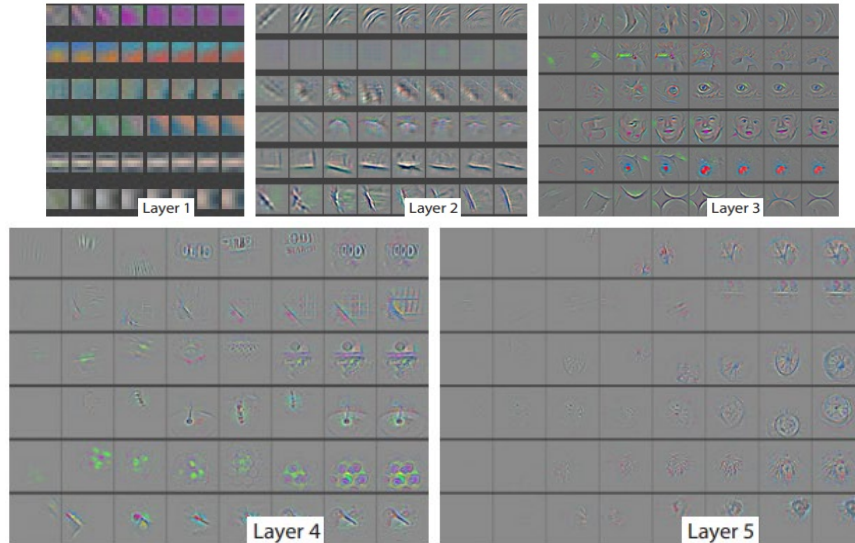


Figura 1.4. Características aprendidas por los filtros a través del entrenamiento [129].

Aunque Zeiler y Fergus mostraron de acuerdo con sus experimentos que las redes convolucionales aprenden de una manera alejada de la aleatoriedad, la tarea específica en la que se aplicó el experimento fue en clasificación de imágenes y se desconoce porque los filtros se comportaron de esta manera. Por lo que, aún se requieren más esfuerzos para investigar y comprender los principios fundamentales de las redes convolucionales y extender la experimentación en otras tareas, como detección de movimiento, segmentación semántica, regresión, entre otras.

1.3.1.1.1. Formalización

Las redes neuronales convolucionales son un tipo especializado de red neural artificial que utilizan la operación matemática de convolución para procesar los datos. Están diseñadas para trabajar con entradas estructuradas en forma tensorial. El ejemplo más claro de esto es una imagen bidimensional que presenta dependencias espaciales. Aunque también se puede trabajar con otras formas de datos estructurados secuenciales como el texto, audio o cualquier otro tipo de señales multidimensionales[130]. No obstante, debido a la naturaleza de este trabajo de tesis, el análisis de las redes convolucionales se centrará en trabajar con imágenes y en ocasiones se darán ejemplos enfocados específicamente a resolver la tarea de detección de movimiento.

Un modelo CNN es una extensión del modelo de una red neuronal artificial (ANN), el cual contiene un conjunto de neuronas artificiales interconectadas entre sí. La neurona artificial es un

modelo matemático cuyos componentes tienen una analogía directa a los de una neurona biológica. La neurona biológica es la unidad celular fundamental del sistema nervioso, es un simple elemento de procesamiento que recibe y combina señales de otras neuronas a través de conexiones de entrada llamadas dendritas. Si la combinación de las señales de entrada es lo suficientemente fuerte, la neurona se dispara, produciendo una señal que se transmitirá a otras neuronas a través del axón. Cada señal que llega a la neurona a través de las dendritas pasa por medio de una unión sináptica, la cual es un espacio infinitesimal lleno de un fluido neurotransmisor que libera sustancias químicas que retardan o aceleran el flujo de las cargas eléctricas hacia el núcleo o soma donde se procesan las señales. El ajuste de impedancia o conductancia del espacio sináptico es un elemento crítico que da lugar a los procesos de aprendizaje y memoria. Conforme las conexiones sinápticas se refuerzan y se ajustan el cerebro aprende y almacena información [131]. Por su parte, la neurona artificial contiene entradas X que son modificadas por pesos We cuyo funcionamiento es análogo al de la unión sináptica de la neurona biológica. Los pesos pueden ser positivos o negativos, emulando a la aceleración o inhibición de las señales de entrada. Además, los pesos se ajustan mediante un algoritmo de entrenamiento dando lugar al proceso de aprendizaje. El procesamiento realizado en el soma corresponde a la agregación de las entradas multiplicadas por los pesos, más un parámetro de polarización b el cual permite un grado de libertad adicional al modelo. Luego, se obtiene la salida de la neurona ζ cuando la agregación de las entradas por los pesos es pasada por una función de activación $f(\cdot)$. En la Figura 1.5 se observa la representación del modelo de una neurona biológica y una neurona artificial.

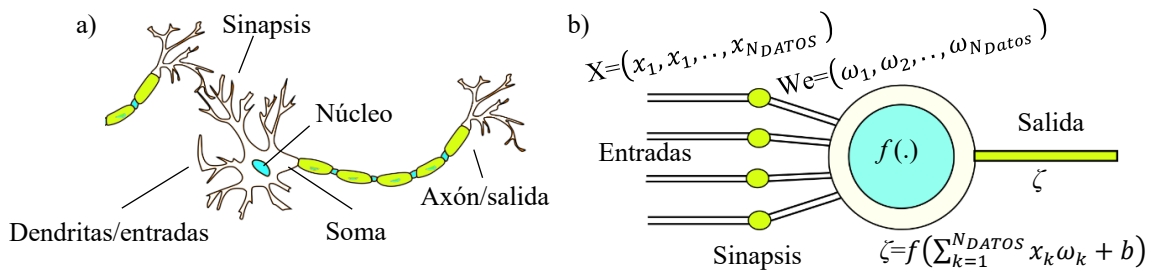


Figura 1.5. Modelos de neuronas: a) neurona biológica, b) neurona artificial.

Las redes ANN son un arreglo multicapa de neuronas artificiales, donde cada neurona está conectada a las salidas de neuronas en capas previas. A diferencia de las redes ANN, donde se realiza la etapa de agregación por medio de la multiplicación de los pesos We con las entradas X , en las redes CNN se utiliza una matriz de pesos compartidos Ke (también llamado filtro o *kernel*) que se recorre a través de la diferentes posiciones de la entrada G realizando la operación de

convolución [109]. La convolución es una operación matemática que describe una regla de como mezclar dos funciones o piezas de información: la entrada G y el filtro de convolución Ke se mezclan para formar un mapa de características fm . A continuación, se presenta la formulación matemática orientada a imágenes a color:

$$fm_{k+1,j}(x,y) = G_k * Ke_{k,j} = \sum_{z=0}^Z \sum_{u=-\frac{H}{2}}^{\frac{H}{2}} \sum_{v=-\frac{W}{2}}^{\frac{W}{2}} G(x+u, y+v, z)_k Ke_{k,j}(u, v, z) \quad (19)$$

El resultado de cada capa de convolución k con el j -ésimo *kernel* Ke es un nuevo mapa de características $fm_{k+1,j}$. Los j filtros utilizados definirán el número de mapas de características fm que forman la profundidad de la capa $k+1$. G es la entrada o el mapa de características de capas de convolución previas. (x, y, z) representan las coordenadas espaciales del mapa de características G , (u, v, z) son coordenadas espaciales del kernel Ke con ancho W y alto H , y Z es el canal de color de la imagen o la profundidad del mapa de características.

A continuación, se describen los elementos e hiperparámetros que participan en el proceso de aprendizaje de la red convolucional [112].

1.3.1.1.2. Capas de agrupamiento o *pooling*

Reducen el tamaño del mapa de características a través de un proceso de muestreo conservando los rasgos más distintivos o prominentes del mismo [109]. Estas capas pueden ser de dos tipos: obtenidas mediante el promedio de los valores de una ventana que se recorre por toda la imagen llamado *average pooling* o mediante la selección del valor máximo *max pooling*. La Figura 1.6 ilustra el proceso de *pooling*, para ambos casos se tiene una ventana de 2×2 que se recorre por la imagen, los colores anaranjado, blanco, verde y azul indican las posiciones por las que se desplaza la ventana. Las matrices resultantes del proceso de *pooling* se muestran a la derecha.

A continuación, se definen formalmente las capas de Max y Average *pooling*, donde se considera el máximo o el promedio de los datos que son considerados dentro del tamaño $W_k \times H_k$ de la ventana.

Max pooling:

$$fmp = \max(fm(x + i_w, y + i_h)); \quad (20)$$

$$i_w = \{1, \dots, W_k\}, \quad i_h = \{1, \dots, H_k\}$$

Average pooling:

$$fmp = \frac{1}{W_k * H_k} \sum_{i_w=1}^{W_k} \sum_{i_h=1}^{H_k} (fm(x + i_w, y + i_h)) \quad (21)$$

Donde fmp es el resultado de la capa de *pooling*, (x, y) posiciones espaciales del mapa de características fm o imagen y (i_w, i_h) son las posiciones que se recorre el operador dentro de la ventana.

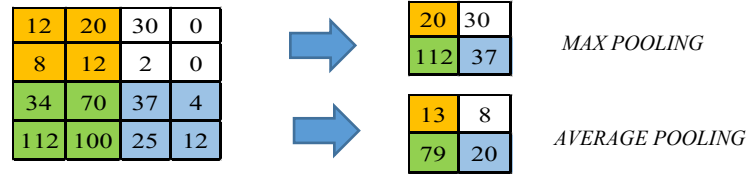


Figura 1.6. Capas de *Average pooling* y *Max pooling*.

Las operaciones de *pooling* proporcionan una invarianza básica o local (de acuerdo al tamaño de la ventana) a las rotaciones y traslaciones[109]. Esto significa, que si movemos o rotamos la entrada en una pequeña cantidad, los valores de la mayoría de las salidas agrupadas no cambian. Por ejemplo, en la Figura 1.7 se tiene un patrón en forma de “u” en la imagen 1 que luego se traslada a otra posición en la imagen 2, después de aplicar la operación de *max pooling* a ambas imágenes el patrón “u” que se genera es el mismo. Por lo que, se tiene una invarianza básica a la translación.

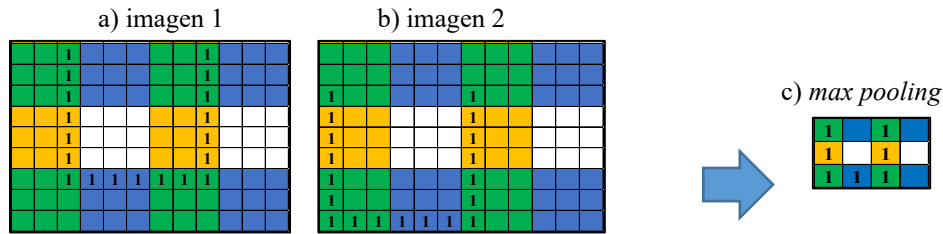


Figura 1.7. Invarianza básica a la translación al aplicar capas de *pooling*. Los pixeles con valor de “1” muestran el patrón y las casillas sin valor numérico en la imagen poseen valor de cero. Los cambios de color muestran las posiciones sobre las que se desplaza la ventana de *max pooling* de tamaño 3x3.

Por otro lado, en la Figura 1.8 se tiene un patrón en forma de “7” en la imagen 1 que luego se traslada a otra posición con cierta rotación (véase imagen 2), después de aplicar la operación de *max pooling* a ambas imágenes, el patrón “7” que se genera es el mismo. Por lo que, se tiene una invarianza básica a la rotación. En las capas de *pooling* cuanto mayor sea el tamaño del área de la ventana más información se condensa, lo que conduce a redes más simples, con menor tamaño y por lo tanto más rápidas. Sin embargo, si se abusa del tamaño de la ventana se desecha demasiada información, lo que provoca una disminución en el rendimiento predictivo de la red.

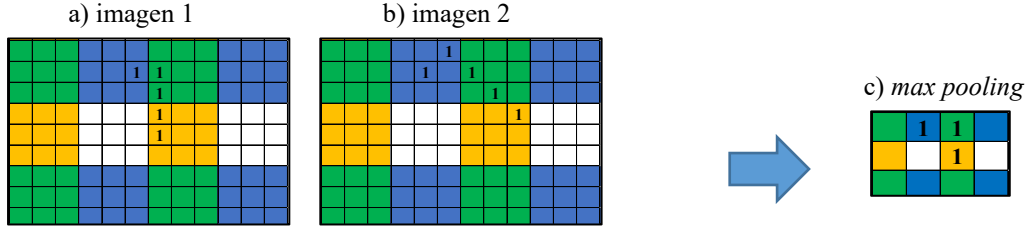


Figura 1.8. Invarianza básica a la rotación al aplicar capas de *max pooling*. Los pixeles con valor de “1” muestran el patrón y las casillas sin valor numérico en la imagen tienen valor de cero. Los cambios de color muestran las posiciones sobre las que se desplaza la ventana de *max pooling* de tamaño 3x3.

1.3.1.1.3. Capas totalmente conectadas (*fully connected*)

Otra de las capas que se utilizan en la construcción de una red convolucional son las capas totalmente conectadas. En la Figura 1.9 se observa la representación de una capa totalmente conectada, donde cada neurona está totalmente conectada a las salidas de las neuronas de la capa previa. La salida de esta capa normalmente se asocia a problemas de clasificación, es decir; cada neurona representa una posible clase. Por lo que la neurona con la mayor activación a su salida será la clase ganadora para el patrón de entrada. Formalmente la ecuación de una capa totalmente conectada es la siguiente:

$$\zeta_o = f\left(\sum_{k=1}^W \sum_{j=1}^H (X_{k,j} W e_{k,j} + b_k)\right)_o \quad (22)$$

Donde para cada o -ésima salida ζ_o se tiene una sumatoria de la multiplicación de las entradas $X_{k,j}$ con los pesos $W e_{k,j}$ más el sesgo b .

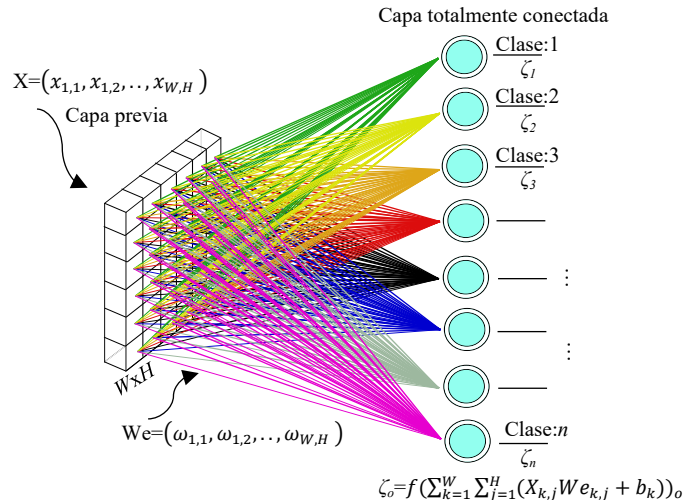


Figura 1.9. Ejemplo de una capa totalmente conectada.

1.3.1.1.4. Funciones de activación

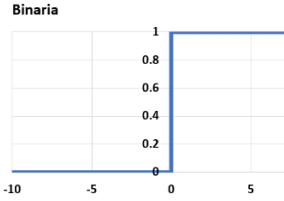
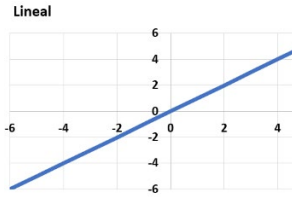
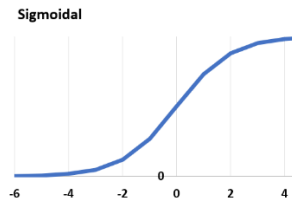
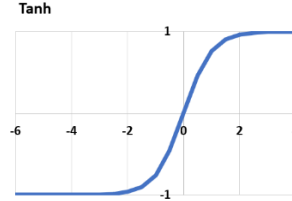
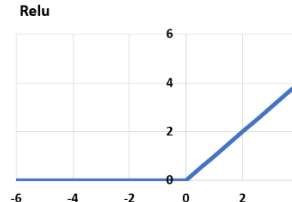
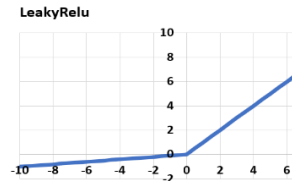
La función de activación determina la tasa de disparo de una neurona de acuerdo con la suma ponderada de los nodos de entrada [112][130], además normaliza la salida de la neurona en un rango, por ejemplo $[1,0]$ o $[-1,1]$. Existen tres tipos de función de activación, binaria, lineal y no lineal.

La función binaria también llamada *hard-limit* consiste en un simple umbral de decisión, si el valor de entrada supera un valor de umbral específico entonces la neurona se dispara. La función lineal otorga una salida proporcional a la entrada. A diferencia de la función binaria, otorga un mayor rango de valores de salidas, ya no es solo 1 o 0, sino un rango como $[-1,1]$. El inconveniente de usar este tipo de funciones de activaciones es que no es posible utilizarlas en el algoritmo de entrenamiento *backpropagation*, ya que su derivada es una constante y no se puede establecer una relación con la entrada que logre reducir el error. Además, si todas las capas poseen funciones de activación lineal (sin considerar operadores no lineales como *maxpooling*), no importa cuantas capas tenga la red, la función de activación final de la última capa será solo una función lineal de la primera, por lo tanto, se dice que la toda la red es equivalente a una simple capa con función de activación lineal[132].

El algoritmo *backpropagation* requiere que las funciones sean diferenciables y no lineales para lograr minimizar el error [127]. Las funciones no lineales permiten crear mapeos no lineales entre las salidas y entradas de la red, lo cual es esencial para que la red aprenda y modele datos complejos como imágenes, video, audio y datos no lineales de alta dimensionalidad. Además, las funciones no lineales son adecuadas para realizar *backpropagation* ya que son derivables y permiten la concatenación de múltiples capas para crear estructuras de red profundas. Algunas de las funciones de activación no lineales más utilizadas son: relu, leakyrelu, tangente hiperbólica y sigmoideal.

En la Tabla 1.6 se describen algunas funciones de activación, así como sus gráficas y su formalización. Como es sabido el gradiente de una función de activación es de suma importancia en el entrenamiento de la red. Por lo que, para seleccionar alguna de las diferentes funciones de activación se debe considerar tanto su funcionamiento en propagación hacia enfrente como en retropropagación en el entrenamiento. La Tabla 1.6 también señala las ventajas y problemas que podrían presentarse al utilizar un tipo de función de activación específica [112][127][130][132].

Tabla 1.6. Características y tipos de funciones de activación.

Función	Ecuación	Gráfica	A favor	En contra
Binaria	$a(x_{in}) = \begin{cases} 0 & \text{si } x_{in} < 0 \\ 1 & \text{si } x_{in} \geq 0 \end{cases}$		Respuesta de salida binaria. Se pueden utilizar para construir compuertas lógicas.	No permite salidas de múltiples valores. No puede ser usada en el entrenamiento debido a que su derivada es cero.
Lineal	$a(x_{in}) = x_{in}$		Permite múltiples valores a la salida. Se utiliza comúnmente a la salida de la red para realizar tareas de regresión.	Su derivada no permite establecer una relación entre la salida y la entrada que ayude a reducir el error por retropropagación.
Sigmoidal	$a(x_{in}) = \frac{1}{1 + e^{-x_{in}}}$		Otorga una salida suavizada. Los valores de salida se normalizan de [0, 1]. Es comúnmente aplicada a la salida de la red para otorgar un valor de probabilidad que puede ser utilizado en clasificación binaria.	Si los valores de x_{in} están lejos del origen, el valor obtenido por la función sigmoide derivada será muy pequeño. Por lo tanto, si existen múltiples capas con la función sigmoide el gradiente se vuelve aproximadamente cero en las primeras capas, esto se conoce como <i>Vanishing gradient</i> .
Tanh	$a(x_{in}) = \frac{e^{x_{in}} - e^{-x_{in}}}{e^{x_{in}} + e^{-x_{in}}}$		Es una versión reescalada de la función sigmoide. Esta centrada en cero y esta propiedad incrementa la velocidad de convergencia. Normaliza la salida de [-1, 1].	Al igual que la función sigmoide Tanh sufre de <i>Vanishing gradient</i> . Lo que conduce a que la red se paralice y deje de aprender.
Relu	$a(x_{in}) = \begin{cases} 0 & \text{si } x_{in} < 0 \\ x_{in} & \text{si } x_{in} \geq 0 \end{cases}$		Computacionalmente eficiente. Permite que la red converja rápidamente. Puede causar que algunas neuronas sean cero, funcionando como una especie de regularizador.	Cuando x_{in} es negativa, el gradiente de la función se convierte en cero, y en una red de múltiples capas no se podrá realizar la retropropagación, este problema se conoce como <i>Dying relu</i> .
Leakyrelu	$a(x_{in}) = \begin{cases} x_{in} * \alpha & \text{si } x_{in} < 0 \\ x_{in} & \text{si } x_{in} \geq 0 \end{cases}$ $\alpha = [0, 1]$		Evita el problema <i>Dying Relu</i> ya que permite la retropropagación incluso para los valores de entrada negativos	Produce resultados similares a Relu debido a que la región positiva es idéntica. Sin embargo, requiere del ajuste del parámetro alfa para obtener mejores resultados.

1.3.1.1.5. Funciones de pérdida

En teoría de la decisión u optimización matemática, una función de costo es una función que asigna un valor que representa intuitivamente algún "costo" o "error" asociado a un evento. El objetivo de la estrategia de optimización es minimizar la función de costo. Aunque términos de función de costo y función de pérdida son utilizados comúnmente como sinónimos, en términos de entrenamiento son diferentes. Una función de pérdida es el error asociado a una simple muestra de entrenamiento y es también llamada función de error. Por otro lado, una función de costo es la pérdida o error promedio sobre todo el conjunto de datos de entrenamiento. Las funciones de pérdida se pueden clasificar dependiendo de la tarea que se desee realizar [133]. En tareas de regresión, las funciones de pérdida más comúnmente utilizadas son: *Squared Error Loss*, *Mean Squared Error*, *Absolute Error Loss* y *Huber Loss*. A continuación, se muestra un ejemplo de una función de pérdida E usando el error cuadrado medio y la función de costo J correspondiente a el promedio del error sobre el número de datos de entrenamiento N_{Datos} :

$$\begin{aligned} E &= \frac{1}{2}(y_p - y_t)^2 \\ J &= -\frac{1}{N_{\text{Datos}}} \sum_{q_D=1}^{N_{\text{Datos}}} E(y_p, y_t)_{q_D} \end{aligned} \tag{23}$$

Donde y_p es la predicción realizada por la red para un dato q_D específico y y_t es el *target* o valor deseado asociado al dato.

Por otra parte, en tareas de clasificación se identifican dos tipos de función de pérdida, multiclase y binaria. En multiclase dos muy populares son *Kullback Leibler Divergence Loss* y *Multi-class Cross Entropy Loss* y para clasificación binaria *Hinge Loss* y *Binary Cross-Entropy*. Estas últimas son descritas a continuación debido a que este trabajo se enfoca en resolver un problema de clasificación binaria (1: objeto en movimiento, 0: fondo).

Hinge Loss: es una función de costo que era mayormente utilizada en las máquinas de soporte vectorial pero después se extendió su uso a las redes neuronales artificiales. Los valores objetivo o *target* para esta función se encuentran en el conjunto $\{-1, 1\}$, por lo que si se tiene una entrada binaria se debe convertir el valor de "0" a "-1". El error entre las clases se calcula mediante:

$$E_{\text{Hinge}} = \max(1 - (y_p * y_t), 0) \tag{24}$$

La función de costo J_{Hinge} asociada a la función del error está dada por:

$$J_{Hinge} = -\frac{1}{N_{Datos}} \sum_{q_D=1}^{N_{DATOS}} \max(1 - (y_p^{q_D} * y_t^{q_D}), 0) \quad (25)$$

La ecuación de error E_{Hinge} anterior se puede expresar también de la siguiente manera, teniendo dos casos, para $y_t=1$ y $y_t=-1$:

$$E_{Hinge} = \begin{cases} \max(1 - (y_p), 0) & \text{si } y_t = 1 \\ \max(1 + (y_p), 0) & \text{si } y_t = -1 \end{cases} \quad (26)$$

Las gráficas de esta función de error se muestran en la Figura 1.10. La función de pérdida Hinge es una idea conceptual basada en clasificadores marginales, por lo que si el valor y_p es correcto y mayor que el margen, no existirá error. Si el valor y_p es cercano a cero existirá un error pequeño, y si el valor y_p es incorrecto y alejado a cero se incrementa el error proporcionalmente [133].

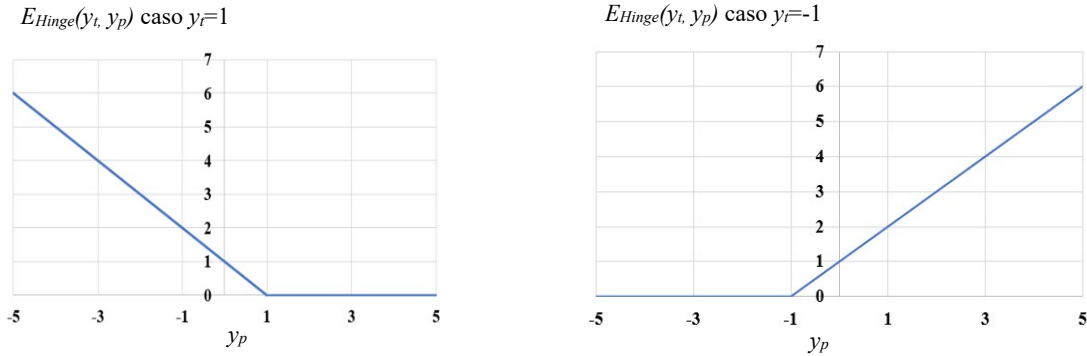


Figura 1.10. Gráficas función de pérdida Hinge.

Binary Cross-Entropy: es la función de pérdida más utilizada en clasificación binaria. Esta función de pérdida básicamente mide la diferencia de la entropía asociada con la distribución asociada a una variable aleatoria respecto a la distribución de *target* u objetivo [134]. En conceptos de la teoría de la información, se puede definir a la entropía de una manera intuitiva como la medida de incertidumbre asociada a una distribución. Por ejemplo, en el caso de arrojar una moneda cargada o sesgada a favor de caer cara, se tendría una entropía baja porque se espera que la moneda caiga cara en cada lanzamiento y no existe incertidumbre. Por otra parte, en el caso de una moneda ordinaria, se tendría una alta entropía porque las salidas del evento “arrojar la moneda” son inesperadas y se tiene incertidumbre sobre si esta caerá cara o cruz. La ecuación del cálculo de entropía H_{Ent} se define por:

$$H_{Ent}(P) = -\sum_n P(n) \log(P(n)) \quad (27)$$

Donde P es la distribución de probabilidad asociada a una variable aleatoria y n indica un evento discreto, del ejemplo anterior de la moneda, n representaría el evento de que la moneda caiga cara o cruz.

Dado lo anterior, si se conoce la distribución de probabilidad de una variable aleatoria se puede calcular su entropía. Entonces, si se desea comparar la entropía de dos distribuciones de probabilidad diferentes, se utiliza la siguiente ecuación de *Kullback-Leibler Divergence* o de entropía cruzada $H_{Ent}(P, Q)$:

$$H_{Ent}(P, Q) = -\sum_n Q(n) \log(P(n)) \quad (28)$$

$H_{Ent}(P, Q)$ es una medida de similitud entre la entropía de la distribución objetivo Q respecto a la distribución P asociada a la variable aleatoria. Si P y Q concuerdan perfectamente, el valor de entropía es 0 y esta es la base del algoritmo de optimización. Se busca la mejor estimación de la distribución P que se ajuste a la distribución Q la cual minimiza el valor de entropía cruzada o error.

Si se lleva la ecuación anterior al problema específico de detección de movimiento $n=\{1:objeto \text{ en movimiento } (Obj.mov), 0:fondo\}$, se tiene la siguiente expresión:

$$H_{Ent}(P, Q) = -Q(Obj.mov) \log(P(Obj.mov)) - Q(Fondo) \log(P(Fondo)) \quad (29)$$

Si se sabe que $P(Fondo)=(1-P(Obj.mov))$ y $Q(Fondo)=(1-Q(Obj.mov))$, $H_{Ent}(P, Q)$ se puede expresar de la siguiente manera:

$$H_{Ent}(P, Q) = -Q(Obj.mov) \cdot \log(P(Obj.mov)) - (1 - Q(Obj.mov)) \cdot \log(1 - P(Obj.mov)) \quad (30)$$

Simplificando $Q = Q(Obj.mov)$ y $P = P(Obj.mov)$, la expresión se convierte en:

$$H_{Ent}(P, Q) = -Q \cdot \log(P) - (1 - Q) \cdot \log(1 - P) \quad (31)$$

Durante el entrenamiento se usa el número total de píxeles $Np=N_{Obj.mov}+N_{Fondo}$ de la imagen de entrenamiento para calcular la función de costo de entropía cruzada $J_{BinaryCrossEntropy}$, por lo tanto, la expresión será:

$$J_{BinaryCrossEntropy} = -\frac{1}{Np} \sum_{i=1}^{Np} (Q(i) \cdot \log(P(i)) + (1-Q(i)) \cdot \log(1-P(i))) \quad (32)$$

Siendo $Q(i)$ el valor de probabilidad del dato i o pixel en la imagen del GT de pertenecer a un objeto dinámico y $P(i)$ la probabilidad estimada por la red para que el pixel i pertenezca a un objeto dinámico. En la Figura 1.11 se muestran las gráficas de esta función de costo, teniendo los siguientes casos:

- 1) Si, $Q(i)=1$ (Obj.movimiento) se usaría la expresión de la parte izquierda de la sumatoria de la ecuación 32: $-\log P(i)$
- 2) Si, $Q(i)=0$ (fondo) se trabaja con la parte de la derecha del sumando: $-\log(1-P(i))$

De las gráficas de la Figura 1.11 se puede apreciar lo siguiente: en el caso 1 conforme la predicción de la red $P(i)$ se aproxima a 1 (que es valor ideal objetivo $Q(i)$), el error de la función se vuelve cero, pero conforme $P(i)$ se aleja del valor ideal, el error aumenta exponencialmente. En el caso 2, ocurre algo similar, cuando $P(i)$ se aproxima al valor ideal $Q(i)=0$, el error tiende a cero y conforme $P(i)$ se aleja del valor ideal el error incrementa.

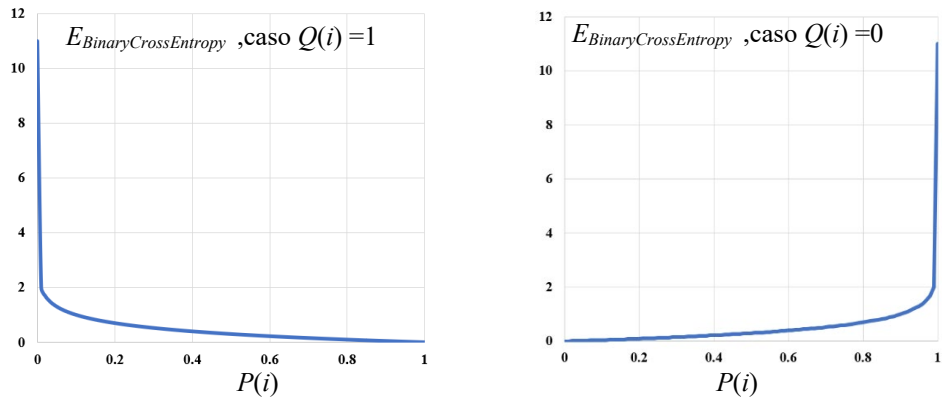


Figura 1.11. Gráficas función de pérdida *BinaryCrossEntropy*.

1.3.1.1.6. Técnicas de regularización

Los regularizadores, de acuerdo con [109], son cualquier modificación que se realiza al algoritmo de entrenamiento con el fin de reducir su error de generalización, pero esto no implica

que se reduzca el error en el entrenamiento. Los errores de generalización y de entrenamiento, aunque ocurren conjuntamente durante el aprendizaje, no tienen un mismo significado. El error de entrenamiento ocurre cuando el algoritmo de optimización busca minimizar la función de costo asociada al error entre la predicción de la red y el GT. No obstante, el verificar el error de entrenamiento no es suficiente ya que el desafío principal en aprendizaje de máquina es obtener un algoritmo que posea la capacidad de generalizar. La generalización, se define como la habilidad del algoritmo para obtener un buen desempeño tanto en las muestras para las que fue entrenado, como en datos nuevos o desconocidos. El error de generalización se puede observar al monitorear el desempeño del algoritmo en las etapas de entrenamiento y validación. Si el desempeño en entrenamiento y validación es bajo, se tiene un caso de *underfitting* o subajuste. El *underfitting* se debe a que el modelo no fue capaz de ajustarse a los datos de entrenamiento, por lo que también los nuevos datos son clasificados incorrectamente. Otro caso es cuando el desempeño de entrenamiento es alto y el de validación bajo, aquí se cae en el problema de *overfitting* o sobreajuste. El *overfitting* sucede cuando el número de parámetros del modelo es muy alto y la red se ajusta demasiado a los datos de entrenamiento, lo que provoca una clasificación incorrecta cuando llegan nuevos datos. Lo ideal es tener un modelo balanceado que tenga un desempeño similar en entrenamiento y validación para que se logre una correcta generalización. La Figura 1.12 ejemplifica los casos de *overfitting*, correcta generalización, y *underfitting*, en un problema aplicado a clasificación de clases; roja y negra.

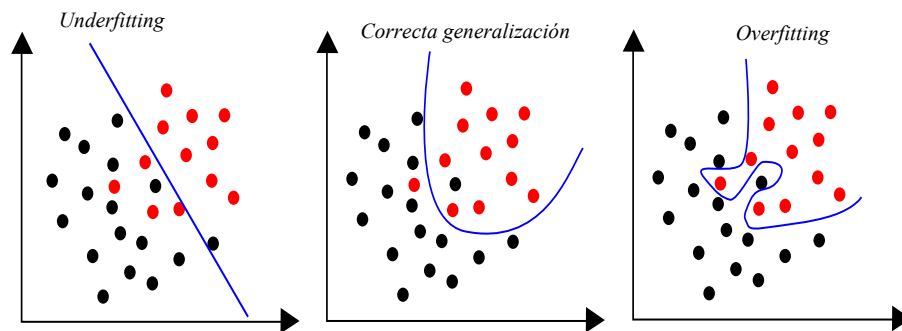


Figura 1.12. Situaciones de *overfitting*, correcta generalización y *underfitting*.

Una forma eficaz de resolver los problemas de *overfitting* y *underfitting*, es utilizar regularizadores, puesto que simplifican la tarea del algoritmo de optimización y ayudan a la red a lograr una mejor generalización de los datos. A continuación, se describen los regularizadores más comúnmente utilizados en el estado del arte.

Normas L1 y L2:

Las normas $L1$ y $L2$ son dos de las técnicas más comunes de regularización, estas modifican la función de costo J al agregar un término de regularización r y un parámetro $\lambda \in [0, \infty)$ que controla la contribución de la regularización r , para obtener una función de costo regularizada L :

$$L = J + \lambda r \quad (33)$$

A grandes valores de λ se tiene un mayor valor de regularización y cuando $\lambda=0$ se regresa a la función de costo original $L=J$.

La norma $L1$ agrega un factor de regularización r en la función de costo definido por:

$$r = \sum_{j=1}^{N_w} |We_j| = |We_1| + |We_2| + |We_3| + \dots + |We_{N_w}| \quad (34)$$

Siendo N_w el número total de pesos o parámetros We .

Para visualizar el comportamiento de este parámetro de regularización r en la función de costo veamos el caso en dos dimensiones $N_w=2$:

$$\sum_{j=1}^2 |We_j| = |We_1| + |We_2|; \quad Ct = |We_1| + |We_2| \quad (35)$$

donde Ct es una constante que corresponde a la suma de ambos pesos We , por lo tanto, $|We_1| \leq Ct$ y $|We_2| \leq Ct$. Entonces, si $Ct=1$, se obtiene la gráfica de la Figura 1.13a. Para el caso en tres dimensiones $\sum_{j=1}^3 |We_j| = |We_1| + |We_2| + |We_3|$ se obtiene la Figura 1.13b.

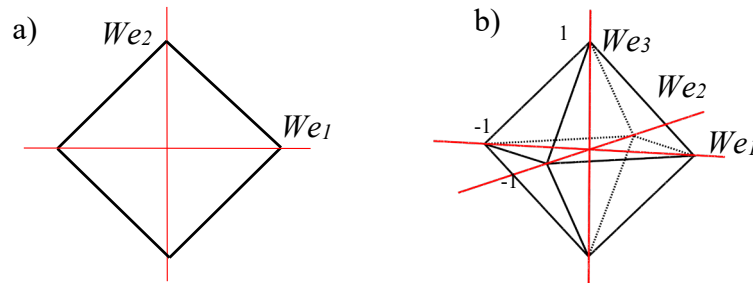


Figura 1.13. Gráfica de la ecuación 33 con $Ct=1$; a) caso dos dimensiones We , b) caso tres dimensiones We .

Como se observa en la Figura 1.13, la norma $L1$ posee una forma convexa (también conocida como forma romboide o diamante). Esta forma hace que la norma $L1$ sea útil para lograr modelos esparcidos. La idea es que, durante la minimización del error, el valor de los pesos estará restringido por el límite definido por la norma $L1$ [109]. Usando la norma $L1$, la función de costo tiene una

mayor posibilidad de golpear las puntas del romboide que los lados, debido a su forma convexa. Por lo tanto, la regularización L1 conduce a que algunos pesos se conviertan en cero. Por ejemplo, en la Figura 1.14 se aprecia que el contorno de la función de pérdida toca el romboide en el eje We_2 , por lo que el valor de We_1 será cero, este comportamiento conduce a la generación de un modelo esparcido donde se requiere un menor número de parámetros para lograr una correcta generalización.

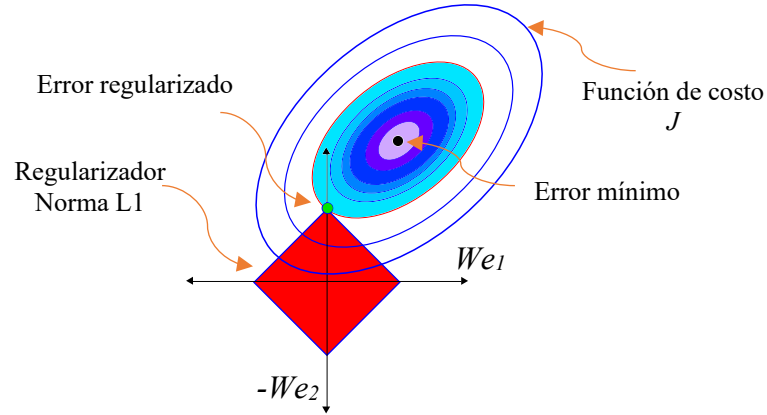


Figura 1.14. Efecto de regularización L1 en función de pérdida.

Como se mencionó anteriormente el sobreajuste ocurre cuando el modelo o algoritmo contiene demasiados parámetros en comparación con los datos de entrenamiento, lo que provoca que el modelo recuerde cada uno de los casos de entrenamiento e incluso si existen datos ruidosos. Por lo que una forma efectiva de evitar esta situación y lograr una mejor generalización es utilizar la norma L1 para descartar la información de algunas neuronas.

Por su parte, en la norma L2 se agrega un factor de regularización r en la función de costo definido por la sumatoria cuadrática de los pesos:

$$r = \sum_{j=1}^{N_w} We_j^2 = We_1^2 + We_2^2 + We_3^2 + \dots + We_{N_w}^2 \quad (36)$$

$$L = J + \lambda r$$

Esta norma al igual que la norma L1 restringe el valor de los pesos We , pero debido a que es una función cuadrática si se analiza en dos dimensiones $N_w=2$, la restricción se da por medio de una forma circular, como se observa en la Figura 1.15. La regularización L2 obliga a los pesos a tener valores cercanos a cero, pero no cero, contrario al regularizador L1. Por lo tanto, la solución no es esparcida.

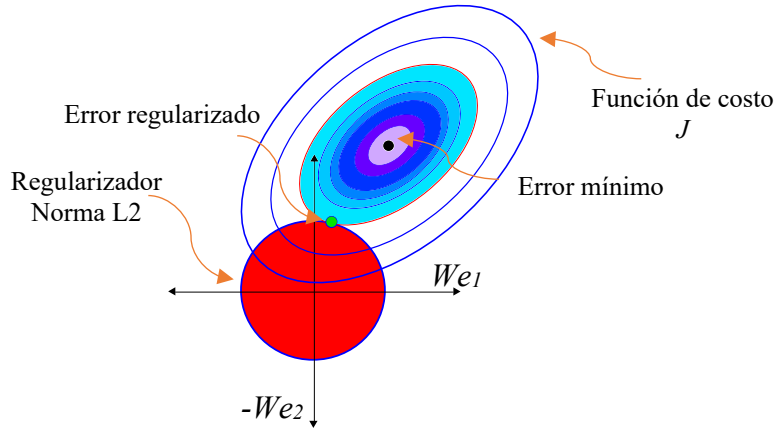


Figura 1.15. Efecto de regularización L2 en función de pérdida.

Dropout:

Es un proceso diseñado para evitar el *overfitting* y ayudar a la generalización de la red donde algunas neuronas de manera aleatoria son ignoradas durante la etapa de entrenamiento, de tal modo que las neuronas en capas ocultas no aprenden información redundante o de cada detalle de los datos de entrenamiento. Por ejemplo, supongamos una red con tres capas ocultas y cinco neuronas en cada capa oculta. Si dos neuronas de la primera capa son ignoradas, sus activaciones no se propagan hacia adelante, por lo tanto, las neuronas de las siguientes capas tienen que aprender de una menor cantidad de información (de solo tres neuronas). En otras palabras, las neuronas de capas más profundas tienen que averiguar cómo ajustarse para generar o encontrar patrones de los datos no vistos. Además, cuando algunas neuronas están inactivas se reducen los cálculos, lo que impacta de manera favorable en el tiempo de entrenamiento. Formalmente de acuerdo con [135] una red que posee el proceso de *dropout* se define mediante:

$$\begin{aligned}
 r_l &\sim \text{Bernoulli}(\rho r) \\
 \tilde{Y}_l &= r_l * Y_l \\
 Sa_{l+1} &= \tilde{Y}_l * We_{l+1} + b_{l+1}
 \end{aligned} \tag{37}$$

Donde r_l es un vector de variables aleatorias independientes de Bernoulli, cada una de estas variables aleatorias tiene una probabilidad ρr de ser 1. $\tilde{Y}$ es la salida “disminuida” o “adelgazada” generada al multiplicar los valores del vector r_l con la salida Y_l de la capa l -ésima de la red, por lo que la salida de la red Sa_{l+1} será la multiplicación de $\tilde{Y}_l$ por los pesos del filtro We_{l+1} más el sesgo b_{l+1} .

En la Figura 1.16 se observa la ejemplificación de una red convencional y una red con el proceso de *dropout*. En la red con *dropout*, r_l actúa como un interruptor que decide si la salida de la neurona Y_l en la capa l es tomada en cuenta, o no, en el proceso de entrenamiento, lo que genera la salida adelgazada o disminuida $\tilde{Y}_l$.

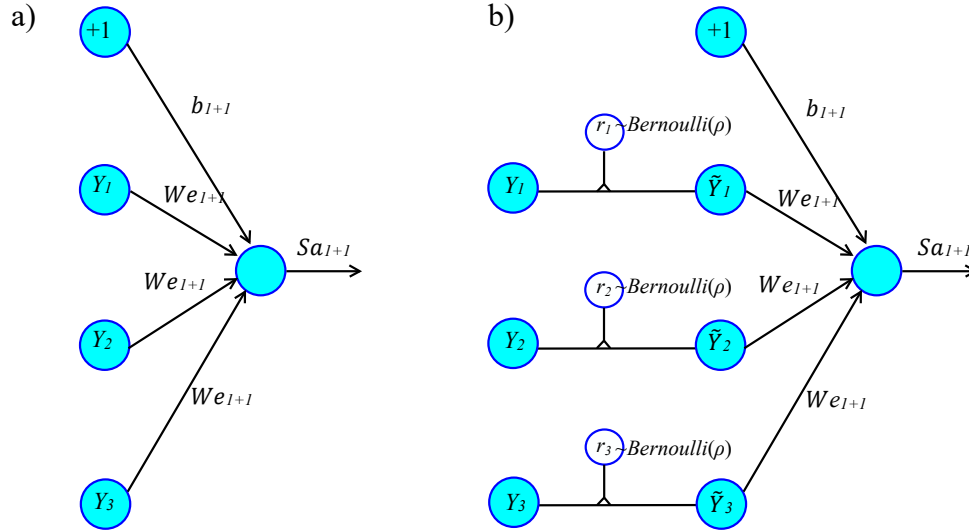


Figura 1.16. Comparación de una red estándar a) y de una red con *dropout* b).

Para utilizar *dropout*, los autores en [135] recomiendan que el parámetro pr sea de 0.5 para capas intermedias y $pr = 0.8$ para la capa de entrada. En la Figura 1.17 se muestra un ejemplo de una red con *dropout* $pr=0.5$ para ilustrar la pérdida de conexiones que evita el sobre entrenamiento y ayuda a obtener una mejor generalización.

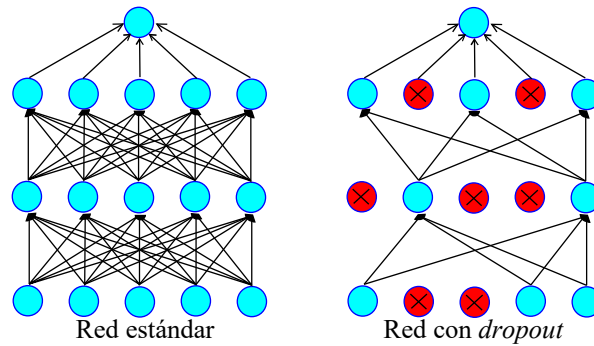


Figura 1.17. Red estándar y red con *Dropout*.

Aumento de datos:

El aumento de datos abarca un conjunto de técnicas que mejoran el tamaño y la calidad de los conjuntos de datos de entrenamiento, de manera que se puedan construir mejores modelos de

aprendizaje profundo con ellos. Los algoritmos de aumento de imágenes incluyen transformaciones geométricas, de espacio de color, filtros, mezcla de imágenes, borrado aleatorio, aumento y disminución de brillo, redes generativas de adversarias, entre otros [130]. El aumento de datos de acuerdo con [136], es similar a la imaginación o los sueños, los humanos imaginan diferentes escenarios basados en la experiencia y esto les ayuda a comprender mejor el mundo.

La técnica de aumento de datos es utilizada para reducir el *overfitting* producido por utilizar un reducido número de imágenes de entrenamiento y arquitecturas complejas que poseen una elevada cantidad de parámetros. El uso de aumento de datos evita que la red memorice cada una de las imágenes de entrenamiento. Además, resulta muy efectivo en dominios de aplicación que no tienen acceso a una gran cantidad de datos, como el análisis de imágenes médicas.

1.3.1.1.7. Técnicas de optimización

De manera general, los optimizadores desempeñan la función de actualizar los parámetros de las neuronas de la red en la etapa de entrenamiento, a manera de minimizar una función de pérdida en la dirección de la solución óptima. Los algoritmos de optimización se clasifican en dos categorías: primer y segundo orden. Los optimizadores de primer orden [137] minimizan la función objetivo usando valores de gradientes. Estos gradientes indican si la función está decreciendo o creciendo; el optimizador de primer orden más utilizado es el de gradiente descendente. Los algoritmos de optimización de segundo orden, como su nombre lo indica, utilizan derivadas de segundo orden también llamadas matrices Hessianas para minimizar la función objetivo. Sin embargo, debido a que el cálculo de estas matrices Hessianas es costoso computacionalmente, su uso no ha sido ampliamente extendido.

Existen varias técnicas de optimización, pero una de las más populares y utilizadas en el aprendizaje, es la de gradiente descendente, la cual a través de los años se le ha incorporado elementos para mejorar su ejecución. Algunos algoritmos derivados de la técnica de gradiente descendente son descritos a continuación:

Gradiente descendente estocástico

El gradiente descendente estocástico o SGD por sus siglas en inglés es el esquema básico del gradiente descendente. Primero, los datos de entrada se pasan a la red y se propagan hacia adelante, calculando el producto de las señales de entrada y los pesos, para luego pasar por una función de

activación que obtiene la salida de la red. Después, el algoritmo de aprendizaje a través de la retropropagación o *backpropagation* propaga el error de atrás hacia adelante y actualiza los parámetros de la red. La actualización de los pesos We se da mediante:

$$We(t+1) = We(t) - \eta \cdot \nabla J(We(t); x_i; y_{o_i}) \quad (38)$$

Donde x_i y y_{o_i} son respectivamente las muestras y objetivos i -ésimos de entrenamiento, de entrada y salida respectivamente, ∇J es el gradiente del error y η el parámetro de aprendizaje. Uno de los principales inconvenientes de SGD es escoger un valor adecuado para el parámetro η con el fin de permitir una óptima convergencia. De modo que, si η es grande la minimización del error será oscilatoria debido a que la actualización de los pesos se realiza a pasos grandes, si η es pequeño el algoritmo de minimización avanzaría hacia al mínimo lentamente sin asegurar una correcta convergencia, (véase Figura 1.18).

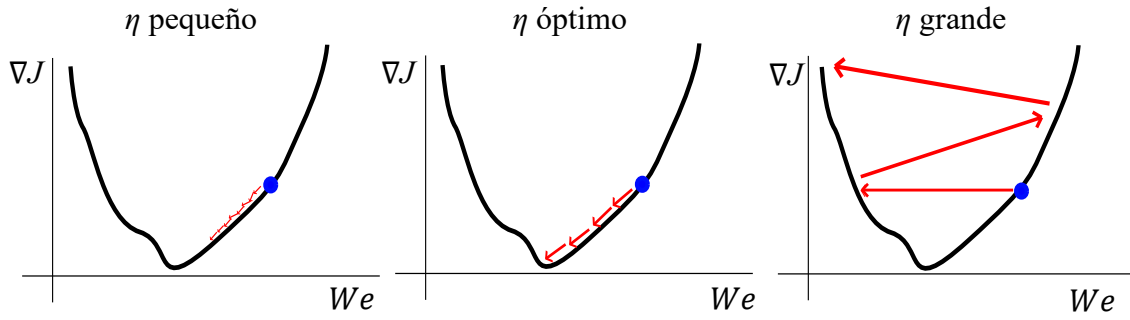


Figura 1.18. Importancia de la selección correcta del parámetro η .

Gradiente descendente estocástico con momento

Un problema muy común con el esquema anterior de entrenamiento SGD, se debe a que la superficie del error puede presentar una cantidad considerable de valles y crestas. Por lo que, es posible caer en mínimos locales que no aseguran un correcto desempeño. Por lo tanto, como una mejora al método SGD se introduce un parámetro llamado momento, que ayuda a acelerar los vectores de gradientes en las direcciones correctas para evitar caer en mínimos locales y proveer una rápida convergencia.

$$\begin{aligned} V(t) &= \gamma \cdot V(t-1) + \eta \cdot \nabla J(We(t); x_i; y_{o_i}); & V(0) &= 0 \\ We(t+1) &= We(t) - V(t) \end{aligned} \quad (39)$$

Donde γ es el parámetro de momento y $V(t)$ es el parámetro de decremento en la actualización de pesos. De la ecuación 40, se observa que la actualización de los pesos We considera los cambios

producidos en pasos anteriores $V(t-1)$. Por lo tanto, si los gradientes anteriores iban en la misma dirección que los nuevos se produce un mayor incremento en $V(t)$. En la Figura 1.19 se ilustra el comportamiento del parámetro momento. La actualización de los pesos sin utilizar momento cae en un mínimo local en la imagen Figura 1.19a. Sin embargo, en la imagen Figura 1.19b se logra salir del valle debido a la fuerza de empuje del parámetro momento a favor de la tendencia de los gradientes.

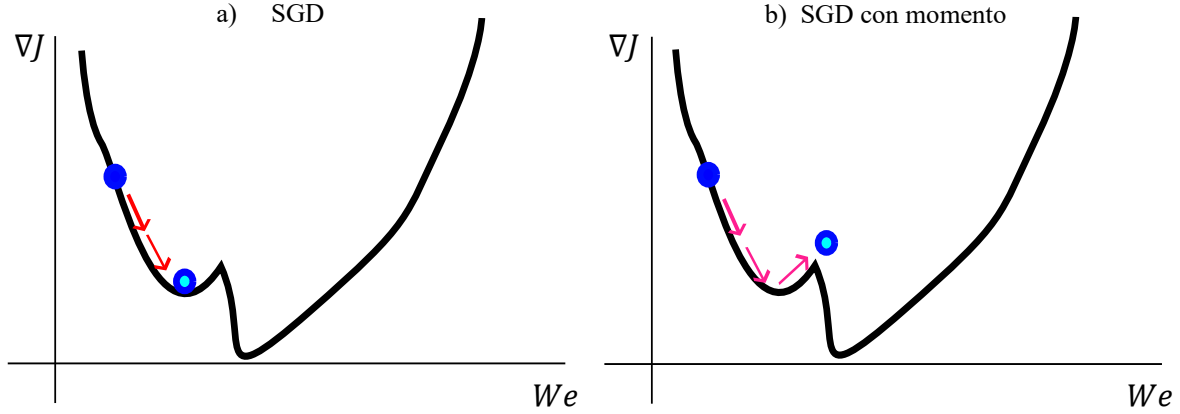


Figura 1.19. Comportamiento del algoritmo SGD con momento.

Gradiente descendente con optimizador Adagrad

El optimizador Adagrad (*Adaptive gradient*) modifica el parámetro de aprendizaje η en cada iteración para cada peso We_i basado en los gradientes cuadrados g previamente calculados, mediante:

$$\begin{aligned}
 g_0 &= 0 \\
 g_i(t) &= g_i(t-1) + \nabla J(We)^2 \\
 We_i(t+1) &= We_i(t) - \frac{\eta_i}{\sqrt{g_i(t) + \varepsilon}} \cdot \nabla J(We)
 \end{aligned} \tag{40}$$

Siendo $\varepsilon \rightarrow 0$ un parámetro que asegura la no división entre cero.

El optimizador Adagrad elimina la necesidad del ajuste manual del parámetro de aprendizaje η ya que cada coeficiente We tiene su propia tasa de aprendizaje. Sin embargo, esta tasa de aprendizaje disminuye monótonicamente, lo que causa que llegue a ser tan pequeña que la red deje de aprender, este problema también es conocido como *diminishing learning rate*.

Gradiente descendente con optimizador RMSprop

RMSprop (*Root mean square propagation*) fue presentado por Geoffrey Hinton en la clase 6e de sus cursos en línea en la plataforma Coursera [138] y actualmente es ampliamente utilizado para afrontar el problema de *diminishing learning rate*. Al igual que Adagrad en RMSProp se utiliza la suma de gradientes al cuadrado, pero en lugar de dejar que esa suma crezca continuamente durante el periodo de entrenamiento, se usa el promedio recursivo de los gradientes.

$$\begin{aligned} g_0 &= 0, \quad \rho \approx 0.9 \\ g_t &= \rho \cdot g_{t-1} + (1 - \rho) \cdot \nabla J(We)^2 \\ We_i(t+1) &= We_i(t) - \frac{\eta}{\sqrt{g_t} + \varepsilon} \cdot \nabla J(We_i) \end{aligned} \tag{41}$$

Siendo ρ el factor de la tasa de decaimiento, RMSprop divide la tasa de aprendizaje η por el promedio exponencial descendente de los gradientes cuadrados g . Hinton sugiere $\rho \approx 0.9$, mientras que un buen valor por defecto para la tasa de aprendizaje es $\eta = 0.001$.

Gradiente descendente con optimizador Adam

Adam (*Adaptive moment estimation*) es un optimizador que combina las ideas del RMSprop y del SGD con momento, conserva la tasa de aprendizaje por medio de la suma de los gradientes cuadrados g del RMSprop y se le incorpora un parámetro adaptivo de momento γ :

$$\begin{aligned} \gamma(t) &= \beta_1 \cdot \gamma(t-1) + (1 - \beta_1) \cdot \nabla J(We(t)) \\ g(t) &= \beta_2 \cdot g(t-1) + (1 - \beta_2) \cdot \nabla J(We(t))^2 \\ We_i(t+1) &= We_i(t) - \frac{\eta}{\sqrt{g_t} + \varepsilon} \cdot \gamma(t) \end{aligned} \tag{42}$$

Siendo $\beta_1 = 0.9$ la tasa de decaimiento para el parámetro adaptivo de momento m y $\beta_2 = 0.999$ el parámetro de decaimiento para la suma de los gradientes cuadrados g .

El optimizador Adam es considerado el mejor algoritmo entre todos los optimizadores discutidos anteriormente. Sin embargo, es importante hacer notar que no existe un optimizador que sea ideal o resuelva todas las tareas o problemas. Cada tarea posee su propia y desconocida superficie del error de n -dimensiones, por lo que, no hay garantía de que un optimizador sea mejor

o peor para localizar rápidamente el mínimo global. Se recomienda, a manera general, en problemas complejos, comenzar el entrenamiento usando el optimizador Adam y si no se observan buenos resultados cambiar por algún otro.

1.4 Percepción de movimiento mediante la visión biológica

De acuerdo con [139] la diversidad de patrones de movimiento que existen en nuestro entorno puede ser dividido en tres tipos de elementos de movimiento o patrones; traslación, expansión/contracción y rotación (Figura 1.20). Estos tres patrones de movimiento juegan un rol importante en la síntesis del movimiento. Por ejemplo, si se observa una llanta moviéndose en dirección horizontal, tendrá patrones de movimiento de traslación y rotación. Aunque como humanos conocemos estos patrones de movimiento e incluso los utilizamos a diario, no estamos generalmente conscientes de que existe un proceso interno que involucra la acción en conjunto de miles de neuronas que no solo detectan los objetos en movimiento, sino que también analizan la velocidad, dirección y profundidad de los patrones. En esta sección se describen los procesos neurofisiológicos que ocurren dentro del cerebro, que dan lugar a la percepción de movimiento, además se describen modelos computacionales de detección de energía del movimiento y funciones que modelan los campos receptivos de las células simples y complejas.

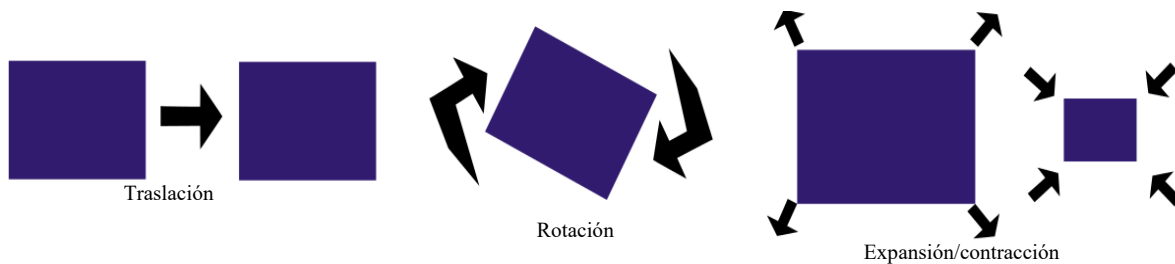


Figura 1.20. Patrones de movimiento.

1.4.1. Aspectos neurofisiológicos de la percepción de movimiento

Unos de los pioneros en realizar estudios en sistemas biológicos fueron Hubel y Wiesel [140], ellos examinaron los campos receptivos de las células en la corteza visual de gatos, observando sus respuestas a estímulos de luz orientados en específicas direcciones y con diferentes patrones de movimiento (Figura 1.21). Mediante estos experimentos, se descubrió que existían neuronas simples, complejas e hipercomplejas con características sensitivas a bordes, orientación, formas, tamaños y velocidad. Después, varios estudios neurofisiológicos en animales como: la mosca, el

escarabajo, la langosta, el pez cebra, el gato, el conejo y primates [141]–[147] contribuyeron a reafirmar ampliamente los resultados de Hubel y Wiesel.

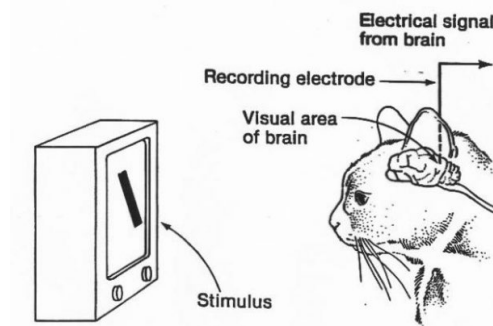


Figura 1.21. Experimentos de Hubel y Wiesel [140], 1959, para entender como el cerebro toma la información de los ojos y la transforma en la percepción visual compleja de nuestro mundo.

El proceso de percepción de movimiento comienza cuando incide la luz en el ojo a través de la córnea, pupila y cristalino hasta llegar a la retina. Luego, los núcleos geniculares lateral (NGL) reciben la información de la retina y envían su respuesta a la corteza visual que se compone de las áreas: primaria (V1), secundaria (V2), terciaria (V3), cuarta (V4) y media temporal (MT) o V5 [148].

La retina es un tejido situado sobre la superficie del ojo que tiene una estructura compleja, ahí la luz atraviesa diferentes capas celulares hasta alcanzar las células fotorreceptoras de conos y bastones. La retina humana contiene aproximadamente 6.5 millones de conos y 120 millones de bastones. Los conos son responsables de la medición del color (rojo, verde y azul), mientras que los bastones permiten una visión en baja condición de luminosidad, ya que son células extremadamente sensibles a la luz. Estos fotorreceptores se encargan de transformar el estímulo luminoso a una señal eléctrica que luego estimula a las células bipolares. Las células bipolares son neuronas que modulan los impulsos recibidos por los fotorreceptores. Cuando los fotorreceptores se encuentran en ausencia de luz, liberan un neurotransmisor llamado glutamato y las células bipolares reaccionan a este estímulo con dos respuestas diferentes, *ON-center* (hiperpolarización del glutamato) y *OFF-center* (despolarización del glutamato). De manera que, en la célula bipolar *ON-center* se tiene una respuesta inhibitoria, mientras que en la célula bipolar *OFF-center* una excitatoria. En caso contrario, cuando los fotorreceptores captan la luz, la liberación de glutamato decrementa, por lo que, las células bipolares *ON-center* ahora poseen una respuesta excitatoria y las *OFF-center* inhibitoria.

1.4.1.1. Detectores de contraste

Las células bipolares, entre sus funciones, conectan la información de los estímulos captados por los fotorreceptores a las células ganglionares, GN (Véase Figura 1.22a). Las células GN contienen una colección de células bipolares y horizontales que permiten dos tipos de estructuras de acuerdo con su campo receptivo *ON-center/OFF-surround* y *OFF-center/ON-surround*. Se denomina campo receptivo a la región del espacio en la cual la presencia de un estímulo altera la respuesta de una neurona. Las células ganglionares *ON-center/OFF-surround* como se muestra en la Figura 1.22b se disparan o tienen una mayor actividad cuando el estímulo de luz se encuentra en el centro y no en la periferia del campo receptivo; si se cambia el estímulo de luz a la periferia del campo receptivo, la neurona no muestra respuesta al estímulo.

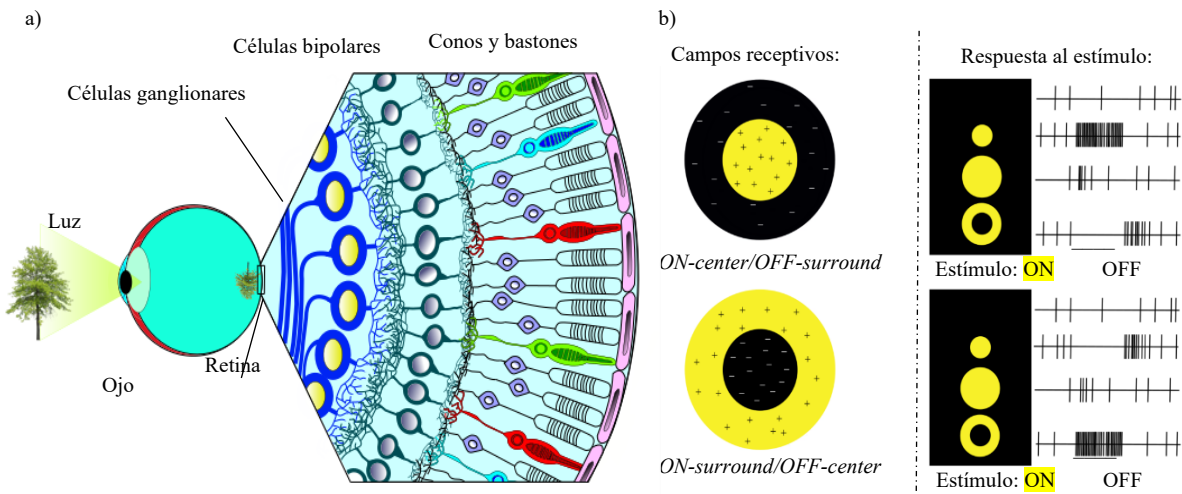


Figura 1.22. a) Representación esquemática de la estructura de la retina. La luz penetra por la izquierda y atraviesa diferentes capas celulares hasta alcanzar las células fotorreceptoras (conos y bastones). b) respuestas de las células ganglionares a un determinado estímulo de luz.

Por otro lado, la célula ganglionar *ON-surround/OFF-center* tiene un comportamiento contrario a la *ON-center/OFF-surround* (Figura 1.22b), arrojando la mayor respuesta cuando la luz esta aplicada en la periferia del campo receptivo y en su centro se tiene ausencia de luz. Las células ganglionares están relacionadas a detectores de contraste [149] debido a que son sensibles a las diferencias de iluminación o color entre dos zonas contiguas en el interior de sus campos receptivos, existiendo los siguientes tipos: luminancia, *ON-center/OFF-surround* y *OFF-center/ON-surround*; cromaticidad verde-rojo, *Green-center/Red-surround* y *Red-center/Green-surround*; cromaticidad azul-amarillo, *Blue-center/Yellow-surround* y *Yellow-center/Blue-surround*.

1.4.1.2. Orientación

La salida de las células ganglionares se transmite a través de sus axones a los NGL en el tálamo que se dividen en tres tipos, *parvocelulares* (P), *magnocelulares* (M) y *koniocelulares* (K). Las células P están más enfocadas a la precepción del color y la forma. Las células M se encargan de la precepción del movimiento, profundidad y pequeñas diferencias en brillo. Por otro lado, el rol de las células K aun no es completamente claro. Sin embargo, hay estudios donde las relacionan a la percepción del color, espacial y temporal [150][151][152].

El procesamiento de los NGL llega hacia el área V1 de la corteza visual que posee neuronas simples (SN). Estas neuronas responden selectivamente a patrones de luz en forma de líneas o bordes en una orientación particular. Los campos receptivos de las células simples involucran un número considerable de células ganglionares y son alargados respecto al eje preferente. En la Figura 1.23 se muestran los campos receptivos de tres neuronas simples con distintas preferencias a la orientación. La neurona a) tiene preferencia a la orientación horizontal o 0° , la neurona b) diagonal a 45° y la neurona c) a la orientación vertical o 90° . La neurona simple produce mayor activación cuando el estímulo o patrón de luz coincide con su campo receptivo, para ilustrar esto, en la Figura 1.24 se observa la activación que produce una neurona con preferencia a la orientación horizontal (0°) al ser excitada con un patrón de luz en forma de barra en diferentes orientaciones, la mayor actividad de la neurona se obtiene en su orientación preferente [153].

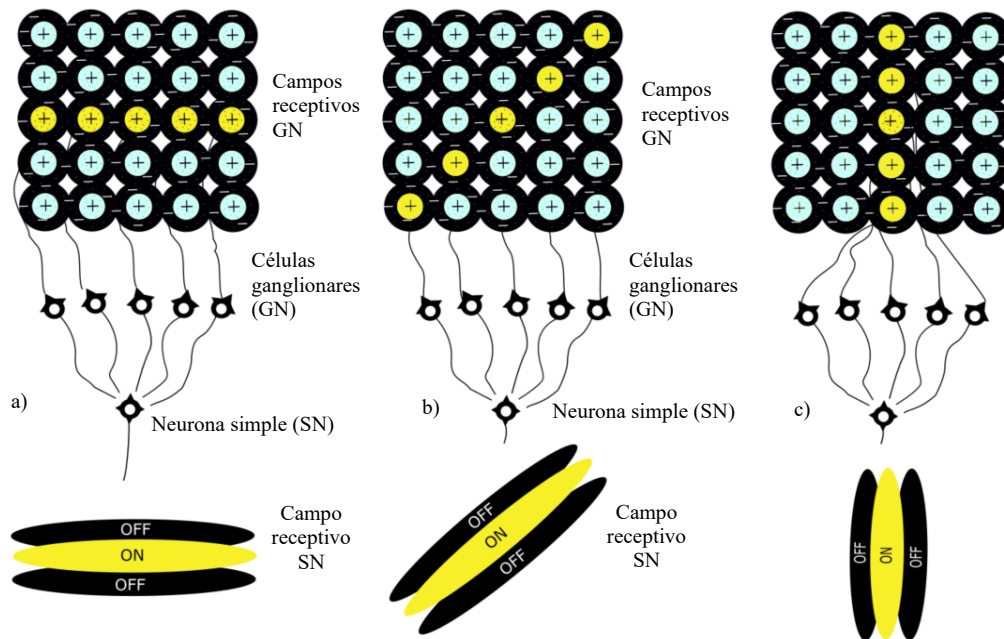


Figura 1.23. Neuronas simples preferentes a distintas orientaciones: a) 0° , b) 45° y c) 90° .

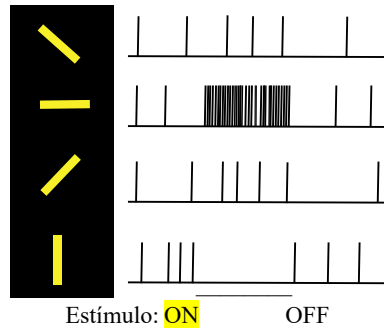


Figura 1.24. Disparos de una neurona simple (con preferencia a la orientación horizontal, 0°) al ser excitada con un patrón de luz en diferentes orientaciones.

1.4.1.3. Movimiento

Las neuronas en el área V1 no solo están involucradas en análisis estático de la escena. Existen también neuronas llamadas “complejas” que son selectivas a la orientación y dirección del movimiento [154]. Estas neuronas responden solo si el patrón de estímulo se presenta en la orientación y dirección preferente. En la Figura 1.25 se observa una célula compleja CN conformada de cinco células simples que responden a un estímulo de luz, en una misma orientación, aplicado en diferentes tiempos. Se puede analizar de la Figura 1.25 que las neuronas simples poseen axones de diferentes longitudes que convergen en la neurona compleja. Si el estímulo de luz se mueve con dirección hacia la derecha las activaciones de las neuronas simples llegan a las dendritas de la neurona compleja en diferentes tiempos, por lo tanto, la neurona CN no se dispara. Por otro lado, si el estímulo de luz se mueve con dirección hacia la izquierda las activaciones de las neuronas simples llegan a las entradas de la neurona compleja de manera síncrona, lo que provoca el disparo de CN. Las activaciones SN llegan al mismo tiempo porque las neuronas SN que se activan primero son las que poseen axones más largos, es decir que poseen un mayor retraso, de esta manera la neurona CN para este ejemplo se dice que es sensible al movimiento hacia la izquierda.

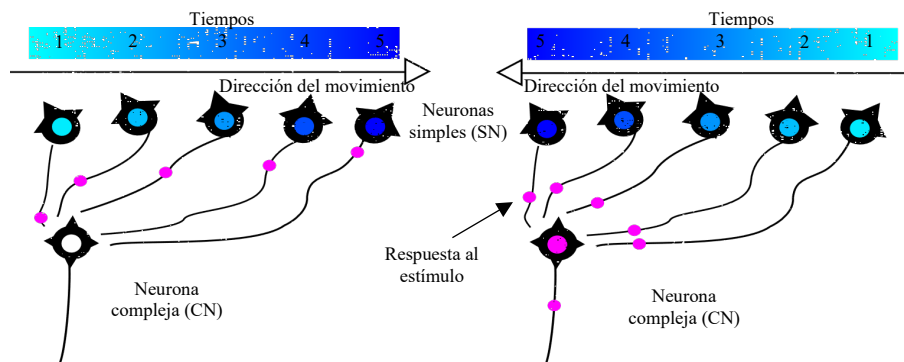


Figura 1.25. Neurona compleja sensible al movimiento hacia la izquierda.

Las neuronas CN selectivas al movimiento se concentran en las zonas VI y V2 de la corteza visual y se proyectan hacia el área MT especializada en el procesamiento del movimiento [155]. Mientras que las neuronas CN responden solo a la dirección preferente de movimiento, en el área MT existen neuronas que realizan análisis más complejos. Las neuronas del área MT, identifican patrones de movimiento en común de varias neuronas CN e inhiben patrones de movimiento no relevantes u oponentes. Una de las principales funciones de las neuronas MT es afrontar el problema de apertura de área. El problema de la apertura se refiere a la ambigüedad inherente del hecho de detectar el movimiento de una estructura espacial (como una barra o un borde) vista a través de una pequeña abertura de modo que los extremos del estímulo no son visibles. Las neuronas del área MT integran las activaciones de distintas neuronas CN especializadas a diferentes tipos de movimiento y codifican sus componentes ortogonales de movimiento para obtener un patrón global de la dirección del movimiento. La Figura 1.26 ilustra el procesamiento realizado por el área MT en un problema de apertura de área. Se puede observar que el estímulo que se mueve horizontalmente sólo es visible parcialmente a través de la ventana circular o apertura. Se observa además, que existirán un grupo de neuronas que se activan de acuerdo con el estímulo, representadas por sus posibles vectores de velocidad mediante múltiples flechas que se extienden desde el centro. Debido a que las neuronas en el área MT integran los componentes individuales de cada una de las neuronas activas, sin ningún tipo de información extra más que lo que se observa a través de la ventana, el movimiento en a) se predice hacia arriba a la derecha y el movimiento en b) hacia abajo a la derecha. c) Si ambos estímulos ocurren al mismo tiempo, se inhiben respuestas oponentes de velocidad dando como resultado una predicción horizontal hacia la derecha.

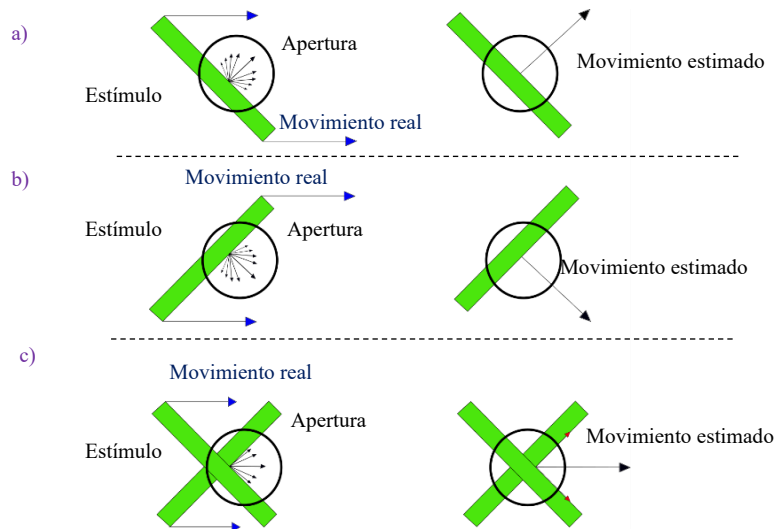


Figura 1.26. Integración de respuestas individuales de neuronas CN en el área MT para la estimación del movimiento.

En las siguientes secciones, se presentan algunos de los modelos de extracción de energía de movimiento basados en filtros. Aunque existen otros modelos de extracción de movimiento en el estado del arte, se decidió optar por describir estos modelos basados en filtros ya que este trabajo se enfoca en redes CNN, las cuales utilizan filtros dentro de su procesamiento. Por lo que, se pensó que tal cómo se observa en la sección 5.4, algunos de los filtros de la CNN se ajustarían durante el entrenamiento para tener un comportamiento semejante a estos modelos de extracción de energía basados en filtros.

1.4.2. Modelo de una neurona simple mediante filtros de Gabor

La transformada de Fourier es utilizada como una herramienta para analizar las propiedades de frecuencia de una determinada señal, sin embargo, usualmente la información acerca del tiempo se pierde y es complicado decir con certeza donde ocurrió dicha frecuencia. Por ejemplo, las funciones sinusoidales están completamente localizadas en la frecuencia, pero totalmente deslocalizadas en el tiempo o una función impulso está completamente localizada en el tiempo, pero totalmente deslocalizada en la frecuencia ya que abarca todo el espectro. Siempre existe incertidumbre entre la resolución en el tiempo y en la frecuencia. Lo anterior llevó a Dennis Gabor a introducir las funciones de Gabor [156]. Estas funciones consisten básicamente en una función compleja que involucra una Gaussiana modulada por una señal Sinusoidal:

$$Gabor_{1D}(t) = \exp(-\vartheta^2 t^2) \exp(j2\pi f_o t) \quad (43)$$

Donde ϑ determina el ancho de la Gaussiana y f_o es la frecuencia modulada.

Las funciones de Gabor utilizan una función gaussiana ya que la transformada de Fourier de una función gaussiana continúa siendo una función gaussiana, lo que ayuda a su localización en el dominio de la frecuencia. Además, si se aplica un análisis mediante la transformada de Gabor, este proporcionaría toda la información de la señal en su conjunto, información localizada tanto en el ámbito temporal como en el de la frecuencia [157].

Daugman en [158] extendió los Filtros de Gabor al dominio espacial e hipotetizó que los filtros de Gabor 2D eran excelentes para modelar y describir los campos receptivos de las células simples. Luego, Jones y Palmer en [159] ampliaron el estudio de Daugman corroborando su hipótesis. En la Figura 1.27 se muestra un ejemplo tomado de [158] donde el campo receptivo de una neurona

simple es modelado por una función de Gabor, se puede observar al comparar ambas gráficas que la diferencia es prácticamente nula. Los filtros de Gabor 2D se definen por:

Parte real:

$$Gabor_{2D}(x, y) = \exp \left[- \left(\left(\frac{x_r^2 + y_r^2}{2\sigma^2} \right) \right) \right] \cos(2\pi \cdot f \cdot x_r + \psi) \quad (44)$$

$$x_r = x \cos \theta + y \sin \theta, \quad y_r = -x \sin \theta + y \cos \theta$$

Parte imaginaria:

$$Gabor_{2D}(x, y) = \exp \left[- \left(\left(\frac{x_r^2 + y_r^2}{2\sigma^2} \right) \right) \right] \sin(j2\pi \cdot f \cdot x_r + \psi) \quad (45)$$

$$x_r = x \cos \theta + y \sin \theta, \quad y_r = -x \sin \theta + y \cos \theta$$

Donde f es la frecuencia de modulación de la onda senoidal, θ es la orientación del eje mayor de la Gaussiana elíptica (en grados), Ψ es un offset de fase (en grados) y σ es la desviación estándar de la Gaussiana.

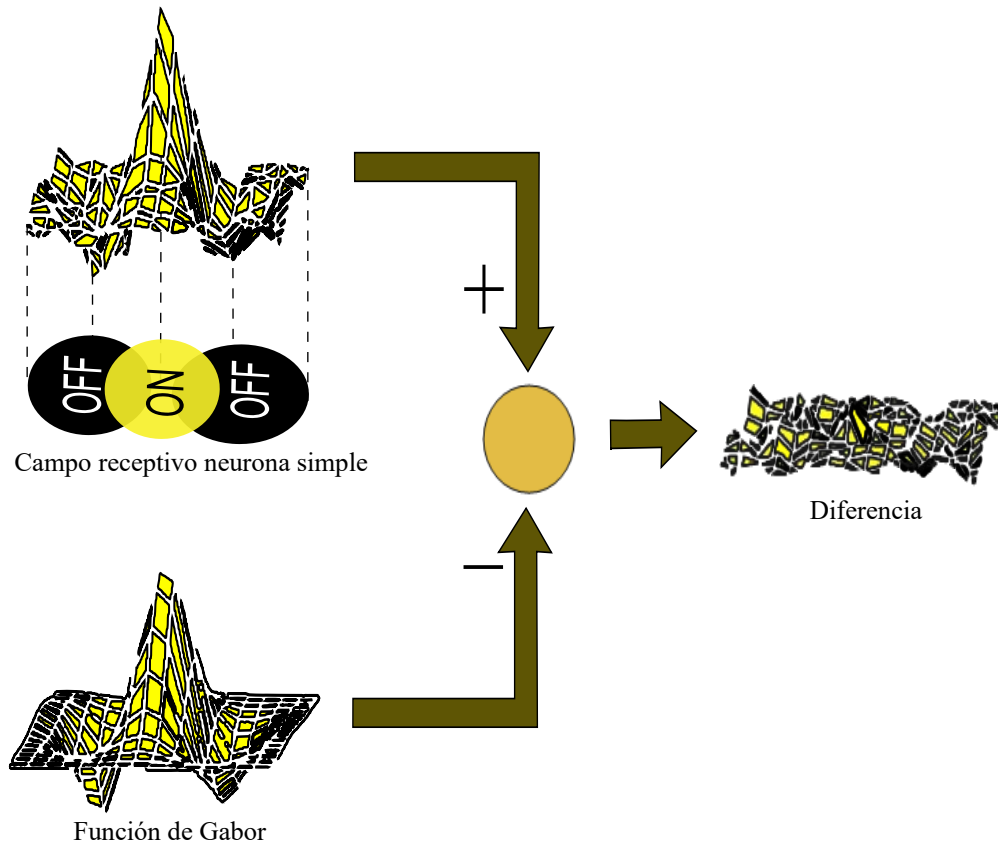


Figura 1.27. Funciones de Gabor 2D modelando los campos receptivos de las neuronas simples [158].

Los filtros de Gabor 2D son muy utilizados en análisis de imágenes para extraer características, debido a que, al igual que las neuronas simples, pueden ser vistos como detectores de bordes sensibles a una frecuencia y orientación específica [160]. En la imagen de la Figura 1.28 se muestra un ejemplo donde se extraen características de diferentes frecuencias y orientaciones de la imagen a) al realizar el proceso de convolución con un banco de filtros de Gabor.

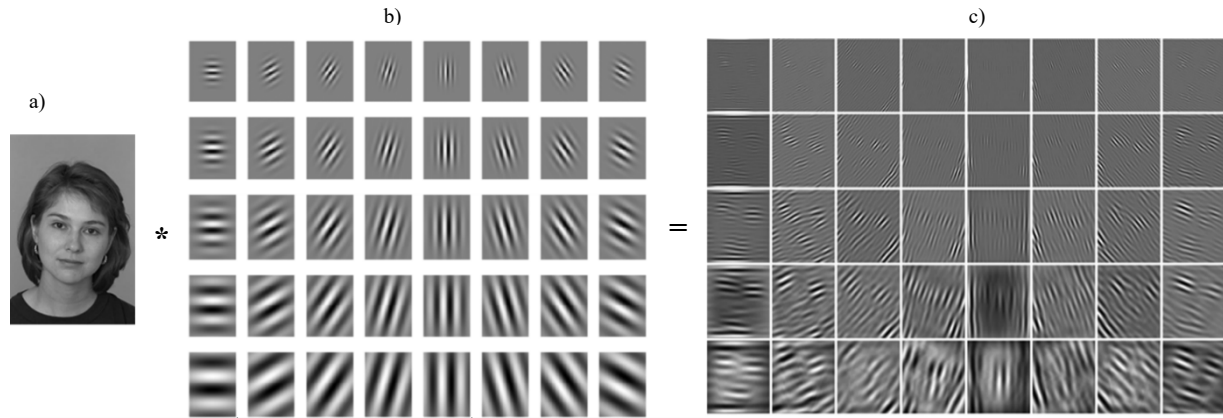


Figura 1.28. Extracción de características mediante filtros de Gabor. a) Imagen de entrada, b) Banco de filtros de Gabor 2D, c) Características extraídas. Imagen modificada de [160].

1.4.3. Modelo de una neurona compleja mediante filtros de Gabor 3D

Adelson y Bergen en [161] establecieron que una secuencia de movimiento puede representarse como un único patrón en el dominio espaciotemporal, que corresponde a una orientación tridimensional igual a la velocidad del movimiento. Por ejemplo, si consideramos un estímulo o barra vertical que se mueve continuamente hacia la derecha en el espaciotiempo estará muestreada en tiempos discretos debido a la tasa de adquisición de cuadros de video, por lo tanto, la barra vertical en movimiento dejará un patrón en forma de sección inclinada en el plano XT, esto se visualiza en la Figura 1.29. Se puede observar que la barra se mueve del punto inicial a) al punto final b) y al trasladarse en el tiempo dibuja una sección con una cierta inclinación (Figura 1.29c y d). Las imágenes e), f), g) y h) ejemplifican el concepto anterior, cambiando el movimiento en el eje horizontal hacia el eje vertical. En ambos casos, el movimiento resulta en una sección inclinada u orientada en los planos; XT o YT, esta inclinación corresponde a la velocidad del desplazamiento de la barra o estímulo [161].

De acuerdo con lo anterior una velocidad de movimiento corresponde a una orientación tridimensional, por lo tanto, la información de movimiento de un escenario puede ser extraída por un sistema que responda a la energía orientada espaciotemporalmente. Previamente en la sección

1.5.2 se ilustró como los filtros de Gabor 2D son una buena opción para detectar bordes en imágenes ya que responden a una orientación y frecuencia preferente. Si ahora consideramos el escenario espaciotemporal donde el movimiento es igual a la orientación se podría utilizar un filtro de Gabor con un campo receptivo orientado espaciotemporalmente para detectar dicho movimiento. Los filtros espaciotemporales de Gabor o filtros de Gabor 3D [162] emulan el comportamiento de las neuronas complejas modelando tanto la preferencia en orientación espacial como la dirección y velocidad del movimiento.

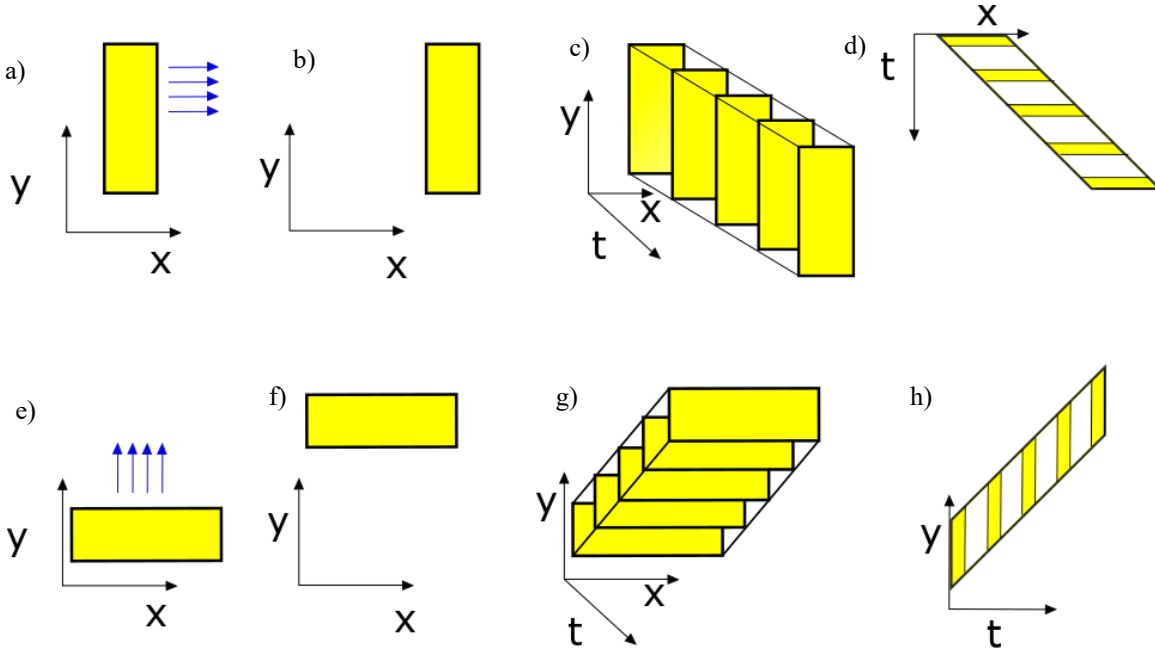


Figura 1.29. Secuencia de movimiento representada en un patrón espacio temporal que corresponde a una velocidad u orientación específica. a) barra vertical en punto inicial antes de moverse, b) barra vertical en su punto final después del movimiento, c) muestreo del movimiento de la barra vertical a través del tiempo, d) patrón espacio temporal generado cuya orientación corresponde a la velocidad de desplazamiento horizontal de la barra vertical. Las imágenes e), f), g) y h) muestran una barra horizontal desplazándose hacia arriba, ejemplificando el caso en desplazamientos verticales.

Los filtros de Gabor 3D se definen mediante la siguiente expresión [162]:

$$Gabor_{3D}(x, y, t) = \frac{a}{2\pi\sigma^2} \exp\left(-\frac{(x_r + v_c t)^2 + (ay_r)^2}{2\sigma^2}\right) \cdot \cos\left(\frac{2\pi}{\lambda_g}(x_r + vt) + \psi\right) \frac{1}{\sqrt{2\pi T}} \exp\left(-\frac{(t - \mu_t)^2}{2T^2}\right)$$

$$x_r = x \cos \theta + y \sin \theta, \quad y_r = -x \sin \theta + y \cos \theta$$

(46)

Siendo σ la desviación estándar del factor Gaussiano, a es la tasa que especifica la elipticidad de la envolvente Gaussiana en el dominio espacial, v es la velocidad de movimiento o fase del factor cosenoidal. La velocidad con la que se mueve el centro de la Gaussiana a lo largo del eje x_r .

es especificada por el parámetro v_c , si $v_c=0$ el centro de la envolvente Gaussiana es estacionario, por otra parte, si $v_c=v$ se obtiene una envolvente en movimiento. El parámetro λ_g es la longitud de onda de la función cosenoidal, obtenido a través de la relación $\lambda_g = \lambda_0 \sqrt{1 + v^2}$, siendo $\lambda_0=2$. El ángulo $\theta \in [0, 2\Pi]$ determina la dirección del movimiento y la orientación espacial del filtro. El parámetro $\Psi \in [-\Pi, \Pi]$ es un desfase que determina la simetría en dominio espacial. La función Gaussiana, con media μ_t y desviación estándar T , es utilizada para modelar los cambios en intensidad de color, los parámetros $\mu_t=1.5$ y $T=2.75$ son constantes basadas en la duración promedio de la mayoría de los campos receptivos de acuerdo con [162].

La Figura 1.30, en la parte a) muestra la representación de un filtro de Gabor 3D o espaciotemporal con sus campos receptivos “*off-surround*” en negro y “*on-center*” en azul. En b) se observa el movimiento de un estímulo a diferentes velocidades v , a mayores velocidades, mayor inclinación tendrá la orientación tridimensional del patrón generado a través del tiempo. En c) se ejemplifica el uso del filtro de Gabor 3D con una orientación preferente a detectar velocidades hacia la derecha de 4 pixeles por cuadro, tanto en el tiempo continuo como en el discreto.

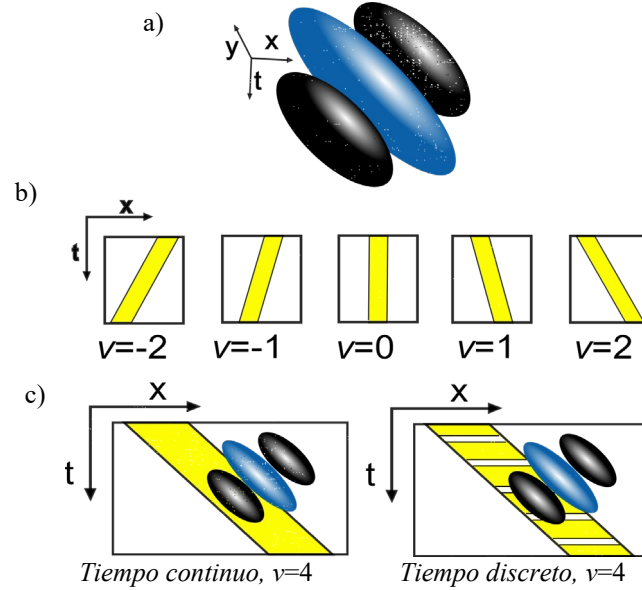


Figura 1.30. Filtro de Gabor espaciotemporal aplicado en detección de movimiento.

La respuesta espaciotemporal $re_{v,\theta,\Psi}(x,y,t)$ de un filtro de Gabor 3D con una secuencia de imágenes se calcula mediante la operación de convolución:

$$re_{v,\theta,\Psi}(x,y,t) = I(x,y,t) * Gabor_{3D}(v,\theta,\Psi) \quad (47)$$

Los filtros espaciotemporales son sensibles a la fase, por lo que su respuesta a un patrón en movimiento depende de cómo este se alinea con su campo receptivo, además la señal de respuesta depende del contraste del estímulo (ej. Un estímulo o borde que cambia de blanco a negro da una respuesta opuesta a uno que cambia de negro a blanco). Por lo tanto, para obtener una respuesta insensible a la fase se debe obtener la energía de movimiento mediante:

$$R_{v,\theta} = \sqrt{re^2_{v,\theta,0}(x,y,t) + re^2_{v,\theta,\Pi/2}(x,y,t)} \quad (48)$$

La ecuación anterior implica la sumatoria de la respuesta de dos funciones de Gabor en cuadratura con una diferencia de fase $\Psi=\Pi/2$. Esta convención matemática, de acuerdo con [161], saca provecho del hecho que el $\sin^2 + \cos^2=1$ y es la base para establecer modelos de energía de movimiento. Las dos funciones de Gabor con funciones seno y coseno ponderadas por la misma ventana gaussiana, permiten extraer energía dentro de una banda de frecuencia espaciotemporal. La respuesta R resultante es siempre positiva y consistente para bordes que cambian de color blanco a negro, que de negro a blanco. Por lo tanto, la respuesta R es independiente al contraste y fase (Figura 1.31).

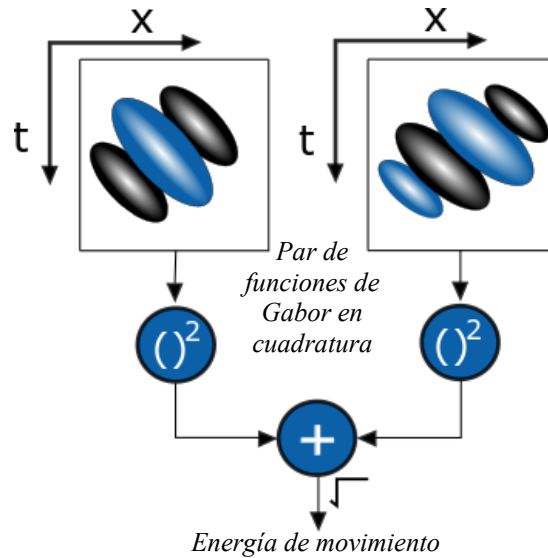


Figura 1.31. Modelo espaciotemporal para la extracción de la energía del movimiento.

CAPÍTULO II. ALGORITMOS DE APRENDIZAJE PROFUNDO ENFOCADOS A DETECCIÓN DE MOVIMIENTO

Últimamente los algoritmos con técnicas de *Deep learning* han mostrado un excelente desempeño en tareas como la clasificación de imágenes, identificación de objetos, segmentación semántica, identificación de escenarios entre otras. Por lo que actualmente son un tema de interés científico que ha llevado a varios autores a hacer contribuciones cada vez más significativas. Una de las contribuciones en *Deep learning* es en el área de detección de movimiento, donde se han publicado varios esfuerzos de algoritmos que intentan afrontar cada una de las circunstancias críticas de los videos. Por lo que, en este capítulo se presenta una recopilación de los trabajos más representativos encontrados en la literatura, algunos de ellos populares por sus propuestas originales o por sus desempeños en bases de datos públicas.

2.1. Red autoorganizadas multicapa en modelado de fondo (SMSOM-BM).

En [84] se presenta el algoritmo SMSOM-BM (*Stacked Multilayer Self-Organizing Map Background Model*). SMSOM-BM consiste en una capa de entrada y varias capas intermedias para cada pixel de la imagen, donde cada capa se utiliza para modelar una parte de las características del fondo. A manera general SMSOM-BM es entrenado usando algunas imágenes de fondo y luego con base en el modelo preentrenado utiliza un modelo recursivo que contempla un módulo de mantenimiento de fondo para ajustar sus parámetros. El esquema general del algoritmo se observa en la Figura 2.1.

El proceso de entrenamiento de SMSOM-BM contiene dos fases principales, un proceso competitivo y otro cooperativo. En el competitivo cada pixel de la imagen $I(t)$ en espacio de color HSV es asociado a 9 nodos en cada capa l de red SMSOM. Luego para cada nodo q se calcula una distancia d_{iq} respecto al pixel i . El nodo con el mínimo valor d_{iq} es llamado ganador y es usado en el proceso subsecuente de cooperación. El nodo ganador muestra que el pixel i actual puede ser considerado como una buena aproximación del fondo. Por lo tanto, debe ser usado para actualizar los pesos del vecindario de nodos ω_{piq}^l mediante:

$$\omega_{piq}^l(t+1) = (1 - \alpha_{train}) \cdot (\rho_q) \cdot \omega_{piq}^l(t) + \alpha_{train} \cdot (\rho_q) \cdot pv_i^l(t) \quad (49)$$

Donde $pv_i^l \in \{h_i^l(t), s_i^l(t), v_i^l(t)\}$, es el valor del pixel i por cada uno de los canales HSV en la capa l . α_{train} es un parámetro de aprendizaje que se requiere ajustar manualmente para regular la velocidad de actualización. $\rho_q = \exp\left(\frac{\|q - q^*\|^2}{2\sigma^2}\right)$ es el valor obtenido al usar un kernel gaussiano aplicado en la distancia del nodo q respecto al nodo ganador q^* , cuando q es el nodo ganador la distancia $\|q - q^*\|$ es 0. Por lo tanto, ese nodo recibe un peso mayor respecto a sus vecinos. σ^2 es la varianza del kernel gaussiano.

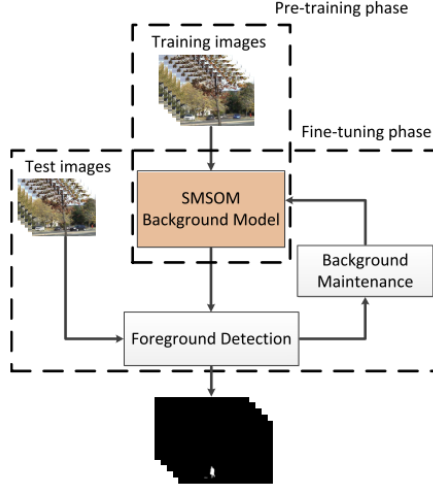


Figura 2.1. Detección de movimiento por SMSOM-BM [84].

Para la etapa de detección de movimiento se obtiene la distancia ganadora d_{iq}^l del modelo SMSOM entrenado en la capa l , si la distancia es menor a un umbral τ , $d_{iq}^l \leq \tau_i^l$, entonces el pixel i asociado al nodo ganador es considerado como fondo y el proceso termina. Por otro lado, si $d_{iq}^l > \tau_i^l$, entonces el pixel i es puesto en la segunda capa y se repite el procedimiento previo para determinar si pertenece al fondo. El proceso es recursivo hasta el alcanzar el número de capas l del modelo. Si ninguna coincidencia al fondo es encontrada entonces el pixel i es clasificado como parte de un objeto dinámico. El modelo SMSOM-BM fue evaluado en la base de datos *CDnet14* en las categorías de video *Baseline*, *DynamicBackground*, *CameraJitter* y *Thermal*, logrando resolver situaciones de movimiento básico de la categoría *Baseline* con buenos desempeños, pero no fue el caso para las demás categorías donde solo unos pocos videos presentaban resultados por encima de 0.7 en la métrica *F-Measure*. Además, otra de las desventajas de SMSOM-BM es que necesita imágenes sin objetos dinámicos para aprender el parámetro τ . Por lo tanto, en algunos videos que no contienen imágenes de entrenamiento sin objetos dinámicos, el modelo no provee una buena estimación inicial y el desempeño decrece.

2.2. Modelo neuronal de respuesta mixta (NeRM)

En [85] se propone un método de detección de movimiento llamado *Neural Response Mixture* (NeRM) el cual, aprovecha las capacidades de las redes neuronales profundas (DNN) estocásticas, manteniendo un bajo costo computacional para su aplicación en sistemas embebidos. En NeRM se extraen características para construir un modelo de mezcla de Gaussianas (GMM) de respuesta neuronal mixta para capturar rasgos únicos de los objetos (personas, vehículos, animales, etc.) en entornos de vigilancia reales.

Para la formación de la red primero se utiliza una red convolucional entrenada en la base de datos para clasificación de imágenes *ImageNet*. Luego, una red llamada *StochasticNet* es formada a través de la selección estocástica de las conexiones sinápticas de la red convolucional previamente entrenada. La selección de las conexiones sinápticas se realiza mediante una función de energía, para obtener las características profundas más representativas. Luego, esas características son pasadas a un modelo de mezclas gaussianas para modelar el fondo o escenario del video y obtener los objetos dinámicos. La Figura 2.2 ilustra el esquema general del NeRM

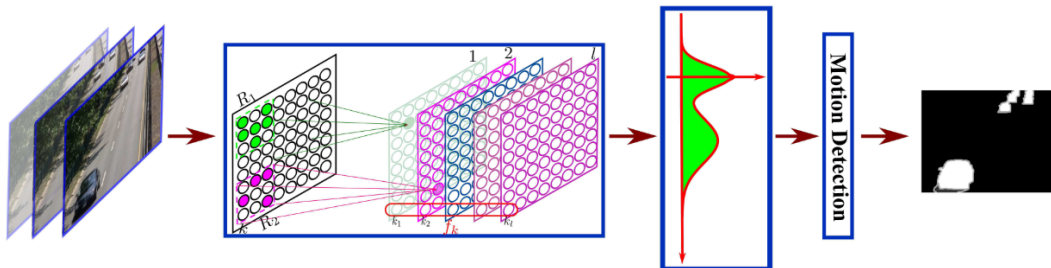


Figura 2.2. Esquema general del NeRM [85].

El algoritmo NeRM fue implementado en la tarjeta embebida Axis Q7436 alcanzando una velocidad de 2 FPS a una resolución de 352x240. Los resultados experimentales mostraron que NeRM logra mejores resultados que el método de mezclas gaussianas en RGB ya que aprovecha las características aprendidas por un enfoque de red convolucional. Sin embargo, el método requiere un extenso conjunto de imágenes de entrenamiento y existe un efecto de ruido en forma de bloques en la detección final del movimiento.

2.3. Sustracción de fondo mediante red neuronal convolucional (ConvNet)

M. Braham y M. Van Droogenbroeck en [86] proponen un método para modelar el fondo a través de un algoritmo basado en características espaciales aprendidas con redes neuronales convolucionales. Este método se compone por cuatro etapas principales: extracción de fondo de la imagen, generación del conjunto de datos, entrenamiento de la red y detección de movimiento. El entrenamiento de la red se logra mediante una arquitectura muy similar a la red LeNet-5 [163] usada para la clasificación de dígitos. El esquema general de ConvNet se muestra en la Figura 2.3, en la primera capa de convolución se utilizan 6 filtros de tamaño 5x5 y en la segunda 16 filtros del mismo tamaño. Por su parte, las capas de *pooling* utilizan filtros de tamaño 3x3. La capa totalmente conectada tiene 120 neuronas ocultas y el valor final de salida *FG probability* consiste en una neurona simple con activación sigmoideal. La imagen de referencia o *Background image* mostrada es obtenida a través del algoritmo de mediana temporal sobre 150 cuadros de video. La red es entrenada con parches de tamaño $T_a \times T_a$ como entrada, siendo $T_a = 27$ y con valores de “1” o “0” correspondientes al pixel central del parche como *target*.

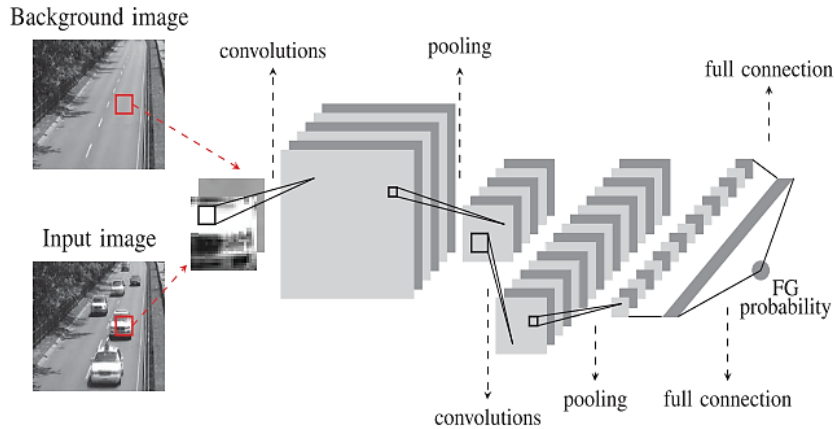


Figura 2.3. Esquema del proceso de clasificación de un pixel centrado en un parche de tamaño $T_a \times T_a$ [86].

La red contiene 20243 pesos que son aprendidos mediante *backpropagation* y la función de costo *cross-entropy*.

$$J = -\sum_{n_p} (y_{t,n_p} \ln(y_{p,n_p}) + (1 - y_{t,n_p}) \ln(1 - y_{p,n_p})) \quad (50)$$

Donde y_t es el valor del *target* para el pixel i en la posición n_p y $y_p = p(Fg|i)$ es la probabilidad de que un pixel i pertenezca a un objeto dinámico (*Fg*).

ConvNet es un algoritmo con un esquema basado en parches para obtener la detección de movimiento. Su evaluación fue realizada en la base datos *CDnet14* con 9 de las 11 categorías disponibles. Por lo que, las categorías *PTZ* e *IntermittentObjectMotion* que contienen movimiento de cámara y objetos dinámicos que permanecen estáticos un largo periodo de tiempo no son resueltas. Además, sus resultados dependen altamente de la imagen de referencia generada por el algoritmo de modelado de fondo y el uso de un gran número de parches y convoluciones por imagen es computacionalmente demandante.

2.4. Arquitectura profunda de apariencia y movimiento (A-Net/M-Net)

El enfoque propuesto en [87] considera características de apariencia y movimiento mediante el diseño de una arquitectura compuesta de dos redes convolucionales. La red de apariencia Apperance net (A-Net) usa la red VGG-16 preentrenada en la base de datos ImageNet en las primeras capas, tal como se observa en la Figura 2.4. La red VGG-16, fue entrenada para clasificación de imágenes, por lo que es ideal para detectar los objetos dinámicos de interés como personas, carros, animales, etc. Por otro lado, la red de movimiento (M-Net) detecta los objetos dinámicos usando como entrada; la imagen actual del video y una imagen de fondo generada por el algoritmo [164]. Luego, ambas redes son combinadas utilizando un método similar a [165], donde se aplica interpolación bilinear a la última capa de la red A-Net para que ambas salidas tengan el mismo tamaño. Una vez que las dos salidas tienen el mismo tamaño se unen empleando una funcion softmax. El método fue evaluado en 5 videos con movimiento de cámara mostrando desempeños regulares. Además, sus resultados dependen en gran parte del algoritmo generador de fondo. Sin embargo, el método tiene una velocidad de procesamiento de 50 FPS en una GPU GTX 1070, lo que lo hace adecuado para su aplicación en tiempo real con el equipo necesario.

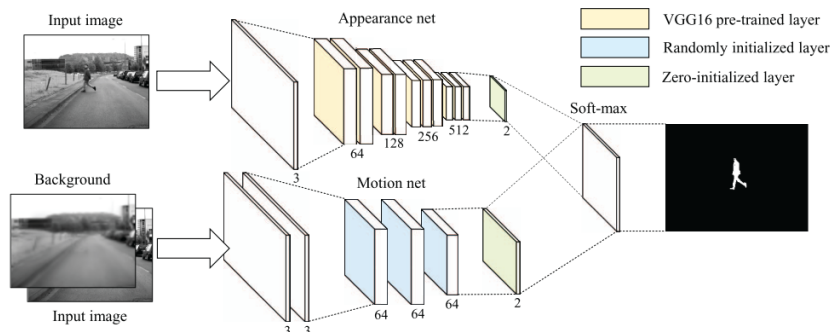


Figura 2.4. Estructura del modelo A-Net y M-Net [87].

2.5. Foreground Segmentation Network (FgSegNet)

El método FgSegNet (*Foreground Segmentation Network*) propuesto en [10] consiste en un enfoque basado en redes convolucionales que adapta una red VGG-16 preentrenada bajo un esquema compuesto de tres redes convolucionales *encoder-decoder* que son alimentadas con imágenes RGB a diferentes escalas.

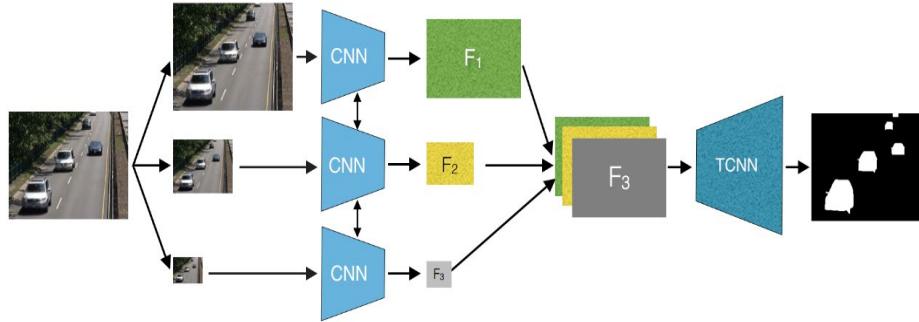


Figura 2.5. Estructura de FgSegNet [10].

Como se ilustra en la Figura 2.5, FgSegNet no considera información temporal del video ya que se alimenta únicamente del cuadro actual, este cuadro luego es convertido a tres diferentes escalas. Asumiendo que la imagen de entrada es de tamaño $W \times H \times 3$, siendo W el ancho, H el alto y 3 la profundidad en RGB, los otros tamaños de imagen serán de $W/2 \times H/2 \times 3$ y $W/4 \times H/4 \times 3$. Este proceso de escalamiento es utilizado por los autores para que su algoritmo logre un entendimiento global del contexto de la escena de video, relacionado la región local donde se aplica el filtro de convolución con sus alrededores. Por su parte, los bloques azules con la leyenda CNN en el diagrama de la Figura 2.5 representan la etapa del *Encoder* o codificador de cada una de las redes con entradas a diferentes escalas. La arquitectura del *Encoder* es la misma para las tres redes y se encarga de la extracción de características mediante la adaptación y uso de la arquitectura previamente entrenada VGG-16. Las salidas de las redes *Encoder* luego son aumentadas en resolución y concatenadas, esto se ilustra en la Figura 2.5 con los mapas de características F_1 , F_2 y F_3 . Estos mapas concatenados son pasados a la parte *Decoder* para que la red aprenda los pesos para decodificarlas, esta etapa considera 5 bloques de convoluciones transpuestas usando filtros de tamaño 1×1 . Estos bloques de convoluciones transpuestas se encargan de aumentar los mapas de características (en ancho y alto) generados de la etapa *Encoder* para generar una salida del tamaño de la resolución original de la imagen.

FgSegNet demostró su robustez y actualmente se encuentra en la primera posición de la base de datos *Change Detection 2014*. Sin embargo, su tiempo de procesamiento es elevado, siendo 18 FPS en imágenes de resolución 320x240 usando GPU.

2.6. Red convolucional en cascada (CascadeCNN)

El algoritmo CascadeCNN [8] es el predecesor de [10] y usa tres redes convolucionales en cascada alimentadas con imágenes a diferentes escalas, teniendo la primera imagen a escala original y las otras dos a escalas de 0.75 y 0.5 respectivamente. Cascade CNN trabaja con parches o regiones de la imagen de tamaño 31x31 aplicados en imágenes a tres diferentes tamaños (Véase Figura 2.6). De esta manera los autores sugieren que se consideran tanto características del contexto como de detalle. El objetivo principal del método es proporcionar resultados lo suficientemente precisos para ser utilizados como Ground truth.

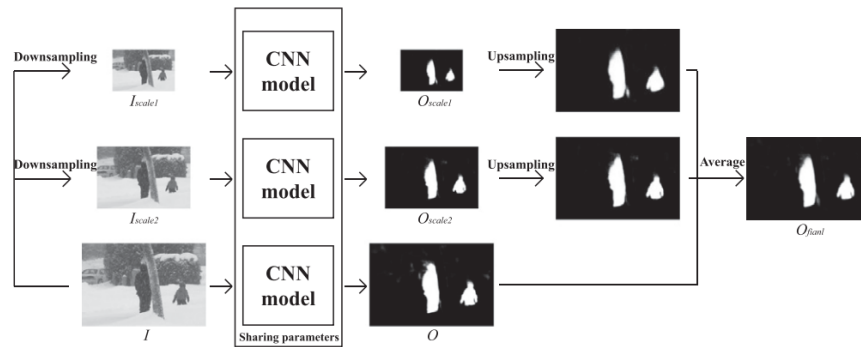


Figura 2.6. Estructura de CascadeCNN [8].

La arquitectura de la red CNN que se aplica para cada parche de la imagen se muestra en la Figura 2.7. Para el entrenamiento se utilizan de 50 a 200 cuadros manualmente etiquetados por video. El número de cuadros de entrenamiento, así como la estrategia de selección depende de la complejidad del vídeo. Los vídeos con iluminación fija que muestran un flujo constante de objetos en movimiento bien contrastado sólo requieren 50 cuadros de entrenamiento elegidos al azar. En el caso de los vídeos más complejos, como los vídeos nocturnos, que contienen objetos de bajo contraste, y los videos PTZ, en los que la cámara se mueve en todas las direcciones, suelen requerir la selección manual de 200 cuadros para alcanzar buenos resultados.

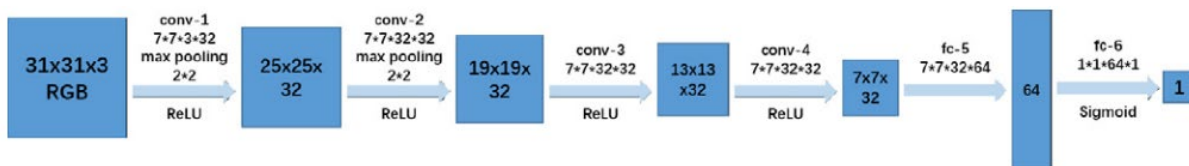


Figura 2.7. Arquitectura de red CNN [8].

El enfoque CascadeCNN no está exento de limitaciones ya que tiene dificultades para resolver situaciones con objetos dinámicos muy pequeños. No obstante, sus buenos resultados en la mayoría de los videos lo han mantenido dentro de los primeros diez lugares de la base de datos *CDnet14*.

2.7. Red convolucional para sustracción de fondo (DeepBS)

Otro trabajo que hace uso de una red convolucional es presentado en [88]. El enfoque es similar al de Braham y Droogenbroeck [86] se tiene una CNN alimentada con parches tomados de una imagen de fondo o referencia y el cuadro actual del video como se muestra en la Figura 2.8.

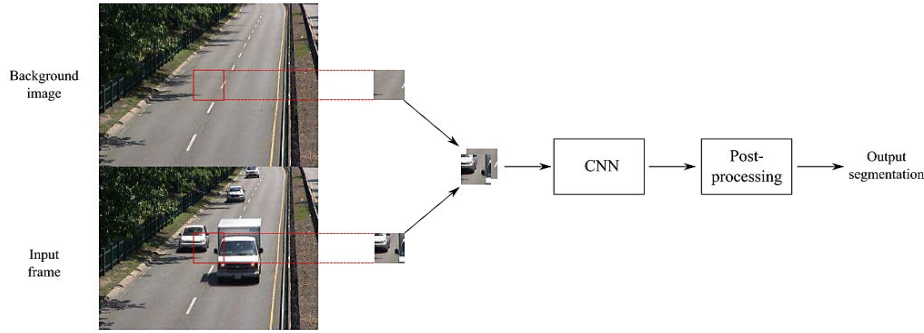


Figura 2.8. Sustracción de fondo con DeepBS [88].

La CNN es responsable de extraer las características de parches de tamaño 37×37 y realizar la segmentación con las características extraídas. La imagen de fondo es obtenida mediante los algoritmos SuBSENSE [58] y FTSG [79]. SuBSENSE utiliza características binarias espaciotemporales, así como en la información de color para detectar cambios en el escenario lo que permite que los objetos dinámicos camuflados se detecten fácilmente mientras que la mayoría de las variaciones de iluminación se ignoran. Además, los parámetros del método se ajustan dinámicamente sin la intervención del usuario. Por otra parte, el algoritmo FTSG es usado como un detector de movimiento a manera de tener una longitud de memoria auto adaptiva en el modelo de fondo. La idea es almacenar en un diccionario bl los últimos 90 valores de los píxeles clasificados por el algoritmo SuBSENSE como píxeles de fondo, cuando se completan $bl=90$ valores del diccionario, el valor del píxel de fondo actual reemplaza al más antiguo y los demás valores se recorren. El modelo del fondo bk se obtiene con el promedio de los valores de los píxeles en el diccionario de fondo bl sobre una cierta longitud de memoria bm :

$$bk(x, y, z) = \frac{\sum_{di=bi-bm}^{bi} bl(x, y, z, di)}{bm}, \quad z = \{R, G, B\} \quad (51)$$

Siendo (x, y, z) locaciones de los pixeles en ancho, alto y profundidad y di un iterador sobre los valores del diccionario bl . Por su parte, bm es una la longitud de memoria adaptiva que se obtiene mediante:

$$bm = \begin{cases} 5 & si \quad Fs \geq 0.25 \\ 5 + \frac{Fs - 0.02}{0.25 - 0.02} * 0.85 & si \quad 0.02 < Fs < 0.25 \\ 90 & si \quad Fs \leq 0.02 \end{cases} \quad (52)$$

Donde Fs es una variable que define que tanto porcentaje de pixeles en movimiento existen en cuadro actual del video de acuerdo con la segmentación binaria obtenida con el algoritmo FTSG.

$$Fs = \frac{Ns}{W \times H} \quad (53)$$

Siendo W y H el ancho y alto de la imagen y Ns es el número de pixeles de objetos dinámicos en la imagen. Grandes valores de Fs significa alto cantidad de movimiento en el escenario y se disminuye bm y si Fs es pequeña significa el fondo es estable y se puede usar una longitud de memoria bm más grande para suprimir el ruido.

En cuanto a la arquitectura de la red, contiene 3 capas convolucionales y 2 capas totalmente conectadas con una función de activación sigmoideal a la salida. Se entrena la red con 150 imágenes por video a través del algoritmo RMSprop y la función de pérdida crosentropía binaria. El método demostró buenos resultados para las categorías de video de vibraciones de cámara, fondos dinámicos, turbulencia y sombras. Sin embargo, en el caso de las categorías de video PTZ o la de baja velocidad de cuadros, la CNN arroja resultados deficientes, ya que las imágenes de fondo que proporciona el algoritmo propuesto son insuficientes para realizar una buena sustracción de fondo.

2.8. Modelado de fondo con redes convolucionales dilatadas (AtrousCNN)

El modelo presentado en [89] utiliza tres redes convolucionales *atrous* con diferente dilatación para extraer información espacial de diferentes vecindarios de pixeles. En términos simples, la convolución *atrous* o dilatada es una convolución con espacios definidos dentro del *kernel* o filtro. Por ejemplo, en una convolución con una imagen, la tasa de dilatación de $dl=1$ es la convolución común; $dl=2$ sería una convolución *atrous*, donde se salta un píxel en la entrada, $dl=3$ sería saltarse

2 píxeles y así sucesivamente. La Figura 2.9 muestra la convolución de un filtro 3x3 (color azul) con diferentes tasas de dilatación sobre una imagen.

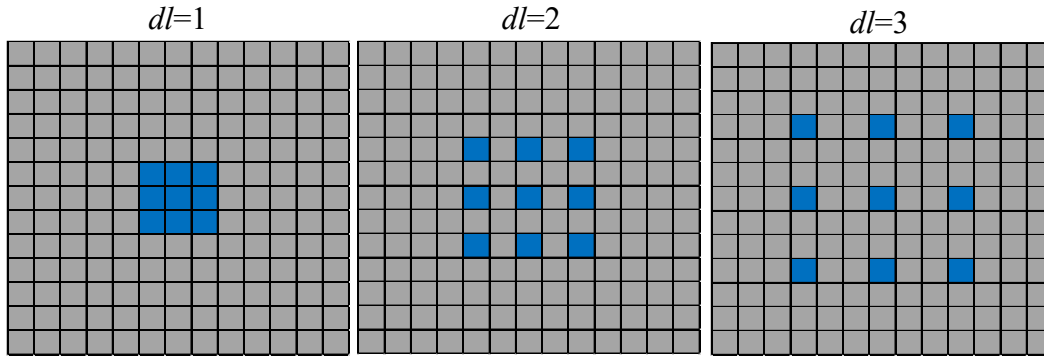


Figura 2.9. Convolución *atrous* con *kernel* de tamaño 3x3 y diferentes tasas de dilatación dl .

El objetivo de las convoluciones *atrous* es aumentar el tamaño del campo receptivo sin aumentar el tamaño de los parámetros del filtro y se utilizan principalmente en el campo de la segmentación semántica.

La entrada a la arquitectura de red mostrada en la Figura 2.10 es la unión de múltiples cuadros de video I en el tiempo t y de esta manera se captura suficiente información temporal del fondo y de los objetos dinámicos. La entrada se define por:

$$entrada = \{I_{t-11}, I_{t-8}, I_{t-5}, I_{t-3}, I_{t-1}, I_t\} \quad (54)$$

La arquitectura utilizada en el modelo de red es un tanto compleja, con las primeras capas tipo *atrous*, también se utilizan capas de *batch normalization* (BN) que normalizan las características de 0 a 1 e incrementan la velocidad de aprendizaje durante el entrenamiento. Además, en la última capa (de clasificación) se añade un módulo de restricción de *conditional random fields* (CRF) que otorga un juicio más preciso, sobre a qué categoría deben de pertenecer los píxeles de acuerdo con la semejanza con sus vecinos. Si los píxeles adyacentes tienen una pequeña diferencia, estos píxeles pertenecerían a una clase diferente con menor probabilidad. Por el contrario, si los píxeles adyacentes tienen una gran diferencia, estos píxeles pertenecerán una clase diferente con mayor probabilidad.

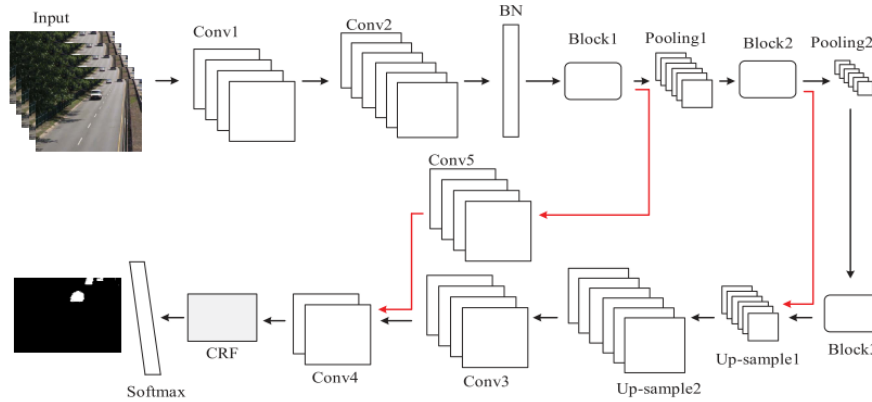


Figura 2.10. Arquitectura de la red con capas tipo *atrous* para detección de movimiento.

La evaluación del algoritmo se realizó sobre 6 videos de *CDnet14* y los resultados en cuanto a desempeño son buenos, con valores por encima de 0.82 en la métrica F-Measure en todos los videos. Sin embargo, son muy pocos videos para demostrar el desempeño correcto del modelo considerando que CDnet cuenta con 53 videos con diferentes situaciones críticas. Además, las comparaciones de los resultados se realizan con algoritmos que no son entrenados, entre ellos algunos modelos estadísticos como GMM, ViBe, PBAS. Por lo que, la evaluación no es particularmente justa.

2.9. Redes convolucionales 3D para detección de movimiento (3DCNN)

El algoritmo [9] incluye información temporal de la escena de video en una red convolucional 3D. En las convoluciones 2D el filtro tiene el mismo tamaño de profundidad que el mapa de características y se desplaza sobre los ejes X-Y. Por otra parte, en las convoluciones 3D el filtro tiene un tamaño menor que el mapa de características y se desplaza en las dimensiones X-Y-Z, lo que genera un nuevo mapa de características de 3 dimensiones. En la Figura 2.11 se ilustran los procesos antes descritos.

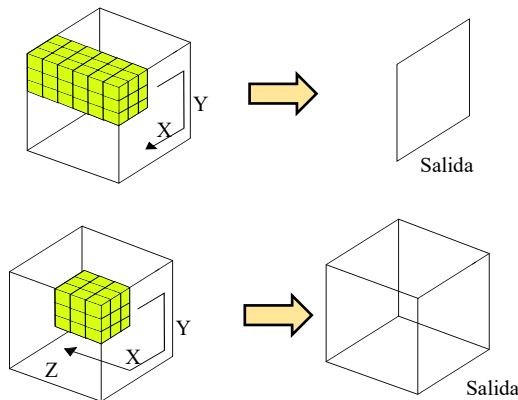


Figura 2.11. Comparación de una convolución 2D y una en 3D.

La arquitectura de red del algoritmo propuesto consiste en seis grupos de capas convolucionales como se observa en la Figura 2.12. La información temporal que recibe la red proviene de 10 imágenes sucesivas de tamaño 320x240, pero en lugar de concatenar todas las imágenes en una sola entrada se dividieron en grupos de 4 cuadros con saltos de 2. En otras palabras, el primer grupo le corresponderán las imágenes del 1 al 4, el segundo grupo las imágenes del 3 al 6, el tercero de la 5 a la 8 y el último de la 7 a la 10. Los bloques rojos mostrados en la Figura 2.12 corresponden a convoluciones 3D, después de cada convolución se aplican capas de activación tipo relu (bloques azules) y capas de *pooling* (bloques verdes). Además, existen capas de muestreos hacia arriba o *upsampling* (bloques amarillos) para ir aumentando la resolución de las estimaciones.

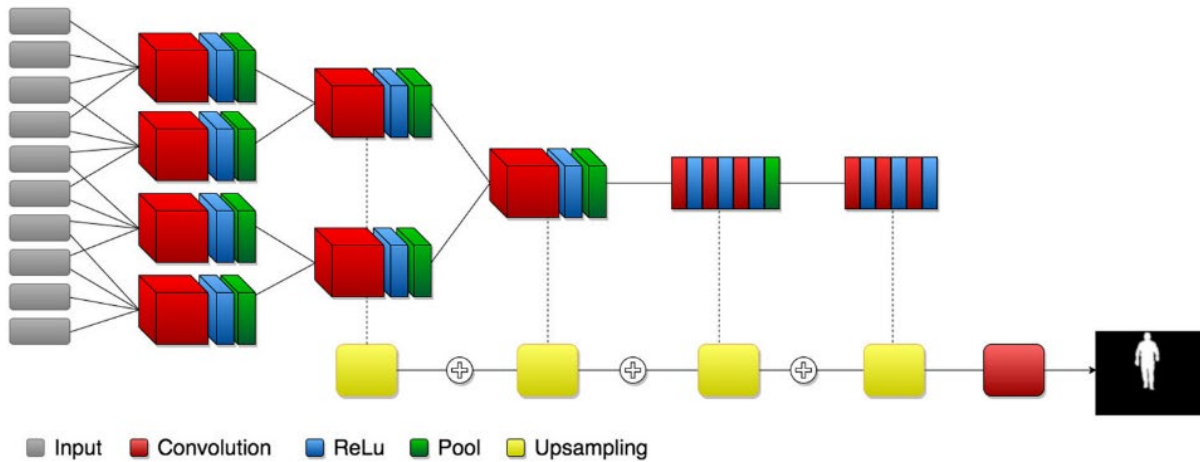


Figura 2.12. Arquitectura de la red 3D para detección de movimiento.

En el entrenamiento de la red se utilizan videos cortos de diez cuadros tomados de la base de datos CDnet. Por lo que una vez que los videos son divididos se toman 70% de los videos cortos aleatoriamente para el entrenamiento y el resto se usa de validación. Se usan 500,000 iteraciones con el algoritmo de gradiente descendente estocástico (GSD) con momento de 0.9 y un factor de aprendizaje inicial de 1×10^{-8} .

La implementación del modelo se realiza en Caffe en una computadora con 64 GB de RAM, con un procesador Intel Core i7-5960X CPU @ 3.00 GHz y cuatro unidades de procesamiento de gráficos GeForce GTX TITAN X.

Los resultados experimentales mostraron que el modelo superaba en las métricas a métodos como CascadeCNN obteniendo un F-Measure de 0.95 en promedio sobre todas las categorías de video. Además, es capaz de manejar diferentes escenas de una manera robusta, teniendo solo un

poco de problemas con las categorías *Nightvideos* y *LowFramerate*. Sin embargo, uno de los inconvenientes del método es que realizar convoluciones 3D en 10 imágenes de tamaño 320x240 es una operación de alta complejidad computacional que demanda una gran cantidad de recursos computacionales, como es el uso de 4 GPU Titan X. por lo que los autores pretenden en un futuro mejorar la eficiencia del modelo empleando convoluciones dilatadas.

2.10. Flujo óptico mediante redes convolucionales (*FlowNet*)

En [13] se propone el método *FlowNet* para estimar flujo óptico mediante el uso de redes convolucionales. El termino flujo óptico fue presentado por James Gibson en su libro *The Perception of the Visual Word* [166] en 1950. Gibson describe el flujo óptico cómo el patrón del movimiento aparente de los objetos, superficies y bordes en una escena causado por el movimiento relativo entre un observador (un ojo o una cámara) y la escena. Básicamente el flujo óptico nos permite identificar los vectores de movimiento entre dos imágenes de una secuencia de video. Para realizar flujo óptico la mayoría de las técnicas asumen que el brillo de cualquier punto es constante y que los puntos cercanos en el plano de la imagen se mueven de manera semejante y con una velocidad suavizada. A continuación, en la Figura 2.13 se observan dos interpretaciones distintas de los vectores de movimiento. A la derecha se muestran los vectores de movimiento de manera convencional, con su magnitud y dirección. Por otra parte, la figura de la izquierda en la Figura 2.13 muestra un código de color que representa la dirección y magnitud del movimiento por medio de la intensidad y tipo de color.

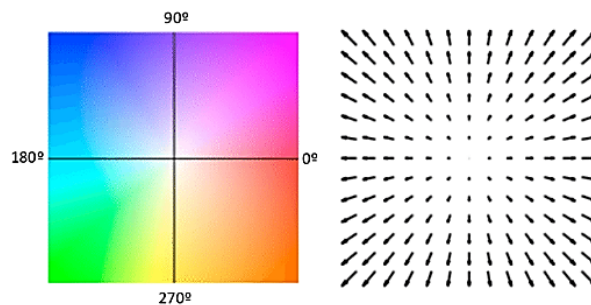


Figura 2.13. Representaciones distintas de vectores de movimiento.

Uno de los inconvenientes de las técnicas convencionales para estimar el flujo óptico es que calcular el vector de movimiento para cada pixel de la imagen es computacionalmente demandante. Además, debido a las asunciones antes descritas (brillo constante y velocidad similar entre vecinos) se cae en el problema de apertura de área, donde se tiene insuficiente información local para

Como se observa en la Figura 2.15, la versión más simple de *Flownet* es llamada *FlowNetSimple* e implica concatenar dos imágenes consecutivas y alimentarlas a la red permitiendo que la red decida por sí misma como procesar el par de imágenes para extraer movimiento. Sin embargo, los autores querían estar seguros de que la arquitectura estuviera capturando las diferencias entre ambas imágenes y diseñaron la arquitectura menos genérica llamada *FlowNetCorr*. En *FlownetCorr* se crean dos vertientes iguales que procesan las imágenes por separado y luego son combinadas en etapas posteriores. Con esta arquitectura la red se ve obligada a producir primero representaciones significativas de las dos imágenes por separado y luego combinarlas en un nivel superior.

Todos los bloques mostrados previamente en las arquitecturas de la Figura 2.15 son procesos de convolucion y se encargan de extraer las características de bajo y alto nivel de ambas imágenes por lo que hasta este punto se tendría la etapa de codificación o encoder. Por otro lado, la parte *refinement* o *decoder* tiene capas de convolución transpuesta y concatenaciones con capas menos profundas para de esta manera preservar información tanto de bajos como de altos niveles de la red y lograr estimaciones de movimiento cada vez más finas (a nivel pixel) conforme se avanza a capas más profundas. Véase Figura 2.16.

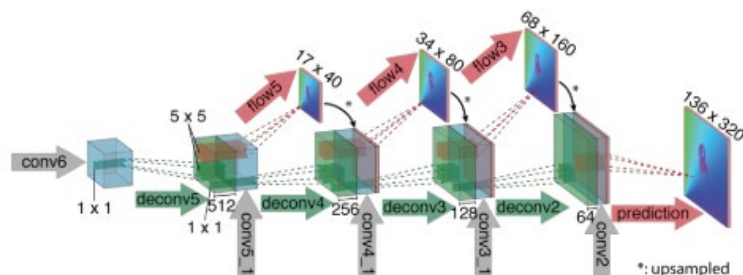


Figura 2.16. Etapa de refinamiento en *Flownet*.

Los resultados experimentales demostraron que ambas arquitecturas lograron realizar una correcta estimación del movimiento e incluso el modelo más simple fue mejor que el modelo *FlownetCorr* en la base de datos *Sintel*. Para el entrenamiento ambas arquitecturas utilizan el optimizador Adam con los parámetros fijos $\beta_1=0.9$ y $\beta_2=0.999$. Se usaron lotes de entrenamiento de 8 imágenes, con tasa de aprendizaje $\eta = 1e-4$ y luego esta se dividía por 2 cada 100,000 iteraciones después de las primeras 300,000 iteraciones. El conjunto de imágenes de entrenamiento

fue de 22,232 para la base de datos *Flyingchairs* y se dejaron 640 imágenes de prueba. Para la base de datos *Sintel* se usaron 908 imágenes de entrenamiento y 133 de validación.

Aunque *FlowNet* presenta un esquema novedoso para la estimación de flujo óptico aun presenta dificultades para detección de movimiento en entornos reales y desplazamientos pequeños en los pixeles de la escena analizada. Como una mejora a *FlowNet* se desarrolló una segunda versión llamada *FlowNet 2.0* [90] que introdujo un esquema que combina múltiples arquitecturas Flownets especializadas a un tipo de movimiento en particular. En la Figura 2.17 se observa la arquitectura de *FlowNet 2.0*, existen varios módulos con las arquitecturas simples FlownetSimple (FlownetS) y de correlación FlownetCorr (FlownetC) cada uno enfocado ya sea a desplazamientos grandes o pequeños. El tiempo de procesamiento aumenta debido a la combinación de múltiples módulos de *FlowNet*, sin embargo, se logra un decremento de más del 50% de error de estimación, que el del original.

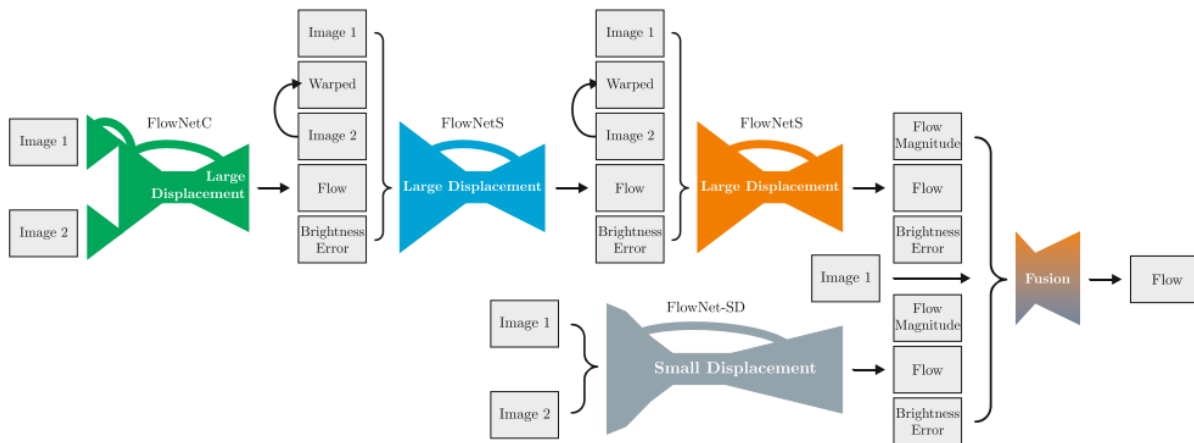
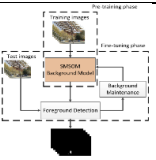
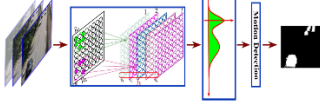
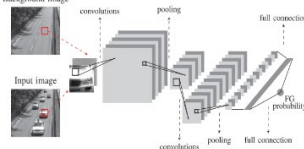
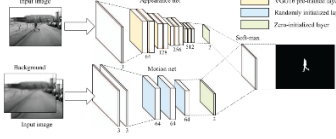
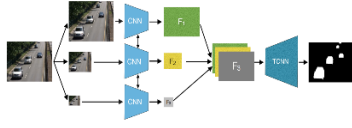
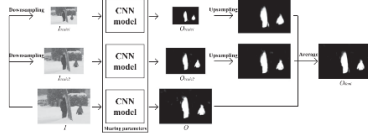
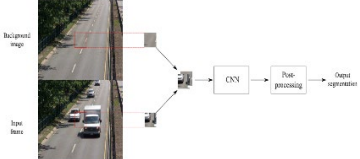
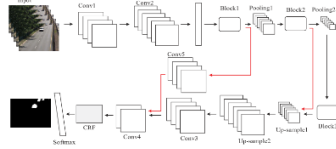
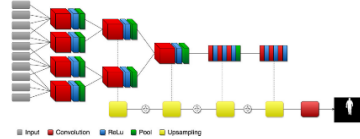
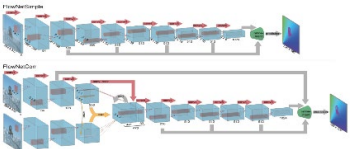


Figura 2.17. Arquitectura FlowNet 2.0.

2.11. Resumen del capítulo

En este capítulo se presentaron algunos ejemplos representativos de los algoritmos del estado del arte enfocados a detección de movimiento mediante el uso de técnicas de aprendizaje profundo. De los algoritmos analizados, se resumió la información más relevante y se generó la Tabla 2.1 que ayuda a visualizar a groso modo las ventajas y desventajas de algunos enfoques frente a otros. Además de que nos permite apreciar ciertas similitudes en los trabajos propuestos en cuanto a arquitectura y modos de trabajo.

Tabla 2.1. Resumen de las técnicas analizadas de aprendizaje profundo para detección de movimiento.

	Arquitectura	Modo de trabajo	A favor	En contra
SMSOM-BM		Emplea una red tipo SOM para generar un modelo de fondo el cual luego es actualizado con cada nuevo cuadro de video.	Adaptación a los cambios de escenario. Buenos resultados en escenarios con movimientos simples.	Se necesitan imágenes sin objetos dinámicos para entrenar el algoritmo y proveer una buena estimación inicial. Desempeños no favorables en movimientos de cámara y fondos dinámicos.
NeRM		Se utiliza una red previamente entrenada para extraer características del escenario que luego son usadas para construir un modelo de fondo mediante GMM.	Factible para su uso en sistemas embebidos. Mejores resultados en situaciones con sombras y mal clima.	Tiempo de procesamiento elevado 2FPS. Existe un efecto de ruido en forma de bloques en la detección final del movimiento. En general sus desempeños son pobres en categorías como fondos dinámicos, videos de noche, turbulencias, vibraciones, etc.
ConvNet		Se utiliza una imagen del fondo y el cuadro actual del video, se divide la imagen en parches y se pasan a una red convolucional que clasificar al pixel central del parche.	Sus mejores resultados son categorías de video con movimiento básico, mal clima, sombras, baja tasa de adquisición de cuadros y turbulencias.	Sus resultados dependen de la imagen de fondo generada por el algoritmo de modelado de fondo. El uso de un gran número de parches y convoluciones por imagen es computacionalmente demandante. No resuelve situaciones de objetos dinámicos intermitentes ni cámaras PTZ.
A-Net/M-Net		Se usa una red, A-net, para detectar objetos de interés y otra red, M-net, para detectar movimiento. Los resultados son combinados y se obtiene la detección de los objetos dinámicos de interés.	Posee una velocidad de procesamiento de 50fps, fue probado en una GPU GTX1070.	Depende de un algoritmo de modelado de fondo para suministrar una imagen de referencia. Depende del uso de la red preentrenada VGG-16. No fue evaluado ampliamente, solo se utilizaron 5 videos.
FgSegNet		Utiliza tres arquitecturas de red tipo Encoder, que se alimentan de la imagen de entrada a diferentes escalas, sus resultados se combinan y pasan a través de una etapa Decoder para obtener la estimación final.	Resuelve una amplia variedad de situaciones de videos, posee la primera posición en la base de datos <i>CDnet14</i> .	No considera información temporal del video. Arquitectura compleja. Tiempo de procesamiento no apto para tiempo real 18FPS. Depende del uso de la red preentrenada VGG-16.
CascadeCNN		Es el predecesor de FgSegNet, usa tres redes convolucionales en cascada alimentadas con imágenes a diferentes escalas. Genera una estimación por cada pixel central de una región de tamaño 31x31.	Resuelve una amplia variedad de situaciones de videos, se encuentra dentro los primeros diez lugares en la base de datos <i>CDnet14</i> .	Dificultades para resolver situaciones con objetos dinámicos muy pequeños. El uso de regiones es computacionalmente demandante. No considera información temporal del video.
DeepBS		La CNN realiza la estimación del movimiento por cada pixel centrado en regiones de tamaño 37x37. Se usa una imagen de fondo obtenida mediante los algoritmos SuBSENSE [58] y FTSG [79].	El método demostró buenos resultados para las categorías de video vibraciones de cámara, fondos dinámicos, turbulencia y sombras.	No resuelve las categorías de video cámaras PTZ o videos a baja tasa de adquisición de cuadros. Sus resultados dependen de la imagen de fondo generada por los algoritmos de modelado de fondo.
AtrousCNN		Utiliza tres redes convolucionales atrous con diferente dilatación para extraer información espacial de diferentes vecindarios de píxeles. La entrada a la red es la unión de múltiples cuadros de video.	Resuelve situaciones de video con escenarios simples. Extrae la información temporal y espacial de los cuadros por medio del aprendizaje.	El método no presenta una evaluación justa ya que se compara contra algoritmos que no son entrenados. Su desempeño general es escaso, el algoritmo se evalúa solo con seis videos de los 53 disponibles en <i>CDnet14</i> .
3DCNN		Se utilizan diez imágenes consecutivas de video para la estimación de cada cuadro. La arquitectura es una combinación de seis niveles de profundidad diferentes con convoluciones 3D.	Maneja diferentes videos de una manera robusta, posee un F-Measure promedio de 0.95 sobre todas las categorías de video de <i>CDnet14</i> .	Realizar convoluciones 3D con 10 imágenes de tamaño 320x240 es una operación de alta complejidad computacional que demanda una gran cantidad de recursos computacionales, como es el uso de 4 GPU Titan X.
FlowNet		El primer algoritmo capaz de estimar flujo óptico mediante un esquema supervisado. En la fase de entrenamiento se le proporciona a la red dos cuadros consecutivos de video de un escenario y el flujo óptico como GT.	Realiza una robusta y correcta estimación del movimiento en la base de datos Sintel y FlyChairs.	El primer modelo propuesto presenta dificultades para detección de movimiento en entornos reales y desplazamientos pequeños. Al introducir la versión que resuelve desplazamientos pequeños, se aumenta la complejidad de la red y el tiempo de procesamiento.

CAPÍTULO III. ALGORITMOS DE DETECCIÓN DE MOVIMIENTO INSPIRADOS EN LA VISIÓN BIOLÓGICA

La naturaleza en su complejidad posee fenómenos y procesos fascinantes que plantean una ruta y fuente de inspiración para resolver una gran cantidad de problemas. En particular, la detección de movimiento biológica ha sido estudiada en campos como la neurociencia, neurofisiología, psicología perceptual, sistemas de visión por computadora, entre otros. Cada una de estas áreas han aportado un avance significativo en el entendimiento, interpretación y aplicación de nuevas técnicas o algoritmos para emular el proceso biológico de la percepción de movimiento. Sin embargo, al tratarse de un sistema tan complejo, aun no existe un algoritmo que emule y cubra en su totalidad todos los aspectos que la visión involucra. Lo anterior ha conducido a los investigadores al constante desarrollo y mejora de algoritmos, con el fin de lograr técnicas más robustas y eficientes para detectar y analizar movimiento. A continuación, se describen algunos de los algoritmos, enfocados a detectar movimiento en video, que se han desarrollado en los últimos años inspirados en la percepción biológica del movimiento.

3.1. Vecindario retinotópico mediante red SOM

En [167] se propone una red autoorganizada (SOM) para el cálculo de la dirección del movimiento. La red contiene dos capas de neuronas: de entrada, L_1 y de procesamiento, L_2 . L_1 consiste en células análogas a las células hipercomplejas de la corteza visual, cada neurona proyecta conexiones de excitación a un vecindario retinotópico de neuronas en la segunda capa L_2 . Las neuronas de L_1 son sensibles al contraste, orientación y longitud a través de su campo receptivo. Sin embargo, no pueden distinguir en cuál de las combinaciones de dirección/velocidad el segmento se está moviendo. Por lo que, las neuronas en la segunda capa L_2 son sensitivas a la orientación y dirección particular del estímulo visual en movimiento que atraviesa su campo receptivo. En la Figura 3.1 se muestra el esquema del algoritmo propuesto, donde se observan las capas L_1 y L_2 , en la parte a) cada célula L_1 proyecta sus conexiones excitatorias a un grupo de cercanas de células en L_2 , por ejemplo, aquí la célula a excita las células b , c y d en L_2 y estas tres células transmitirán su excitación con un cierta latencia o retardo a las células vecinas, esto es representado por las flechas en b , c y d en la figura. En la parte b) de la imagen se observa el segundo tiempo t donde el estímulo ya no activa la célula a , ahora activa la célula e en L_1 y esta

excita a las células h , g y f en L_2 , en ese mismo momento las excitaciones con latencia del tiempo anterior alcanzan sus destinos en las células f , i y j . La célula f recibe dos estímulos excitatorios simultáneos de e y b , por lo tanto, f se vuelve fuertemente activa y suprime las activaciones de sus células vecinas por inhibición lateral. Por su parte las excitaciones en las células i y j son activadas muy débilmente para propagar sus excitaciones. Por consiguiente, solo la célula f puede propagar su activación a sus células sucesoras laterales. La activación de la única célula f representa la decisión recién calculada de la red de que el segmento de entrada se mueve horizontalmente. Así, las latencias de transmisión de la señal permiten la representación de la dirección del movimiento.

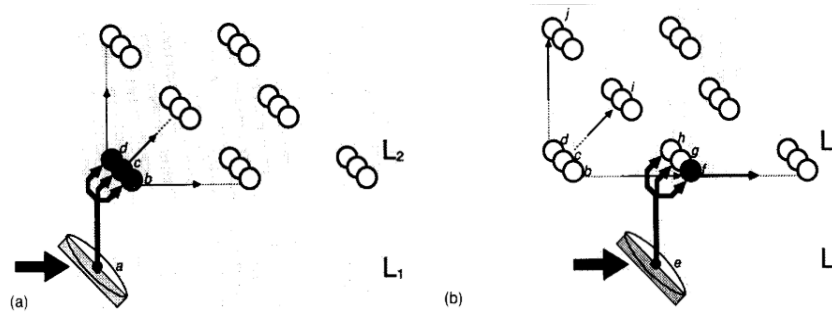


Figura 3.1. Estimulo moviéndose de forma horizontal en el tiempo a) activa la célula a que proyecta conexiones excitatorias a un grupo cercano de células en la capa L_2 . En el tiempo b) el estímulo cambia de posición y activa la célula e que propaga su excitación a las células h , g y f . Solo f es fuertemente estimulada por las células b y e indicando la dirección del movimiento horizontal. Imagen tomada de [167].

3.2. Red autoorganizada con adaptación de retraso

En [168] se utiliza una red autoorganizada con adaptación de retraso, llamada *self-organizing delay adaptation* (SODA). El mapeo de SODA es compuesto de neuronas organizadas en dos capas totalmente conectadas (de retina y de corteza visual). Las conexiones de la capa de retina a la capa de corteza visual tienen todas las mismas constantes de peso, pero con diferente retraso de aprendizaje. La regla de aprendizaje puede ser vista análoga a la regla Hebb, solo que, en lugar de pesos siendo ajustados para crear mapeos de coincidencia de activación espacial, los retrasos son ajustados para crear coincidencia de activación temporal. En la Figura 3.2 se observa cómo se realiza el ajuste en las neuronas, cada figura muestra retrasos de tiempo en las conexiones, donde a) es el ajuste inicial, b) es una neurona ajustada a detectar movimiento en dirección (oeste), esta neurona corresponde a la coordenada (3,3) en el mapa de la Figura 3.3b y c) es una neurona que se ajustó a dos diferentes direcciones de movimiento (este-oeste), esta neurona corresponde a la coordenada (5,7) de la Figura 3.3b.

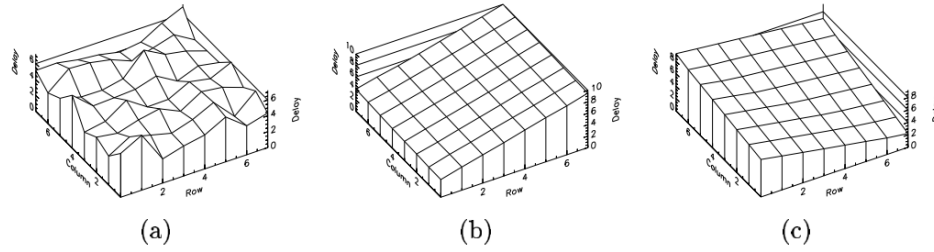


Figura 3.2. Ajuste de posición de retraso, figura tomada de [168]. Cada figura muestra retrasos de tiempo en las conexiones de cada locación de la capa de retina con una neurona particular en la capa de corteza visual.

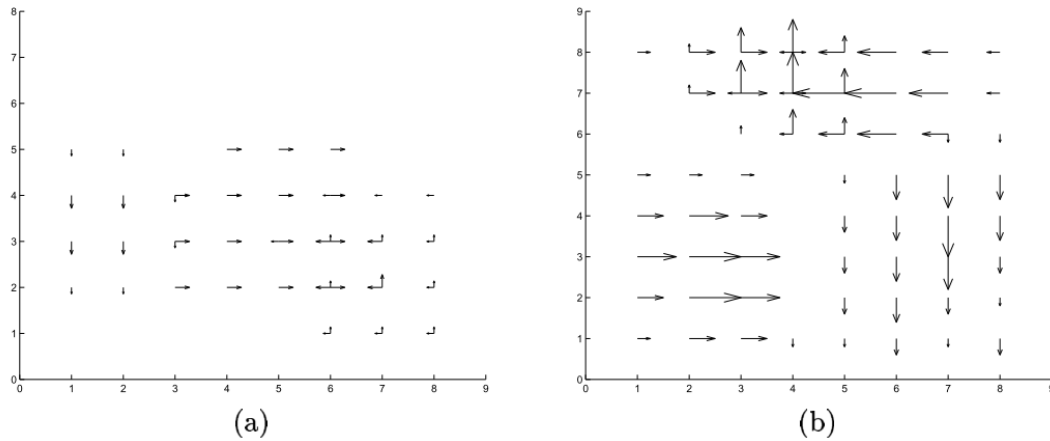


Figura 3.3. Mapeo de selectividad direccional, figura tomada de [168]. El eje x y y indican locación en la capa de corteza visual. En cada locación, se representa la respuesta de una neurona cortical. La longitud de la flecha representa la fuerza de la respuesta de la neurona al estímulo que se mueve en la dirección de esta. Las neuronas sin flechas no obtuvieron respuesta al estímulo. a) mapa inicial, un poco desorganizado. b) Mapa después del entrenamiento, el mapa se vuelve más organizado con diferentes áreas que responden a distintos estímulos de entrada y con transiciones suaves entre áreas.

3.3. Red neuronal de percepción de movimiento rotacional

En [169] se propone un algoritmo basado en la forma de percepción de patrones de movimiento de la visión biológica mediante una red neuronal selectiva a dirección o DSNN por sus siglas en inglés. La red propuesta consiste en dos partes, presináptica y post sináptica. En la parte presináptica, hay una serie de DSNN para extraer señales visuales de dirección. La Figura 3.4 muestra una neurona selectiva a la dirección de movimiento hacia la izquierda L , se puede observar las cuatro capas (P , E , I , S) conectadas a la neurona L . La capa P trabaja como neuronas fotorreceptoras, cada neurona en esta capa calcula los cambios de iluminancia entre dos cuadros consecutivos de video. La capa E o de excitación contiene una copia de las neuronas excitadas de la capa P y por su parte la capa I contiene las neuronas inhibidas. La capa S recibe las capas de inhibición I y excitación E , para calcular un grado o fuerza de excitación para cada neurona S . Por otro lado, en la parte postsináptica existen neuronas selectivas a la dirección en un orden cíclico a

manera de que se perciba señales de movimiento rotacional. La red contiene neuronas con 16 direcciones selectivas al movimiento las cuales ayudan a detectar el sentido de la rotación del movimiento.

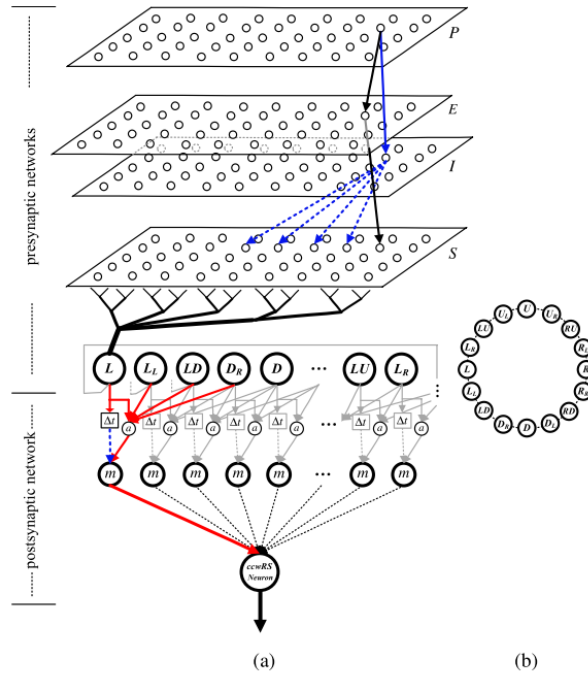


Figura 3.4. Ilustración esquemática del algoritmo [169] con arreglo de 16 neuronas selectivas a la dirección del movimiento.

3.4. Prevención de colisiones basado en el detector de movimiento LGMD

En [170], Shigang Ye y F. Claire Rind proponen una red basada en el detector de movimiento *Lobula Giant Movement Detector* (LGMD), una neurona identificada en el cerebro de la langosta que responde con mayor excitación a movimientos de objetos que se aproximan (como un depredador). La red está compuesta de cuatro grupos de células: foto receptivas (P), excitatorias (E), inhibitorias (I), sumatorias (S) y de agrupamiento (G). Las neuronas foto receptivas de la primera capa, toman el valor de luminancia de cada pixel y se calcula el cambio de luminancia entre la imagen actual y la previa. Si no hay diferencia entre las imágenes sucesivas las neuronas no se excitan. En la capa excitatoria E , se toman los valores del cambio de luminancia de la capa P y por su parte la capa inhibitoria permite obtener mayor dispersión entre las neuronas que se dispararon y las que no mediante un peso de inhibición W_1 . La capa S corresponde a la resta de la capa de excitación contra la capa de inhibición multiplicada por su peso W_1 . Finalmente, la capa de agrupamiento G asocia la excitación de varias neuronas E . El algoritmo fue integrado a un robot autónomo con el fin de que este pudiese evitar posibles colisiones, detenerse, girar en otra dirección

y continuar su trayectoria. Los resultados demostraron que las colisiones inminentes fueron exitosamente detectadas debido a que se producía mayor excitación en las neuronas LGMD a mayor velocidad.

3.5. Detección de movimiento inspirada en la estructura visual de la mosca

Otros trabajo enfocado en percepción de movimiento [171], propone un sistema de detección de colisión inspirado en la estructura de visión de la mosca. El sistema se compone de tres módulos principales; detección de potenciales objetos en movimiento, decisión de la dirección del movimiento y envío de alarma de colisión. La red de detección dirección de movimiento está compuesta por cuatro capas neuronales que se ilustran esquemáticamente en la Figura 3.5.

La primera capa (la retina) recibe la luminancia de la luz o los niveles de gris de los píxeles de imágenes sucesivas. La segunda capa (lámmina) recibe la luminancia de la capa de la retina y envía las actividades de las neuronas a la tercera capa (médula). La tercera capa envía las actividades locales de las neuronas a la última capa (lóbulo) por medio de los detectores de Reichardt. La última capa suma todos los potenciales de capas anteriores para generar el potencial general de la red. Los detectores de Reichardt son circuitos neurales hipotéticos sobre cómo el cerebro puede rastrear el movimiento. En un detector de Reichardt, una célula del cerebro recibe la entrada de dos receptores en el ojo, llamados A y B. La entrada A contiene un retraso. Si el movimiento de los receptores A hacia B se sincroniza de tal manera que la célula del cerebro recibe las entradas de A y B al mismo tiempo, entonces la célula del cerebro se disparará y se asume que la célula posee preferencia a ese movimiento específico.

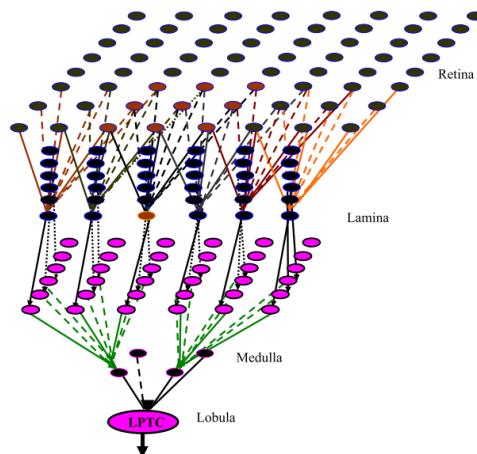


Figura 3.5. Red neuronal artificial basada en la estructura visual de la mosca [171].

3.6. Algoritmo basado en neurona STDM sensible a pequeños movimientos

El algoritmo [172] se encuentra inspirado en un específico tipo de neurona dentro de la mosca llamado *Small Target Motion Detector* (STMD) la cual ha sido identificada como una neurona sensible a pequeños objetivos en movimiento (selectividad al tamaño). Además, las STDM muestran fuertes respuestas al movimiento en la dirección preferida (selectividad a la dirección). La red neuronal del algoritmo propuesto está compuesta por cuatro capas neuronales: retina, lámina, médula y lóbulo (véase Figura 3.6). Estas cuatro capas neuronales ordenadas secuencialmente tienen funciones específicas y cooperan juntas para la detección de movimientos de objetivos pequeños. La capa de retina contiene un gran número de ommatidios. Como se muestra en la Figura 3.7, cada ommatidio está compuesto por 8 fotoreceptores, por lo que una ventana cuadrada es usada para aproximar el campo receptivo hexagonal biológico de esta neurona. En la capa de lámina existen neuronas que muestran respuestas fuertes a incrementos y decrementos de luminancia y actúan como un filtro temporal extrayendo información de movimiento. En la capa de la médula existen neuronas Tm1, Tm2, Tm3 y Mi1. Las neuronas Mi1 y Tm3 responden selectivamente a los incrementos de brillo, con la respuesta de Mi1 retrasada en relación con Tm3. Por el contrario, Tm1 y Tm2 responden selectivamente a los decrementos de brillo, con la respuesta de Tm1 retrasada en comparación con Tm2. En la capa de lóbulos, las neuronas integran la información transmitida desde la capa de la médula respondiendo fuertemente al movimiento de los objetivos pequeños.

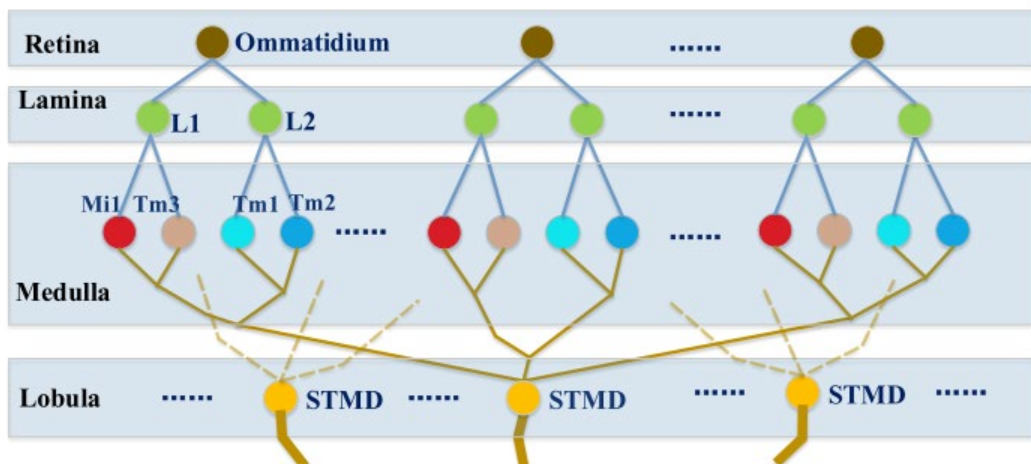


Figura 3.6. Red neuronal artificial STDM [172].

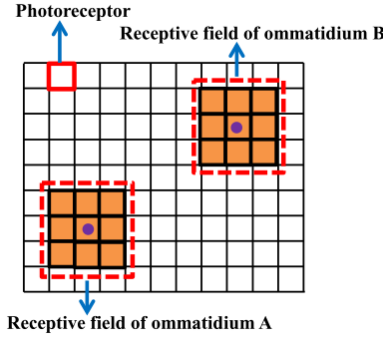


Figura 3.7. Ilustración esquemática del mapeo de los pixeles o fotoreceptores. Cada cuadrícula denota un píxel que corresponde a un fotoreceptor, los rectángulos punteados en rojo representan el campo receptivo de un ommatidio [172].

3.7. Red Retinotópica Autoorganizada (RESOM)

El enfoque *Retinotopic SOM* (RESOM) [33] propone una arquitectura que se inspira en las interacciones de las neuronas en la corteza visual. RESOM cuenta con tres redes SOM asociadas a cada uno de los pixeles k en los canales $z=\{R, G, B\}$ de la imagen $I^t_{k,m}$ en espacio de color RGB. Lo anterior se ilustra en la Figura 3.8, donde cada W^z_m es un vector que contiene 4 neuronas $\omega^z_{k,m}$ con k pesos asociados a cada uno de los pixeles en cada canal de la imagen.

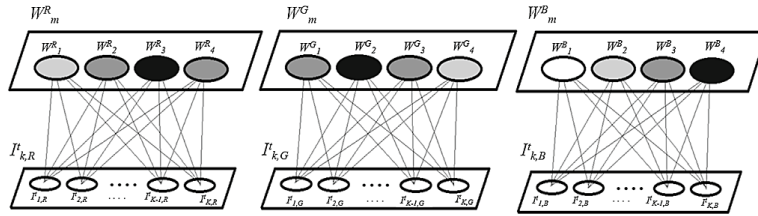


Figura 3.8. Arquitectura RESOM [33], cada vector W^z_m contiene 4 neuronas

La idea general es que los parámetros de la red RESOM son ajustados para modelar o aprender las características del escenario o fondo del video. En cada red del canal z , se tienen η_m^z neuronas que compiten, de tal forma que la ganadora será la que tenga mayor ponderación respecto a su vecindario. Por lo que, los pesos de estas son actualizadas en cada interacción mediante la regla de aprendizaje:

$$\begin{aligned}\omega_{k,m}^{z,ni+1} &= \omega_{k,m}^{z,ni} + \Delta\omega_{k,m}^{z,ni} \\ \Delta\omega_{k,m}^{z,ni} &= \eta^t [I_{k,z}^t - \omega_{k,m}^{z,ni}] \beta_m^z\end{aligned}\tag{55}$$

Donde η^t es la tasa de aprendizaje, ni es el número de iteraciones que la red procesa con los cuadros $I^t_{k,m}$, β_m^z es un parámetro que caracteriza el comportamiento del vecindario retinotópico, mediante:

$$\beta_m^z = \exp \left(-\frac{(\eta_\omega^z - m)^2}{(\sigma^t)^2} \right) \quad (56)$$

Donde η_ω^z es la neurona ganadora y σ^z es el radio del vecindario.

Una vez entrenado el modelo RESOM con el primer cuadro de video, los pesos de las redes se actualizan con cada cuadro de video mediante la siguiente regla:

$$\omega_{k,m}^{z,Nf} = (1 - \beta_m^z \eta^t)^{Nf} \omega_{k,m}^{z,0} + [1 - (1 - \beta_m^z \eta^t)^{Nf}] I_{k,z}^t \quad (56)$$

Donde Nf es un número de iteraciones que define la capacidad de aprendizaje por cuadro, si Nf tiende a infinito las neuronas aprenderán completamente el escenario del cuadro actual del video, conforme las iteraciones pasan los pesos son actualizados y se adaptan al contenido del cuadro.

Las neuronas W_m^z forman las imágenes $V_{x,y,z}^{m,t} = \{ W_m^R, W_m^G, W_m^B \}$ que luego se utilizarán para definir la imagen $Ib_{x,y,z}^t$ como el promedio de $V_{x,y,z}^{m,t}$ y la imagen $Id_{x,y}^t$ como la suma de las diferencias en $V_{x,y,z}^{m,t}$. $Id_{x,y}^t$ es usada para localizar e inhibir la información aprendida de los objetos dinámicos en $Ib_{x,y,z}^t$ lo que conduce a un correcto modelado de fondo. Finalmente, la detección de movimiento de los objetos dinámicos se realiza umbralizando una imagen $Ie_{x,y}^t$ obtenida mediante:

$$Ie_{xy}^t = \|I_{xyz}^t - Ib_{xyz}^t\|_2 \quad (57)$$

El diagrama de la Figura 3.9 ilustra el proceso completo de la detección de movimiento mediante el algoritmo RESOM que se utiliza como modelo de fondo para el algoritmo DR-SOM. En los resultados experimentales DR-SOM demostró su habilidad para resolver situaciones de cambios de iluminación, problemas de *bootstrapping*, fondos dinámicos y objetos en movimiento que se vuelven estacionarios durante varios cuadros de video. Sin embargo, no logra resolver situaciones de camuflaje y el proceso de adaptación en cambios bruscos de iluminación toma alrededor de 16 cuadros de video, lo cual conduce a una cantidad considerable de errores en la clasificación de píxeles.

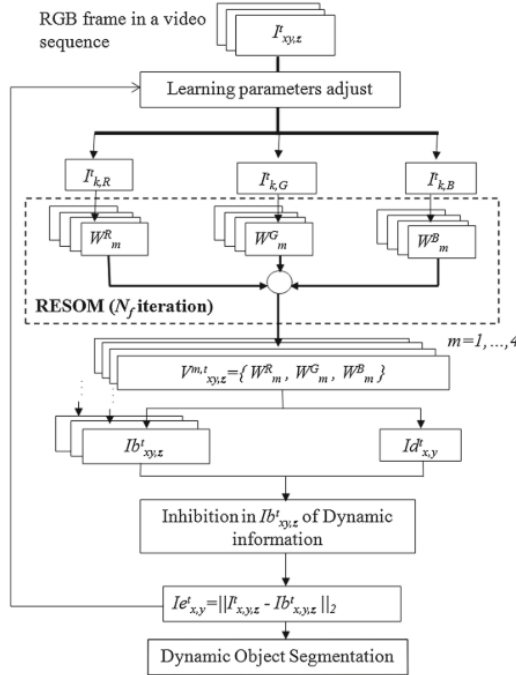


Figura 3.9. Detección de movimiento mediante DR-SOM [33].

3.8. Red de sumatoria binocular para detección dinámica (X-NET)

El enfoque [173], fue el único algoritmo encontrado, que está basado en un aspecto de la visión biológica y además utiliza redes convolucionales para realizar la detección del movimiento. El algoritmo es llamado X-Net ya que posee una arquitectura en forma de equis, basada en la estructura del nervio óptico del sistema binocular de visión humana, tal como se observa en la Figura 3.10. Como se muestra en la imagen se tienen dos vertientes o ramas, la del campo visual del ojo derecho y la del ojo izquierdo que reciben diferentes estímulos visuales e intercambian información de estos. La arquitectura de red propuesta se muestra en la Figura 3.11, la vertiente del ojo derecho recibe el estímulo visual del cuadro $I(t-1)$ en tres diferentes escalas y la del ojo izquierdo el cuadro $I(t)$. Las tres imágenes a distintas escalas del ojo izquierdo y derecho son procesadas de manera simultánea y pasadas a través de la etapa *Encoder* de la red, la cual combina la información de ambos ojos para la extracción de características. Luego en la parte *Decoder* se aumenta la resolución de las características extraídas hasta llegar a una estimación a la resolución original del cuadro de entrada.

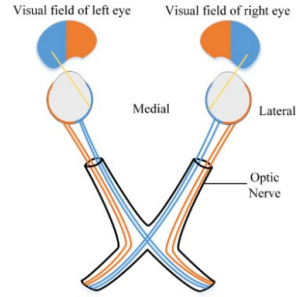


Figura 3.10. Forma de equis en el nervio óptico del sistema de visión binocular humana, [173].

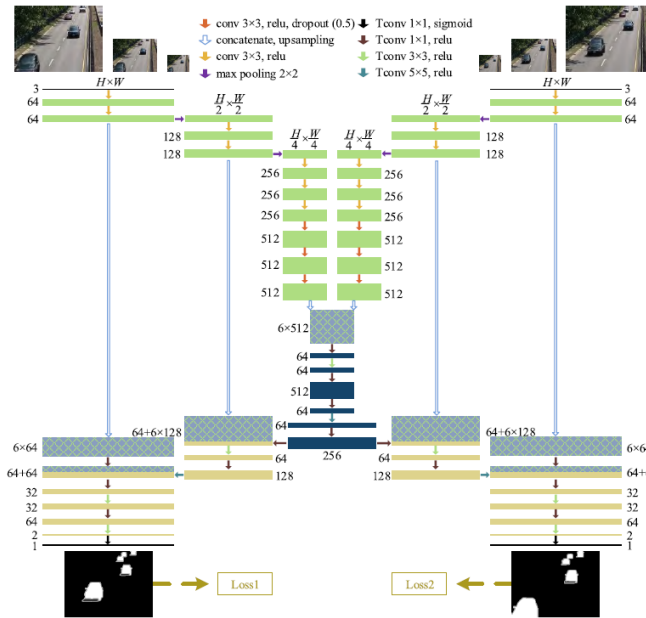


Figura 3.11. Arquitectura X-Net, fusión de capas Encoder-Decoder emulando la estructura binocular del nervio óptico [173].

X-Net demostró su originalidad y robustez para manejar los diferentes escenarios de video de la base de datos *CDnet14* con altos índices de desempeños. Sin embargo, al igual que algunas de las técnicas analizadas en el capítulo II, su tiempo de procesamiento no es apto para tiempo real y depende del uso de la red preentrenada VGG-16 para lograr un correcto entrenamiento.

3.9. Resumen del capítulo

Los algoritmos presentados en este capítulo muestran el avance en los últimos años en el área de algoritmos bioinspirados para realizar detección del movimiento. Los primeros enfoques aquí analizados, [167] y [168], utilizaban redes SOM con adaptaciones de latencia en las conexiones del vecindario para obtener la descripción del movimiento. Después se realizaron estructuras más complejas en cascada que ya no eran de solo dos capas, sino de cuatro o más que permitían detectar movimientos rotacionales [169]. De estas estructuras complejas en cascada, cada célula en capas

menos profundas al ser disparada transmitía su excitación a células en capas más profundas, lo que emulaba de alguna manera las conexiones complejas de sistemas de visión inspirados en neuronas sensibles al movimiento observadas en langostas [170] y moscas [171] [172]. Sin embargo, los algoritmos antes mencionados se limitaban a proponer un modelo matemático y los que se probaban o evaluaban lo hacían de una manera escasa en videos muy simples. Uno de los algoritmos bioinspirados que se evaluó parcialmente con una base de datos pública (*CDnet14*) y propuso un modelo más robusto para separar los objetos dinámicos fue [33]. Hasta este punto, los investigadores en su mayoría recurrían al uso de las redes SOM para proponer sus arquitecturas, luego en [173] se exploró el uso de las redes convolucionales para el diseño de una arquitectura bioinspirada en la visión binocular demostrando altos índices de desempeño y mostrando una evaluación completa en la base de datos *CDnet14*.

De acuerdo con el análisis realizado, los algoritmos bioinspirados presentan buenas alternativas para la creación y desarrollo de nuevos modelos con mejores características. Sin embargo, esta área continúa sin ser explorada en su totalidad, por lo que se recomienda ampliamente continuar la búsqueda y estudio acerca de mecanismos o adaptaciones en la visión para el desarrollo de mejores y más robustos algoritmos enfocados a detectar movimiento.

CAPÍTULO IV. RED NEURONAL CONVOLUCIONAL PARA DETECCIÓN DE MOVIMIENTO CON INFORMACIÓN TEMPORAL (2FraCNN)

La detección de movimiento es uno de los desafíos más importantes en aplicaciones de visión por computadora. Por lo que, en los últimos años se han desarrollado una gran variedad de algoritmos con diferentes técnicas, siendo el uso de redes neuronales convolucionales una de las más prometedoras, cuyos modelos [8]–[10] se posicionan en los primeros lugares de la base de datos *Change detection 2014* y últimamente han obtenido desempeños similares, e incluso mejores que el humano, de acuerdo con el trabajo de tesis “Análisis y comparación de algoritmos de modelado de fondo” [11].

En el capítulo II, se presentaron los algoritmos del estado del arte que utilizan aprendizaje profundo para realizar detección de movimiento con sus principales características y limitaciones. Algunos modelos se ayudan de algún algoritmo de modelado de fondo que les proporciona un cuadro de referencia que es usado junto con el cuadro actual del video como entrada a la red convolucional [86][87][88]. Sin embargo, los resultados de estos algoritmos dependen en gran parte del algoritmo generador de fondos. Además, algunos de estos algoritmos, también utilizan un esquema de parches o ventanas, donde se genera una predicción por cada pixel central contenido en una ventana de $M \times N$, lo que demanda una cantidad considerable de recursos computacionales. Por otro lado, existen técnicas que utilizan una arquitectura en cascada, donde la entrada de la red se alimenta del cuadro actual del video a diferentes escalas [8][10][174]. Estos trabajos, aunque logran buenos resultados no consideran información temporal y dependen de pesos de redes previamente entrenadas como la red VGG-16.

Para aliviar las problemáticas antes mencionadas y considerar información espacio temporal para estimar el movimiento, en este trabajo se presenta el desarrollo de un algoritmo de aprendizaje profundo basado en la arquitectura de *FlowNet* [13]. El algoritmo *FlowNet*, descrito en la sección 2.10, cambió el paradigma de la estimación de flujo óptico al ser el primero en utilizar una arquitectura de red neuronal convolucional para realizar estimación de flujo óptico. La arquitectura *FlowNet* al estimar el flujo óptico también otorga una estimación cruda del movimiento. Por lo que, se decidió basarse en el diseño en esta arquitectura para proponer un nuevo algoritmo enfocado a detectar el movimiento en los objetos de interés. El modelo propuesto de red es llamado Red Neuronal Convolucional de Dos cuadros Sucesivos o *Two-Frame Convolutional Neural Network*

(2FraCNN) ya que emplea la información temporal de dos cuadros consecutivos del video en una arquitectura de red convolucional. Aunque la arquitectura de 2FraCNN se basa en *FlowNet*, existen diferencias marcadas entre ambos algoritmos las cuales son mencionadas en las siguientes secciones.

Básicamente, 2FraCNN consiste en una red convolucional tipo codificador-decodificador (*Encoder-Decoder*). El codificador es alimentado con información temporal, espacial y de color mediante la concatenación de dos cuadros consecutivos de video, esta concatenación luego es propagada a través de las capas convolucionales que se encargan de la extracción de características. En el decodificador existe un arreglo *Coarse to fine* [175], donde la información de capas previas se concatena a capas más profundas y se utilizan convoluciones transpuestas para expandir el mapa de características y producir una detección en alta resolución.

4.1. Arquitectura inspirada en *FlowNet* para detección de movimiento

La arquitectura de red del algoritmo propuesto 2FraCNN posee las siguientes contribuciones:

- A diferencia del esquema multiclase de *FlowNet* para la estimación del flujo óptico, 2FraCNN es una arquitectura para la clasificación binaria destinada a detectar el movimiento de los objetos de interés. Por lo tanto, se realizaron cambios en su arquitectura, entrenamiento e implementación para cumplir con este propósito.
- Los modelos posicionados en los primeros lugares en *CDnet14* utilizan redes preentrenadas en sus arquitecturas [8][10], lo que significa que esos modelos requerirían largas horas de entrenamiento si no tuvieran esos pesos preentrenados. Por el contrario, 2FraCNN no requiere aprendizaje de transferencia, y sólo emplea 200 cuadros de vídeo para su entrenamiento. Además, 2FraCNN incluye un nuevo esquema de red que considera la información espaciotemporal del video.
- Se redujo el número de capas de convolución en 2FraCNN en comparación con la arquitectura *FlowNet*, lo que repercute directamente en el costo de los cálculos y reduce el tiempo de entrenamiento. Además, se introdujeron capas de *Dropout* durante el entrenamiento para lograr una mejor generalización.
- Por último, 2FraCNN tiene un método novedoso de balance de clase que ayuda durante la etapa de entrenamiento a proporcionar una rápida convergencia.

La arquitectura propuesta *Encoder-Decoder* se muestra en la imagen de la Figura 4.1, presentando un ejemplo de un vídeo con una resolución de 320x240. La idea básicamente es dotar a la entrada de la red convolucional información temporal obtenida de dos cuadros sucesivos y su respectivo GT a la salida para realizar el entrenamiento, lo que permitirá a la red aprender por si sola a extraer las características del movimiento en la escena. A continuación, se explica formalmente las partes involucradas en esta arquitectura.

a) Encoder

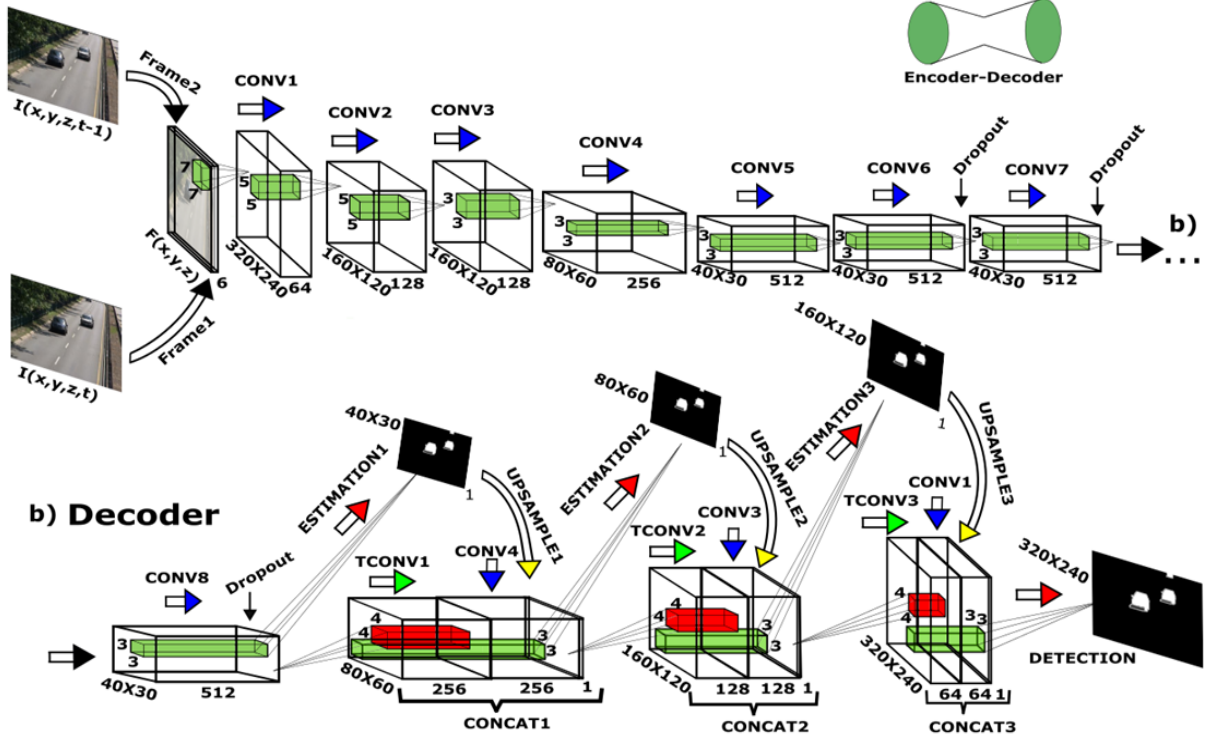


Figura 4.1. Esquema general del algoritmo propuesto 2FraCNN; a) etapa Encoder, b) etapa Decoder.

De la Figura 4.1, la parte codificadora realiza la extracción de características de alto nivel mediante capas convolucionales CONV y la parte decodificadora permite ampliar gradualmente los mapas de características para proporcionar una detección de alta resolución. El proceso de expansión de los mapas de características también se denomina *coarse to fine* [13][176]. Los principales elementos de este tipo de disposición son; capas de muestreo hacia arriba, UPSAMPLE, capas de estimación, ESTIMACIÓN, capas de convolución transpuesta, TCONV, y capas de concatenación, CONCAT. Las capas de UPSAMPLE y TCONV son procesos de convolución transpuesta que permiten ampliar los mapas de características[177]–[179]. Las capas CONCAT ayudan en el proceso de extracción de características, preservando la información de alto y bajo

nivel y conduciendo a una estimación más precisa del movimiento. En cada capa de ESTIMACIÓN se aumenta la resolución al doble hasta obtener la resolución original del vídeo.

Los prismas rectangulares verdes y rojos de la Figura 4.1, ilustran filtros de convolución Ke y convolución transpuesta Kd respectivamente para cada capa de la red. Las flechas azules y rojas indican procesos de convolución, mientras que las flechas amarillas y verdes indican procesos de convolución transpuesta. Las flechas descendentes ($\downarrow$) indican el uso de las convoluciones anteriores, mientras que las flechas horizontales ($\rightarrow$) indican el flujo de datos. A continuación, se dará una descripción matemática detallada del método propuesto.

En la parte *Encoder* (Figura 4.1a) la red es alimentada con la información espaciotemporal $F(x,y,z)$. $F(x,y,z)$ es un tensor obtenido de la concatenación de dos imágenes o cuadros de video sucesivos, el anterior $I(x,y,z,t-1)$ y el actual $I(x,y,z,t)$ en el espacio de color LAB, debido a que es un espacio de color basado en la teoría de color oponente de los campos receptivos del color en la corteza visual. Lo anterior permite que se produzcan cambios o ajustes de luminosidad sin alterar los colores, o ajustes en los colores sin alterar la luminosidad, similar al procesamiento humano.

$$F(x, y, z) = I(x, y, z, t) \parallel_z I(x, y, z, t-1) \quad (58)$$

Donde $\parallel_z$ representa la concatenación respecto al eje z , y (x,y,z) las coordenadas espaciales de ancho, alto y profundidad de la imagen I en el tiempo t . Por lo tanto, se formará una entrada $F(x,y,z)$ de 6 canales de profundidad ($L^*, a^*, b^*, L^*, a^*, b^*$) por los canales en el espacio de color LAB de cada imagen concatenada. La concatenación $F(x,y,z)$ contiene información espacial, temporal y de color de la escena, que será procesada por la red.

El modelo 2FraCNN, como se ilustra en la Figura 4.1, posee 12 capas convolucionales; 8 en la etapa de codificación (CONV), 3 de aproximación (ESTIMACIÓN) y 1 para la detección final (DETECCIÓN). El resultado de cada capa convolucional k con el j -ésimo filtro Ke se denomina mapa de características $fm(x,y)_{k,j}$ y se obtiene por:

$$fm(x, y)_{k,j} = G_k * Ke_{k,j}$$

$$fm(x, y)_{k,j} = \sum_{z=1}^D \sum_{u=1}^{Hf} \sum_{v=1}^{Wf} G(x+u-1, y+v-1, z)_k Ke(u, v, z)_{k,j}$$

$$\begin{aligned} x &= \{1 + nx_1S, 1 + nx_2S, \dots, Wn\}, \quad nx \in \mathbb{N}^{1 \times Wn/S} \\ y &= \{1 + ny_1S, 1 + ny_2S, \dots, Hn\}, \quad ny \in \mathbb{N}^{1 \times Wn/S} \end{aligned} \quad j = \{1, 2, 3, \dots, Nf\} \quad (59)$$

Para la primera capa convolucional ($k=1$), $G(x,y,z)$ será la concatenación de los cuadros sucesivos $F(x,y,z)$ y para capas k posteriores, $G(x,y,z)$ será los mapas de características generados de capas de convolución previas $fm(x,y)_{k-1,j}$. Donde (u,v,z) son coordenadas espaciales de ancho, alto y profundidad del filtro Ke con tamaño $W_f \times H_f \times D$, de ancho, alto y profundidad. N_f es el número de filtros Ke . S es el tamaño del paso o *stride* y nx y ny son índices que indican la posición a la cual se desplaza el filtro con *stride*. Por lo que, el mapa de características generado $fm(x,y)_{k,j}$ será de tamaño:

$$fm(x,y)_{k,j} \in \mathbb{R}^{W_n \times H_n \times N_f} \quad (60)$$

Los valores, N_f , W_n y H_n definirán el tamaño de profundidad, ancho y alto del nuevo mapa de características, W_n y H_n , se obtienen mediante:

$$W_n = \frac{W + 2pad - W_f}{S} + 1 \quad H_n = \frac{H + 2pad - H_f}{S} + 1 \quad (61)$$

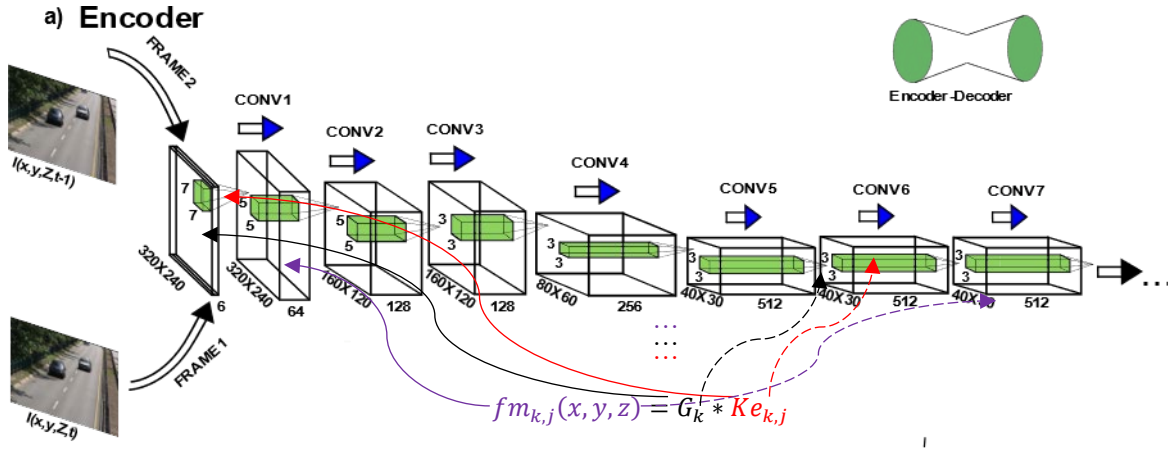
Siendo W y H el tamaño de ancho y alto del mapa de características G y pad el tamaño de *padding* considerado en cada capa, obtenido mediante:

$$pad = \frac{W - W_n + W_f - 1}{2} \quad (62)$$

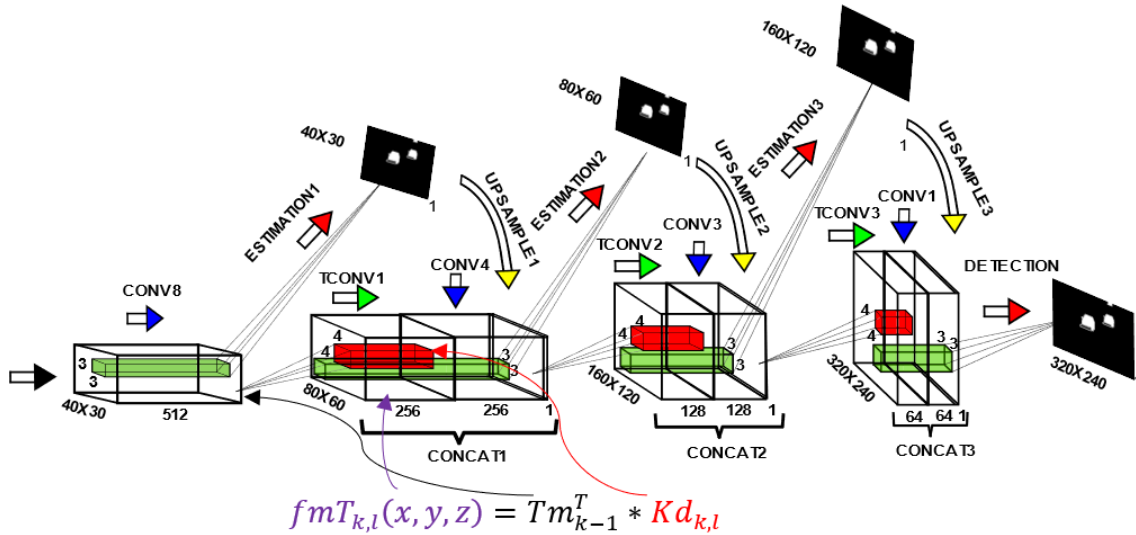
Considerando un *padding same* en todas las capas, tenemos que $W=W_n$, entonces:

$$pad = \frac{W_f - 1}{2} \quad (63)$$

La Figura 4.2 ilustra el proceso de propagación hacia adelante a través de la etapa *Encoder* de la red, donde el filtro Ke (prisma rectangular verde) es convolucionado con el mapa de características G produciendo el nuevo mapa de características fm .

Figura 4.2. Propagación hacia adelante etapa *Encoder*.

Por otro lado, en la parte del *Decoder* las características de alto nivel obtenidas previamente en la etapa *Encoder* son transformadas mediante capas de convolución, convolución transpuesta y concatenaciones a manera de generar estimaciones (capas de ESTIMATION) del movimiento. Estas estimaciones se realizan en aumentos del doble de resolución en ancho y alto, hasta llegar a la resolución de la imagen de entrada. En la Figura 4.3 se observan los elementos involucrados en el proceso *Decoder*. 1

Figura 4.3. Elementos involucrados en la parte *Decoder* de la arquitectura.

Las salidas de las capas TCONV y UPSAMPLE se obtienen por la convolución transpuesta denotadas por:

$$fmT_{k,l}(x,y,z) = Tm_{k-1}^T * Kd_{k,l} \quad (64)$$

Donde $Kd_{k,l}$ es el l -ésimo filtro de convolución transpuesta en la capa *Decoder* k y Tm_{k-1}^T es el mapa de características transpuesto en la capa $k-1$.

Por su parte, las capas de concatenación (CONCAT) o Con_m , se obtienen mediante:

$$\begin{aligned} Con_m &= (T_m) \parallel_z (C_m) \parallel_z (U_m), \quad m = \{1, 2, 3\} \\ T_m &= \{Tconv1, Tconv2, Tconv3\}, \quad C_m = \{Conv4, Conv3, Conv1\} \\ U_m &= \{Upsample1, Upsample2, Upsample3\} \end{aligned} \quad (65)$$

Donde m es el índice que indica qué capas de convolución transpuesta Tm , de convolución Cm o de muestro hacia arriba Um se emplean para hacer la concatenación $\parallel_z$. Las capas CONCAT se encargan de unir información de bajo y alto nivel de la red para conducir a una estimación más precisa del movimiento y a una convergencia más rápida en entrenamiento.

Por último, la capa de DETECTION, al igual que las capas CONV y ESTIMATION, se obtiene también mediante la operación de convolución y luego se pasa a través de una función de activación sigmoideal para conseguir la salida Det_{pe} . Det_{pe} otorga para cada pixel una probabilidad dentro del rango $[0,1]$ (siendo “0” pertenencia a fondo y “1” a objeto dinámico).

$$DETECTION(x, y, z) = G_k * K_{k,j}, \quad Det_{pe}(x, y, z) = \frac{1}{1 + e^{-DETECTION(x, y, z)}} \quad (66)$$

Previamente se definieron los procesos involucrados en la estructura de la red, pero es preciso señalar algunas diferencias entre capas. A continuación, se describe la configuración de los hiperparámetros utilizados en cada capa de la red para la parte *Encoder*. Los hiperparámetros se definen antes de comenzar el entrenamiento y se utilizan para indicar cuál será la configuración del modelo de red. Estos hiperparámetros se definieron inicialmente con base en la arquitectura *FlowNet* y después fueron modificados experimentalmente hasta obtener un entrenamiento estable y eficiente. La metodología utilizada para definir el número de capas de la red fue ir probando el desempeño del modelo y eliminar capas finales de la parte *Encoder* hasta obtener una buena generalización sin sobreentrenamiento, ya que según [179][180][181][182][183] las últimas capas poseen características de detalle de los objetos, lo cual puede producir que el modelo se sobreajuste a los datos de entrenamiento. Por otro lado, para la parte *Decoder* primero se definió el tamaño de la primera capa ESTIMATION1 como 1/8 de la resolución original de la imagen, esto para

conservar la relación espacial de los objetos en movimiento. Así, por ejemplo, objetos dinámicos pequeños con dimensiones de alrededor de 50 píxeles de área, tendrían alrededor de 6 píxeles para que aun fueran detectados en su estimación inicial. Una vez definida esta resolución inicial, las capas de convolución transpuestas, upsample y concatenaciones, se diseñaron a manera que en cada etapa se aumentara la resolución al doble por practicidad hasta obtener la resolución inicial del video.

Específicamente, se eliminaron dos capas convolucionales en la parte *Encoder* y se cambió el número de filtros de la tercera capa, por su parte la etapa *Decoder* sufrió una mayor modificación, se cambió el tamaño de filtros en todas las capas para obtener los tamaños deseados y se eliminó una capa completa de concatenación (véase referencia [13] con fines de comparación). La Tabla 4.1 muestra los hiperparámetros definidos para la parte *Encoder*, los tres parámetros resaltados en negro corresponden al ancho: W , alto: H y profundidad o número de filtros utilizados para generar dicha capa. Por su parte, Fi es el tamaño del filtro, S es la cantidad de pasos o *strides* y Act es la función de activación utilizada en cada capa. Respecto al tamaño de los filtros Fi , por lo general se utilizan dimensiones de número impar $Fi=\{7 \times 7, 5 \times 5, 3 \times 3\}$ debido a que se mantiene simetría respecto al pixel central y se evita problemas para realizar la operación de *padding*. Además de acuerdo con [184][185], los filtros de tamaño superior (Ej. 7×7) cuentan con un campo receptivo mayor que permiten capturar características globales del contexto de la imagen y los filtros de menor tamaño debido a su campo receptivo pequeño se encargan de la extracción de detalles (Ej. 3×3). Por lo que comúnmente las primeras capas de la red poseen filtros de mayor tamaño.

Tabla 4.1. Configuración de Hiperparámetros parte Encoder.

Capa	Hiperparámetros
Entrada $F(x,y,z)$ (cuadros concatenados)	$W \times H \times 6$
CONV1	$W \times H \times 64$, $Fi=7 \times 7$, $S=1$, $Act=relu$
CONV2	$W/2 \times H/2 \times 128$ $Fi=5 \times 5$, $S=2$, $Act=relu$
CONV3	$W/2 \times H/2 \times 128$ $Fi=5 \times 5$, $S=1$ $Act=relu$
CONV4	$W/4 \times H/4 \times 256$ $Fi=3 \times 3$, $S=2$, $Act=relu$
CONV5	$W/8 \times H/8 \times 512$ $Fi=3 \times 3$, $S=2$, $Act=relu$
CONV6 , Dropout(0.25)	$W/8 \times H/8 \times 512$ $Fi=3 \times 3$, $S=1$, $Act=relu$
CONV7 , Dropout(0.25)	$W/8 \times H/8 \times 512$ $Fi=3 \times 3$, $S=1$, $Act=relu$
CONV8 , Dropout(0.25)	$W/8 \times H/8 \times 512$ $Fi=3 \times 3$, $S=1$, $Act=relu$

Para una mejor comprensión e interpretación de la Tabla 4.1, se muestra la Figura 4.4 donde se ejemplifica como se genera la capa CONV1. En la figura, se observa la entrada $F(x,y,z)$ que posee dimensiones $W \times H \times 6$, después F se convoluciona con 64 filtros de tamaño $Fi=7 \times 7$ para generar 64 mapas de características, estos mapas de características se concatenan en profundidad y forman la capa CONV1 de dimensión $W \times H \times 64$. Por otro lado, en la Tabla 4.2 se muestra la configuración

de los hiperparámetros utilizados en cada capa de la red de la parte *Decoder* siguiendo la nomenclatura de la etapa *Encoder*.

Tabla 4.2. Configuración de Hiperparámetros parte Decoder.

Capa	Hiperparámetros
TCONV1	$W/4 \times H/4 \times 256$, $Fi=4 \times 4$, $S=2$, $Act=None$
TCONV2	$W/2 \times H/2 \times 128$, $Fi=4 \times 4$, $S=2$, $Act=None$
TCONV3	$W \times H \times 64$, $Fi=4 \times 4$, $S=2$, $Act=None$
CONCAT1	$W/4 \times H/4 \times 513$
CONCAT2	$W/2 \times H/2 \times 257$
CONCAT3	$W \times H \times 129$
UPSAMPLE1	$W/4 \times H/4 \times 1$, $Fi=4 \times 4$, $S=2$, $Act=None$
UPSAMPLE2	$W/2 \times H/2 \times 1$, $Fi=4 \times 4$, $S=2$, $Act=None$
UPSAMPLE3	$W \times H \times 1$, $Fi=4 \times 4$, $S=2$, $Act=None$
ESTIMATION1	$W/8 \times H/8 \times 1$, $Fi=3 \times 3$, $S=1$, $Act=None$
ESTIMATION2	$W/4 \times H/4 \times 1$, $Fi=3 \times 3$, $S=1$, $Act=None$
ESTIMATION3	$W/2 \times H/2 \times 1$, $Fi=3 \times 3$, $S=1$, $Act=None$
DETECTION	$W \times H \times 1$, $Fi=3 \times 3$, $S=1$, $Act=Sigmoid$

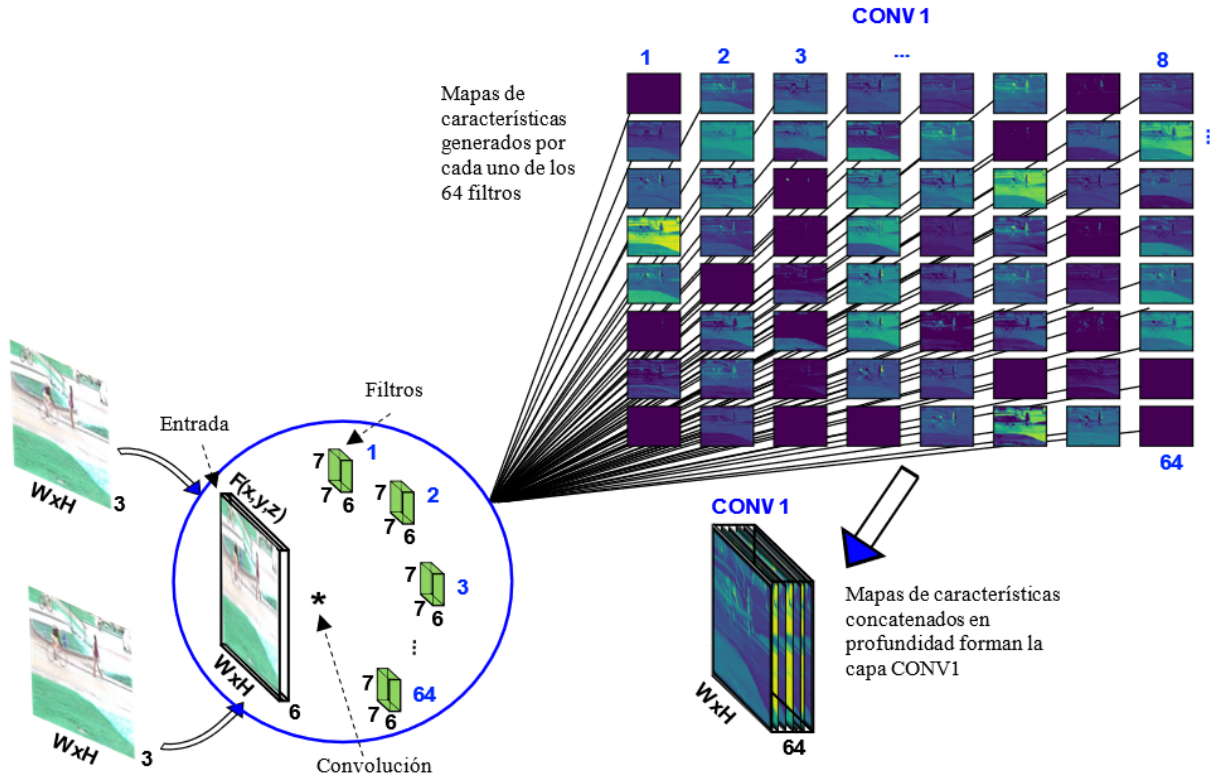


Figura 4.4. Proceso de obtención de la capa CONV1.

4.2. Esquema de entrenamiento

El entrenamiento de la red es uno de los puntos más importantes al trabajar con modelos de aprendizaje profundo, en la sección 1.4.1.1 se abordaron temas referentes a los elementos involucrados para llevar a cabo este propósito, tales como: técnicas de regularización, técnicas de optimización, funciones de pérdida, entre otros. Cada uno de estos elementos posee una

importancia para llevar a cabo un entrenamiento exitoso y obtener un modelo óptimo. Por lo que a continuación se define el proceso de entrenamiento del algoritmo propuesto 2FraCNN.

La actualización de los coeficientes We de los filtros de la red se realiza mediante el algoritmo *backpropagation* y el optimizador RMSProp [113], que divide el factor de aprendizaje η por un factor de promediado exponencial ϕ_t de los gradientes cuadrados ∇J^2 mediante:

$$\phi_t = \rho \cdot \phi_{t-1} + (1 - \rho) \cdot \nabla J^2 \quad (67)$$

$$We(t+1) = We(t) - \frac{\eta}{\sqrt{\phi_t + \varepsilon}} \cdot \nabla J \quad (68)$$

Siendo $\rho = 0.9$ y $\varepsilon = 1 \times 10^{-8}$ parámetros definidos por default, $\eta = 1 \times 10^{-4}$ es el parámetro inicial de aprendizaje definido experimentalmente tras analizar el desempeño de la red en entrenamiento, ∇J el gradiente de la función de costo de entropía cruzada binaria J y $We(t+1)$ la actualización de los coeficientes de los filtros.

Para llevar a cabo el entrenamiento, se seleccionaron 200 cuadros de *Ground truth* (GT) para cada escenario de vídeo utilizado de la base de datos *CDnet14* con sus respectivos cuadros originales $I(x,y,t)$ y $I(x,y,t-1)$. Los GT fueron seleccionados de tal manera que ambas clases de píxeles, fondo y objeto dinámico estuvieran presentes en la escena. Se utilizaron 100 épocas de entrenamiento, con una división para muestras de validación de 10% y un *batch size* igual a 1. No obstante, debido a que en tareas de clasificación binaria por lo general se tiene mayor cantidad de píxeles de fondo que de objetos en movimiento (en proporciones de hasta más de 10000:1), se realizó una penalización en la función de pérdida L , para lograr un mejor balanceo de clases dando menor ponderación a los píxeles pertenecientes al fondo y un peso mayor a los píxeles de objetos dinámicos.

La ponderación Wi_{nc} asociado a cada clase nc se obtuvo usando la distribución de los píxeles de fondo y de objetos dinámicos de cada cuadro de GT de la siguiente manera:

$$Wi_{nc} = \frac{Np}{(N_{classes}) \cdot x(Pc_{nc})}, \quad nc = \{1: obj.dinámicos, 2: fondo\} \quad (69)$$

Siendo Np el número total de píxeles en el GT, N_{clases} el número total de clases presentes nc (de fondo y objetos dinámicos) y P_{nc} es el número de píxeles en la clase nc . Esta ponderación W_{inc} se incorporó en la función de costo de entropía cruzada binaria, J_l :

$$J_l = -\frac{1}{Np} \sum_{k=1}^{Np} (W_{i_1} y_k^l \log(p_k^l) + W_{i_2} (1-y_k^l) \log(1-p_k^l)), \quad l = \{1, 2, 3, \dots, Ng\} \quad (70)$$

donde l es el índice del cuadro que se evalúa, Ng el total de cuadros de GT, W_{i_1} es el peso asociado a la clase objetos dinámicos y W_{i_2} es el peso asociado a la clase de fondo, p_k^l es el valor de probabilidad predicho de un pixel hacia objeto dinámico en la posición k del cuadro l y y_k^l es el valor real del pixel de GT en la posición k del cuadro l .

No obstante, aun y con el proceso de balanceo de clases, durante las primeras corridas algunos de los videos que representaron un mayor reto para lograr un correcto entrenamiento fueron *turbulence01* y *turbulence02* de la categoría *Turbulence* y *port_0_17fps* de la categoría *LowFramerate*. Estos videos contienen GT con una cantidad muy reducida de píxeles pertenecientes a objetos dinámicos y en consecuencia la red no lograba realizar una correcta clasificación. Por otro lado, en videos como *CopyMachine* de la categoría *Shadow* o *library* de la categoría *Thermal* cuyos GT cuenta con mayor cantidad de píxeles de objetos dinámicos, la red lograba excelentes resultados.

Teniendo en cuenta el problema anterior y con el fin de mejorar el rendimiento general del modelo propuesto, se llevó a cabo un nuevo análisis para conocer la tasa de desequilibrio de clases. El nuevo análisis comparó el rendimiento en la métrica F-Measure y la tasa promedio del número de píxeles de fondo por cada pixel de objetos dinámicos en los videos. Este proceso se realizó para seis videos diferentes en todos los cuadros GT que tenían tanto píxeles de primer plano como de fondo, lo que dio como resultado el gráfico de la Figura 4.5.

Se puede observar en la Figura 4.5 que a medida que la proporción de píxeles de fondo y de primer plano disminuye, el rendimiento en la métrica F-Measure aumenta. Por lo tanto, en el proceso de entrenamiento se incorporó un mecanismo que descartaba los cuadros de entrenamiento cuyos GT tuvieran una relación promedio superior a 100:1. Esta tasa fue seleccionada para asegurar al menos un F-Measure de alrededor de 0.7.

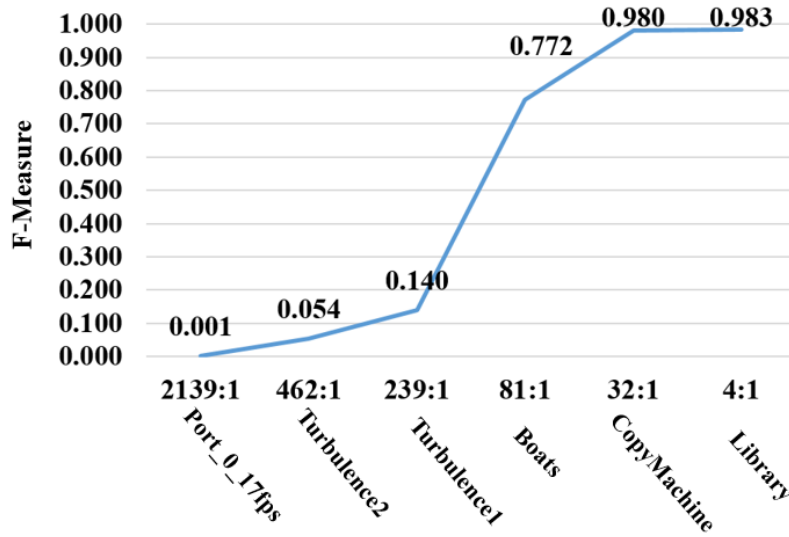


Figura 4.5. Desempeño en F-Measure contra la proporción promedio de número de píxeles de fondo por cada píxel de objeto dinámico en cada video.

Después se observó que el vídeo *ZoominZoomOut* de la categoría *PTZ* y el vídeo *port_0_17fps* de la *LowFramerate*, contenían predominantemente cuadros de GT con esa circunstancia. Sobre la base de esta observación, se decidió añadir una técnica de aumento de datos para videos con esta condición, ya que se ha demostrado que reduce el *overfitting*, el error de generalización y mejora el rendimiento de los modelos de aprendizaje profundo [136], [186], [187]. Para este trabajo, se propuso una novedosa técnica de aumento de datos a nivel de píxel, para aumentar el número de píxeles de objetos dinámicos y de este modo proporcionar una mayor cantidad de información dinámica a la red. El esquema de aumento de datos consistió en realizar copias del objeto de interés en el GT a diferentes posiciones espaciales y aplicar el mismo procedimiento a los cuadros de entrada (anterior y actual). En la Figura 4.6 se observa este procedimiento, las imágenes a), b) y c) corresponden a los cuadros anterior, actual y el GT del cuadro actual del video. Luego, se generan copias del objeto de interés para cada una de las imágenes, las copias de la imagen a) se muestran en la imagen d), las de b) en e) y las de c) en f). Las copias de los objetos de interés conservan el desplazamiento original entre cuadros, por ejemplo, el objeto señalado en un círculo rojo en la imagen a) se desplaza hacia b) y genera la imagen c). Los objetos copias de d) se desplazan hacia e) y generan la imagen de GT f).

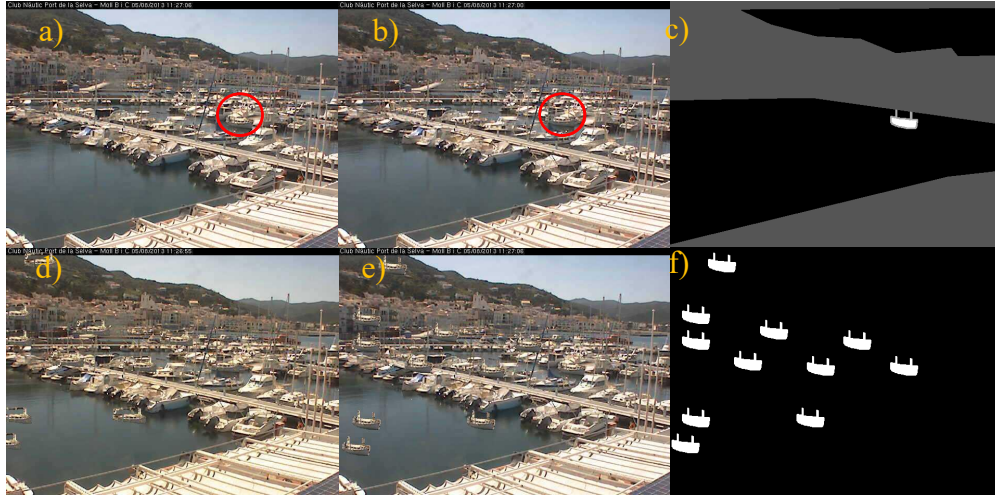


Figura 4.6. Aumento de datos realizando copias del objeto de interés. a) cuadro anterior, b) cuadro actual y c) GT original. d), e) y f) imágenes modificadas con copias.

Debido a que los GT de *Change detection* contienen zonas grises de no interés que no aportan ningún tipo de información, el movimiento en estas zonas fue suprimido repitiendo la información de la escena del cuadro actual al anterior. De esta manera, se aseguró que la zona de no interés no contuviera ningún movimiento. Además, estas zonas fueron cambiadas a color negro en el GT como parte de las zonas de fondo.

4.3. Implementación

La implementación se realizó usando python versión 3.5.2 ya que es un lenguaje de programación de código abierto y cuenta con herramientas especializadas en el manejo de aprendizaje profundo. Se utilizó la plataforma Keras versión 2.2.2 con Tensorflow backend 1.1.0, un GPU NVIDIA GTX 970 en una Workstation con 12 GB de RAM y un procesador Intel Xeon E5620 @2.40GHz. Se obtuvo una velocidad de procesamiento de 15 FPS aproximadamente para una resolución de 320x240. El espacio de color utilizado fue LAB, se tomaron dos imágenes de los cuadros originales del video (anterior y actual) y luego estas fueron concatenadas y pasadas directamente a las capas convolucionales de la red. Por otro lado, debido a que la red otorga para cada pixel una probabilidad dentro del rango $[0,1]$ (siendo “0” pertenencia a fondo y “1” a objeto dinámico), se utilizó un parámetro umbral definido experimentalmente Th para obtener la máscara de detección de red final. Después de varios experimentos con diferentes valores de umbral, el parámetro se fijó en $Th=0.8$ ya que otorgaba el mejor desempeño general para todos los vídeos. Así, un pixel pertenecerá a un objeto dinámico cuando su probabilidad sobrepase el valor Th y pertenecerá al fondo en caso contrario.

4.4. Evaluación del modelo propuesto y análisis de los resultados

La evaluación del método se realizó mediante la base de datos *CDnet14* de uso común y la metodología de evaluación de algoritmos propuesta en [12]. Por lo que, además de las medidas tradicionales para la evaluación del desempeño del algoritmo, como las métricas cuantitativas, se consideran otros criterios como, la velocidad, autoadaptabilidad, documentación y la novedad del algoritmo, que son aspectos importantes desde el punto de vista científico destinados a aplicaciones prácticas en el mundo real. La metodología [12] será referenciada con la siglas PVADN que proviene de los cinco criterios en evaluación *Performance*, *Velocity*, *Autoadaptability*, *Documentation* y *Novelty*. A continuación, se presentan los resultados obtenidos en las metodologías de evaluación *CDnet* y PVADN.

4.4.1 Evaluación en *CDnet14*

La base de datos *CDnet14* considera 53 videos distribuidos en 11 categorías: *Baseline*, *NightVideos*, *Camerajitter*, *BadWeather*, *DynamicBackground*, *Turbulence*, *LowFramerate*, *Shadow*, *Thermal*, *IntermittentObjectMotion*, y *PTZ* (*Panning Tilting Zooming*). *CDnet14* contempla 7 métricas en total (*Recall*, *Specificity*, *FPR*, *FNR*, *PWC*, *F-Measure* y *Precision*) y además permite realizar comparaciones con otros algoritmos del estado del arte. Los resultados obtenidos por 2FraCNN para todas las categorías de video de *CDnet14* se muestran en la Tabla 4.3, en la que se observan los primeros quince lugares de la base de datos. En general 2FraCNN obtuvo un F-Measure de 0.871, logrando colocarse en la séptima posición en el ranking considerando alrededor de 50 algoritmos evaluados en *CDnet14*, en el tiempo que se desarrollo esté trabajo de tesis. Los resultados en las métricas muestran un desempeño alto, en especial en las métricas *Specificity*, *FPR*, *PWC* y *F-Measure*, superando ampliamente los resultados de algoritmos como IUTIS-5, SemanticBGS, BSUV-Net, IUTIS-3, DeepBS, WisenetMD, PAWCS, entre otros. Para ampliar estos resultados, en la Tabla 4.4, se observa la comparación de desempeño respecto a la métrica F-Measure con los métodos del estado del arte mayormente conocidos FgSegNet_v2, FgSegNet_S, FgSegNet, CascadeCNN, SuBSENSE, DeepBS y PAWCS. Se puede observar que 2FraCNN en general obtuvo resultados superiores en comparación a los algoritmos DeepBS, SuBSENSE y PAWCS, y alcanza resultados competitivos respecto a los cuatro métodos de redes convolucionales FgSegNet_v2, FgSegNet_S, FgSegNet y CascadeCNN, los cuales se encuentran dentro de los primeros lugares de *CDnet14*. Además, 2FraCNN obtuvo resultados superiores a

CascadeCNN en los vídeos de Sombra, Térmico y Movimiento de Objetos Intermitentes. Las categorías de videos en las que 2FraCNN obtuvo los resultados más bajos fueron PTZ y *Nightvideos*. Esto se debe a que esas categorías presentan muchos falsos positivos en el primer plano, como luces, y vibraciones de cámara. Sin embargo, comparado con los algoritmos DeepBS, SuBSENSE y PAWCS los resultados de 2FraCNN son bastante superiores.

Tabla 4.3. Resultados del algoritmo propuesto 2FraCNN para todos los videos de la base de datos *CDnet14*.

Algoritmo	Recall promedio	Especificidad promedio	FPR promedio	FNR promedio	PWC promedio	FMeasure Promedio	Precisión promedio
FgSegNet_v2	0.9891	0.9998	0.0002	0.0109	0.0402	0.9847	0.9823
FgSegNet_S	0.9896	0.9997	0.0003	0.0104	0.0461	0.9804	0.9751
FgSegNet	0.9836	0.9998	0.0002	0.0164	0.0559	0.977	0.9758
BSPVGAN	0.9544	0.999	0.001	0.0456	0.2272	0.9501	0.9472
BSGAN	0.9476	0.9983	0.0017	0.0524	0.3281	0.9339	0.9232
CascadeCNN	0.9506	0.9968	0.0032	0.0494	0.4052	0.9209	0.8997
2FraCNN	0.8854	0.9972	0.0028	0.1146	0.5777	0.871	0.8798
IUTIS-5	0.7849	0.9948	0.0052	0.2151	1.1986	0.7717	0.8087
SemanticBGS	0.789	0.9961	0.0039	0.211	1.0722	0.7892	0.8305
BSUV-Net	0.8203	0.9946	0.0054	0.1797	1.1402	0.7868	0.8113
BSUV-net + SemanticBGS	0.8179	0.9944	0.0056	0.1821	1.1326	0.7986	0.8319
IUTIS-3	0.7779	0.994	0.006	0.2221	1.2985	0.7551	0.7875
DeepBS	0.7545	0.9905	0.0095	0.2455	1.992	0.7458	0.8332
WisenetMD	0.8179	0.9904	0.0096	0.1821	1.6136	0.7535	0.7668
PAWCS	0.7718	0.9949	0.0051	0.2282	1.1992	0.7403	0.7857

Tabla 4.4. Comparación de los desempeños de 2FraCNN en la métrica F-Measure.

	PTZ	badWeat...	baseline	cameral...	dynamic...	intermi...	lowFram...	nightVi...	shadow	thermal	turbule...	Promedio
FgSegNet_v2	0.986	0.990	0.998	0.997	0.995	0.996	0.934	0.974	0.996	0.994	0.973	0.985
FgSegNet_S	0.988	0.990	0.998	0.996	0.996	0.994	0.897	0.971	0.993	0.994	0.968	0.980
FgSegNet	0.984	0.985	0.997	0.995	0.996	0.995	0.879	0.966	0.994	0.992	0.965	0.977
CascadeCNN	0.917	0.943	0.979	0.976	0.966	0.851	0.837	0.897	0.959	0.896	0.911	0.921
2FraCNN	0.730	0.866	0.978	0.950	0.926	0.855	0.808	0.690	0.980	0.966	0.837	0.871
DeepBS	0.313	0.830	0.958	0.899	0.876	0.610	0.600	0.584	0.930	0.758	0.846	0.746
SuBSENSE	0.348	0.862	0.950	0.815	0.818	0.657	0.645	0.560	0.899	0.817	0.779	0.741
PAWCS	0.462	0.815	0.940	0.814	0.894	0.776	0.659	0.415	0.891	0.832	0.645	0.740

En la Figura 4.7, se presentan algunos ejemplos de los resultados obtenidos para los videos donde inicialmente se obtuvo una segmentación pobre y como estos mejoraron después del proceso de entrenamiento al descartar cuadros con tasa de pixeles negros sobre blancos mayor a 100:1 y aumento de datos. Además, en la Figura 4.8 se observan las segmentaciones obtenidas después de las mejoras en el entrenamiento en un video por cada una de las once categorías de *CDnet14*. Se puede apreciar que los resultados de 2FraCNN son robustos y muy similares al GT, logrando afrontar situaciones de mal clima, sombras, videos de noche, vibraciones de cámara, fondos dinámicos, turbulencia, entre otros.

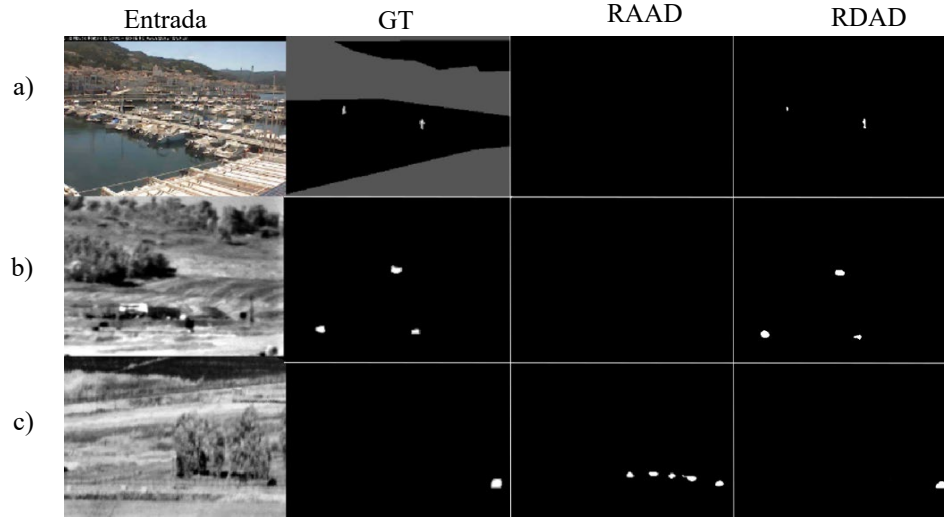


Figura 4.7. Mejoras en los resultados después de aumento de datos y descarte de cuadros. RAAD: resultados antes de aumento de datos y descarte de cuadros, RDAD: resultados después de aumento de datos y descarte de cuadros, videos: a) port_0_17fps, categoría lowFramerate, b) turbulence1 y c) turbulence2, categoría turbulence.

Otras ventajas de 2FraCNN sobre algoritmos que emplean redes convolucionales FgSegNet_v2, FgSegNet_S, FgSegNet, CasadeCNN, BSUV y DeepBS son las siguientes. DeepBS y BSUV utilizan una imagen de referencia o fondo en la entrada de la red. Por lo tanto, estos métodos dependen en gran medida de lo bueno que sea el fondo generado por algún otro algoritmo de sustracción de fondo. Además, DeepBS utiliza un procesamiento por parches, donde por cada parche o ventana que se localiza sobre la imagen de entrada se obtiene una clasificación para el pixel central del parche, si este pertenece o no a un objeto dinámico. Sin embargo, el uso de parches de tamaños fijos puede causar la pérdida de información del contexto de la imagen, especialmente con objetos de interés que son más grandes que el parche. Además, el procesamiento por parches es computacionalmente demandante. Por el contrario, 2FraCNN no requiere de algoritmos de generación de fondo y sólo necesita dos cuadros concatenados en una estructura Codificador-Decodificador. Así, se considera todo el contexto de la escena para obtener la segmentación del primer plano de forma eficiente. Las arquitecturas de FgSegNet_v2, FgSegNet_S, FgSegNet y CasadeCNN emplean una disposición en cascada en la que se fusiona la información de una sola imagen (cuadro actual del vídeo) en diferentes resoluciones espaciales, pero no se considera la información temporal. En cambio, 2FraCNN trabaja con la concatenación de dos cuadros sucesivos, lo que proporciona pistas de información temporal a la red y permite el diseño de una arquitectura más simple que no requiere transformaciones de escala. El tiempo de procesamiento de 2FraCNN es de 15 FPS a una resolución de video de 320x240, que es más rápido que los 12,5

FPS de CascadeCNN. Otra superioridad de 2FraCNN sobre los otros métodos es que, debido al esquema de entrenamiento realizado, 2FraCNN no depende del aprendizaje por transferencia o *transfer learning* para cumplir la tarea de detección de objetos en movimiento.

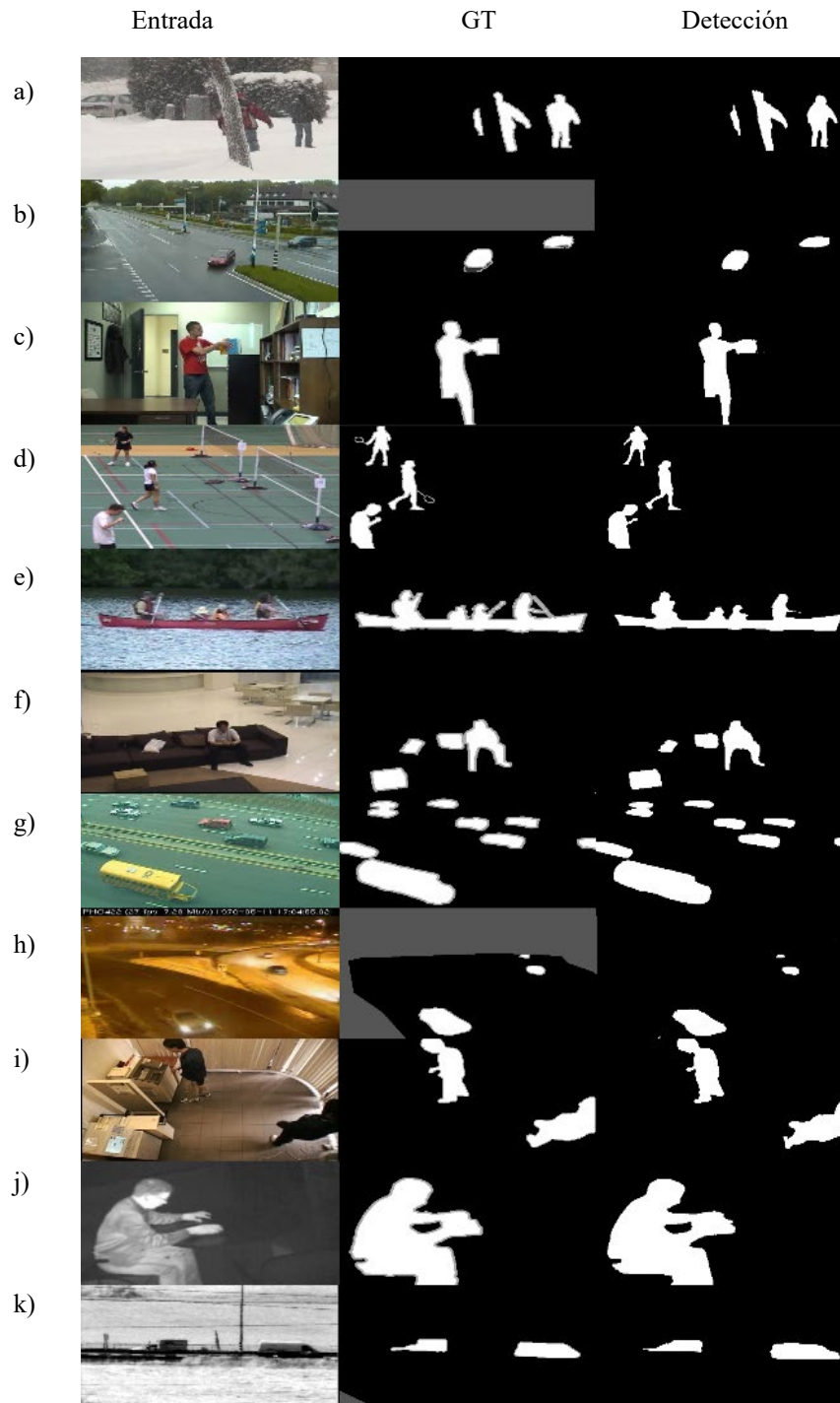


Figura 4.8. Resultados de segmentación de 2FraCNN en los videos, a) skating, b) twoPositionPTZCam, c) office, d) badminton, e) canoe, f) sofa, g) turnpike_0_5fps, h) winterStreet, i) copyMachine, j) library y k) Turbulence3.

4.4.2 Evaluación en PVADN

Además de realizar la evaluación de 2FraCNN en la base de datos *CDnet14*, se utilizó el instrumento de evaluación PVADN [12], para realizar una evaluación más completa. El instrumento PVADN considera aspectos relacionados a la documentación del método, velocidad, autoadaptabilidad, desempeño y novedad de la propuesta. Esta metodología de evaluación analiza la información y características de los algoritmos y de tres bases de datos (*CDnet14*, *CDnet12* y *BMC*) para otorgar una puntuación formulada matemáticamente a manera de realizar comparaciones entre algoritmos más justas. De los criterios considerados por PVADN la documentación evalúa si hay suficiente información para reproducir o implementar el algoritmo. La autoadaptabilidad evalúa la capacidad del algoritmo para autoadaptarse o ajustar sus parámetros a los cambios ocurridos en la secuencia de vídeo, sin necesidad de entrenamiento o intervención humana para procesar nuevos vídeos. El desempeño es medido respecto al número de bases de datos utilizadas para evaluar el algoritmo y la clasificación obtenida en las mismas. El criterio velocidad evalúa y resalta los algoritmos que pueden procesar vídeos a una velocidad de al menos 25 FPS. Finalmente, el criterio novedad evalúa la fecha de publicación del algoritmo otorgando mayor puntuación a los métodos más recientes.

La Tabla 4.5 muestra los resultados obtenidos en PVADN mostrando los primeros veinte lugares en el ranking y la Figura 4.9 es usada para una mejor visualización de las puntuaciones obtenidas en cada uno de los criterios en evaluación. Se puede observar que los primeros lugares pertenecen a algoritmos autoadaptivos como SuBSENSE, AAPSA y PAWCS. Sin embargo, estos métodos poseen una menor puntuación en desempeño en comparación con las técnicas supervisadas. 2FraCNN no logra colocarse dentro las primeras posiciones, debido mayormente a que es un método supervisado por lo que obtiene baja puntuación en autoadaptabilidad y además su procesamiento no alcanza una velocidad en tiempo real. Hay que señalar que ninguno de los métodos supervisados logra buenos resultados en estos criterios porque no son modelos adaptativos, sus parámetros no cambian a través del tiempo, y su tiempo de procesamiento es muy exigente. Sin embargo, debido al alto desempeño de 2FraCNN en la base de datos *CDnet14*, su novedad y documentación, logra situarse en el lugar 12 del ranking, la cual es una posición buena considerando que en PVADN se comparan 119 algoritmos en evaluación. Por lo que 2FraCNN,

demuestra ser un algoritmo competente en una evaluación más completa que considera aspectos prácticos y teóricos.

Tabla 4.5. Resultados de 2FraCNN en el instrumento de evaluación PVADN. Los valores en corchetes son el intervalo de valores de la métrica.

Ranking	Algoritmo	Documentación [0-15]	Autoadaptabilidad [0-20]	Velocidad [0-25]	Novedad [0-10]	CD2012-P [0-15]	CD2014-P [0-20]	BMC-P [0-10]	Desempeño [0-30]	Puntuación final [0-100]	Código disponible
1	SuBSENSE	10.97	17.95	1.06	10.00	14.70	17.37	0.00	17.37	57.36	✓
2	AAPSA	11.74	19.50	0.00	10.00	0.00	14.65	0.00	14.65	55.89	✓
3	PAWCS	9.26	17.70	0.77	10.00	14.85	17.73	0.00	17.73	55.46	✓
4	WeSamBE	10.97	16.53	0.00	10.00	0.00	17.55	0.00	17.55	55.05	✗
5	SBM	9.86	16.53	4.63	10.00	13.98	0.00	0.00	13.98	55.00	✗
6	SharedModel	10.11	15.90	1.79	10.00	0.00	17.03	0.00	17.03	54.84	✗
7	FgSegNet_v2	15.00	8.33	0.00	10.00	0.00	20.00	0.00	20.00	53.33	✓
8	FgSegNet	15.00	8.33	0.00	10.00	0.00	19.60	0.00	19.60	52.93	✓
9	Cascade CNN	11.40	11.93	0.00	10.00	0.00	19.02	0.00	19.02	52.34	✓
10	WisenetMD	10.54	11.00	2.74	10.00	0.00	17.91	0.00	17.91	52.19	✗
11	BSGAN	10.97	11.93	0.00	10.00	0.00	19.21	0.00	19.21	52.11	✗
12	2FraCNN	10.97	11.93	0.00	10.00	0.00	18.83	0.00	18.83	51.73	✗
13	(MBS V0)	10.54	15.03	0.00	10.00	14.12	15.71	0.00	15.71	51.28	✗
14	DeepBS (supervised method)	11.57	11.55	0.00	10.00	0.00	18.09	0.00	18.09	51.21	✗
15	GMM KaewTraKulPong	3.09	12.68	14.76	0.00	11.32	0.00	9.32	20.64	51.16	✓
16	CTU	6.94	11.55	22.50	10.00	0.00	0.00	0.00	0.00	50.99	✗
17	STBM	9.86	16.53	0.00	10.00	14.55	0.00	0.00	14.55	50.94	✗
18	AMBER	0.00	0.00	25.00	10.00	0.00	14.94	0.00	14.94	49.94	✗
19	IUTIS-5	8.83	11.55	0.00	10.00	0.00	18.64	0.00	18.64	49.02	✗
20	IUTIS-3	8.83	11.55	0.00	10.00	0.00	18.27	0.00	18.27	48.65	✗

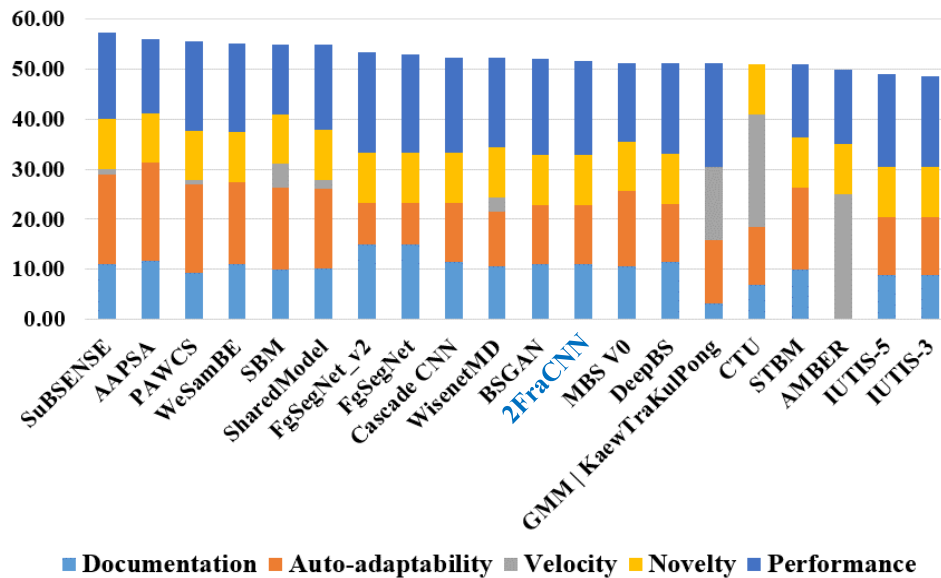


Figura 4.9. Resultados de 2FraCNN en PVADN.

4.5. Conclusiones del método 2FraCNN

En este capítulo fue presentado el algoritmo 2FraCNN para realizar la detección de movimiento adaptando y mejorando la arquitectura de la red *FlowNet* utilizada en las tareas de estimación de

flujo óptico. Entre las ventajas y diferencias de 2FraCNN se encuentran: la cantidad y tipo de información de entrenamiento, un menor número de capas, un menor tiempo de entrenamiento y la reducción de la posibilidad de sobreajuste. La entrada a la red utilizada por 2FraCNN es la concatenación de dos cuadros sucesivos, lo que proporciona pistas de información temporal a la red en una arquitectura simple, que no requiere transformaciones de escala o algoritmos generadores de fondo. Además, como en los GT de vídeo de CDnet14 había situaciones en las que existe una cantidad reducida de píxeles en primer plano, se propuso un esquema novedoso de aumento de datos en el entrenamiento y balanceo de clases. Gracias a este esquema de entrenamiento 2FraCNN no depende de la transferencia de aprendizaje de otras redes para lograr un aprendizaje correcto y se entrena desde cero con sólo 200 cuadros del video.

Los resultados experimentales demostraron que el algoritmo propuesto por 2FraCNN obtuvo resultados competentes en todas las categorías de vídeo de la base de datos de *CDnet14*, situándose en la séptima posición del ranking. Los mejores resultados se obtuvieron en las categorías de vídeo *Baseline*, *CameraJitter*, *DynamicBackground*, *Shadow* y *Thermal* con una F-Measure superior a 0,926, que son altas y competitivas en comparación con los primeros puestos de *CDnet14*. Además, 2FraCNN se clasificó en el 12º lugar en el instrumento de evaluación PVADN que considera los aspectos de desempeño, velocidad, autoadaptabilidad, documentación y novedad. Por lo tanto, 2FraCNN demostró su robustez al ser evaluado contra 119 algoritmos con una metodología que considera tanto los aspectos prácticos como teóricos y es capaz de resolver situaciones críticas en los videos como fondos dinámicos, ruido en la escena debido al mal tiempo, sombras, objetos intermitentes, entre otros.

CAPÍTULO V. ALGORITMO DE DETECCIÓN DE MOVIMIENTO BASADO EN LA VISIÓN BIOLÓGICA DE LAS AVES RAPACES (DeepFoveaNet)

En el capítulo anterior se describió el modelo 2FraCNN que utiliza la información de dos cuadros sucesivos para realizar detección de movimiento. Este algoritmo mostró resultados robustos en una gran cantidad de escenarios. Sin embargo, uno de sus principales problemas fue llegar a un correcto aprendizaje o convergencia en el entrenamiento cuando se tenían videos con objetos dinámicos pequeños. Además, en los videos con estas situaciones cuando se llegaba a una correcta convergencia, se observó que en algunas ocasiones las detecciones de objetos dinámicos pequeños tendían a ser intermitentes. Con el fin de resolver esta situación, se realizó un análisis de modelos de redes neuronales artificiales basados en la visión biológica con el propósito de aprovechar algunas de las características presentes en la naturaleza y ayudar a obtener un algoritmo más robusto.

En la revisión de la literatura en el capítulo III se presentaron algunos algoritmos que emulan la percepción biológica de movimiento mediante redes neuronales [33], [34], [169]–[173]. Estos trabajos enfocaron sus esfuerzos al estudio de la percepción de movimiento de la visión humana y de animales como la mosca o langosta. Gracias a esto, surgió el interés por buscar otros mecanismos, adaptaciones o estructuras de la visión biológica en animales. En particular, se buscó algún tipo de estructura que permitiera mejorar la detección de los objetos dinámicos pequeños. De esta búsqueda se encontró que la visión biológica de las aves rapaces es considerada dentro del grupo de animales con el mayor grado de agudeza visual, superando hasta por 5 veces la agudeza visual humana. Las aves rapaces cuentan con dos tipos de fovea, una superficial utilizada en la visión binocular y una profunda utilizada en la visión monocular que les permite ver objetivos a largas distancias para identificar posibles presas, esto es clave para la detección de objetos dinámicos pequeños. Hay que señalar que ninguno de los trabajos encontrados en la literatura se inspira o explora la estructura peculiar de la visión de las aves rapaces para sacar provecho de las ventajas que ofrece su agudeza visual, además solo en el trabajo [173] se ha evaluado en una base de datos completa, los demás utilizan evaluaciones parciales y bases de datos propias.

Por lo anterior, inspirados en el funcionamiento de la visión biológica de las aves rapaces se propone DeepFoveaNet, un modelo de redes convolucionales para detección de objetos dinámicos en video que se basa en el funcionamiento de la fovea profunda. DeepFoveaNet emula la visión

monocular de las aves rapaces mediante dos módulos de redes convolucionales *Encoder-Decoder*, que combinan la capacidad de magnificación de la fovea profunda y la información del contexto contenida en la periferia. Gracias a esto, DeepFoveaNet permite detectar incluso los objetos dinámicos más pequeños.

5.1. Visión biológica de las aves rapaces

En los últimos años se ha confirmado que las aves rapaces especialmente águilas y halcones están dotadas de una agudeza visual considerablemente superior a la humana, incluso hay estudios sobre el águila audaz que señalan que esta posee una agudeza visual hasta 4 o 5 veces superior a la del humano en notación de Snellen [188][189] [190][191]. Esta agudeza visual es posible por tres principales adaptaciones. En primer lugar, cuentan con un número denso de fotorreceptores, pueden llegar a tener al alrededor de $1,000,000/\text{mm}^2$, en comparación al número de fotorreceptores del ojo humano que es de $200,000/\text{mm}^2$, esto les proporciona una capacidad de resolución superior a la nuestra [192] [193]. En segundo lugar, su pupila es considerablemente mayor a la humana con un diámetro de 6 mm, lo que reduce considerablemente el efecto de difracción [194]. Además, como se muestra en la imagen de la Figura 5.1 su ojo es tubular y largo en comparación a su cabeza proveyéndole una amplificación importante de la imagen. El ojo de estas aves es casi inmóvil, por lo que, en lugar de girar los ojos para enfocar, giran la cabeza.

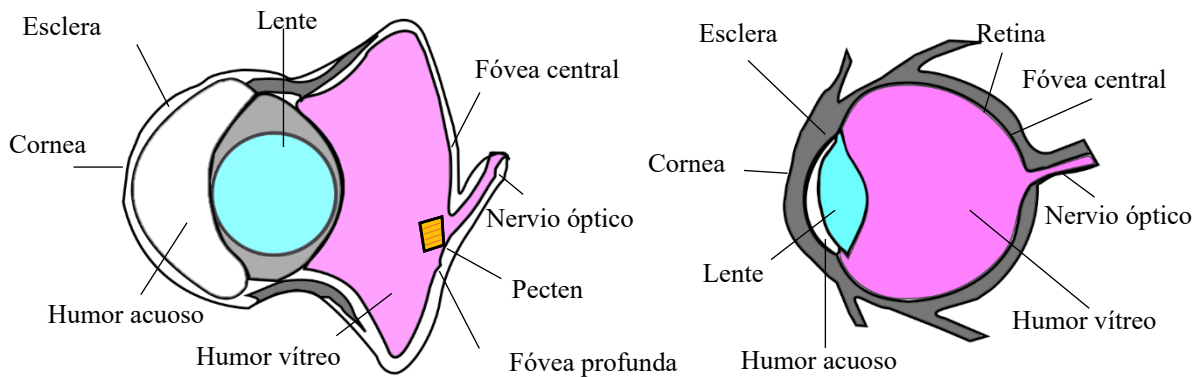


Figura 5.1. Comparación de la estructura de un ojo de águila contra un ojo humano.

De la Figura 5.1 se puede observar que al igual que la de otros vertebrados, la córnea de las aves es transparente y constituye la porción anterior de la capa fibrosa del glóbulo ocular. Los ojos tubulares, como los del águila, tienen una córnea fuertemente curvada con una superficie relativamente pequeña. La córnea funciona como soporte del contenido intraocular, refracta la luz por su curvatura y transmite la luz por su transparencia. Además, otra de las adaptaciones en el ojo

de las aves es una estructura llamada pecten, la cual provee los nutrientes y la oxigenación necesaria al ojo. El pecten es una estructura altamente pigmentada y de forma variable, que se extiende desde el nervio óptico hasta el humor vítreo. En el ojo humano, los vasos sanguíneos se disponen en frente de la retina, lo que evita el paso directo de los rayos de luz y oscurece parcialmente la imagen. Sin embargo, en las aves rapaces, el pecten resuelve este problema, permite el paso de luz directa a través del cristalino y dota al ave de una visión extremadamente aguda. La pigmentación del pecten se cree que protege los vasos sanguíneos contra los daños de la luz ultravioleta. Por último, hay que añadir que estas aves poseen dos foveas [195] [196] como se observa en la Figura 5.2 la fovea llamada temporal o poco profunda (*shallow fovea*) que se enfoca principalmente en la visión binocular y ayuda a obtener una imagen de presas cercanas y la fovea profunda o *deep fovea* que cuenta con una alta densidad de fotorreceptores y una forma convexa que dota al animal de un mecanismo de magnificación o acercamiento para observar objetos a la distancia [197] [198].

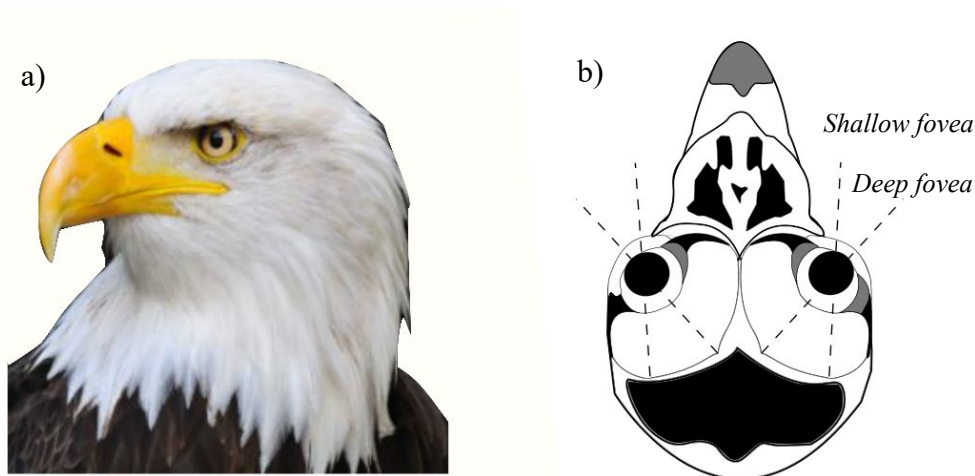


Figura 5.2. Estructura de doble fovea en el ojo del águila. (a) águila calva. (b) vista superior de la cabeza seccionada del águila para mostrar ubicación de ambas foveas.

En la imagen de la Figura 5.3 se muestran la estructura interna de una fovea profunda (*deep fovea*) y una poco profunda (*shallow fovea*). Las imágenes muestran secciones transversales de la fovea profunda de un ave Sand martin europea (izquierda) y una poco profunda de un águila dorada (derecha). Se puede observar en los *brackets* azules que en la fovea poco profunda existe un área menos densa de células ganglionares GCL (*bracket* azul), en comparación a la fovea profunda donde además existe una densa capa de fotorreceptores (PRS). Según [190][197][198] la forma convexa de la fovea profunda logra una expansión de la imagen en su camino por los tejidos de la retina, como se observa en la Figura 5.4.

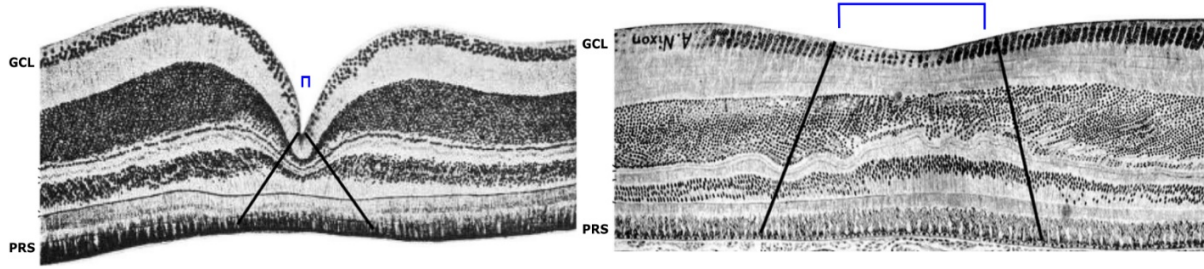


Figura 5.3. Imágenes histológicas de foveas seccionadas. Izquierda: fovea profunda de un Sand martin europeo. Derecha: fovea poco profunda de un águila dorada. Imagen de [192].

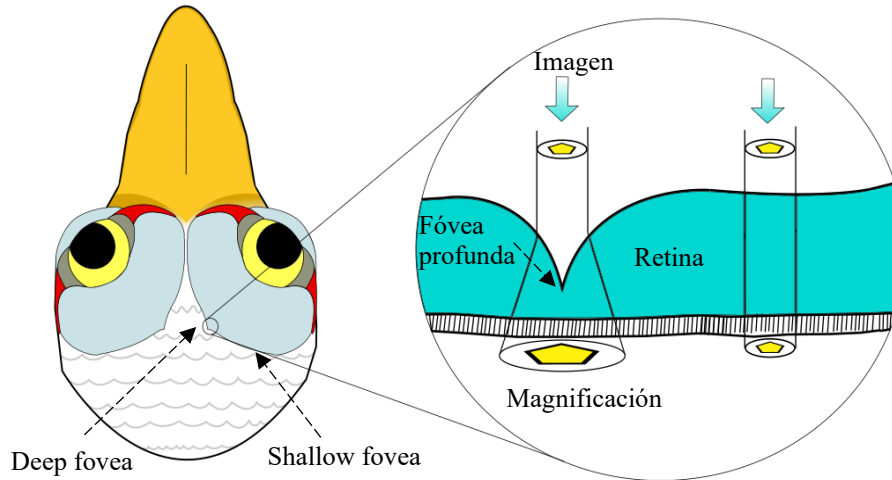


Figura 5.4. Vista superior del sistema de visión del águila calva haciendo énfasis en la estructura del ojo y la magnificación de imagen producida en la fovea profunda.

El efecto de magnificación es relacionado a los lentes de un telefoto [197], como se observa en la siguiente Figura 5.5, el lente negativo del telefoto es remplazado por la superficie convexa de la fovea profunda con radio R_a .

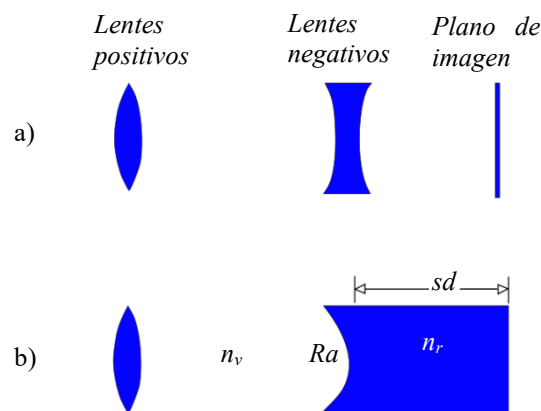


Figura 5.5. Comparación de los lentes de una telefoto contra la estructura de la fovea profunda de un ave rapaz. a) lentes de una telefoto. b) El lente negativo es remplazado por la superficie convexa de la fovea profunda [197].

La relación de magnificación mag se obtiene por medio de la siguiente expresión:

$$mag = 1 + \frac{sd}{Ra} \left(\frac{n_r - n_v}{n_v} \right) \quad (71)$$

Donde n_v y n_r son índices refractivos del humor vítreo y de la superficie de la fovea profunda respectivamente, Ra el radio de la curvatura de la superficie y sd la distancia de la cúspide a la curvatura de la fovea profunda.

Usando como ejemplo la fovea profunda de un halcón de lomo rojo (*red-backed hawk*), con $sd=110 \mu\text{m}$, $Ra=6 \mu\text{m}$, asumiendo un índice refractivo de humor vítreo $n_v=1.336$ y un índice refractivo de la fovea profunda de $n_r=1.369$, se tendría una magnificación de $mag=1.45$. Sin embargo, el índice refractivo, el tamaño del ojo y la profundidad de la fovea profunda varia de una especie a otra.

De acuerdo con lo anterior, es evidente que las aves rapaces cuentan con un sistema visual superior al de un humano, en especial por la forma de su fovea profunda utilizada en su visión monocular que les permite ver objetos a la distancia. Por lo cual, inspirados en esta característica se planteó el desarrollo de una nueva propuesta de algoritmo de detección de movimiento que fuera robusto a varias situaciones críticas en video y con un buen desempeño para la detección de objetos dinámicos, incluso los pequeños.

5.2. Algoritmo bioinspirado en la visión biológica de las aves rapaces

El esquema del modelo inspirado en la visión monocular de las aves rapaces se muestra en la Figura 5.6, la idea principal es emular la capacidad de magnificación producida por la fovea profunda en las aves y combinar esta información con la detección de la periferia del ojo. Por lo que, el primer elemento será una red que detecta el contexto o el movimiento general de la escena, emulando la visión periférica llamada *Peripheral-CNN* y el segundo módulo es una red llamada *Deepfovea-CNN*, que detecta el movimiento en el lugar de interés o de atención visual donde es colocada la fovea profunda. La ubicación de la atención visual se define inicialmente de manera aleatoria y después se realiza un proceso automático considerando la localización del objeto más pequeño, ya que es en esa zona donde se requiere mayor atención para no perder a ese objeto y los de su alrededor. La combinación de ambas detecciones de redes será la salida final del sistema. A continuación, en las siguientes secciones se detalla el algoritmo.

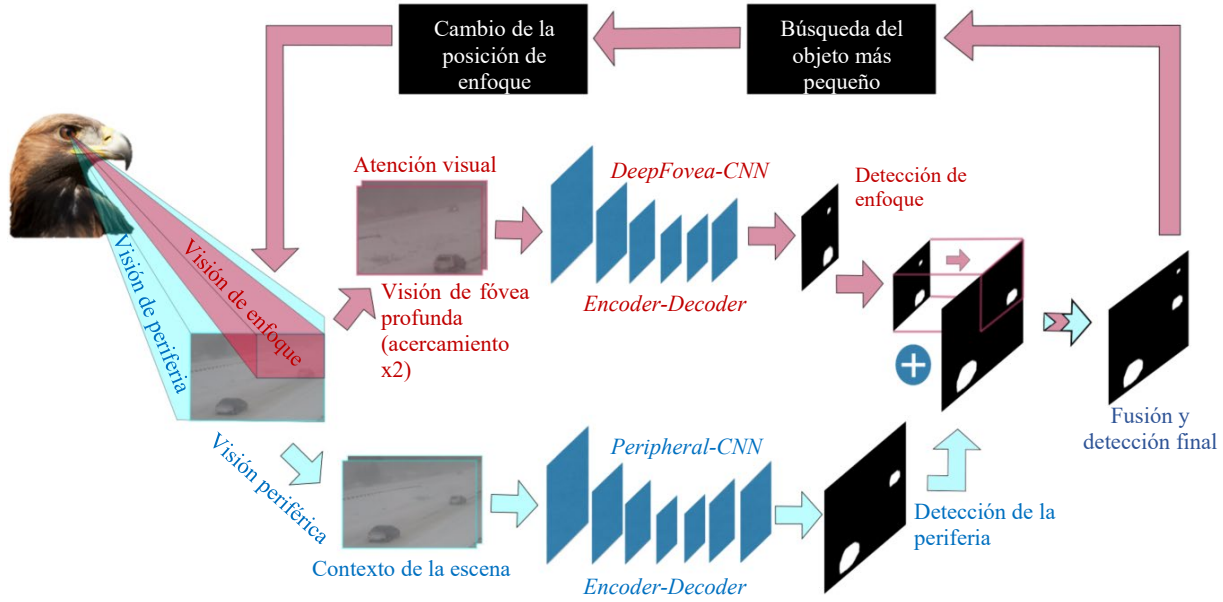


Figura 5.6. Esquema general del algoritmo propuesto basado en la visión biológica de las aves rapaces.

5.2.1. Red convolucional de periferia/Arquitectura (*Peripheral-CNN*)

La red de periferia o *Peripheral-CNN* recibe los cuadros del video normalizados a una resolución de 320x240 en el espacio de color LAB. Se toman los últimos dos cuadros consecutivos del video $I(x,y,z,t)$ y $I(x,y,z,t-1)$, siendo x,y,z coordenadas espaciales de ancho, alto y profundidad (canales L^*,a^*,b^*), y t corresponde a la variable temporal. Se utilizan dos cuadros consecutivos del video, ya que de acuerdo a [199] estimar movimiento o flujo óptico puede ser considerado como un caso especial de estimación de disparidad donde se tienen dos imágenes desplazadas en el tiempo, en lugar del espacio. Además, al igual que el algoritmo *FlowNet* [13] se espera que al proporcionarle a la red información temporal del cambio de posición de los píxeles en el tiempo, la red aprenda a extraer el patrón de movimiento por sí misma.

La arquitectura utilizada para emular la periferia del ojo y su entrenamiento son iguales a los propuestos en el capítulo IV del algoritmo 2FraCNN. Se usan 200 cuadros de entrenamiento, el esquema propuesto de aumento de datos (en los videos con objetos dinámicos pequeños; *ZoominZoomOut* y *port_0_17fps*) y balanceo de clases, el optimizador RMSprop y Crossentropía binaria como función de pérdida (véase sección 4.2). La única diferencia es que la red de *Peripheral-CNN* se entrenó para trabajar a una resolución de 320x240, puesto que, no fue posible realizar el entrenamiento de ambas redes (periferia y fovea profunda) con resoluciones mayores a 320x240 debido a limitaciones de hardware. Sin embargo, esto permite mejorar los tiempos de

procesamiento del algoritmo en la etapa de propagación hacia adelante y no presenta una desventaja en los desempeños obtenidos por el algoritmo.

La capa de DETECTION de la red de periferia, como se describió previamente en la sección 4.2 para el algoritmo 2FraCNN, se obtiene mediante la operación de convolución y luego se pasa a través de una función de activación sigmoideal para conseguir la salida Det_{Pe} que otorga un valor de probabilidad de pertenencia a objeto en movimiento.

$$\begin{aligned} \text{DETECTION}(x, y, z) &= G_k * Ke_{k,j} \\ Det_{Pe}(x, y, z) &= \frac{1}{1 + e^{-\text{DETECTION}(x, y, z)}} \end{aligned} \quad (72)$$

5.2.2. Red convolucional de fovea profunda/Arquitectura (*Deepfovea-CNN*)

Para la red de fovea profunda se utilizan los cuadros $I(x, y, z, t)$ y $I(x, y, z, t-1)$ y se dividen en cuatro cuadrantes C ; arriba izquierda (C_1), arriba derecha (C_2), abajo izquierda (C_3) y abajo derecha (C_4). Cada cuadrante es un posible lugar para colocar el enfoque de la fovea profunda. El efecto de acercamiento sobre el cuadrante de la imagen emula el enfoque producido por la fovea profunda de la visión biológica de las aves rapaces y a su vez imita el comportamiento del ojo donde se presta mayor atención y se observa con mejor detalle la zona hacia donde se dirige el interés de la visión o enfoque. En la Figura 5.7 se observa una imagen de la cual se seleccionó el cuadrante C_2 , el cual luego es aumentado en resolución.

En el primer cuadro de video, el cuadrante se elige de manera aleatoria y luego este se actualiza de manera recursiva con cada cuadro de video. El procedimiento de actualización se describe en la sección 5.2.6 de implementación. Una vez que el cuadrante es seleccionado se realiza un acercamiento o *zoom* para obtener una imagen del doble de resolución, generando la imagen ZI .

$$ZI(x, y, z, t) = \text{zoom}(I(x, y, z, t), C) \quad (73)$$

Siendo $\text{zoom}()$ una función que realiza la magnificación de la imagen en el cuadrante C , mediante interpolación bicúbica. Se realiza el mismo procedimiento de *zoom* para la imagen posterior $I(x, y, t-1)$ en el cuadrante C seleccionado.

$$ZI(x, y, z, t-1) = \text{zoom}(I(x, y, t-1), C) \quad (74)$$

finalmente se hace la concatenación de $ZI(x,y,z,t)$ y $ZI(x,y,z,t-1)$ en el eje de profundidad z para obtener el tensor ZF que alimenta a la red de fovea profunda

$$ZF(x,y,z,t) = ZI(x,y,z,t) \parallel_z ZI(x,y,z,t-1) \quad (75)$$

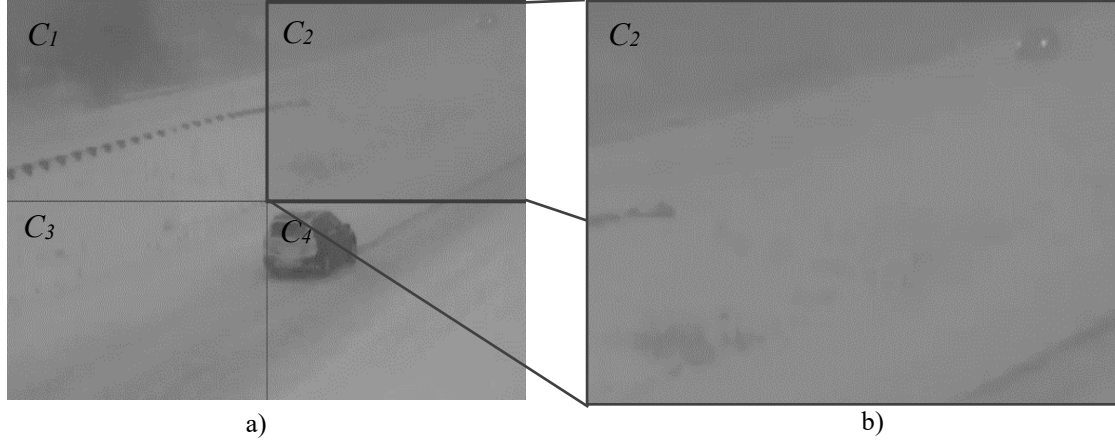


Figura 5.7. Ubicaciones posibles de atención visual. a) división de los cuadros de entrada en cuadrantes, b) acercamiento aplicado a la imagen de entrada en el cuadrante C_2 seleccionado, emulando el efecto producido por la fovea profunda de las aves rapaces.

La arquitectura *Encoder-Decoder* de la red de fovea profunda se muestra en la Figura 5.8, su estructura es idéntica a la de la red *Peripheral-CNN* removiendo la última capa en la parte *Decoder*, para obtener la salida deseada al tamaño del cuadrante seleccionado C , ya que en ese cuadrante es donde existirá un mayor detalle en la segmentación, emulando el enfoque de la visión al hacer la agregación con el resultado de la red de periferia. En la Tabla 5.1, se observa la configuración de los hiperparámetros utilizados en cada capa de la red de fovea profunda para la parte *Encoder* y en la Tabla 5.2 para la parte *Decoder*. Los hiperparámetros se conservaron con los mismos valores que para la red *Peripheral-CNN* o 2FraCNN con la excepción de que la salida de la red Det_{Fo} será la capa de ESTIMATION3, que es la mitad de la resolución original de los cuadros del video, pasada a través de la función sigmoideal.

$$ESTIMATION3(x,y,z) = G_k * Ke_{k,j}$$

$$Det_{Fo}(x,y,z) = \frac{1}{1 + e^{-ESTIMATION3(x,y,z)}} \quad (76)$$

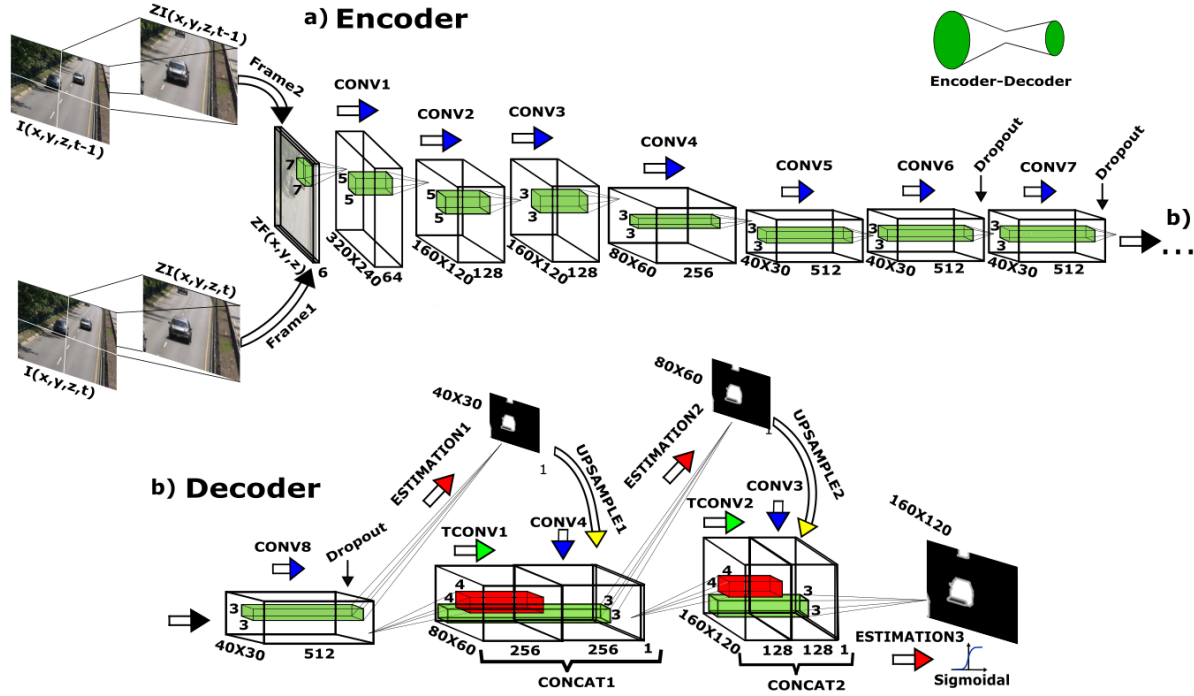


Figura 5.8. Esquema de arquitectura de la red de fovea profunda.

Las Tabla 5.1 y Tabla 5.2 conservan la nomenclatura previamente descrita en la Tabla 4.1 y Tabla 4.2, los parámetros $W=320$ y $H=240$, son el ancho y alto de los cuadros de entrada respectivamente y el tercer elemento corresponde al número de filtros. F_i corresponde al tamaño del filtro, S es la cantidad de pasos o *strides* y Act la función de activación.

Tabla 5.1. Configuración de Hiperparámetros de la red de fovea profunda en la parte Encoder.

Capa	Hiperparámetros
Entrada ZI	$W \times H \times 6$
CONV1	$W \times H \times 64$, $F_i=7 \times 7$, $S=1$, $Act=relu$
CONV2	$W/2 \times H/2 \times 128$, $F_i=5 \times 5$, $S=2$, $Act=relu$
CONV3	$W/2 \times H/2 \times 128$, $F_i=5 \times 5$, $S=1$, $Act=relu$
CONV4	$W/4 \times H/4 \times 256$, $F_i=3 \times 3$, $S=2$, $Act=relu$
CONV5	$W/8 \times H/8 \times 512$, $F_i=3 \times 3$, $S=2$, $Act=relu$
CONV6 , Dropout(0.25)	$W/8 \times H/8 \times 512$, $F_i=3 \times 3$, $S=1$, $Act=relu$
CONV7 , Dropout(0.25)	$W/8 \times H/8 \times 512$, $F_i=3 \times 3$, $S=1$, $Act=relu$
CONV8 , Dropout(0.25)	$W/8 \times H/8 \times 512$, $F_i=3 \times 3$, $S=1$, $Act=relu$

Tabla 5.2. Configuración de Hiperparámetros de la red de fovea profunda en la parte Decoder.

Capa	Hiperparámetros
TCONV1	$W/4 \times H/4 \times 256$, $F_i=4 \times 4$, $S=2$, $Act=None$
TCONV2	$W/2 \times H/2 \times 128$, $F_i=4 \times 4$, $S=2$, $Act=None$
TCONV3	$W \times H \times 64$, $F_i=4 \times 4$, $S=2$, $Act=None$
CONCAT1	$W/4 \times H/4 \times 513$
CONCAT2	$W/2 \times H/2 \times 257$
CONCAT3	$W \times H \times 129$
UPSAMPLE1	$W/4 \times H/4 \times 1$, $F_i=4 \times 4$, $S=2$, $Act=None$
UPSAMPLE2	$W/2 \times H/2 \times 1$, $F_i=4 \times 4$, $S=2$, $Act=None$
UPSAMPLE3	$W \times H \times 1$, $F_i=4 \times 4$, $S=2$, $Act=None$
ESTIMATION1	$W/8 \times H/8 \times 1$, $F_i=3 \times 3$, $S=1$, $Act=None$
ESTIMATION2	$W/4 \times H/4 \times 1$, $F_i=3 \times 3$, $S=1$, $Act=None$
ESTIMATION3	$W/2 \times H/2 \times 1$, $F_i=3 \times 3$, $S=1$, $Act=None$

5.2.3. Esquema de entrenamiento

Para realizar el entrenamiento de la red de periferia fueron seleccionados 200 cuadros de *Ground Truth* (GT_i) para cada uno de los escenarios de video de la base de datos *CDnet14* como *targets*. Los GT_i seleccionados indican cuáles cuadros originales de video formarán el tensor E , que corresponde a la concatenación del cuadro I_t e I_{t-1} , siendo t la correspondencia en el tiempo entre el GT y los cuadros I . Por lo que, el arreglo de tensores de entrenamiento utilizado será de tamaño $200 \times W \times H \times 6$ (siendo 200: número de tensores E , W : ancho de 320, H : alto de 240 y 6: profundidad de la concatenación de los canales L^*a*b^* de las dos imágenes). Por su parte, el arreglo de *targets* es de tamaño $200 \times W \times H \times 1$ (siendo 200: el número de *Ground Truth* GT_E de cada tensor E , W : ancho de 320, H : alto de 240 y 1: profundidad de la imagen binaria). Véase Figura 5.9.

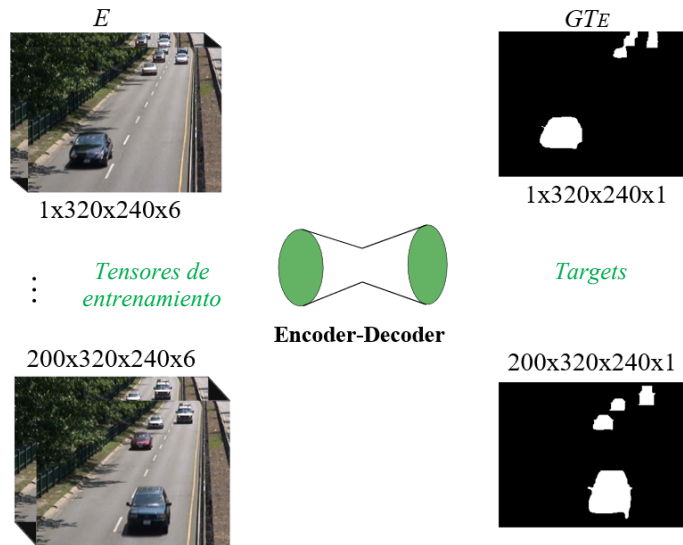


Figura 5.9. Cuadros de entrenamiento utilizados en la red de periferia.

Por otro lado, para la red de fovea profunda se realiza un proceso más complejo para obtener los tensores de entrenamiento. Primero se copian los 200 cuadros de GT_E y tensores E usados previamente en la red de periferia. El utilizar estos tensores E y cuadros GT_E permite que la red considere el contexto general y no pierda información global del video en el entrenamiento. Sin embargo, para obtener el tamaño deseado a la salida de *Deepfovea-CNN* se cambia la resolución de los cuadros GT_E o *targets*, a la mitad en ancho ($W/2$) y alto ($H/2$). Luego, para otorgarle la información de enfoque a la red *Deepfovea-CNN* en el entrenamiento, se realiza un proceso de obtención de tensores considerando los cuatro cuadrantes C . Por lo tanto, para el primer cuadrante

se obtiene el recorte de cada uno de los tensores E en el cuadrante C_1 generando 200 tensores EC_1 . Además, se obtiene el recorte de cada uno de los GT_E en el cuadrante C_1 generando los 200 *targets* o cuadros GT_{C1} . Los tensores EC_1 al ser un recorte de E poseerán la mitad de resolución ($W/2 \times H/2$), por lo tanto, se realiza un ajuste de dimensión al doble en ancho y alto para obtener los tamaños deseados, este efecto de acercamiento permite emular el efecto de atención en el ojo de las aves rapaces. El proceso anterior se repite para los demás cuadrantes C , por lo tanto, de los recortes a los tensores E en el segundo cuadrante C_2 se generan 200 tensores EC_2 y se obtienen sus respectivos *targets* GT_{C2} , y así sucesivamente para los cuadrantes restantes. Por lo tanto, el arreglo de entrenamiento para la entrada será de tamaño $1000 \times W \times H \times 6$ compuesto por cada uno de los 200 tensores E , EC_1 , EC_2 , EC_3 , y EC_4 . A su vez, el arreglo de *target* será de tamaño $1000 \times W \times H \times 6$ compuesto por cada uno de los 200 cuadros de GT_E , GT_{C1} , GT_{C2} , GT_{C3} y GT_{C4} . La Figura 5.10 ejemplifica el proceso de extracción de cuadros de entrenamiento para la red de fovea profunda.

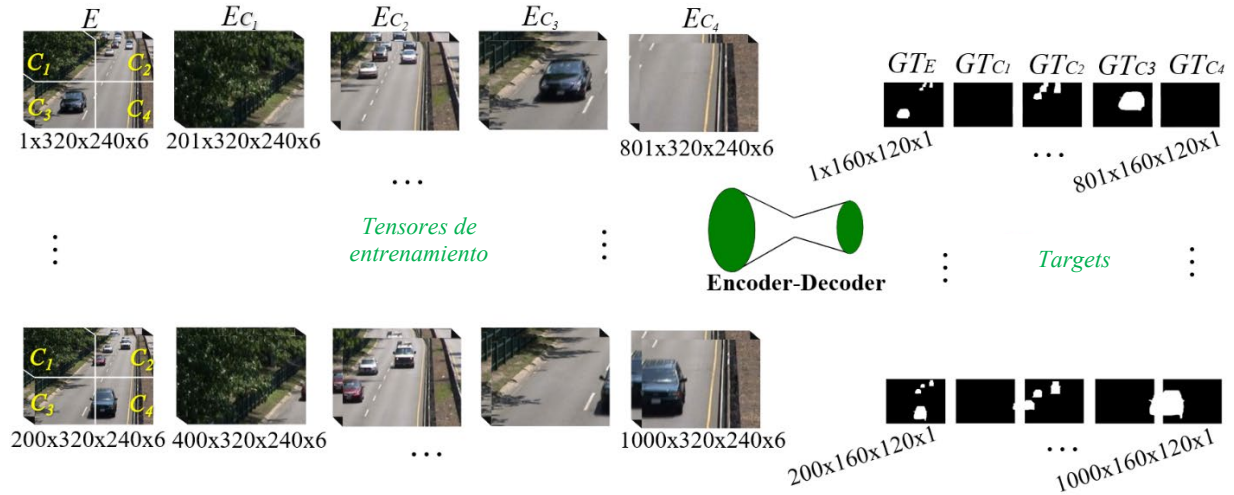


Figura 5.10. Proceso de extracción de cuadros de entrenamiento utilizados en la fovea profunda.

La selección de cuadros de GT para las dos redes, periferia y fovea profunda se realizó considerando situaciones donde hubiera ambas clases, tanto pixeles pertenecientes al fondo como pixeles de objetos dinámicos en la escena, esto para lograr un mejor balanceo de datos. Además, se aplicó el mismo esquema de entrenamiento previamente descrito en la sección 4.2 de aumento de datos y descarte de cuadros con tasa mayor a 100:1; pixeles de fondo por cada uno de objeto dinámico.

5.2.4. Etapa de agregación

Las salidas de la red de periferia $Det_{Pe}(x,y,t)$ y de fovea profunda $Det_{Fo}(x,y,t)$, una vez entrenadas, otorgan un valor de probabilidad gracias a la función sigmoideal en la capa de salida, siendo 1 la mayor pertenencia hacia objetos dinámicos. Por lo que la integración de las salidas de las redes $Det_M(x,y,z)$ se define mediante el máximo de las salidas $Det_{Fo}(x,y,t)$ y $Det_{Pe}^C(x,y,t)$ en el cuadrante C de donde se obtuvo la salida $Det_{Fo}(x,y,t)$:

$$Det_M(x,y,t) = Max(Det_{Pe}^C(x,y,t), Det_{Fo}(x,y,t)) \quad (77)$$

La clasificación final de pixeles como objetos en movimiento o fondo, $D(x,y,t)$, es la binarización de $Det_M(x,y,t)$ considerando un umbral th . Este parámetro se fijó a $th=0.8$ ya que tras realizar una serie de experimentos variando th y verificando el desempeño general del algoritmo, fue donde se obtuvo un mejor desempeño general para todos los videos de *CDnet14*.

$$D(x,y,t) = \begin{cases} 1 & \text{si } Det_M(x,y,t) \geq th \\ 0 & \text{si } \text{otro} \end{cases} \quad (78)$$

5.2.5. Reubicación del cuadrante de interés

Para reubicar el lugar de interés o de atención visual donde es colocada la fovea profunda, se utiliza la detección D para buscar el conjunto de objetos Obj que cumplan:

$$Obj = \{u \mid (u \in D) \wedge (10 \leq A_u \leq 100)\} \quad (79)$$

Siendo u un objeto que pertenece a la detección D cuya área A_u se encuentra en el rango de 10 y 100 pixeles. Este rango se definió analizando videos con objetos dinámicos pequeños y el desempeño obtenido por la red de periferia. La idea es que la segmentación obtenida por la red de fovea profunda ayude a la red de periferia cuando se le presentan objetos dinámicos pequeños. Dentro de este análisis, se observó que la red de periferia podía detectar objetos dinámicos pequeños conformados de hasta por 10 pixeles de área, sin embargo, eran fácilmente perdidos o su detección era intermitente. Además, considerando un área mayor a 10 pixeles se evita que el enfoque de la fovea profunda se aplique a zonas ruidosas ocasionadas por desenfoque de

movimiento (*blurring*). Por otro lado, objetos con un área de aproximadamente 100 píxeles eran correctamente detectados con la red de periferia por lo que no se necesitaba ubicar la fóvea en donde se tuvieran objetos de estas dimensiones.

Una vez localizados los objetos con áreas dentro de los rangos elegidos, se utilizó la función $\arg \min()$ para obtener el objeto con área menor O_M . Si existen dos o más objetos con la misma área, la función $\arg \min$ devolverá el objeto de la primera ocurrencia. Sin embargo, esta circunstancia rara vez ocurre debido al comportamiento aleatorio de los píxeles en el video y es muy difícil que dos objetos posean la misma área.

$$O_M = \arg \min(A_{Obj}) \quad (80)$$

La posición (x,y) del objeto O_M se usa para identificar a que cuadrante C se debe cambiar la zona de enfoque de la fóvea profunda mediante las siguientes ecuaciones:

$$C = \begin{cases} C_1 & si & (x_{O_M} \leq W/2) \wedge (y_{O_M} \leq H/2) \\ C_2 & si & (x_{O_M} > W/2) \wedge (y_{O_M} \leq H/2) \\ C_3 & si & (x_{O_M} \leq W/2) \wedge (y_{O_M} > H/2) \\ C_4 & si & (x_{O_M} > W/2) \wedge (y_{O_M} > H/2) \end{cases} \quad (81)$$

Siendo x_{O_M} y y_{O_M} las coordenadas (x,y) del objeto O_M . El proceso de cambio de posición de enfoque se repite con cada nuevo cuadro de video.

5.2.6. Implementación del algoritmo propuesto

La implementación se realizó usando python (versión 3.5.2), la plataforma Keras (versión 2.2.2) con Tensorflow backend 1.1.0, un GPU NVIDIA GTX 1070 en una Workstation con 12 GB de RAM y un procesador Intel Xeon E5620 @2.40GHz. Se obtuvo una velocidad de procesamiento de 16 FPS aproximadamente para una resolución de 320x240. El diagrama a bloques de la implementación del algoritmo propuesto se muestra en la Figura 5.11. Este diagrama ilustra el funcionamiento del algoritmo con los modelos de red, de fóvea profunda y periferia entrenados, es decir; funcionamiento en línea. A continuación se describen los pasos involucrados en el proceso.

1. Se otorga un cuadrante C inicial en forma aleatoria, el cual funcionará como un área de enfoque.

2. Entren los dos cuadros consecutivos de video I_t e I_{t-1} .
3. Se realiza un acercamiento en ambas imágenes I_t e I_{t-1} en el área de enfoque o cuadrante C .
4. Las imágenes con acercamiento son concatenadas para formar el tensor ZF y este alimenta a la red de fovea profunda. Del mismo modo, las imágenes consecutivas I_t e I_{t-1} sin modificación son concatenadas para formar el tensor F que es pasado a través de la red de periferia.
5. Se obtiene la salida de ambas redes, de periferia Det_{Pe} y de fovea profunda Det_{Fo} y se realiza el proceso de agregación Det_M considerando el valor máximo de probabilidad
6. Luego Det_M se umbraliza y genera la salida D que es la detección final del algoritmo.
7. A continuación, se realiza la búsqueda de los objetos con un área entre los 10 y 100 pixeles y se localiza el objeto con área menor O_M .
8. Finalmente, se realiza el cambio de la localización del cuadrante C de enfoque, se vuelven a adquirir los siguientes cuadros del video I_t e I_{t-1} y se repiten los pasos del 2 al 8.

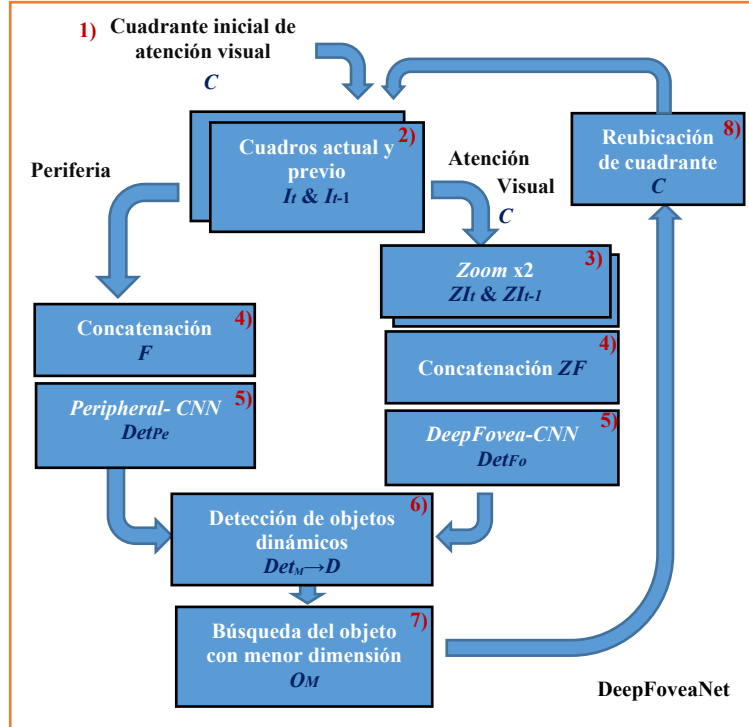


Figura 5.11. Diagrama funcional del algoritmo propuesto basado en la visión biológica de las aves rapaces.

5.3. Evaluación del modelo propuesto y análisis de los resultados

Con el propósito de realizar una evaluación robusta del algoritmo propuesto DeepFoveaNet, se utilizaron las bases de datos *CDnet14* y LASIESTA que contienen una amplia variedad de videos con diferentes situaciones críticas. En las siguientes secciones se describen y presenta los resultados obtenidos.

5.3.1. Evaluación en *CDnet14*

La evaluación del método se efectuó primeramente mediante la herramienta que otorga *CDnet14* para hacer una evaluación parcial con los 53 videos de la base datos. Esto con el fin de observar el efecto de la red fovea profunda en el algoritmo propuesto y comprobar si ayudaba a la red de periferia en la detección de objetos dinámicos, en especial aquellos que son pequeños y difíciles de detectar. En la Tabla 5.3 se muestra la evaluación sobre todas las categorías de video; *Peripheral-CNN* refiere a la red de periferia y *DeepFoveaNet* al algoritmo completo haciendo uso de la red de periferia y de fovea profunda.

Tabla 5.3. Comparación de resultados de los modelos propuestos, evaluando la red de periferia *Peripheral CNN* y el modelo completo *DeepFoveaNet* (con ambas redes, de periferia y de fovea profunda) para todos los videos de la evaluación parcial de *CDnet14*.

Video category	Recall		Specificity		FPR		FNR		PWC		Precision		FMeasure	
	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>	<i>DeepFoveaNet</i>	<i>Peripheral-CNN</i>
PTZ :	0.953	0.936	1.000	1.000	0.000	0.000	0.047	0.064	0.063	0.064	0.964	0.976	0.958	0.954
badWeather	0.892	0.860	1.000	1.000	0.000	0.000	0.108	0.140	0.154	0.197	0.988	0.991	0.931	0.913
baseline	0.973	0.957	1.000	1.000	0.001	0.000	0.027	0.043	0.150	0.146	0.989	0.999	0.981	0.977
cameraJitter	0.972	0.938	1.000	1.000	0.000	0.000	0.028	0.062	0.150	0.238	0.990	0.994	0.981	0.965
dynamicBackground	0.955	0.931	1.000	1.000	0.000	0.000	0.045	0.069	0.052	0.058	0.983	0.994	0.968	0.960
intermittentObjec...	0.860	0.818	0.998	0.998	0.002	0.002	0.140	0.182	1.179	1.572	0.888	0.889	0.838	0.807
lowFramarate	0.898	0.890	1.000	1.000	0.000	0.000	0.102	0.110	0.072	0.094	0.783	0.784	0.799	0.796
nightVideos	0.833	0.788	1.000	1.000	0.000	0.000	0.167	0.212	0.294	0.318	0.970	0.985	0.892	0.870
Shadow	0.992	0.990	1.000	1.000	0.000	0.000	0.008	0.010	0.039	0.038	0.997	0.998	0.995	0.994
Thermal	0.986	0.985	1.000	1.000	0.000	0.000	0.014	0.015	0.048	0.051	0.996	0.998	0.991	0.991
Turbulence	0.901	0.740	1.000	1.000	0.000	0.000	0.099	0.260	0.038	0.031	0.890	0.927	0.882	0.778
Promedio general	0.929	0.894	1.000	1.000	0.000	0.000	0.071	0.106	0.203	0.255	0.949	0.958	0.929	0.910

De los resultados mostrados en la Tabla 5.3 se infiere que el introducir la red de fovea profunda ayudó a mejorar los resultados generales de la red de periferia *Peripheral CNN* sobre todas las métricas en las categorías de video, disminuyendo solo un poco en la métrica precisión debido a que en ocasiones la red de fovea profunda realza al objeto de interés encerrándolo en un blob más grande que el del GT. Esto ocurre debido al aumento o *zoom* que se realiza en los cuadrantes de la imagen, este *zoom* permite que la red identifique mejor el objeto dinámico y con videos de baja calidad algunos píxeles en áreas de desenfoque o *blurring* son considerados como parte del objeto dinámico y esto impacta principalmente en la métrica precisión. Sin embargo, como se observa en la Tabla 5.3, en particular en la categoría Turbulence que cuenta con objetos dinámicos muy pequeños se mejora considerablemente el desempeño del valor de la métrica F-Measure de 0.778 a 0.882.

En la Figura 5.12, se muestra el funcionamiento del algoritmo en algunos cuadros de los videos *blizzard*, *turbulence0* y *turbulence2*, donde el recuadro azul indica el cuadrante donde se aplica la red de fovea profunda. Se puede observar que mediante la red de fovea profunda se detectan correctamente los objetos dinámicos pequeños que la red de periferia *Peripheral-CNN* pierde o detecta con dificultad en el video. Por otro lado, la categoría con menor desempeño fue *LowFramerate*, lo cual era de esperarse ya que la arquitectura de la red *DeepFoveaNet* trabaja con dos cuadros consecutivos concatenados y resulta difícil para la red estimar el movimiento en videos a baja tasa de adquisición, debido a los cambios drásticos de un cuadro a otro, teniendo incluso objetos en movimiento totalmente distintos en el cuadro I_t respecto al I_{t-1} .

Una vez comprobado que la red de fovea profunda obtuvo un correcto desempeño. Se realizó la evaluación del algoritmo *DeepFoveaNet* en la base de datos completa de *CDnet14*, que cuenta con alrededor de 50 algoritmos evaluados y catalogados con base en su desempeño. En la Tabla 5.4 se muestra el desempeño del modelo propuesto respecto a los mejores algoritmos del estado del arte enfocados a detección de movimiento en la base de datos *CDnet14*.

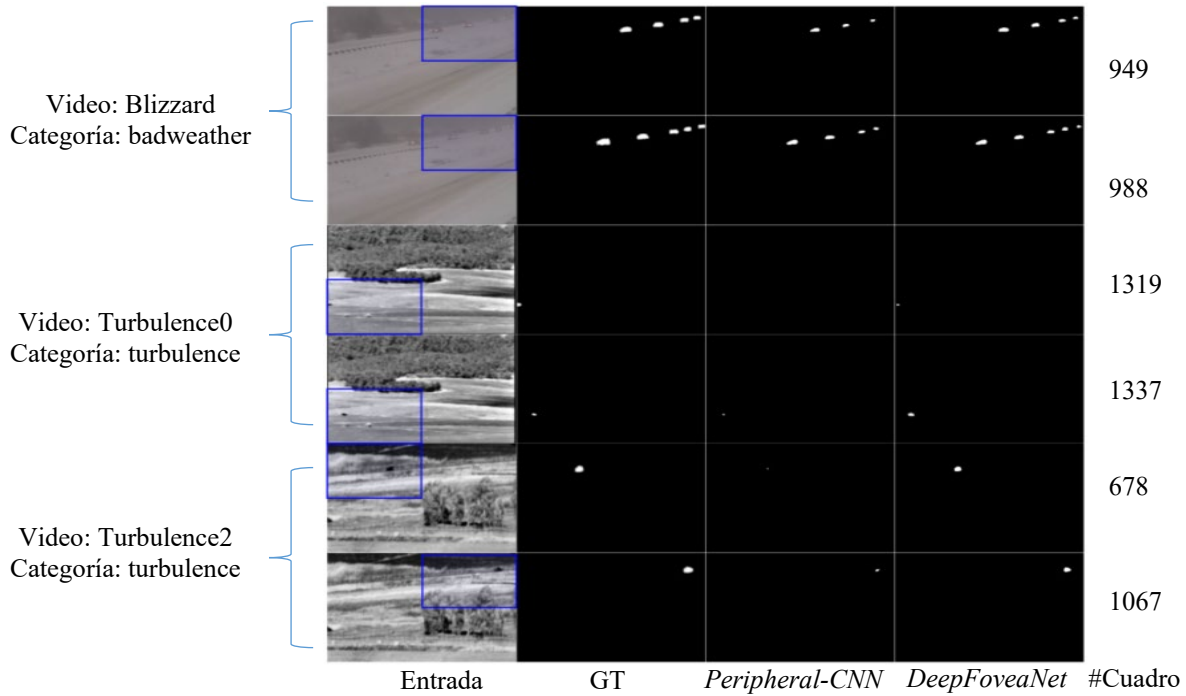


Figura 5.12. Comparación de detección de los resultados del modelo *Peripheral-CNN* (periferia) y *DeepFoveaNet* usando fovea profunda.

Tabla 5.4. Resultados del modelo propuesto sobre todos los videos de *CDnet14*.

Método	Promedio <i>Recall</i>	Promedio <i>Specificity</i>	Promedio <i>FPR</i>	Promedio <i>FNR</i>	Promedio <i>PWC</i>	Promedio <i>FMeasure</i>	Promedio Precisión
FgSegNet_v2[200]	0.989	1.000	0.000	0.011	0.040	0.985	0.982
FgSegNet_S[174]	0.990	1.000	0.000	0.010	0.046	0.980	0.975
FgSegNet[10]	0.984	1.000	0.000	0.016	0.056	0.977	0.976
BSPVGAN[201]	0.954	0.999	0.001	0.046	0.227	0.950	0.947
BSGAN[202]	0.948	0.998	0.002	0.052	0.328	0.934	0.923
CascadeCNN[8]	0.951	0.997	0.003	0.049	0.405	0.921	0.900
<i>DeepFoveaNet</i>	<i>0.793</i>	<i>0.999</i>	<i>0.001</i>	<i>0.207</i>	<i>0.599</i>	<i>0.842</i>	<i>0.936</i>
IUTIS-5[81]	0.785	0.995	0.005	0.215	1.199	0.772	0.809
SemanticBGS[203]	0.789	0.996	0.004	0.211	1.072	0.789	0.831
BSUV-Net[204]	0.820	0.995	0.005	0.179	1.140	0.787	0.811

Se puede observar en la Tabla 5.4 que los resultados en las métricas decaen un poco a los mostrados en la Tabla 5.3, esto debido principalmente a que *DeepFoveaNet* fue entrenado a una resolución 320x240 por limitaciones de hardware y para poder realizar la evaluación en la base de datos *CDnet14* se hizo un reescalamiento de los resultados a los tamaños de los videos originales mediante interpolación bicúbica. Por lo que, es de esperarse que se produzcan falsos positivos y una disminución en las métricas. No obstante, *DeepFoveaNet* demostró su robustez para manejar una gran variedad de escenarios al logra colocarse en la séptima posición en la base de datos, llegando a estar a la par con los primeros lugares e incluso superando a los algoritmos *CascadeCNN*

y BSGAN en las métricas *Precision*, *FPR* y *Specificity*. Otras ventajas de DeepFoveaNet, en comparación a los modelos de redes convolucionales del estado del arte presentados en el capítulo II, se enlistan a continuación. 1) Los algoritmos ConvNet[86], A-Net/M-Net[87] y DeepBS[88] usan un modelo de fondo de referencia a la entrada de la red y el procesamiento de ConvNet y DeepBs es basado en parches por cada pixel de la imagen. Por lo tanto, los resultados obtenidos por estos métodos dependerán de que tan bueno sea el algoritmo generador de fondos y debido al uso de parches existirá un aumento en la complejidad computacional y pérdida de la información global del contexto de la imagen. Por otro lado, DeepFoveaNet es un modelo que no depende de algoritmos generadores de fondo y trabaja directamente con la concatenación de dos cuadros sucesivos. 2) El algoritmo propuesto considera tanto el contexto general de la imagen como el lugar de atención visual o enfoque de la visión para obtener una detección más robusta. Contrario a los primeros lugares de *CDnet* FgSegNet[10], FgSegNet_S[174], FgSegNet_V2[200] y CascadeCNN[8] que solo consideran información espacial, la concatenación de los cuadros consecutivos a la entrada de la red de DeepfoveaNet facilita que aprenda representaciones espaciotemporales de la escena. 3) Debido al esquema de entrenamiento propuesto, DeepFoveaNet no depende de transferencia de aprendizaje para dar una correcta detección de movimiento, a diferencia de los modelos FgSegNet, FgSegNet_S, FgSegNet_V2 y X-NET [173].

Para demostrar la robustez de los resultados de DeepFoveaNet se realizó la evaluación cualitativa en un video por cada una de las categorías de la base de datos *CDnet14* con los métodos FgSegNet_v2, BSUV-Net, CascadeCNN y IUTIS-5, el primero, último y los dos más próximos a los resultados de DeepFoveaNet mostrados en la Tabla 5.4. Véase Figura 5.13.

De la Figura 5.13, se observa que los resultados de DeepFoveaNet son muy similares a los primeros lugares de *CDnet14*, FgSegNet y CascadeCNN, pero DeepFoveaNet presenta una menor dilatación que impacta en un mejor realce de la figura o forma del objeto de interés. Lo anterior se puede observar mayormente en los videos BW, LF, e IO. En BW se puede apreciar la separación de las piernas en uno de los sujetos patinando, en LF se observa a detalle el mástil del barco y en el video IO se puede identificar claramente a la persona dejando un objeto. Por otro lado, FgSegNet y CascadeCNN pierden un poco la forma o muestran una mayor dilatación e incluso CascadeCNN en algunos videos como NV, PT y LF presenta una segmentación más ruidosa. Sin embargo, como el ruido cae dentro de zonas de no interés, no es considerado en las métricas. Por otro lado, en

contraste con los métodos IUTIS-5 y BSUV-Net, DeepFoveaNet presenta claramente resultados más robustos, logrando incluso detectar los objetos dinámicos pequeños en el video TU que IUTIS-5 y BSUV-Net pierden.

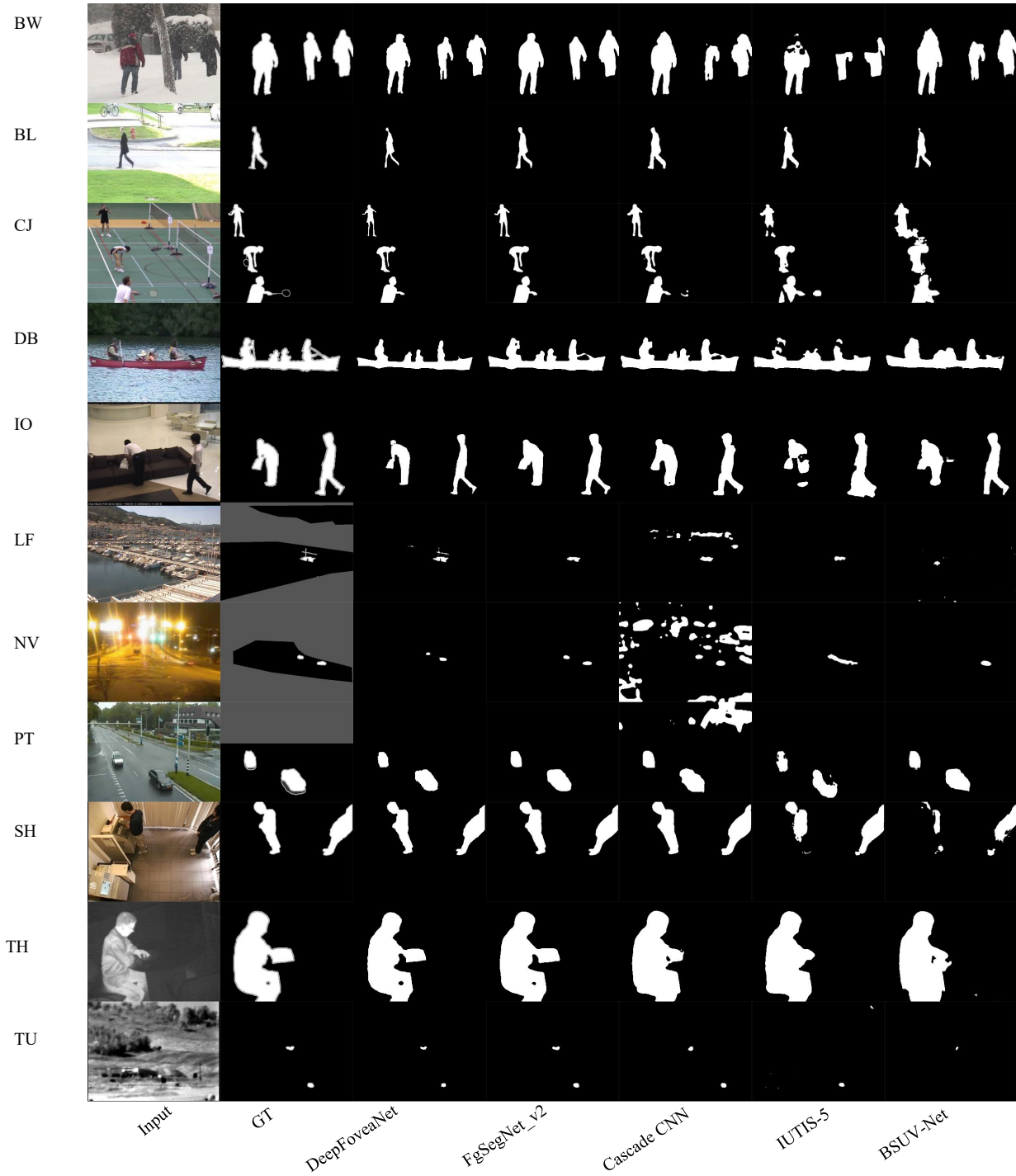


Figura 5.13. Comparación de los resultados de detección de DeepFoveaNet contra algoritmos del estado del arte en un video por cada categoría de la base de datos *CDnet14*. Videos, BW: Badweather, BL: Baseline, CJ: CameraJitter, DB: DynamicBackground, IO: Intermittent Object Motion, LF: LowFramerate, NV: Night Videos, PT: PTZ, SH: Shadow, TH: Thermal, TU: Turbulence.

5.3.2. Evaluación en LASIESTA

Para demostrar la robustez del algoritmo DeepFoveaNet se realizó su evaluación en una segunda base de datos. Esta base de datos fue LASIESTA (*Labeled and Annotated Sequences for Integral Evaluation of Segmentation Algorithms*). LASIESTA está compuesta por 26 secuencias reales de interior y 22 de exterior organizadas en diferentes categorías. Las categorías de escenarios interiores son: *indoorsequences*, *movingcamera*, *modifiedbackground*, *bootstrap*, *camouflage*, *illuminationchanges*, *occlusions* y *simulatedmotion*. Por su parte, las categorías de escenarios en exterior son: *cloudyconditions*, *movingcamera*, *rainyconditions*, *simulatedmotion*, *snowyconditions* y *sunnyconditions*. A diferencia de otras bases de datos, LASIESTA se encuentra completamente anotada tanto a nivel de píxeles como a nivel de objetos (todas las secuencias fueron etiquetadas a través de la herramienta de etiquetado semiautomático TSLAB). Por lo tanto, es adecuada no sólo para las tareas de detección de objetos en movimiento, sino también para tareas de seguimiento. Contiene secuencias grabadas con cámaras estáticas y en movimiento y también proporciona información sobre los objetos en movimiento que permanecen temporalmente estáticos.

Uno de los inconvenientes de esta base de datos, es que por lo general los videos son cortos, de alrededor de 300 cuadros y DeepFoveaNet utiliza 200 cuadros para entrenamiento. Por lo que, para no utilizar más de la mitad de cuadros en esos videos y demostrar la capacidad de DeepFoveaNet, en el entrenamiento fueron considerados únicamente 100 cuadros de video para videos cortos de menos de 500 cuadros. Para videos con 500 cuadros o más se procedió a utilizar los 200 cuadros que se usan normalmente en el modelo DeepFoveaNet propuesto. Todos los parámetros y esquemas de entrenamiento mencionados en las secciones previas fueron conservados para este experimento.

A continuación, en la Tabla 5.5 se muestran los resultados cuantitativos obtenidos sobre todos los videos de la base de datos LASIESTA, se puede observar que estos son bastante favorables, con valores en la métrica F-Measure por encima de 0.80 en casi todas las categorías de video. Los mejores resultados se obtuvieron en las categorías *bootstrap*, *camouflage*, *modifiedbackground*, *E_cloudyconditions* y *E_snowyconditions* con valores por encima a 0.90 en F-Measure. La categoría que representó un mayor desafío fue *E_movingcamera* que contiene movimiento de cámaras en videos con tomas en el exterior.

Tabla 5.5. Resultados del modelo propuesto sobre todos los videos de *LASIESTA*.

Categoría	Subcategoría	<i>Recall</i>	<i>Specificity</i>	<i>FPR</i>	<i>FNR</i>	<i>PWC</i>	<i>Precisión</i>	<i>FMeasure</i>
Interior	<i>bootstrap</i> :	0.905	0.999	0.001	0.095	0.264	0.918	0.911
	<i>camouflage</i> :	0.835	0.994	0.006	0.165	1.920	0.915	0.871
	<i>illuminationchanges</i> :	0.893	0.997	0.003	0.107	0.603	0.880	0.887
	<i>indoorsequences</i> :	0.898	0.994	0.006	0.102	0.967	0.875	0.886
	<i>modifiedbackground</i> :	0.943	0.996	0.004	0.057	0.719	0.937	0.940
	<i>movingcamera</i> :	0.852	0.996	0.004	0.148	0.641	0.776	0.812
	<i>occlusions</i> :	0.851	0.996	0.004	0.149	0.726	0.857	0.854
	<i>simulatedmotion</i> :	0.835	0.988	0.012	0.165	2.100	0.825	0.830
	<i>E_cloudyconditions</i> :	0.926	0.998	0.002	0.074	0.317	0.913	0.920
	<i>E_movingcamera</i> :	0.737	0.997	0.003	0.263	0.661	0.828	0.777
Exterior	<i>E_rainyconditions</i> :	0.881	0.998	0.002	0.119	0.363	0.860	0.870
	<i>E_simulatedmotion</i> :	0.804	0.996	0.004	0.196	0.770	0.807	0.805
	<i>E_snowyconditions</i> :	0.970	1.000	0.000	0.030	0.066	0.962	0.966
	<i>E_sunnyconditions</i> :	0.830	0.999	0.001	0.170	0.348	0.864	0.846
	Promedio	0.869	0.996	0.004	0.131	0.747	0.872	0.870

De acuerdo con la Tabla 5.5, el promedio general sobre todas las categorías de video se considera bastante bueno y tiene consistencia con los resultados obtenidos en la evaluación de la base de datos *CDnet14*. En la Tabla 5.6 se muestra el *benchmark* actual de la base de datos *LASIESTA*, el cual utiliza la métrica F-Measure promedio sobre todas las categorías de video excepto *movingcamera*, *simulatedmotion*, *E_movingcamera* y *E_simulatedmotion*. Se puede observar que el mejor desempeño lo obtuvo el algoritmo [205], con 0.80. Sin embargo, su evaluación no es sobre todos los videos a diferencia de DeepFoveaNet, que obtuvo 0.87 de F-Measure promedio considerando todos los videos, superando a todos los algoritmos evaluados en el *benchmark*. Para ampliar los resultados se muestra la Tabla 5.7 donde se incluye el resultado en la métrica F-Measure para cada uno de los videos de la base de datos. De la Tabla 5.7 se puede inferir que DeepFoveaNet logra resolver diferentes situaciones críticas de video aun y cuando en la gran mayoría de videos solo se usaron 100 cuadros para su entrenamiento, lo que habla muy bien del nivel de su generalización para resolver nuevos desafíos de video.

Tabla 5.6. Resultados de algoritmos evaluados en la base de datos *LASIESTA*.

Método	F-Measure Promedio
<i>DeepFoveaNet</i>	0.8700 (<i>Considerando todos los videos</i>)
Cuevas2[205]	0.8051
Maddalena2[206]	0.7842
Haines[207]	0.7826
Maddalena1[208]	0.7528
Cuevas1[209]	0.7386
Wren[210]	0.7234
Zivkovik[91]	0.7173
Stauffer[211]	0.6953

Tabla 5.7. Resultados de DeepFoveaNet en la métrica F-Measure sobre todos los videos de *LASIESTA*.

Escenario	Categoría	Video	FMeasure	Número de cuadros de entrenamiento
Interior	bootstrap	I_BS_01	0.933	100
		I_BS_02	0.889	100
		Promedio	0.911	
	camouflage	I_CA_01	0.955	100
		I_CA_02	0.787	200
		Promedio	0.871	
	illuminationchanges	I_IL_01	0.869	100
		I_IL_02	0.905	200
		Promedio	0.887	
	indoorsequences	I_SI_01	0.837	100
		I_SI_02	0.935	100
		Promedio	0.886	
	modifiedbackground	I_MB_01	0.947	100
		I_MB_02	0.932	100
		Promedio	0.940	
	movingcamera	I_MC_01	0.834	100
		I_MC_02	0.790	100
		Promedio	0.812	
	occlusions	I_OC_01	0.862	100
		I_OC_02	0.846	100
		Promedio	0.854	
	simulatedmotion	I_SM_01	0.834	100
		I_SM_02	0.891	100
		I_SM_03	0.888	100
		I_SM_04	0.824	100
		I_SM_05	0.825	100
		I_SM_06	0.819	100
		I_SM_07	0.821	100
		I_SM_08	0.823	100
		I_SM_09	0.815	100
		I_SM_10	0.806	100
		I_SM_11	0.805	100
		I_SM_12	0.805	100
		Promedio	0.830	
Exterior	E_cloudyconditions	O_CL_01	0.959	100
		O_CL_02	0.880	100
		Promedio	0.920	
	E_movingcamera	O_MC_01	0.724	100
		O_MC_02	0.830	100
		Promedio	0.777	
	E_rainyconditions	O_RA_01	0.935	200
		O_RA_02	0.806	100
		Promedio	0.870	
	E_simulatedmotion	O_SM_01	0.883	100
		O_SM_02	0.854	100
		O_SM_03	0.820	100
		O_SM_04	0.880	100
		O_SM_05	0.844	100
		O_SM_06	0.780	100
		O_SM_07	0.816	100
		O_SM_08	0.809	100
		O_SM_09	0.774	100
		O_SM_10	0.730	100
		O_SM_11	0.748	100
		O_SM_12	0.723	100
		Promedio	0.805	
	E_snowyconditions	O_SN_01	0.953	200
		O_SN_02	0.979	200
		Promedio	0.966	
	E_sunnyconditions	O_SU_01	0.837	100
		O_SU_02	0.855	100
		Promedio	0.846	
	General		0.870	

5.4. Análisis de las características aprendidas por los filtros de las redes *Peripheral-CNN* y *Deepfovea-CNN* que componen al algoritmo *DeepFoveaNet*

En esta sección se realiza un análisis de la visualización de los filtros o *kernels* aprendidos por la red para comprender un poco más acerca del proceso interno de la red convolucional para lograr la detección del movimiento. Para realizar el análisis se mostrará la información aprendida por los filtros de las redes de periferia (*Peripheral-CNN*) y fovea profunda (*Deepfovea-CNN*) en cada una de las capas de su arquitectura, tanto de la parte *Encoder* como la Parte *Decoder*. La visualización de las características aprendidas se realiza de la siguiente manera: 1) se alimenta la red entrenada con dos imágenes consecutivas de video, 2) se realiza la propagación a través de la red, 3) se muestran las imágenes de los mapas de características generados en cada capa de la arquitectura de red. Los mapas de características indican a que información son sensibles los filtros, por ejemplo, supongamos que se alimenta una red entrenada, con una imagen que contiene de dos carros, uno color rojo y otro azul, si se extrae un primer mapa de características y al observarlo este resalta la información del carro rojo se podría decir que es un filtro sensible al color rojo. Si se extrae otro mapa de características y resalta la información de ambos carros, se podría inferir que es un filtro sensible a la forma.

En la Figura 5.14, se hace referencia las dos arquitecturas en análisis (*Peripheral-CNN* y *Deepfovea-CNN*) que fueron descritas en secciones previas, ambas redes trabajan en conjunto en el algoritmo *DeepFoveaNet*.

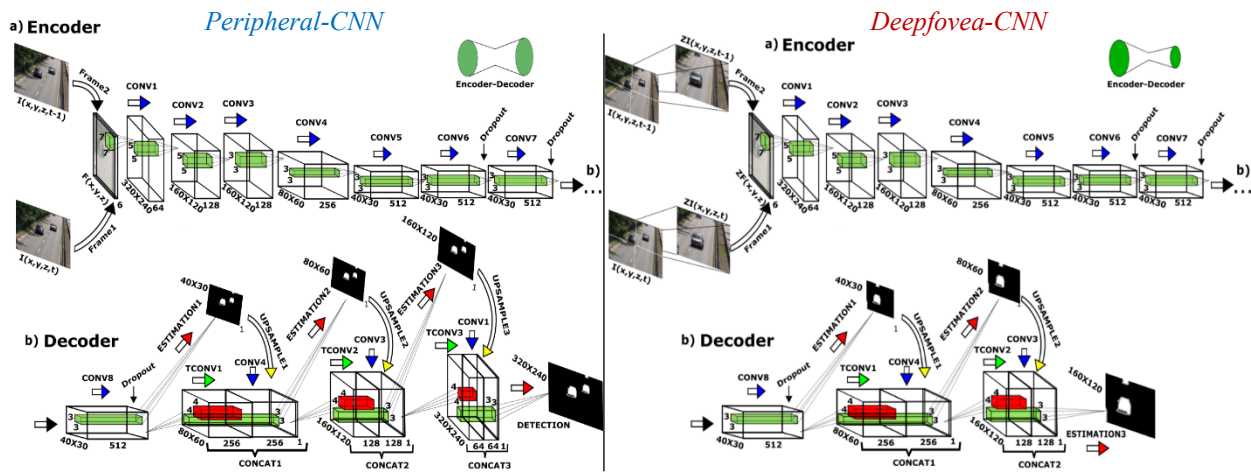


Figura 5.14. Arquitecturas de las redes *Peripheral-CNN* y *Deepfovea-CNN*.

En la Figura 5.15 y Figura 5.16 se observan los primeros mapas de características que se generan por los 64 filtros de $7 \times 7 \times 6$ de la primera capa convolucional CONV1 al pasar dos imágenes consecutivas del video *pedestrians* de *CDnet14*, este video fue elegido debido a su simplicidad ya que facilita el análisis, sin embargo, los resultados de este experimento no se limitan al video en particular, el orden en el que aparecen las características a través de los distintos niveles de la red es similar para otros videos. El mapa de color elegido para mostrar los mapas de características es “viridis”, por su uniformidad perceptual que va de colores morados a amarillos que representan la menor y mayor activación respectivamente.

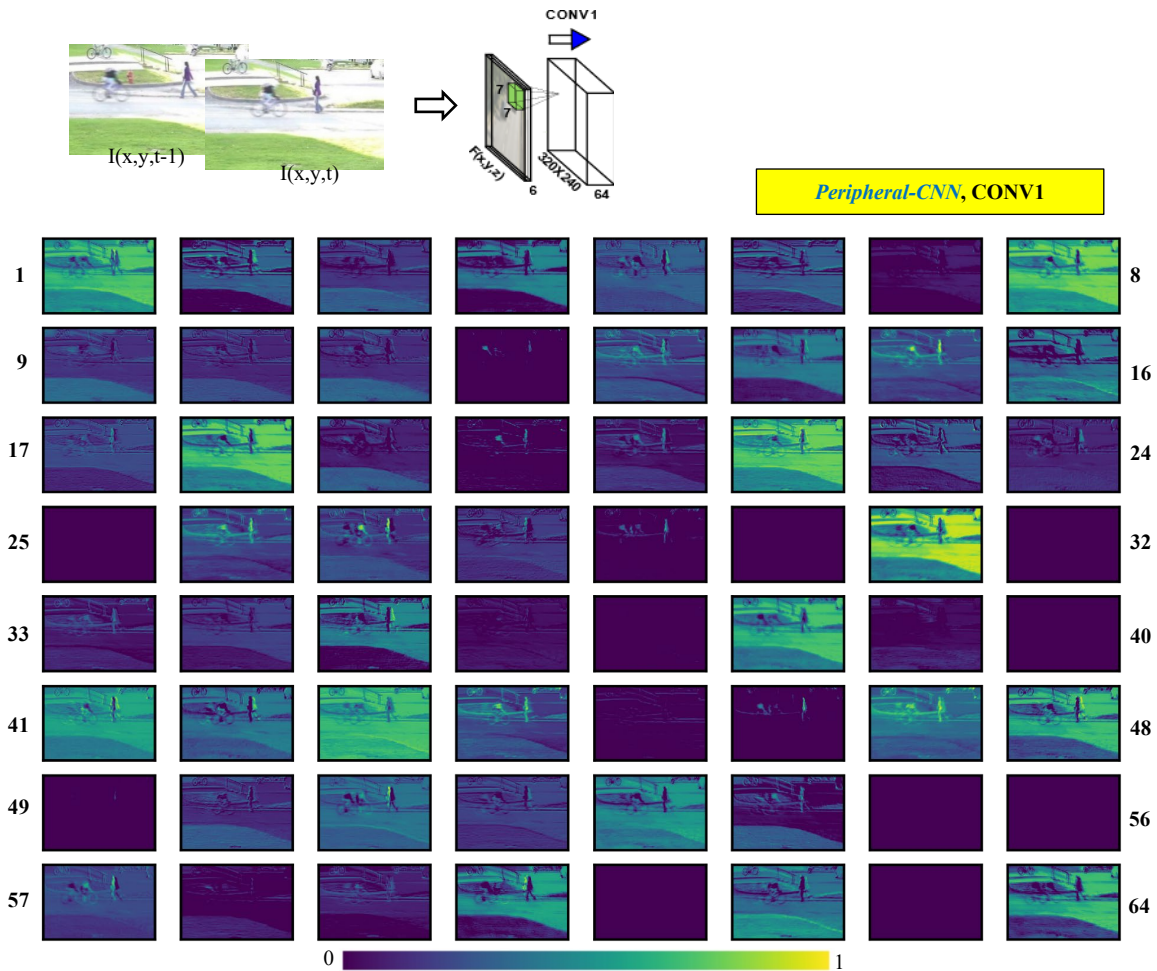


Figura 5.15. Mapas de características primera capa convolucional CONV1 red *Peripheral-CNN*.

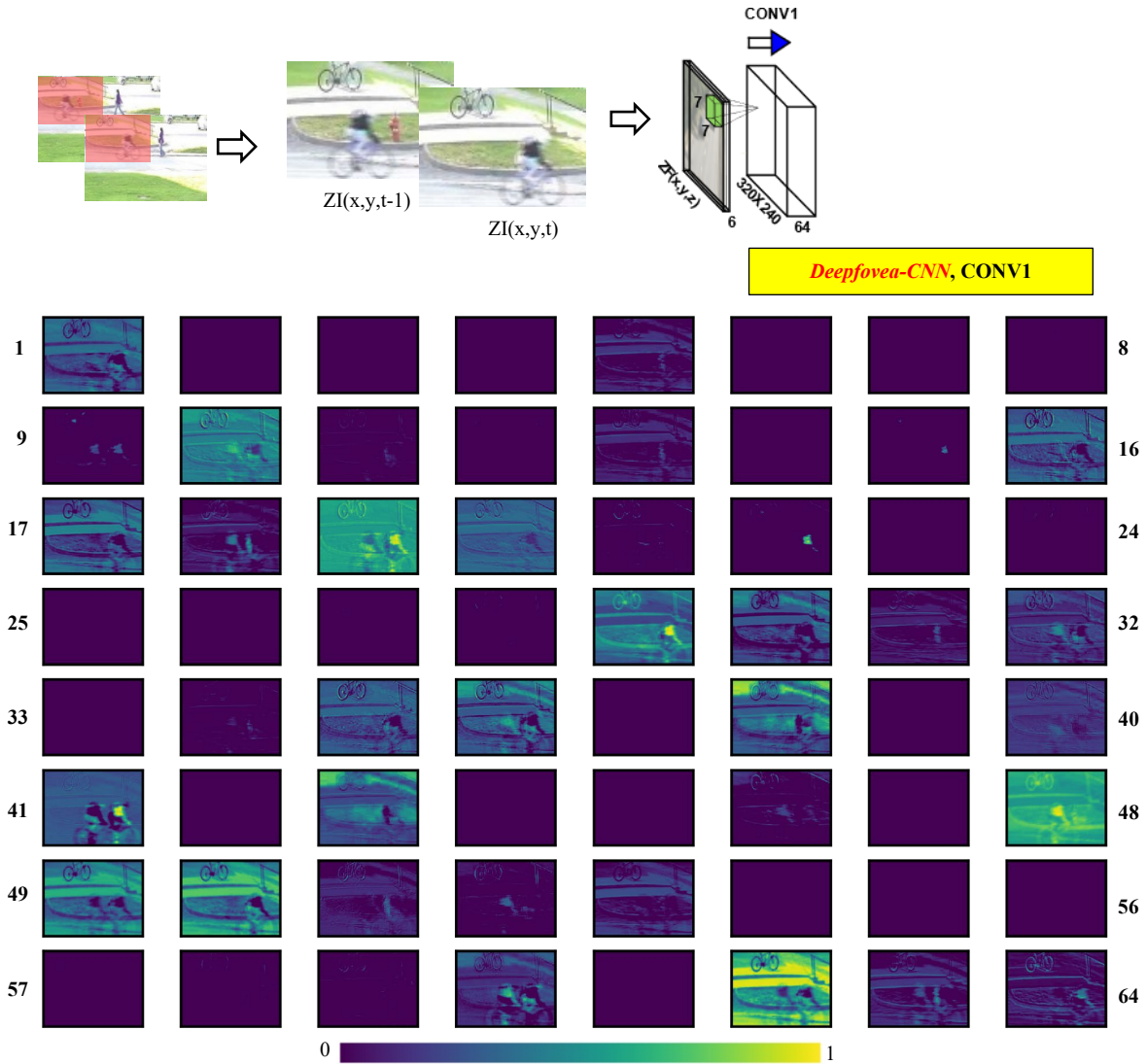


Figura 5.16. Mapas de características primera capa convolucional CONV1 red *Deepfovea-CNN*.

De la Figura 5.15 y Figura 5.16 se pueden observar diversos mapas de características que generan los filtros, los números al costado de la imagen indican el número de filtro de la primera y última columna respectivamente. A continuación, se muestran imágenes ampliadas de algunos de ellos, para visualizar y analizar mejor su comportamiento.

Algunos de los mapas de características que más sobresalen a simple vista son aquellos que son generados por filtros que resaltan bordes, que corresponden entre ellos a los números 36, 45, 58 y 59 de la red *Peripheral-CNN* (Figura 5.17) y los filtros 5, 32, 46 y 64 de la red *Deepfovea-CNN* (Figura 5.18).

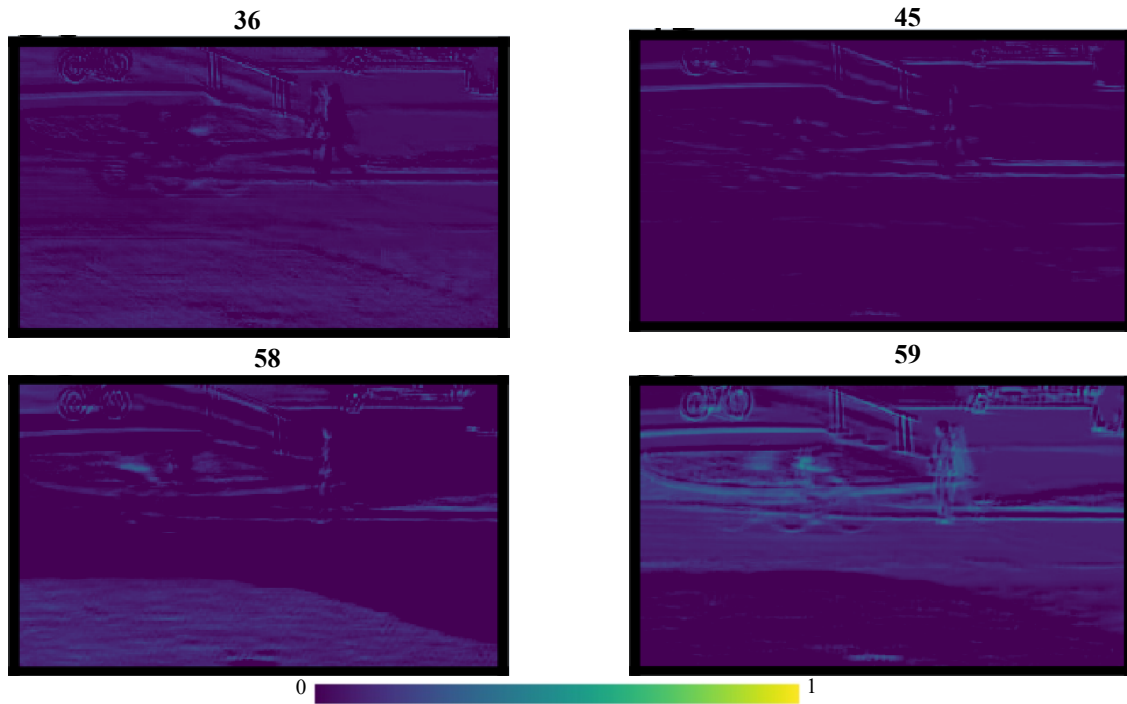


Figura 5.17. Ampliación: mapas de características generados por filtros sensibles a bordes primera capa convolucional CONV1 red *Peripheral-CNN*.

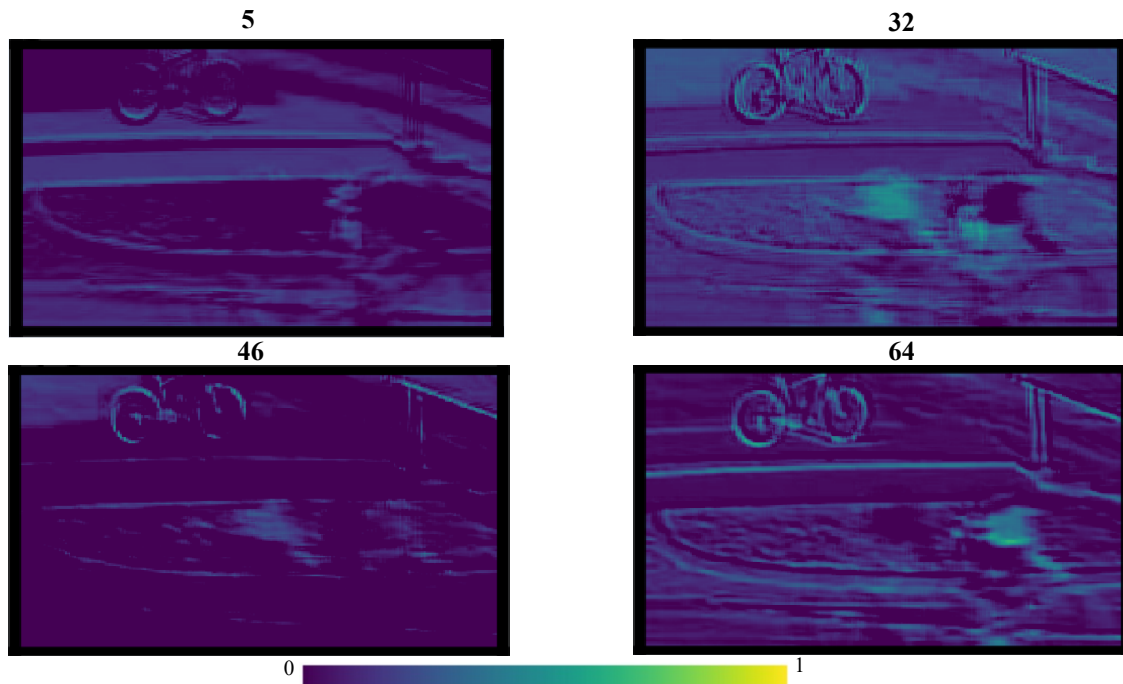


Figura 5.18. Ampliación: mapas de características generados por filtros sensibles a bordes primera capa convolucional CONV1 red *Deepfovea-CNN*.

Hay otros filtros en la capa CONV1 que son sensibles a colores o texturas como los filtros 31 y 64 (Figura 5.19) que resaltan el color gris o la textura del pavimento, o los filtros 38 y 62 (Figura 5.20) que destacan el pasto y pavimento de la imagen respectivamente.

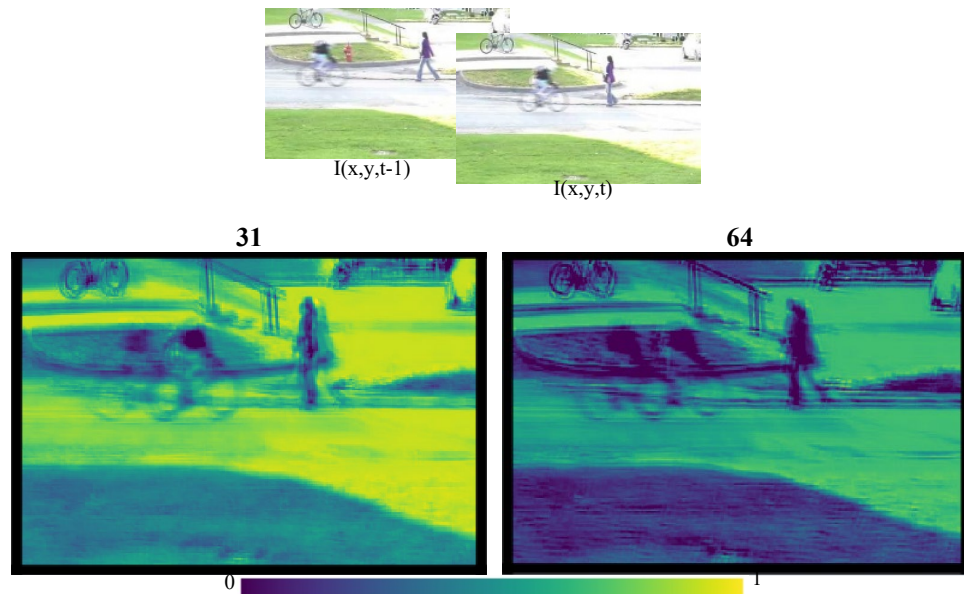


Figura 5.19. Ampliación: mapas de características generados por filtros sensibles al color o textura en la capa convolucional CONV1 red *Peripheral-CNN*.

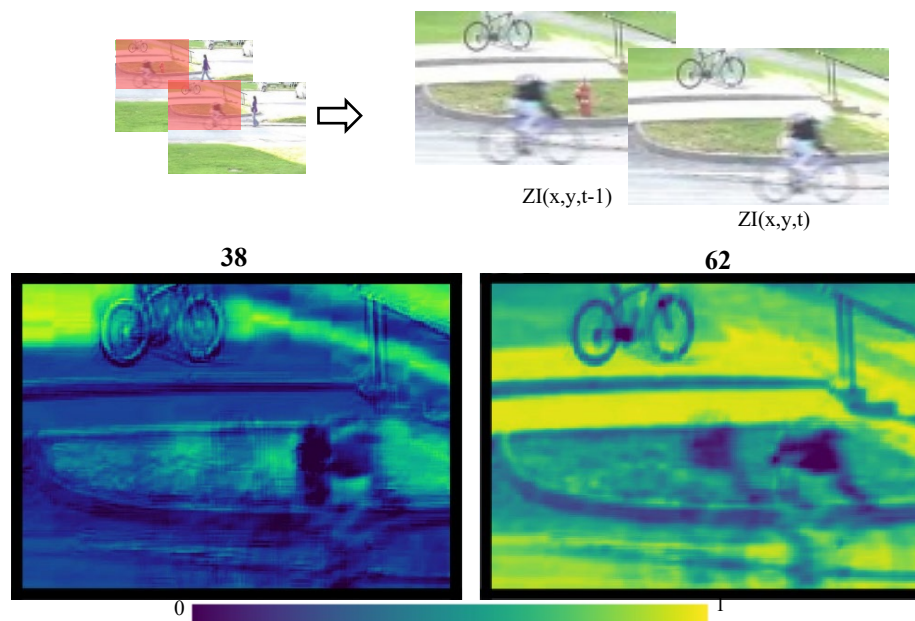


Figura 5.20. Ampliación: mapas de características generados por filtros sensibles al color o textura en la capa convolucional CONV1 red *Deepfovea-CNN*.

También hay filtros que inhiben ciertas características del escenario y resaltan las de los objetos de interés. Por ejemplo, en la red *Peripheral-CNN* los filtros 12 y 46 destacan la figura de la persona que va caminando y la del ciclista (Figura 5.21) y en la red *Deepfovea-CNN*, el filtro 9 extrae la figura del ciclista y los filtros 15 y 22 realzan lo que parece ser la playera de este (Figura 5.22). Es importante señalar que en la imagen del filtro 9 se observan dos figuras que corresponden con el

ciclista del cuadro $ZI(t-1)$ y $ZI(t)$ y en los filtros 15 y 22 las figuras corresponden únicamente al ciclista del cuadro $Z(I)$, por lo que, de cierta manera, estos últimos filtros solo prestan atención a los objetos del cuadro actual de video.

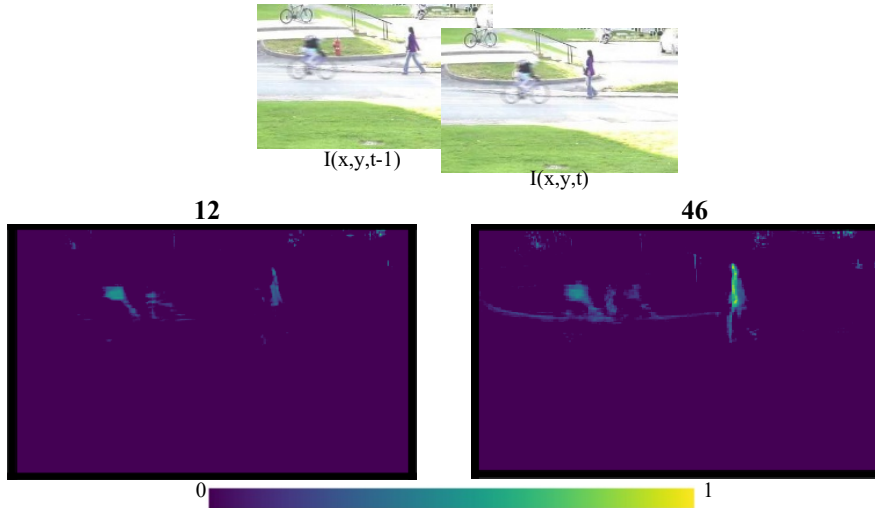


Figura 5.21. Ampliación: mapas de características generados por filtros sensibles a rasgos de los objetos de interés en la capa convolucional CONV1 red *Peripheral-CNN*.

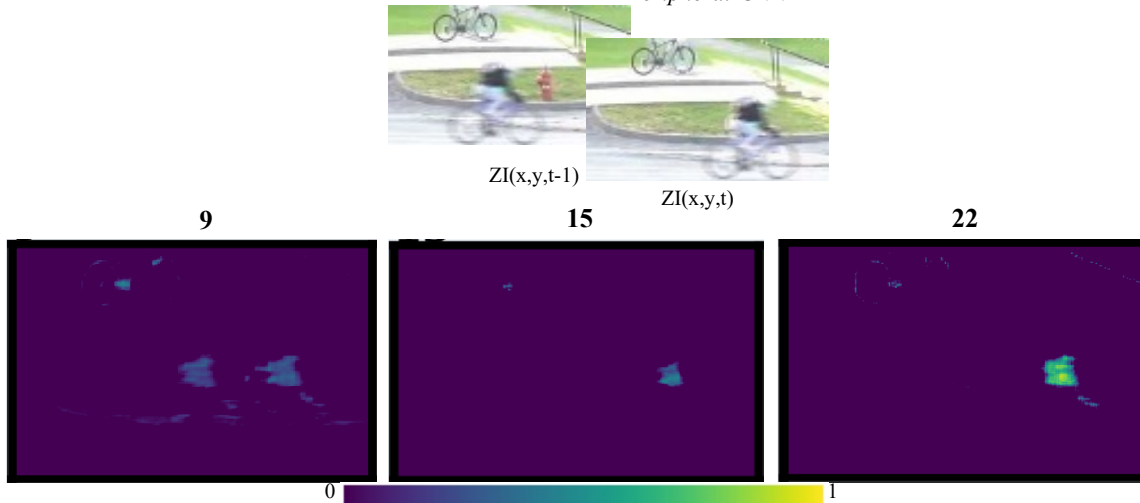


Figura 5.22. Ampliación: mapas de características generados por filtros sensibles a rasgos de los objetos de interés en la capa convolucional CONV1 red *Deepfovea-CNN*.

Otro tipo de filtros que se observan en su mayoría en la capa CONV1 de la red *Peripheral-CNN* son el 5, 13, 15, 17, 18, 19, 21, 24, 27, 42 que como se explicará más adelante, se piensa son los encargados de la obtención del movimiento de objetos de interés, así como los filtros 1, 10, 19, 32, 41, 60 de la red *Deepfovea-CNN*. En las Figura 5.23 y Figura 5.24 se muestran algunos ejemplos de los mapas de características generados por este tipo de filtros.

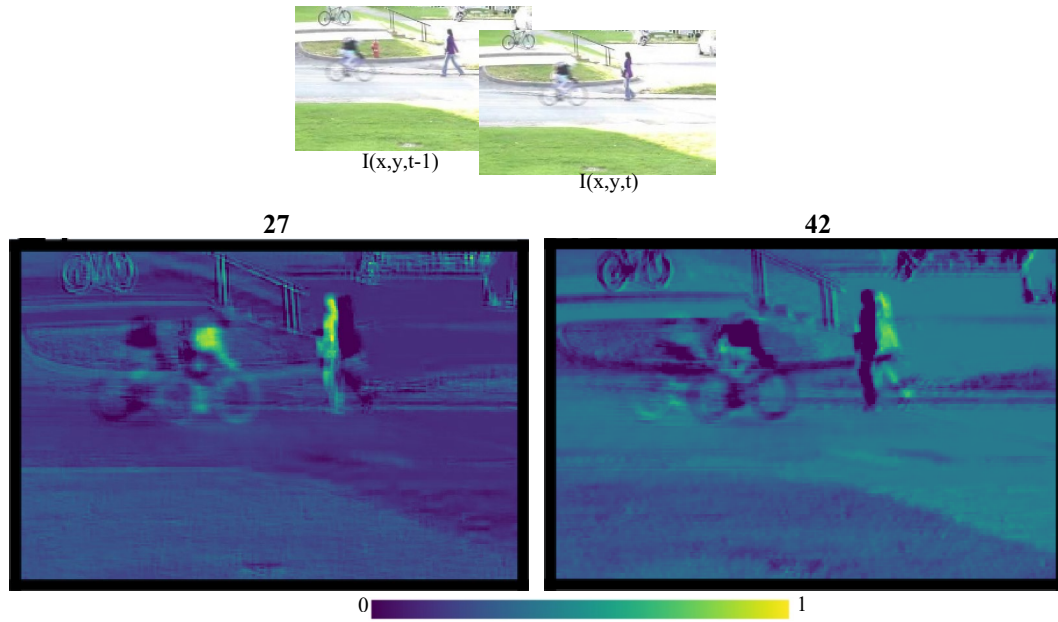


Figura 5.23. Ampliación: mapas de características generados por filtros sensibles al movimiento en la capa convolucional CONV1 red *Peripheral-CNN*.

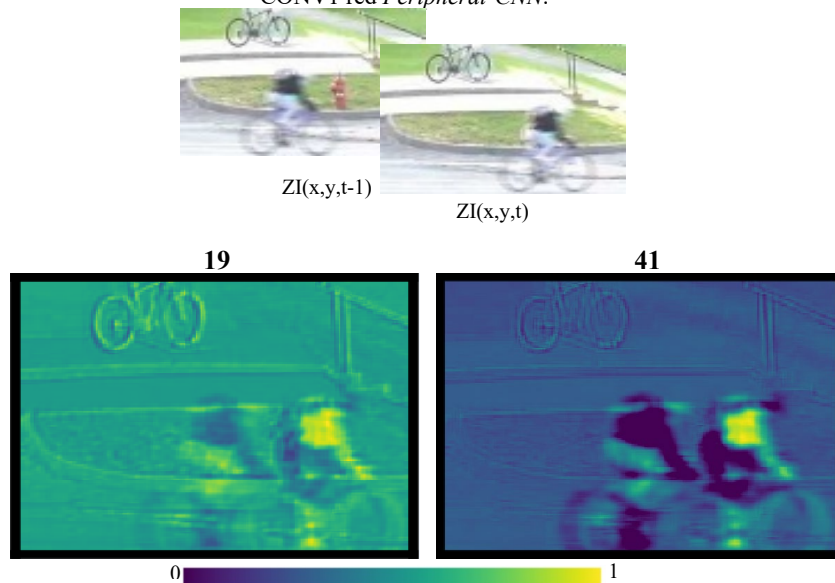


Figura 5.24. Ampliación: mapas de características generados por filtros sensibles al movimiento en la capa convolucional CONV1 red *Deepfovea-CNN*.

De la Figura 5.23, se puede observar que el mapa de características generado por el filtro 27 resalta la información del movimiento en el cuadro actual e inhibe la información de los objetos en el cuadro anterior. Por el contrario, el mapa de características generado por el filtro 42 resalta la información del cuadro previo e inhibe la del cuadro actual. Por lo tanto, se puede inferir que estos filtros extraen características temporales o son sensibles a la información del objeto de interés en un tiempo preferente.

Con el fin de ampliar el análisis y reforzar la idea de que este tipo de filtros extraen características temporales, se utilizó la herramienta [162] para generar filtros espaciotemporales de Gabor (ETG) en el programa MATLAB®. Se generaron 4 filtros ETG con preferencia al movimiento horizontal con dirección hacia la izquierda a diferentes velocidades v y luego estos se convolucionaron con la concatenación de las dos imágenes sucesivas del video $F(x,y,z)$ para generar sus respectivos mapas de características, tal como se muestra en la Figura 5.25.

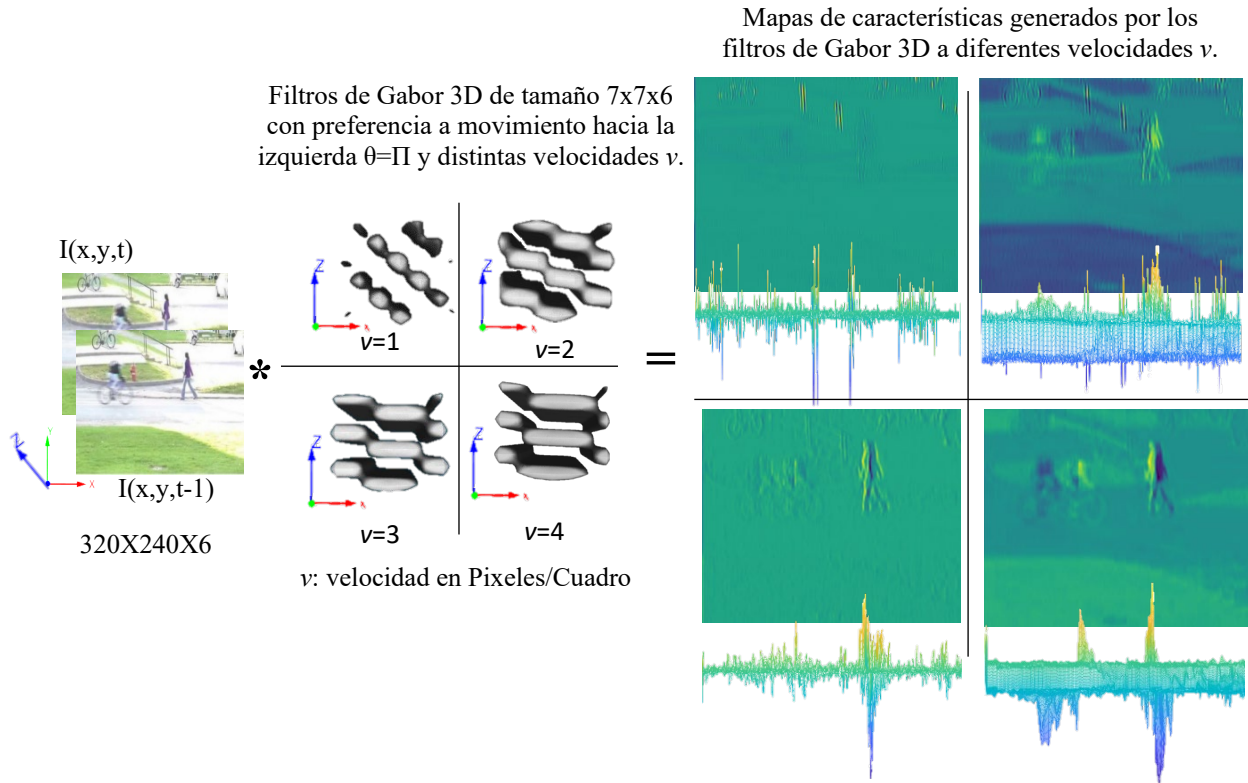


Figura 5.25. Mapas de características generados por filtros ETG con preferencia al movimiento hacia la izquierda a diferentes velocidades: $v=1$, $v=2$, $v=3$ y $v=4$.

De la Figura 5.25 los mapas de características obtenidos por los filtros ETG realzan el movimiento en la dirección y velocidad preferente del filtro. En especial si compramos el mapa de características generado por el filtro ETG con $v=4$ y $\theta=\Pi$ contra el mapa de características generado por el filtro 27 aprendido por la red, estos son visualmente muy similares entre si (Figura 5.26) ya que ambas imágenes realzan el movimiento de los objetos de interés en el cuadro actual del video $I(t)$ e inhiben la información del cuadro anterior. Además, la mayor activación en ambas imágenes se localiza en la persona caminando que se mueve hacia la izquierda. Sin embargo, si se observa de cerca y se compara el mapa de características generado por el filtro 27 y el mapa generado por

el filtro ETG, el filtro que la red aprende es aún más complejo ya que además de filtrar el movimiento realza los bordes en la imagen y es más sencillo ver los cambios de textura.

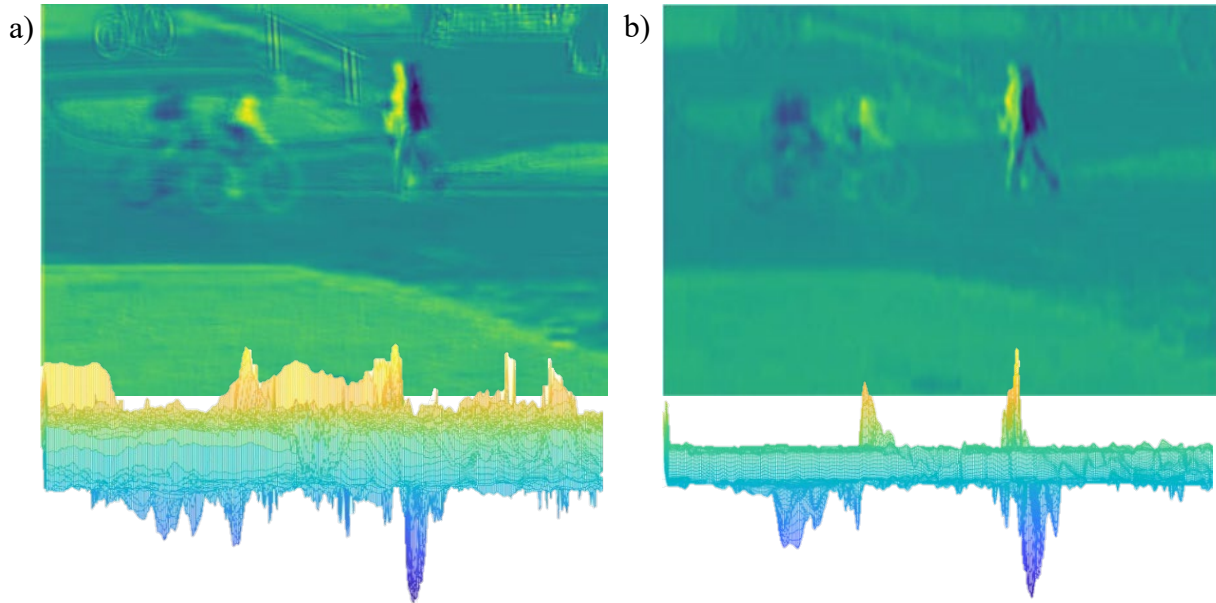


Figura 5.26. Comparación de mapas de características; a) mapa generado por el filtro 27 aprendido por la red *Peripheral-CNN* en la capa CONV1, b) mapa generado con un filtro ETG para extracción de movimiento.

La visualización del filtro 27 aprendido por la red se realizó mediante la herramienta *Volumen viewer* de MATLAB®, donde XYZ son el ancho, alto y el eje temporal (Figura 5.27). Se puede observar que el filtro 27 posee una estructura variable que no parece tener algún patrón específico. Algo intrigante sobre los filtros aprendidos por la red es que, a pesar de poseer una estructura a simple vista desorganizada, logran emular el comportamiento de un filtro ETG para detectar movimiento y al mismo tiempo extraen otras características como bordes o texturas. A continuación, se muestran los mapas de características generados en las siguientes capas de la parte *Encoder* de las arquitecturas *Deepfovea-CNN* y *Peripheral-CNN*. De la capa CONV2 de la Figura 5.28, se pueden observar filtros que son preferentes a los objetos de interés (filtros: 2, 11, 32, 56, etc.) y continúan apareciendo filtros sensibles a texturas, bordes, color y movimiento, por ejemplo, los filtros 18, 31, 37 y 70 respectivamente. El mismo comportamiento se observa para la red *Deepfovea-CNN* en la Figura 5.29.

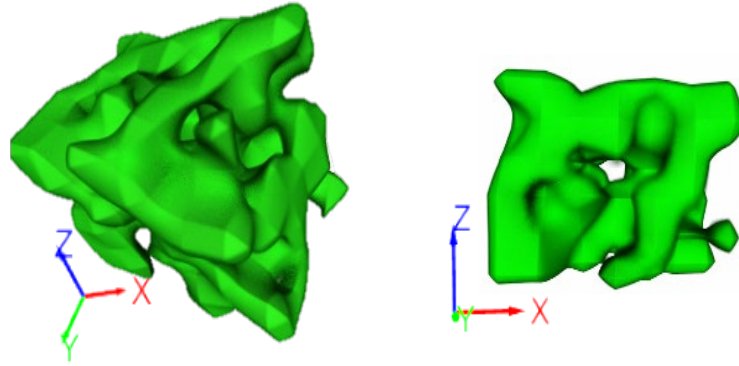


Figura 5.27. Estructura del filtro 27 aprendido por la red en la capa CONV1; a) plano XYZ, b) plano XZ.

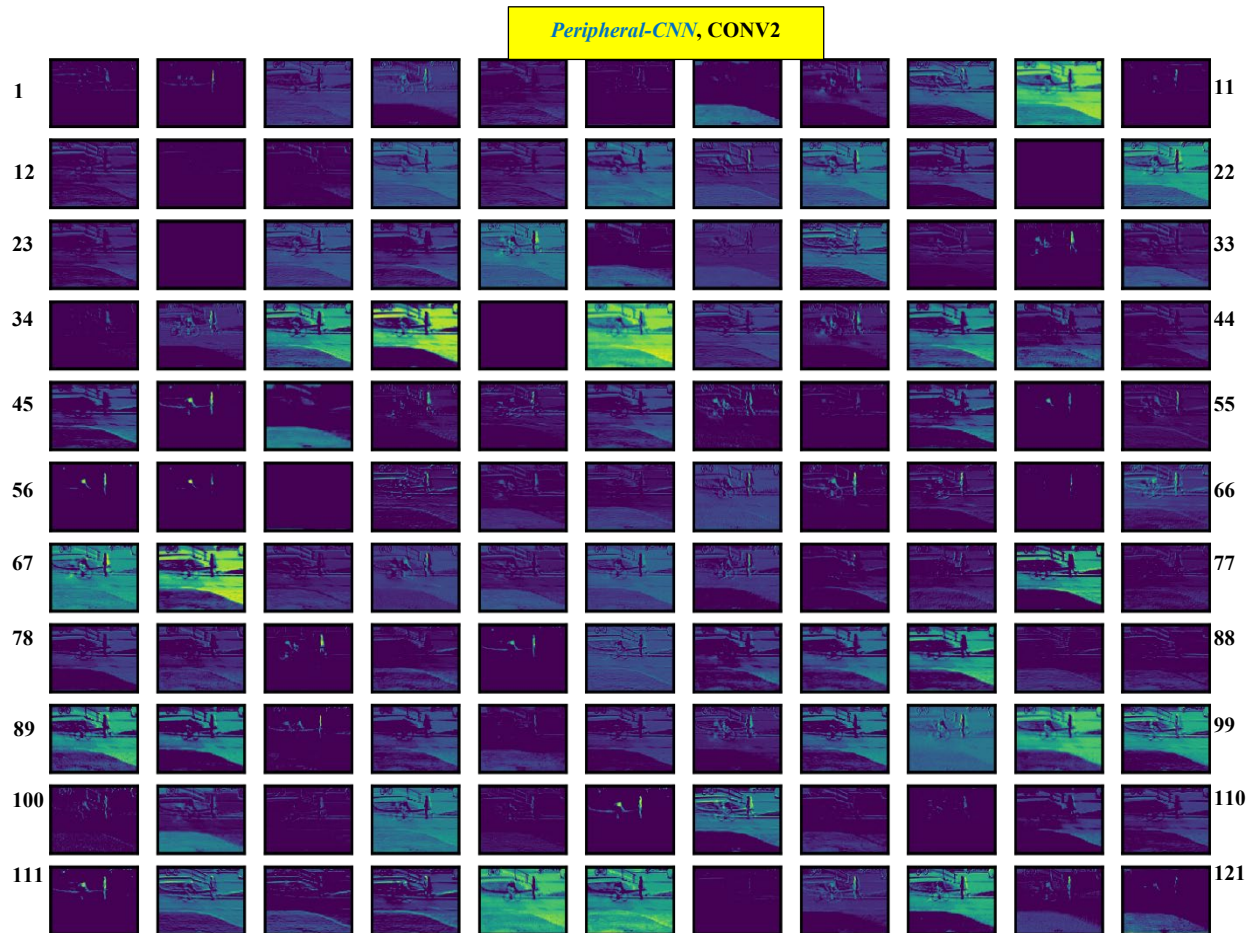


Figura 5.28. Mapas de características capa convolucional CONV2 red *Peripheral-CNN*.

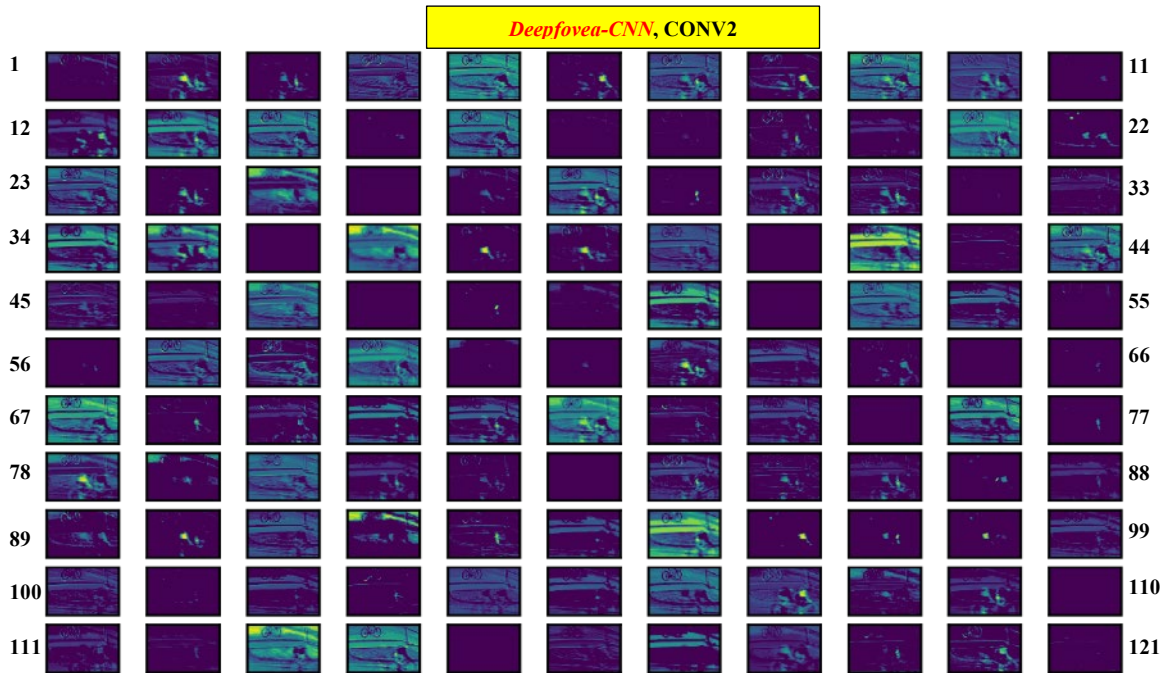


Figura 5.29. Mapas de características capa convolucional CONV2 red *Deepfovea-CNN*.

De los mapas de características de la capa CONV3, Figura 5.30 y Figura 5.31, se observan mayormente filtros detectores de bordes y texturas para la red *Peripheral-CNN* y filtros detectores de bordes y sensibles a los objetos de interés (formas) para la red *Deepfovea-CNN*.

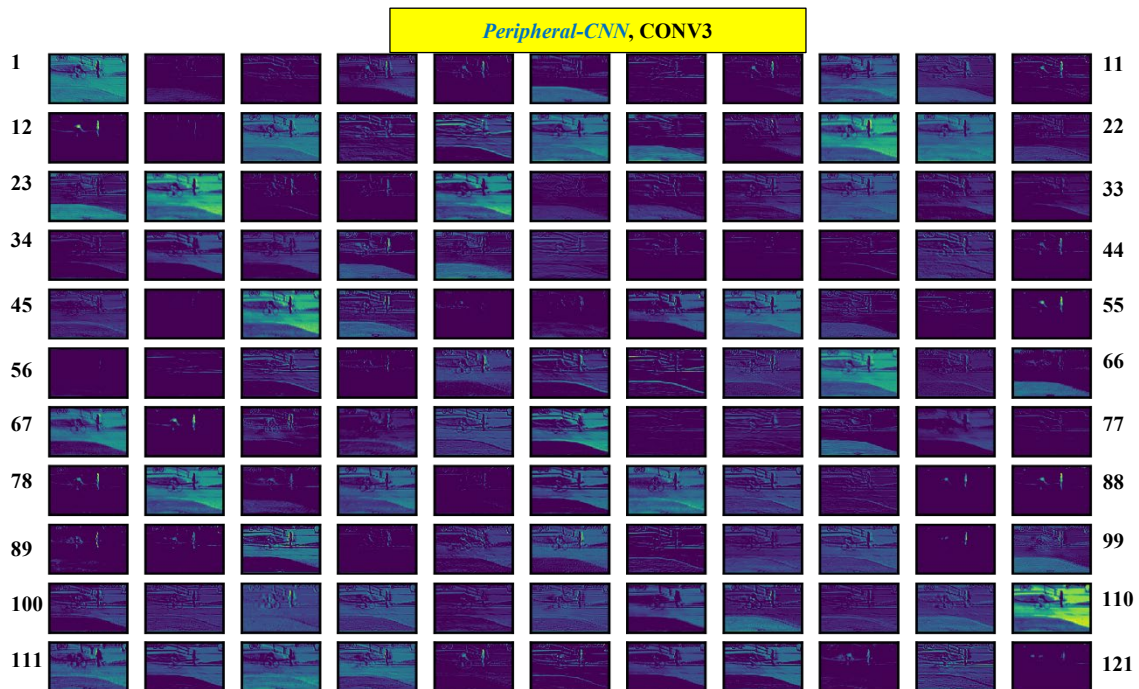


Figura 5.30. Mapas de características capa convolucional CONV3 red *Peripheral-CNN*.

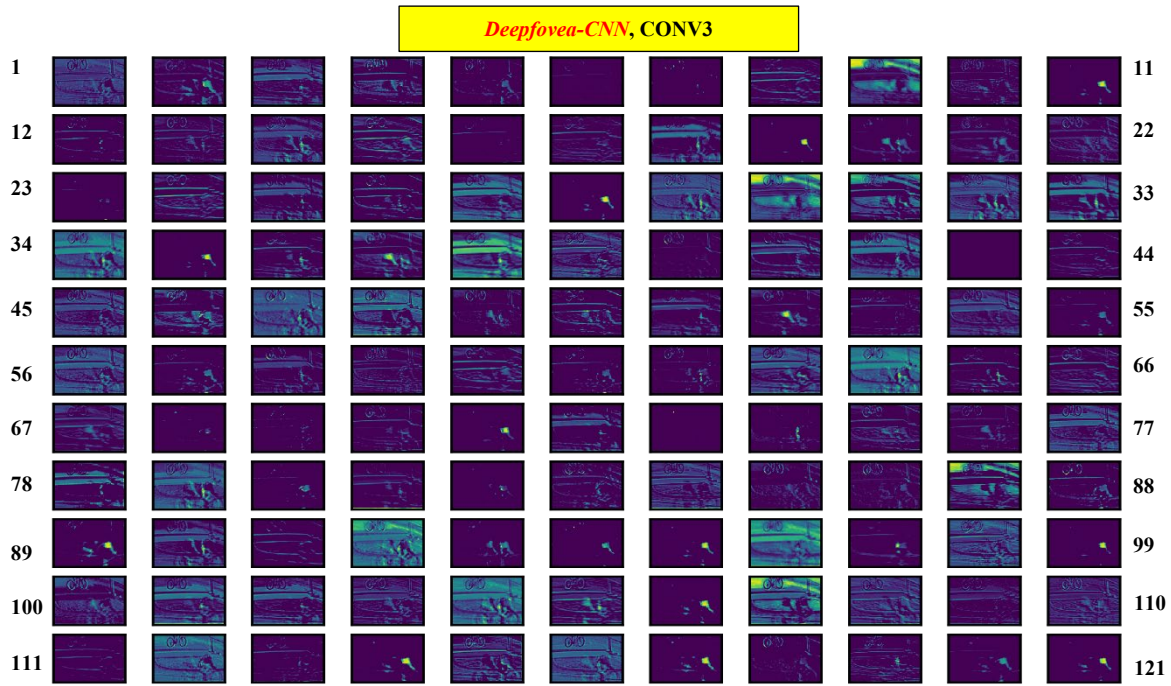


Figura 5.31. Mapas de características capa convolucional CONV3 red *Deepfovea-CNN*.

A continuación se muestran los mapas de características de las capas CONV4 (Figura 5.32 y Figura 5.33), CONV5 (Figura 5.34 y Figura 5.35) y CONV6 (Figura 5.36 y Figura 5.37).

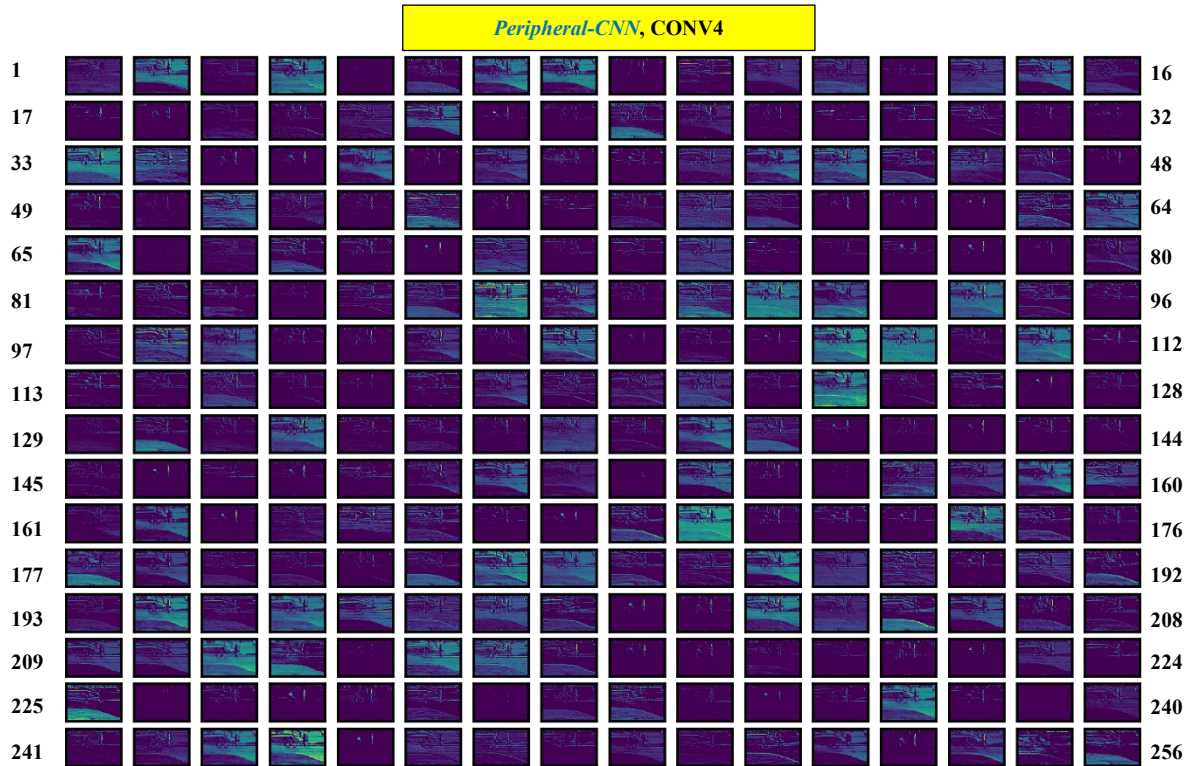
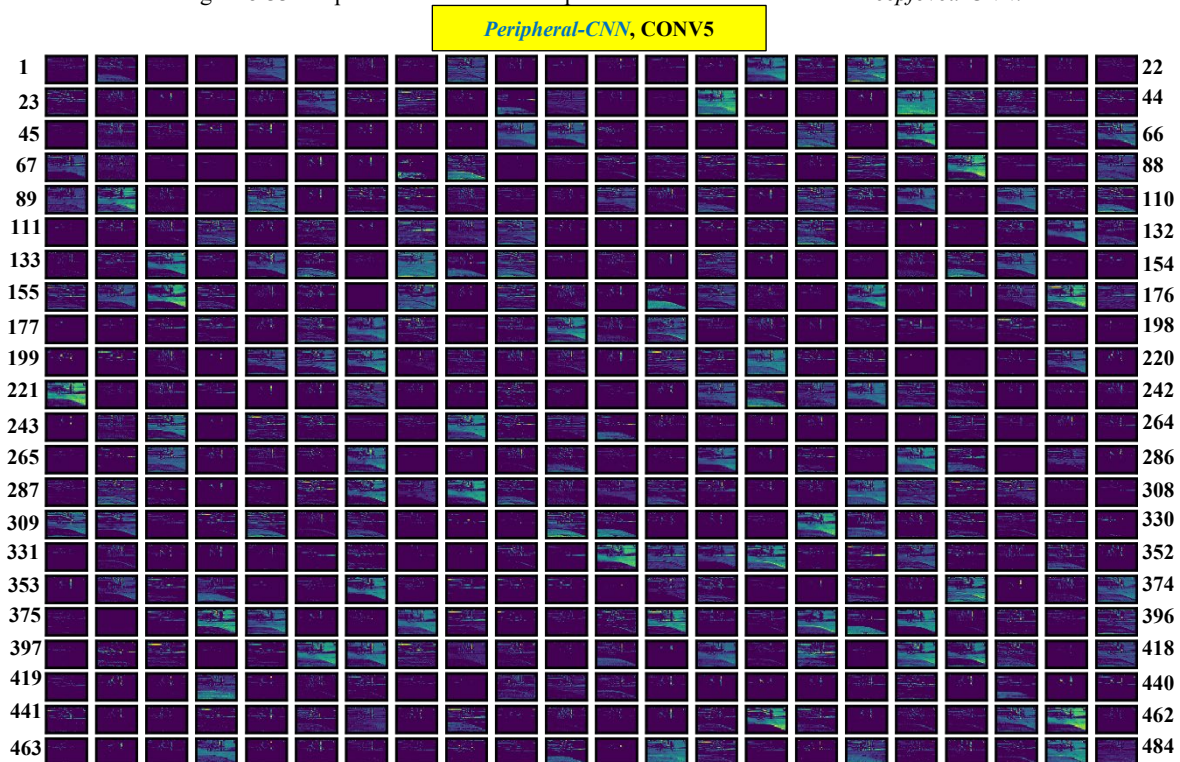
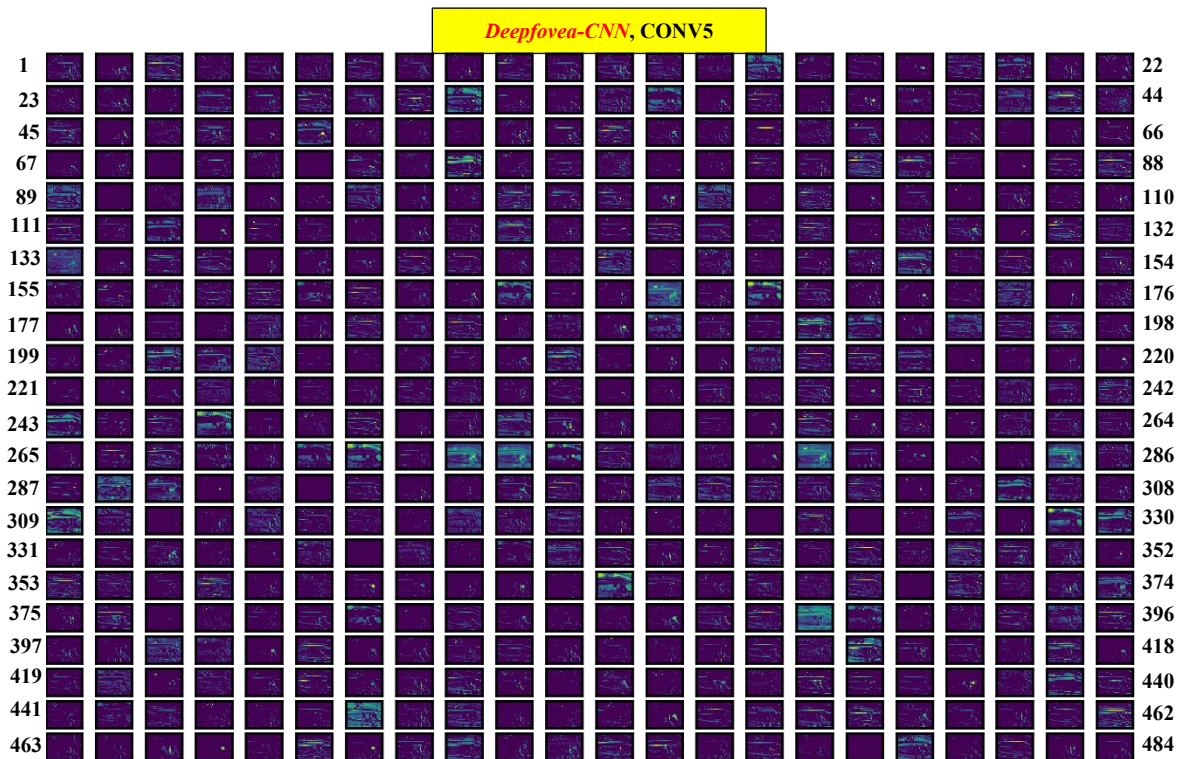
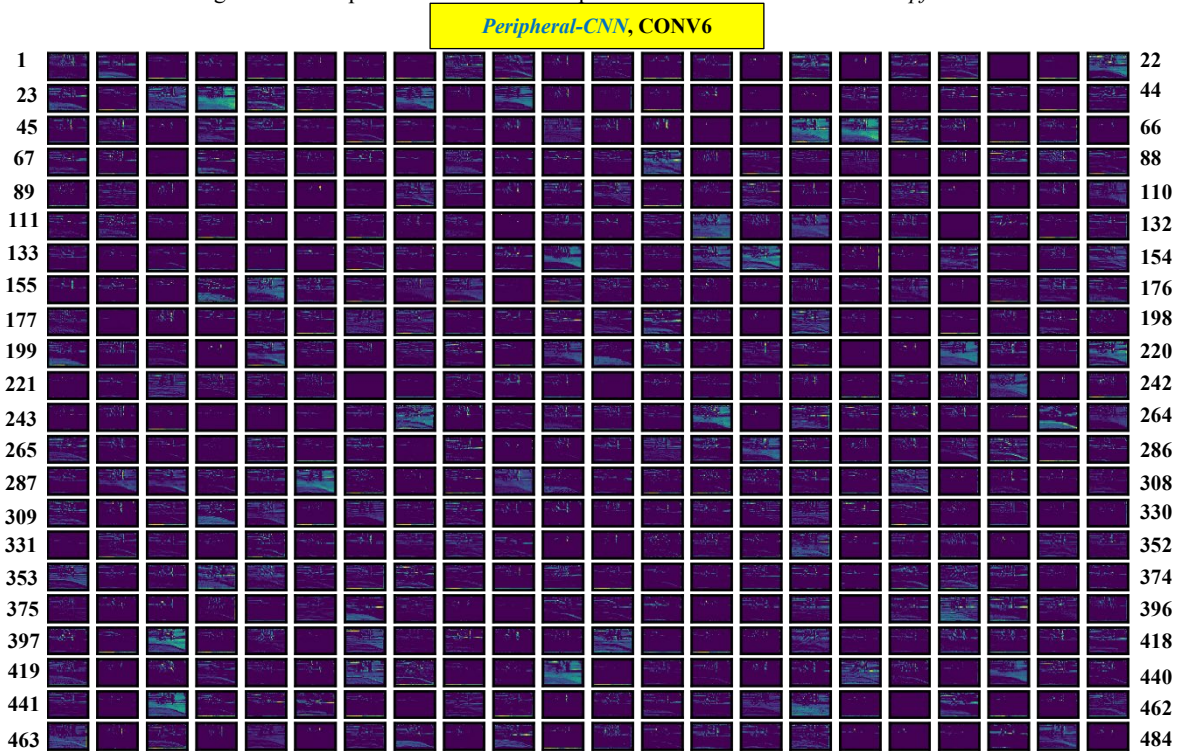


Figura 5.32. Mapas de características capa convolucional CONV4 red *Peripheral-CNN*.

Figura 5.33. Mapas de características capa convolucional CONV4 red *Deepfovea-CNN*.Figura 5.34. Mapas de características capa convolucional CONV5 red *Peripheral-CNN*.

Figura 5.35. Mapas de características capa convolucional CONV5 red *Deepfovea-CNN*.Figura 5.36. Mapas de características capa convolucional CONV6 red *Peripheral-CNN*.

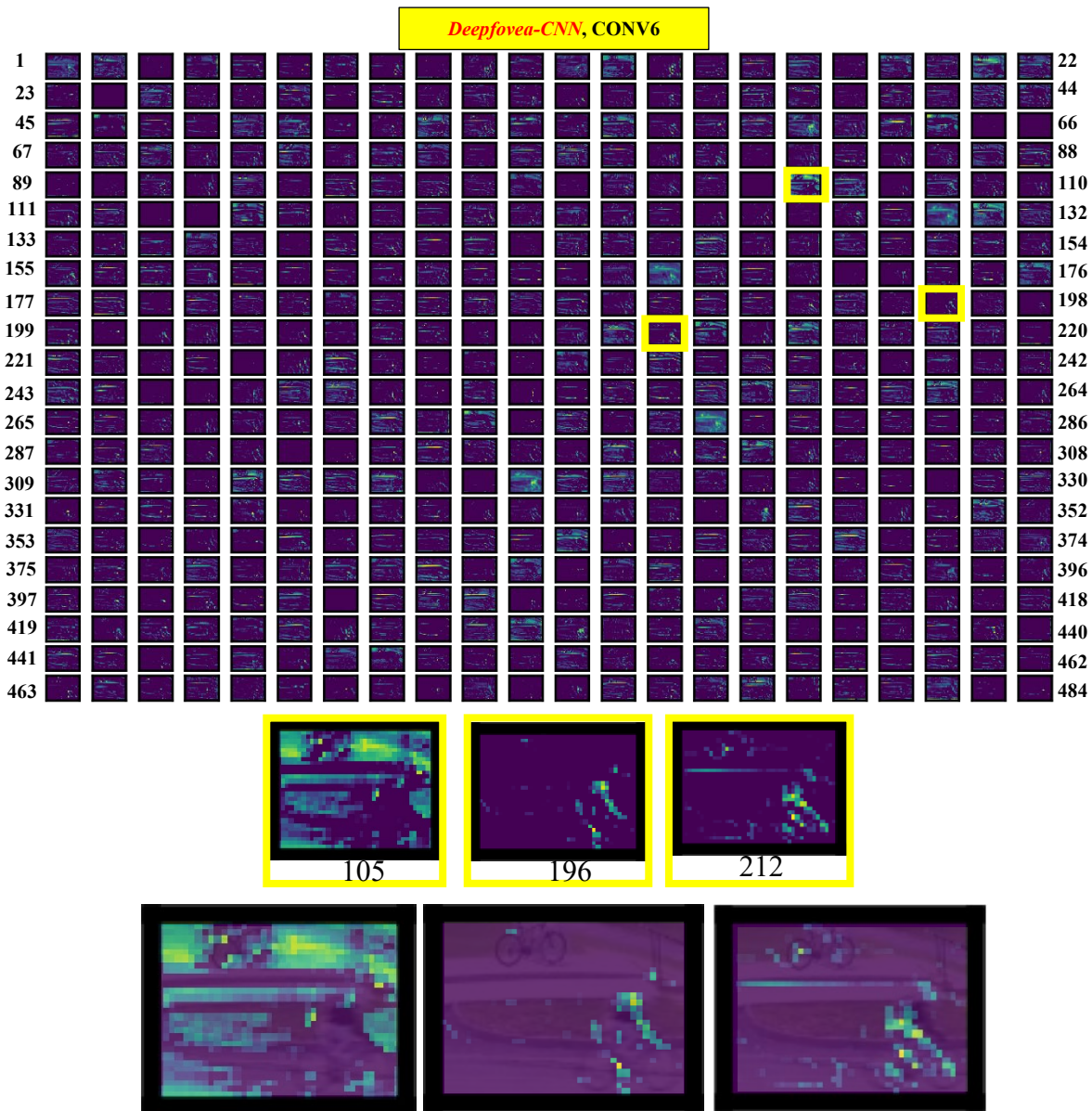


Figura 5.37. Mapas de características capa convolucional CONV6 red *Deepfovea-CNN*.

De la Figura 5.37, se puede observar que conforme se avanza a capas más profundas aumentan el número de filtros enfocados a detectar las características de los objetos de interés (Ej. filtros 212, 196) y además se reduce el tamaño del mapa de características. Por otra parte, existen también filtros que realzan características del escenario e inhiben los objetos de interés, dicho de otra manera, son filtros preferentes a alguna característica del escenario como el pasto o el pavimento (Ej. filtro 105). En la Figura 5.37 anterior y en algunas otras figuras que se muestran a continuación, se incluyó un efecto de acercamiento y se traslapó las imágenes de entrada con un efecto de

transparencia a los mapas de características, para una mejor visualización y observar a detalle que características realzan los filtros.

Las siguientes figuras muestran las capas CONV7 (Figura 5.38 y Figura 5.39) y CONV8 (Figura 5.40 y Figura 5.41) de las redes *Peripheral-CNN* y *Deepfovea-CNN*.

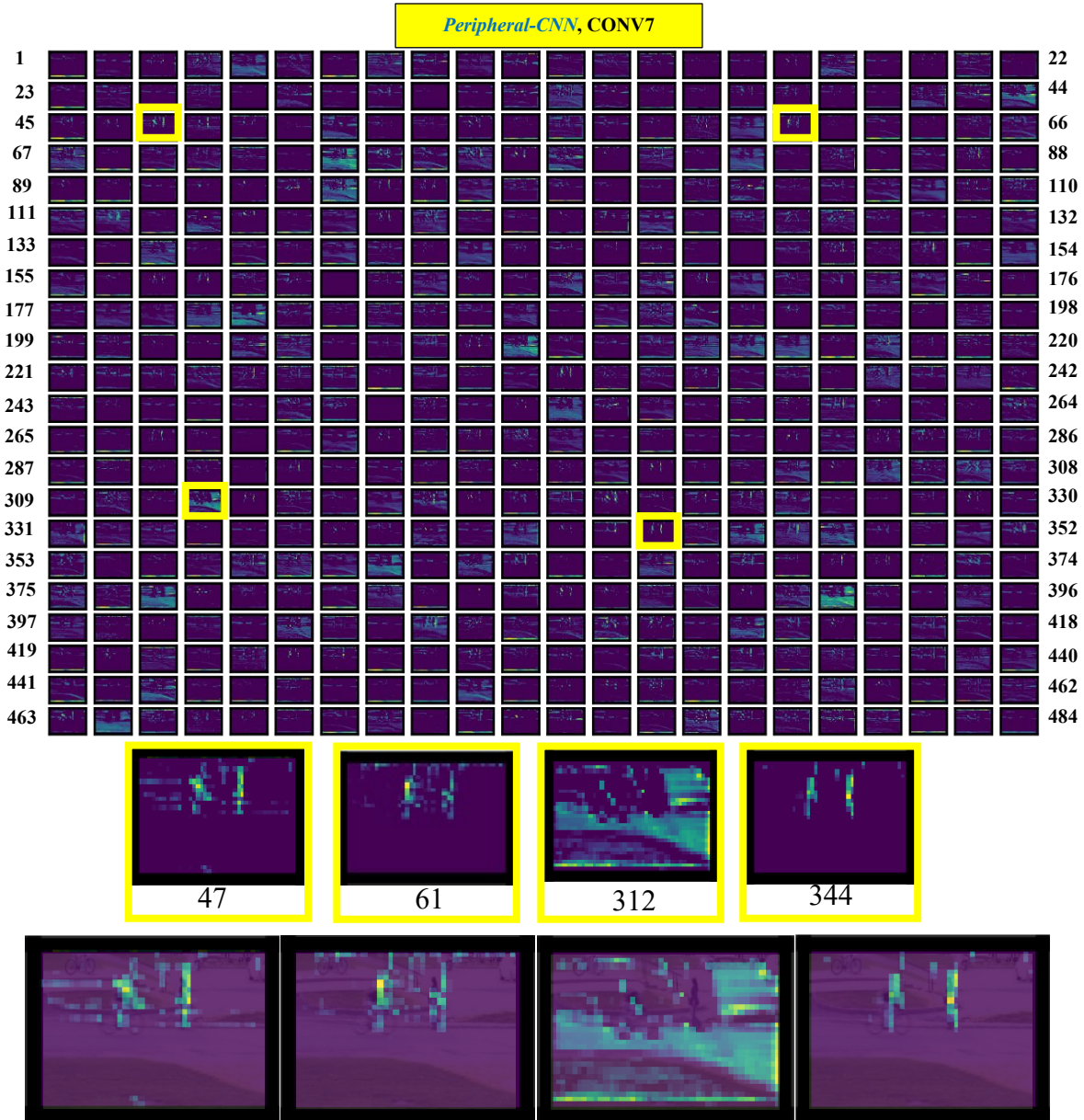
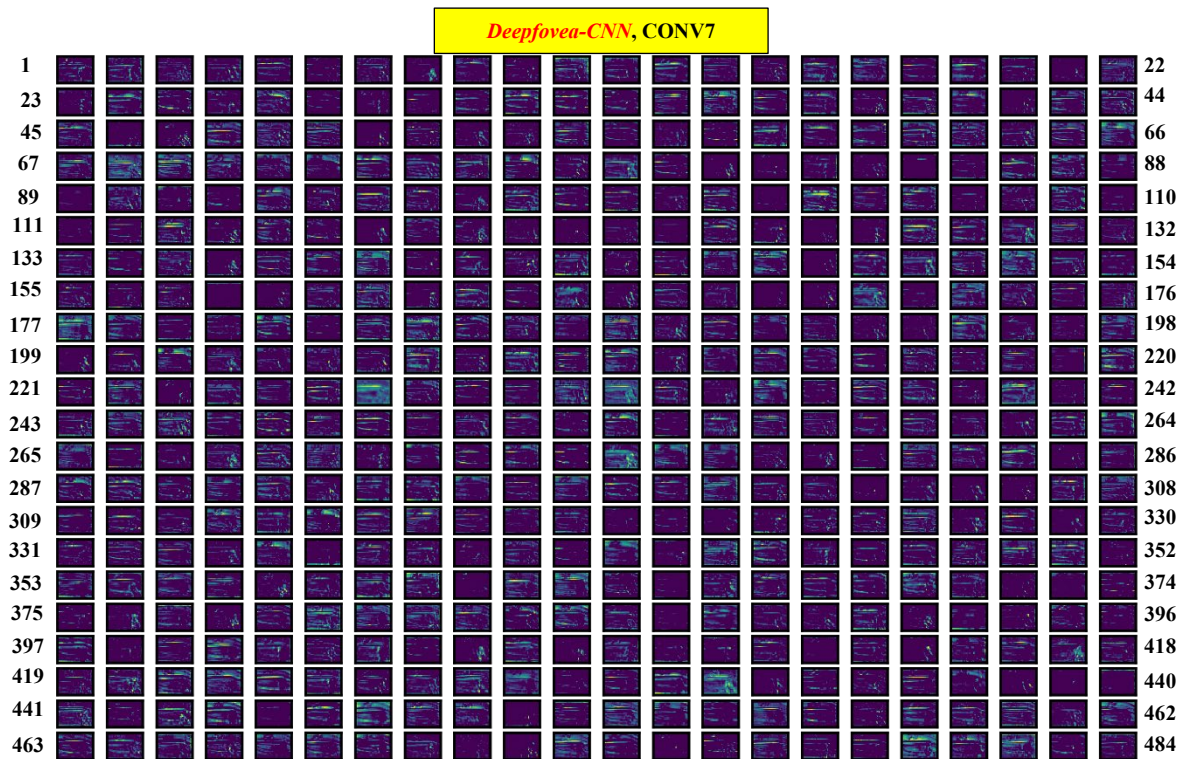
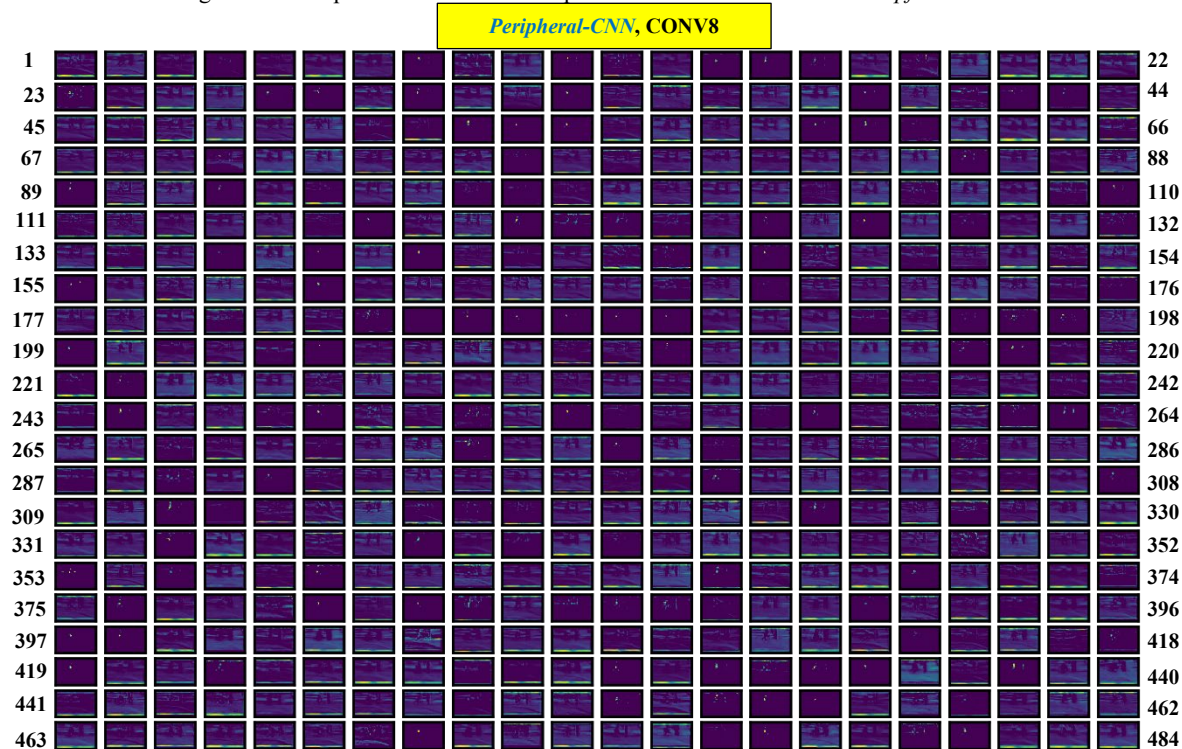


Figura 5.38. Mapas de características capa convolucional CONV7 red *Peripheral-CNN*.

Figura 5.39. Mapas de características capa convolucional CONV7 red *Deepfovea-CNN*.Figura 5.40. Mapas de características capa convolucional CONV8 red *Peripheral-CNN*.

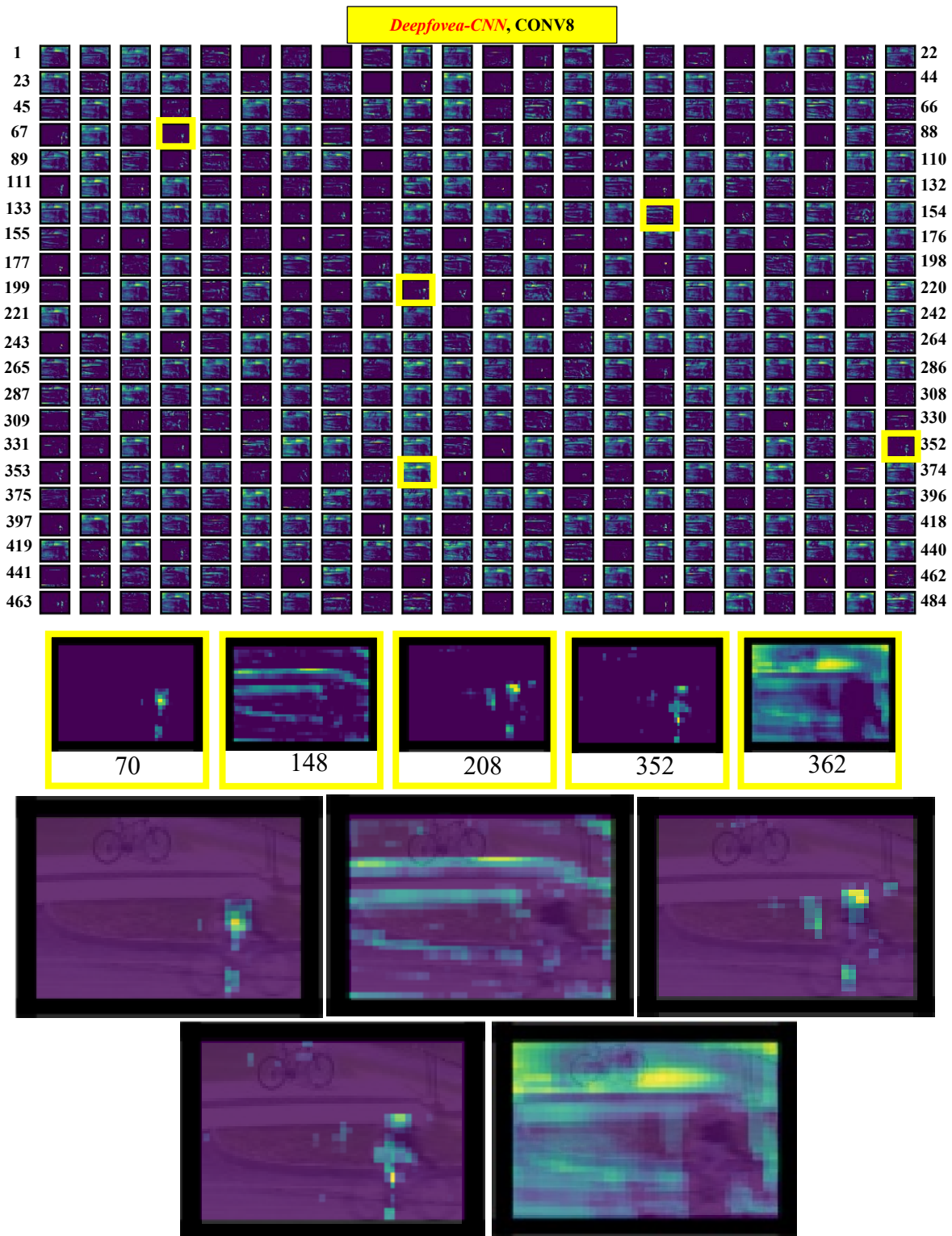


Figura 5.41. Mapas de características capa convolucional CONV8 red *Deepfovea-CNN*.

Las imágenes anteriores (Figura 5.40 y Figura 5.41) corresponden a la última capa CONV8 de la parte *Encoder*. Por lo tanto, a continuación, se muestran los mapas generados en la etapa *Decoder*. La primera capa de la etapa *Decoder* es ESTIMATION1, que podría considerarse una estimación preliminar o a baja resolución de la detección del movimiento de los objetos de interés. En la Figura 5.42 se aprecia la capa ESTIMATION1, contrario a lo esperado, en la red *Peripheral-CNN* esta estimación inhibe la información de los objetos de interés y se observa la mayor activación cerca de las orillas de arriba y abajo de la imagen, las cuales pueden ser debido a la agregación de *padding* entre capas para obtener los tamaños de resolución deseados. Por otra parte, en la red *Deepfovea-CNN* se observa una imagen un tanto caótica.

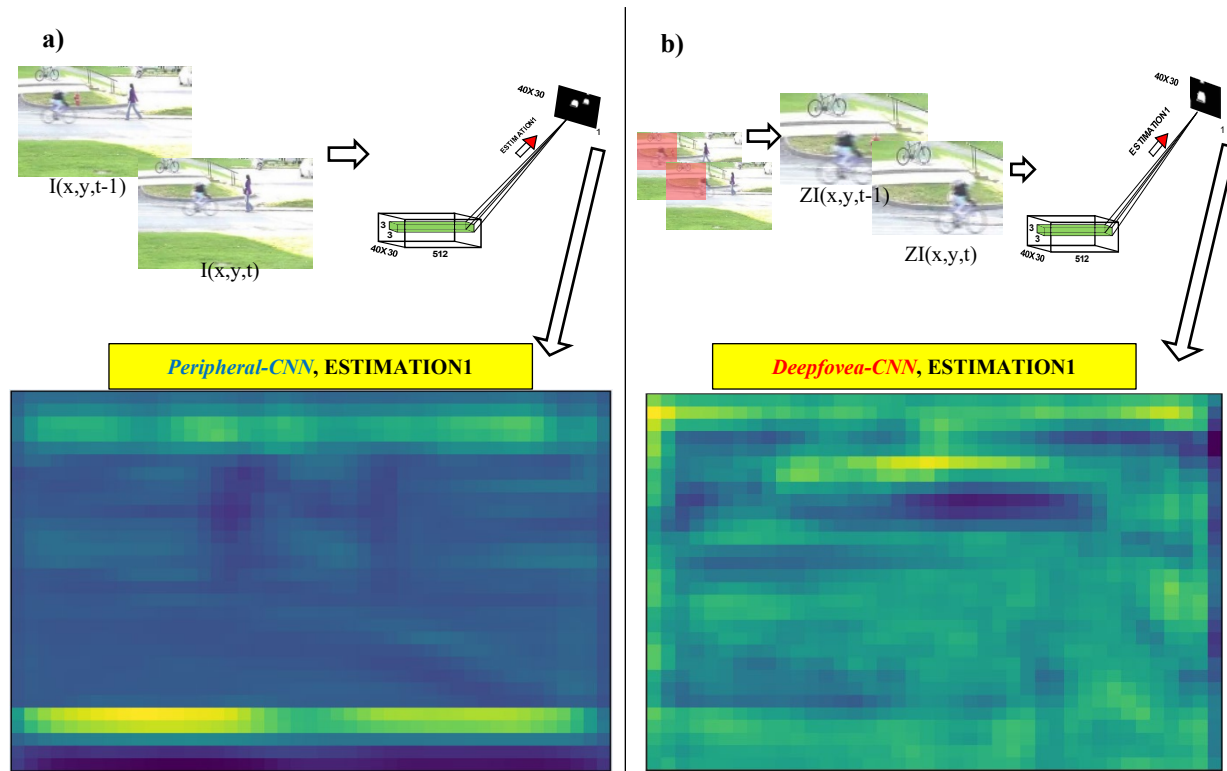


Figura 5.42. Mapa de características capa ESTIMATION1. Redes: a) *Peripheral-CNN*, b) *Deepfovea-CNN*

La siguiente capa de la etapa *Decoder* es CONCAT1, que involucra la combinación de información de varios niveles de profundidad de la red (Figura 5.43). La capa CONCAT1 (

Figura 5.46) para la red *Peripheral-CNN* se forma de la unión de la convolución transpuesta TCONV1 de la capa CONV8 (Figura 5.44), la capa de muestro hacia arriba UPSAMPLE1 de la capa ESTIMATION1 (Figura 5.45) y la capa CONV4 (Figura 5.32). Por su parte, la capa

CONCAT1 para la red *Deepfovea-CNN* se observa en la Figura 5.49, formada por la capa TCONV1 (Figura 5.47), CONV4 (Figura 5.33) y UPSAMPLE1 (Figura 5.48).

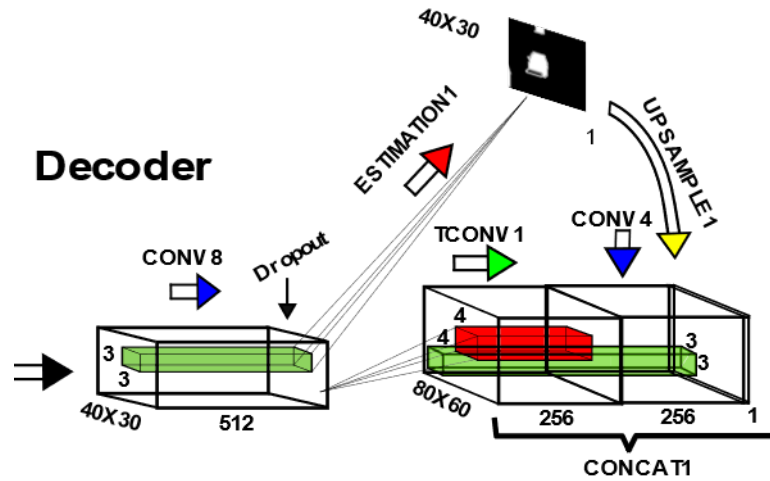


Figura 5.43. Capa CONCAT1 de las redes *Peripheral-CNN* y *Deepfovea-CNN*.

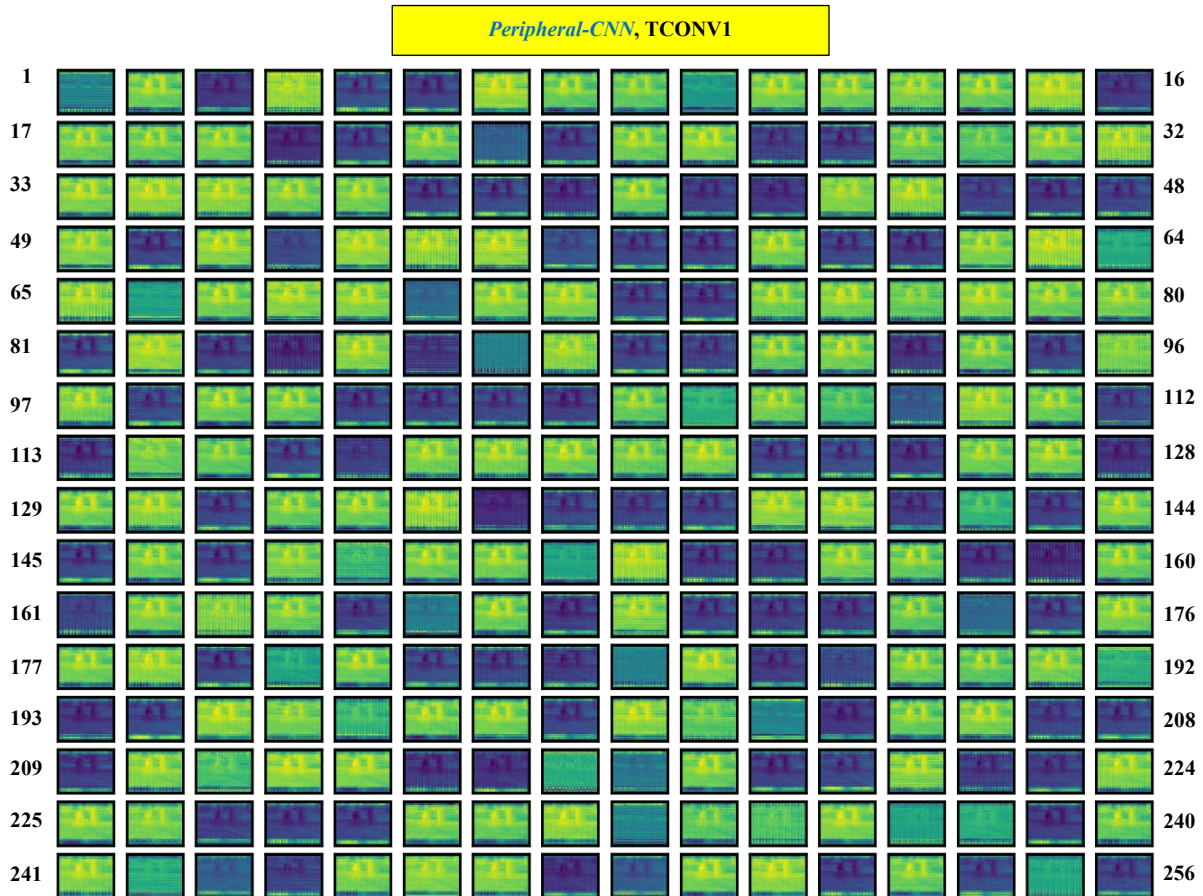


Figura 5.44. Mapas de características capa TCONV1 red *Peripheral-CNN*.

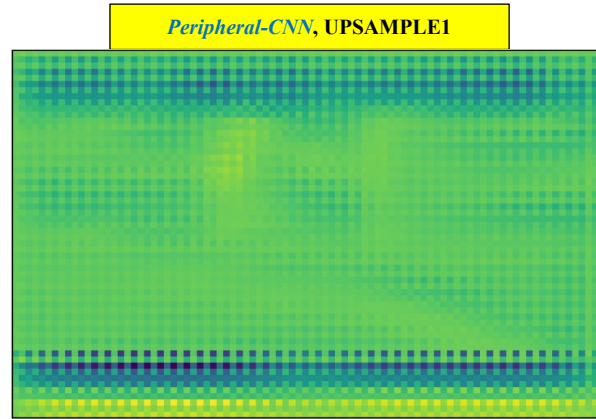


Figura 5.45. Mapas de características capa UPSAMPLE1 red *Peripheral-CNN*.

En la Figura 5.46 se muestra la concatenación CONCAT1 de los mapas de características TCONV1, CONV4 y UPSAMPLE1 de la red *Peripheral-CNN*. Se puede observar claramente la separación entre los mapas de características de la convolución CONV4 y los de TCONV1, ya que estos últimos tienen una mayor activación. La unión de estos mapas de características generará la siguiente estimación ESTIMATION2, la cual posee una mayor resolución.

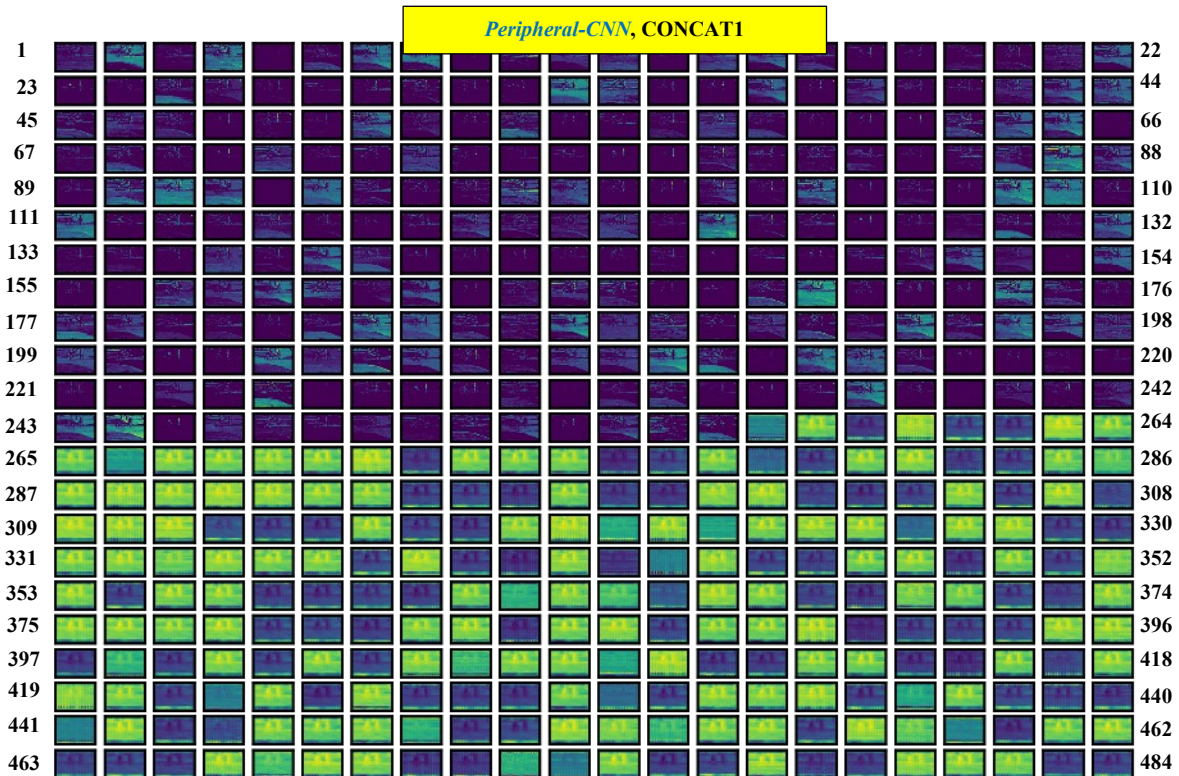


Figura 5.46. Concatenación CONCAT1 de los mapas de características TCONV1, CONV4 y UPSAMPLE1.

A continuación, se observan los mapas de características de las capas TCONV1 (Figura 5.47) y UPSAMPLE1 (Figura 5.48) involucrados en la capa de concatenación CONCAT1 de la red de fóvea profunda *Deepfovea-CNN* (Figura 5.49).

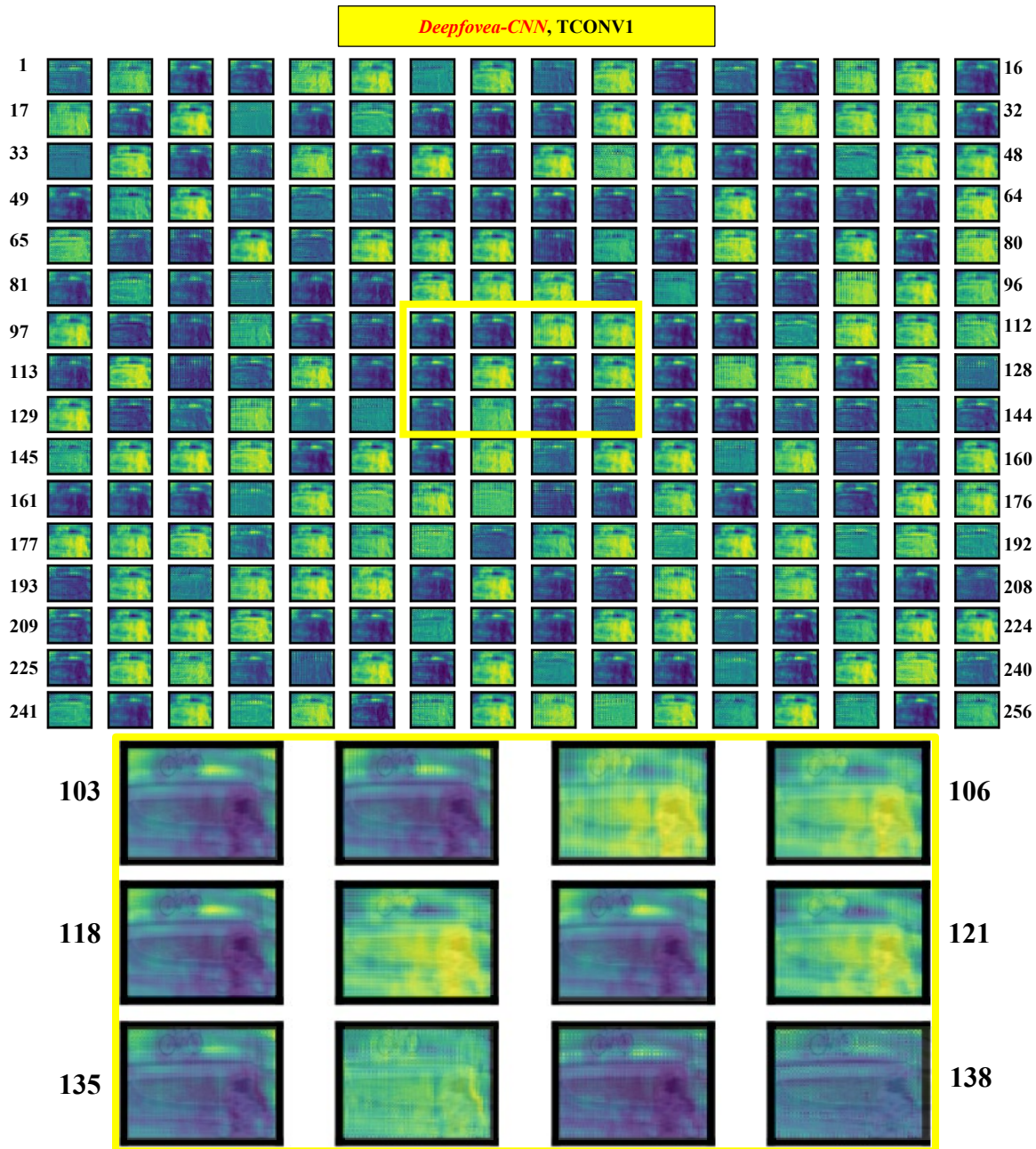


Figura 5.47. Mapas de características capa TCONV1 red *Deepfovea-CNN*.

Se puede observar que la capa TCONV1 de la Figura 5.47 cuenta en su mayoría con dos clases de filtros; los que realzan la información del objeto de interés, en este caso el ciclista y los que lo inhiben.

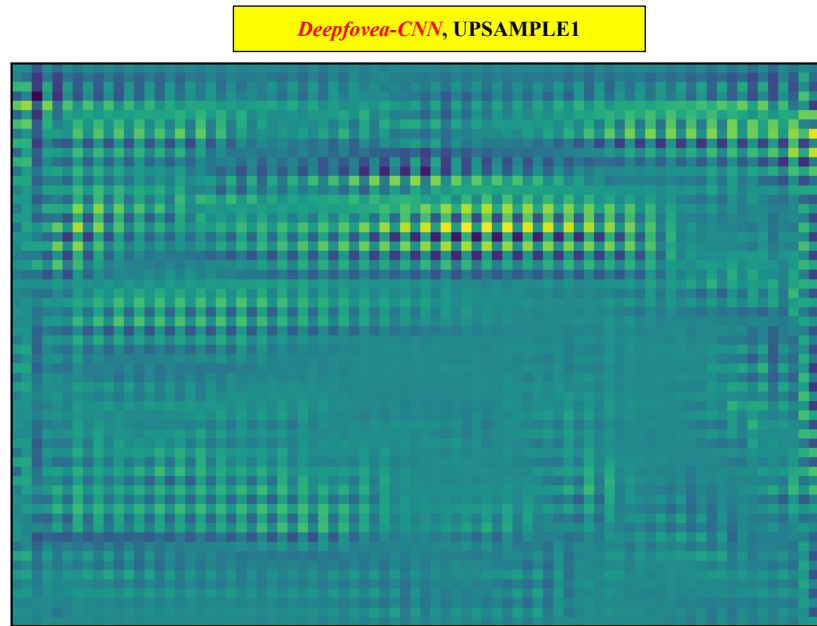


Figura 5.48. Mapas de características capa UPSAMPLE1 red *Deepfovea-CNN*.

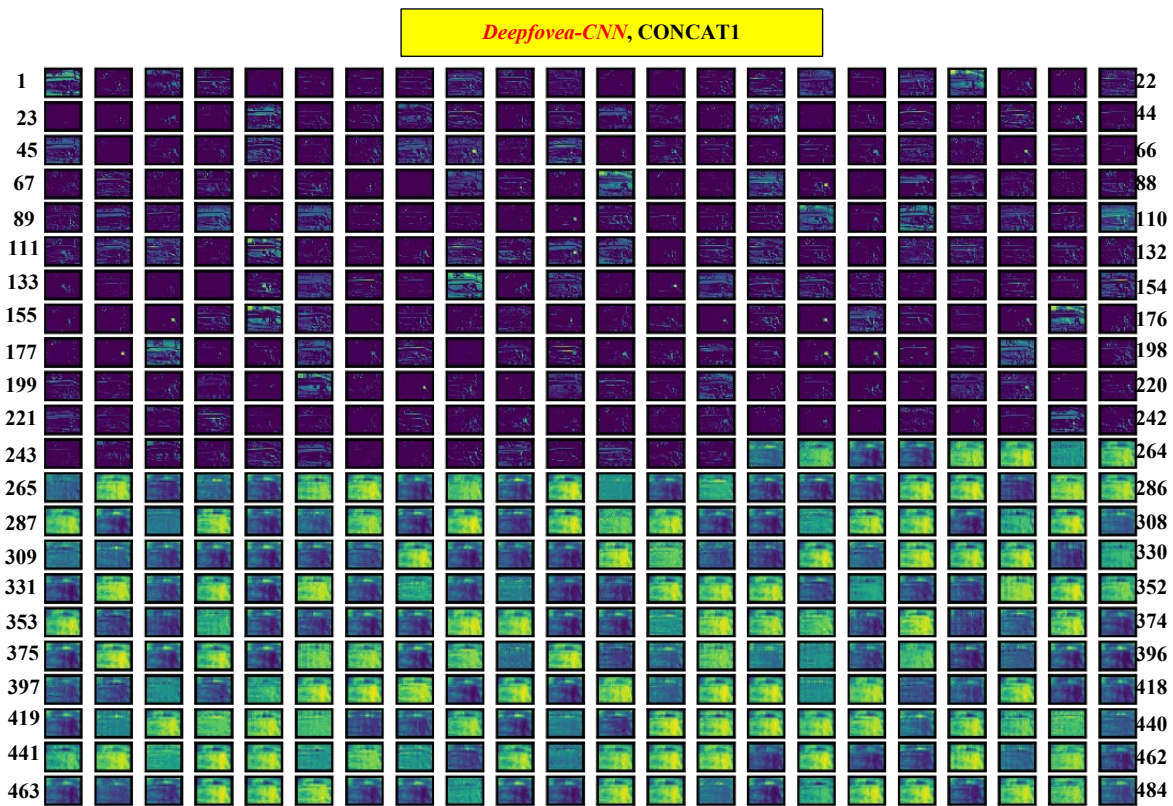


Figura 5.49. Mapas de características capa CONCAT1 red *Deepfovea-CNN*.

A continuación, en la Figura 5.50 se observa la segunda estimación ESTIMATION2 de cada una de las redes, la cuál es generada por la convolución de la capa CONCAT1 y un filtro 3x3.

Para la siguiente capa de concatenación CONCAT2 (*Peripheral-CNN*: Figura 5.54, *Deepfovea-CNN*: Figura 5.55) se usan las capas TCONV2 (*Peripheral-CNN*: Figura 5.51, *Deepfovea-CNN*: Figura 5.52), CONV3 (*Peripheral-CNN*: Figura 5.30, *Deepfovea-CNN*: Figura 5.31) y UPSAMPLE2 (*Peripheral-CNN* y *Deepfovea-CNN*: Figura 5.53). Para ambas redes, la capa TCONV2 se obtiene de la convolución transpuesta de su capa CONCAT1, la capa CONV3 proviene de la parte *Encoder* y la capa UPSAMPLE2 es el muestreo hacia arriba de la ESTIMATION2.

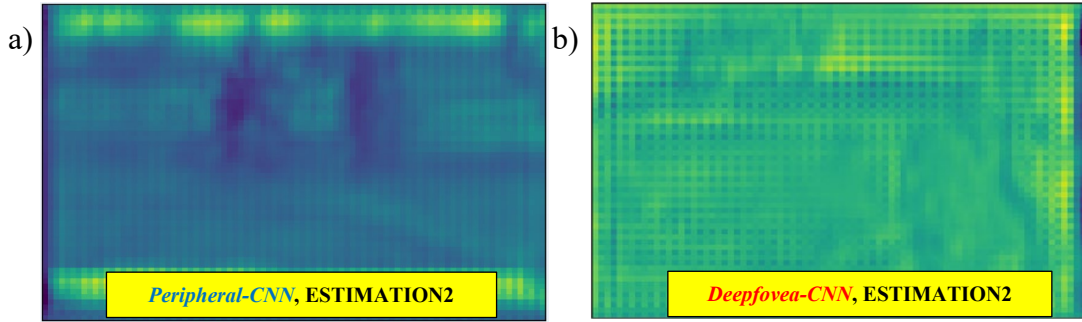


Figura 5.50. Mapas de características, capa ESTIMATION2: a) *Peripheral-CNN*, b) *Deepfovea-CNN*.

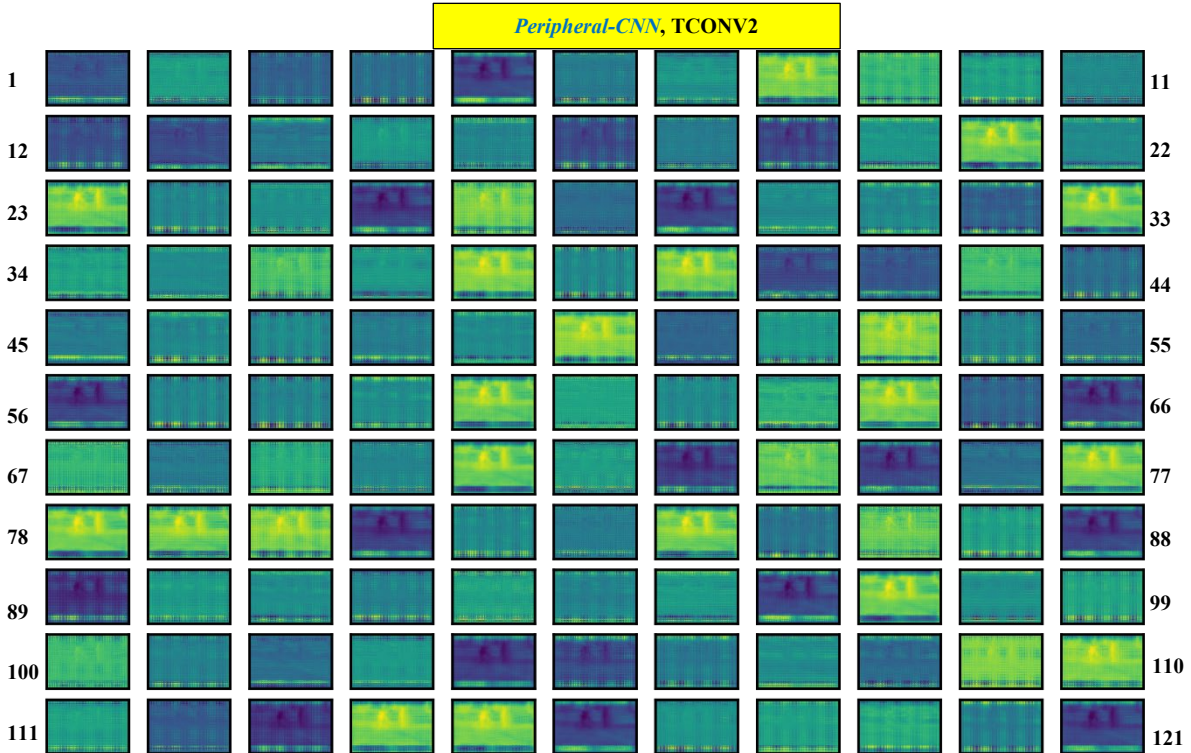


Figura 5.51. Mapas de características capa TCONV2 red *Peripheral-CNN*.

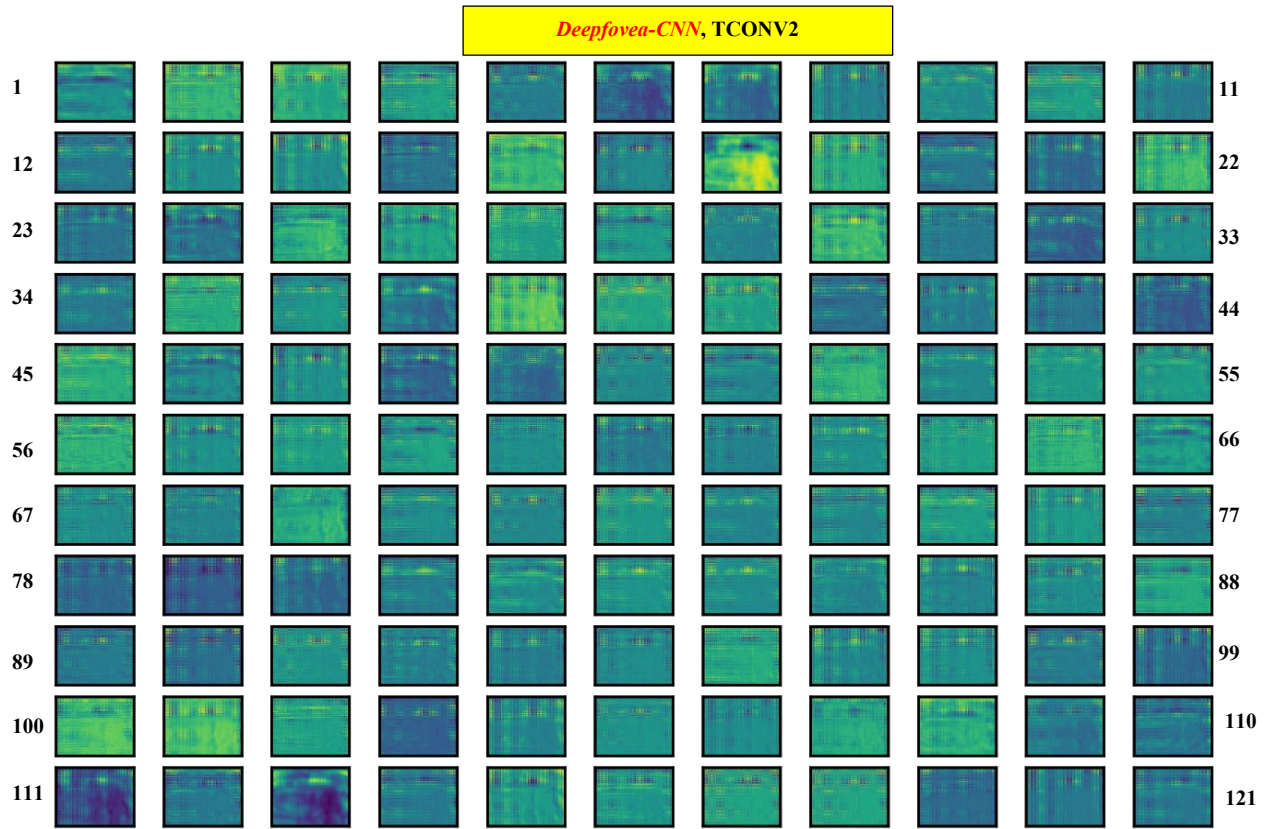


Figura 5.52. Mapas de características capa TCONV2 red *Deepfovea-CNN*.

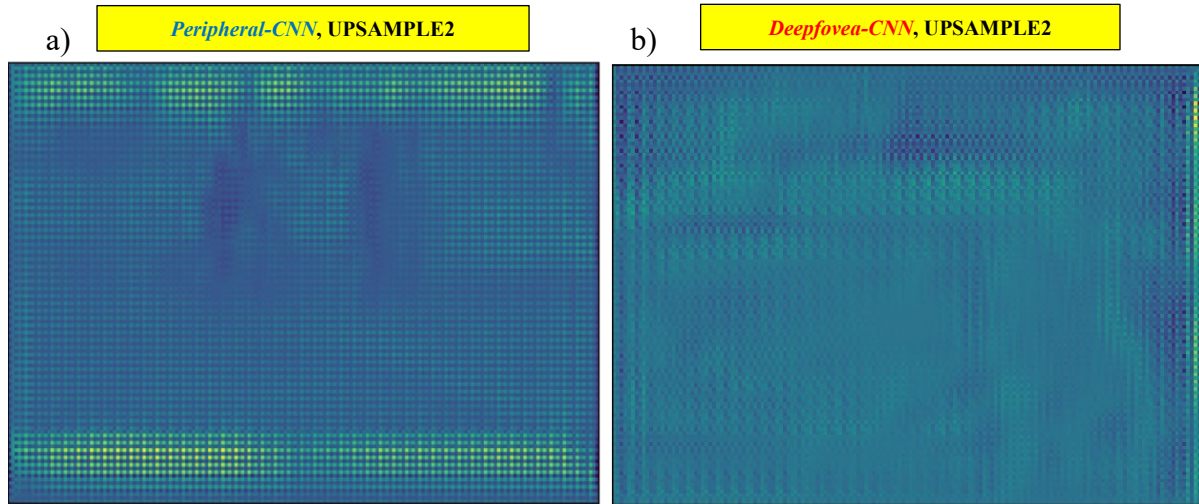


Figura 5.53. Mapas de características, UPSAMPLE2: a) red *Peripheral-CNN*, b) *Deepfovea-CNN*.

En las siguientes figuras se muestran las capas de CONCAT2 para la red *Peripheral-CNN* y *Deepfovea-CNN*, se puede apreciar a detalle las características de bajo y alto nivel que están involucradas para generar la tercera capa de estimación, ESTIMATION3.

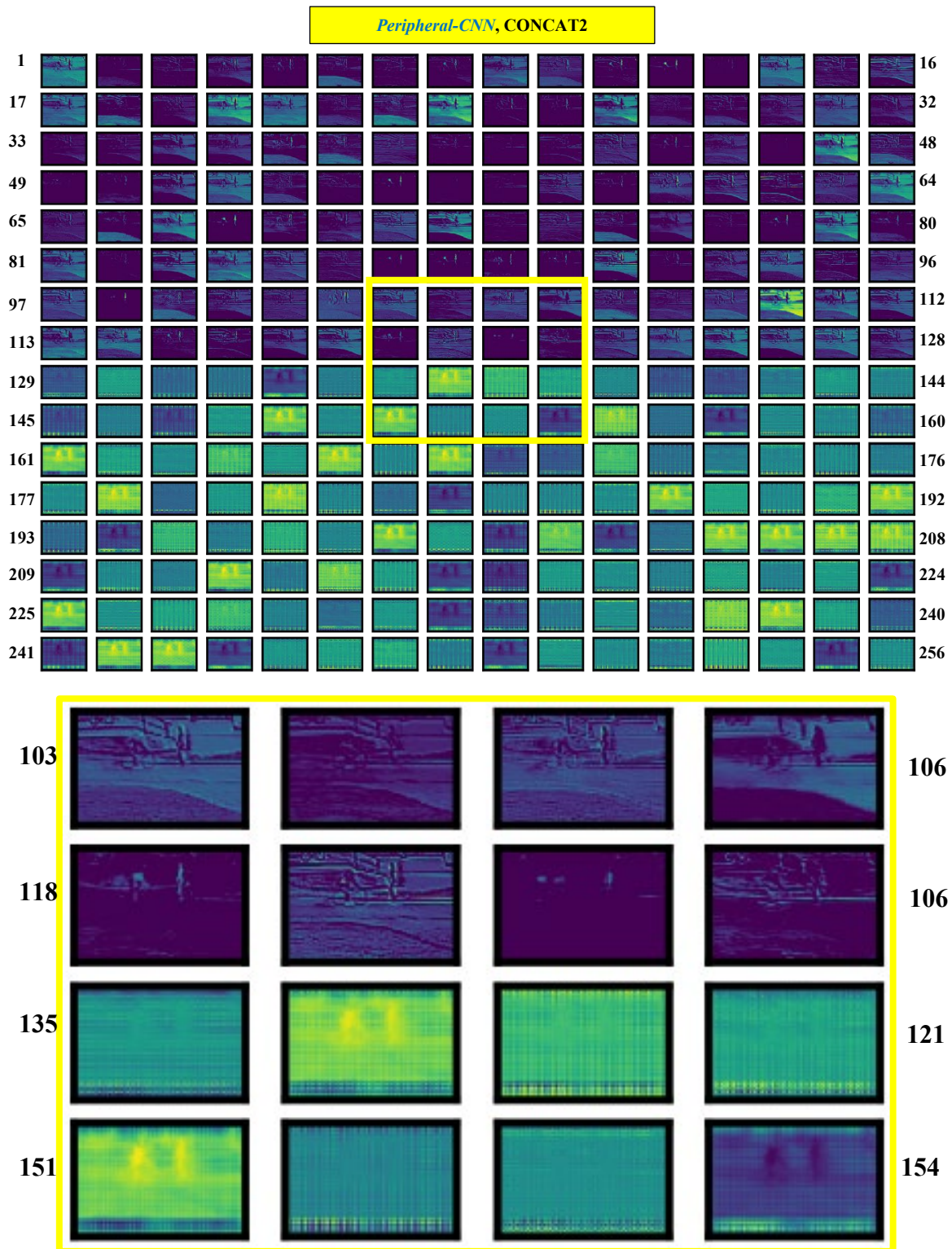


Figura 5.54. Información de bajo y alto nivel utilizada en la capa de concatenación, CONCAT2 de los mapas de características TCONV2, CONV3 y UPSAMPLE2 en la red *Peripheral-CNN*.

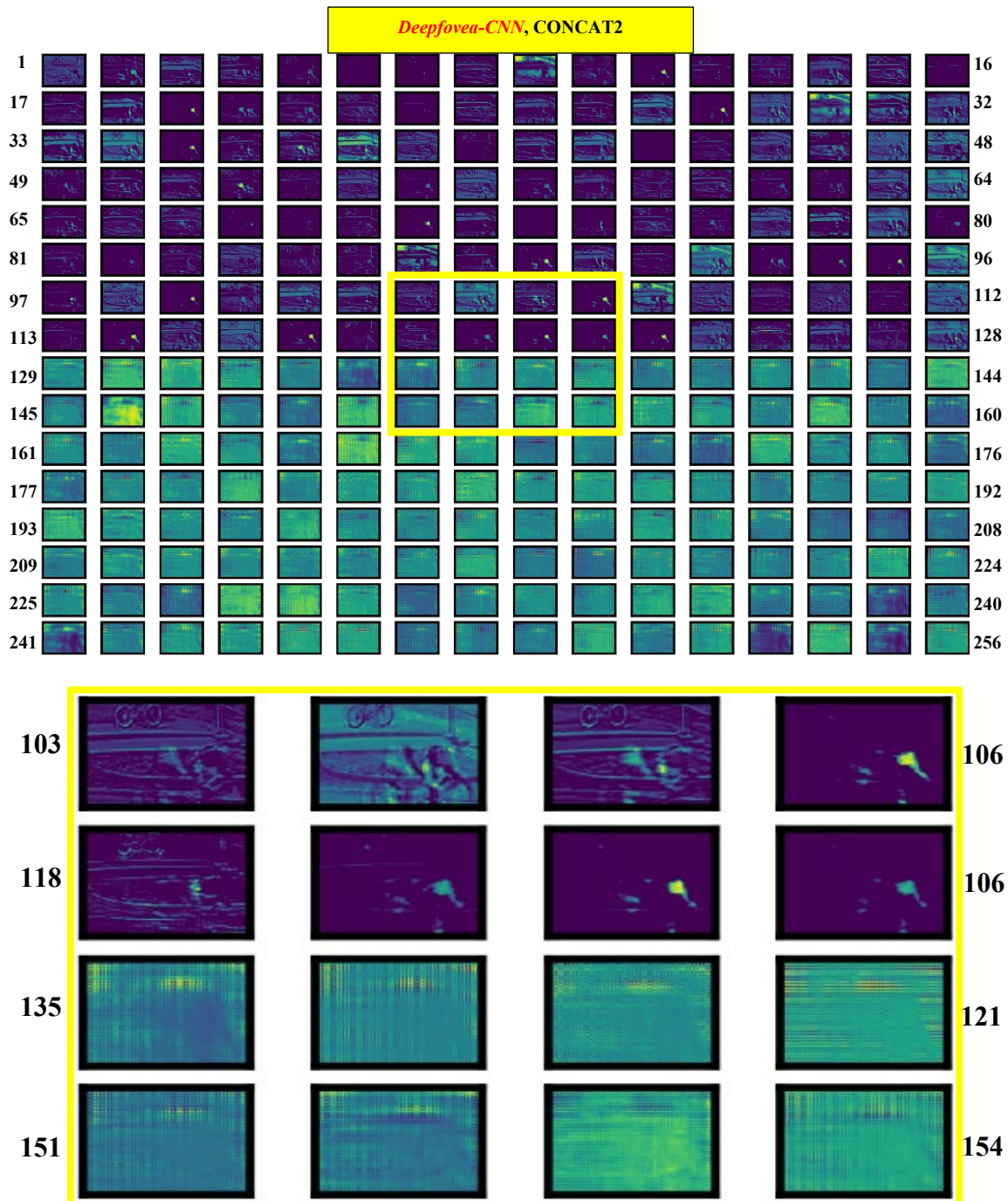


Figura 5.55. Información de bajo y alto nivel utilizada en la capa de concatenación CONCAT2 en la red *Deepfovea-CNN*.

En la Figura 5.56 se muestra el mapa generado por la capa ESTIMATION3 para ambas redes, se puede observar que esta estimación se encuentra más definida, con una mayor resolución que resalta el movimiento de los objetos de interés. ESTIMATION3 es la última capa de la red *Deepfovea-CNN*, la cual pasa después a través de una función de activación sigmoideal que otorga un valor de probabilidad de 0 a 1, de pertenencia a objeto en movimiento, el mapa de características

final de *Deepfovea-CNN* se muestra en la Figura 5.58b. Para la red *Peripheral-CNN* la siguiente capa es la concatenación *CONCAT3* (Figura 5.59), que usa las capas *TCONV3* (Figura 5.57), *CONV1* (Figura 5.15) y *UPSAMPLE3* (Figura 5.58a). La capa *TCONV3* se obtiene de la convolución transpuesta de *CONCAT2*, la capa *CONV1* proviene de la parte *Encoder* y la capa *UPSAMPLE3* es el muestro hacia arriba de la *ESTIMATION3*. Todas las capas anteriores se visualizan a continuación.

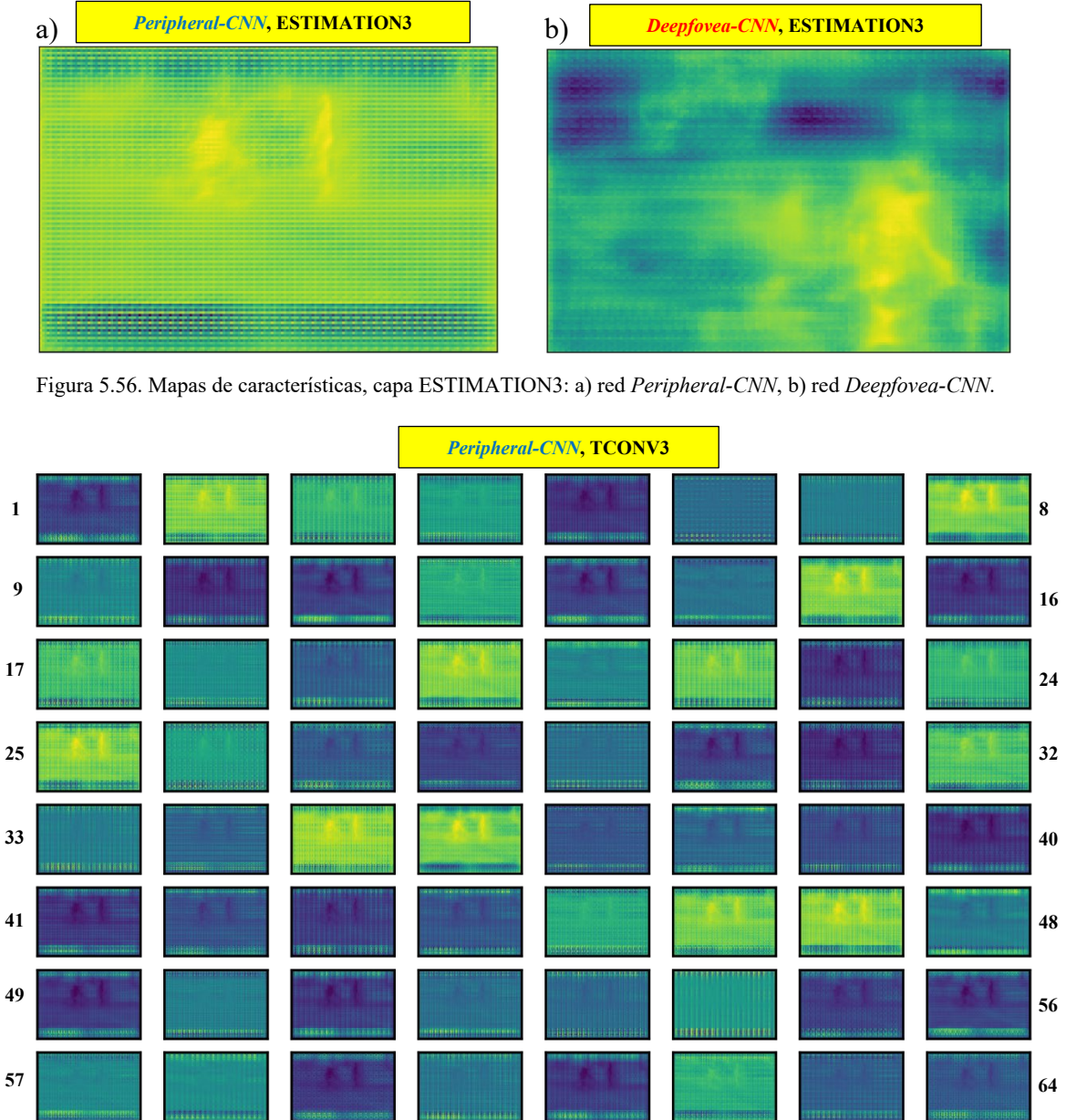


Figura 5.57. Mapas de características capa *TCONV3* red *Peripheral-CNN*.

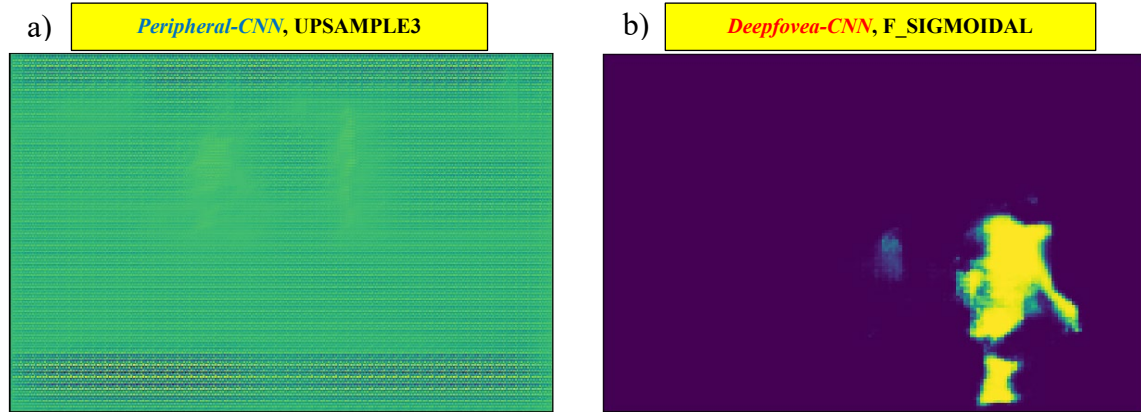


Figura 5.58. Mapas de características capa UPSAMPLE3 red *Peripheral-CNN* y capa final, red *Deepfovea-CNN*.

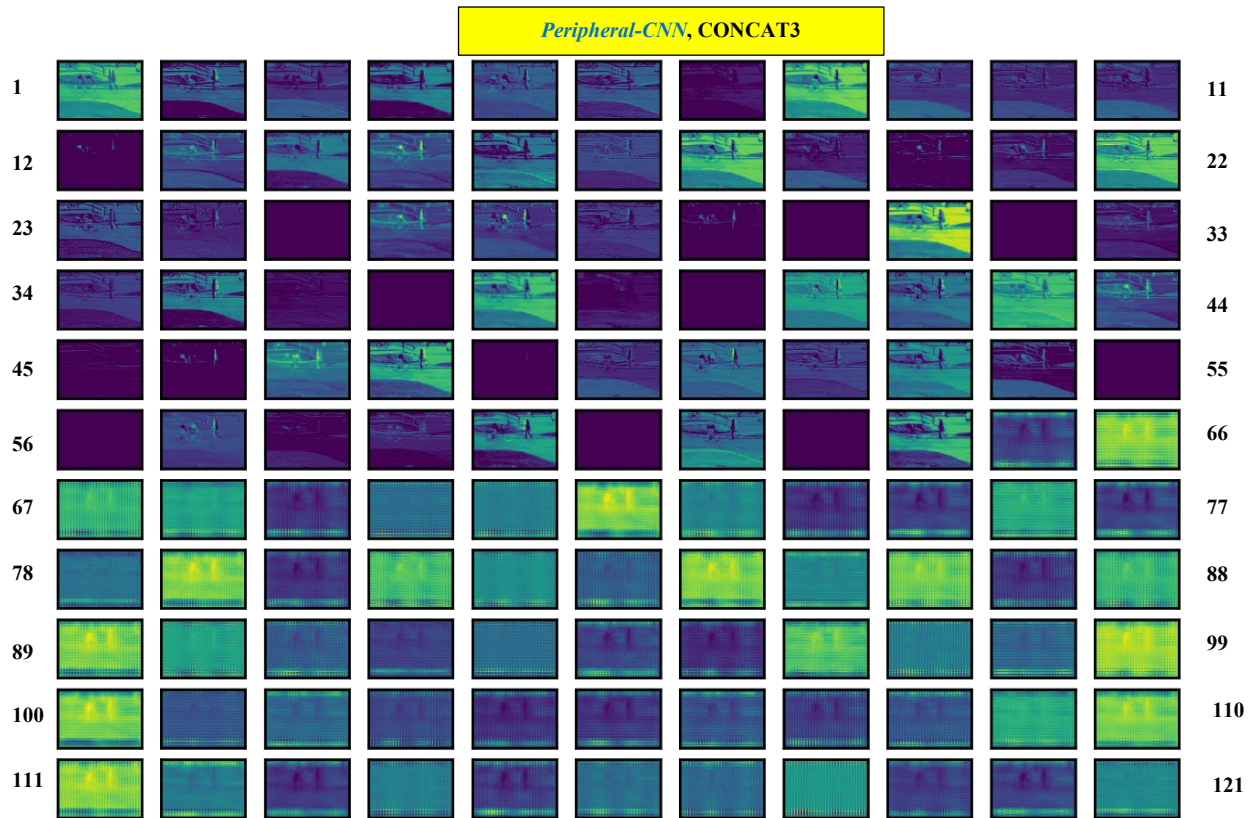


Figura 5.59. Información de bajo y alto nivel utilizada en la capa de concatenación CONCAT3 en la red *Peripheral-CNN*.

A continuación, la Figura 5.60 muestra la última capa (DETECTION) de la red *Peripheral-CNN*, esta capa es la estimación final de la red y se obtiene de realizar la convolución de la capa CONCAT3 con un filtro de tamaño 3x3. Se puede observar que esta estimación es más fina y a este punto la imagen del mapa de características generado tiene una resolución igual a la imagen de entrada. Además, en esta estimación se aprecia fácilmente una mayor activación donde se localizan los objetos dinámicos de interés.

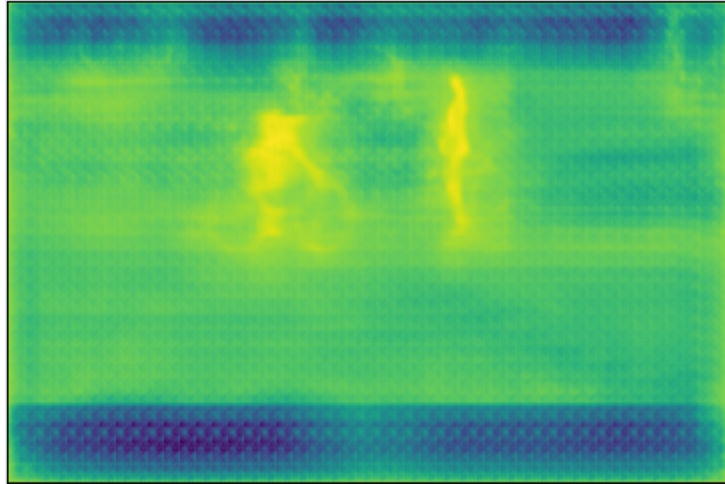


Figura 5.60. Mapas de características capa DETECTION.

Finalmente se pasa la estimación final por una función de activación sigmoideal que resalta los pixeles con mayor activación y otorga un valor de probabilidad de 0 a 1 de pertenencia al a objeto en movimiento. Figura 5.61.



Figura 5.61. Mapas de características capa DETECTION después de la función sigmoideal en red *Peripheral-CNN*.

La Figura 5.62, muestra la capa final de ambas redes *Peripheral-CNN* y *Deepfovea-CNN*, así como la combinación de estas capas para obtener la detección final del movimiento. Se observa que la estimación realizada es bastante acertada para detectar correctamente los objetos dinámicos en el video.

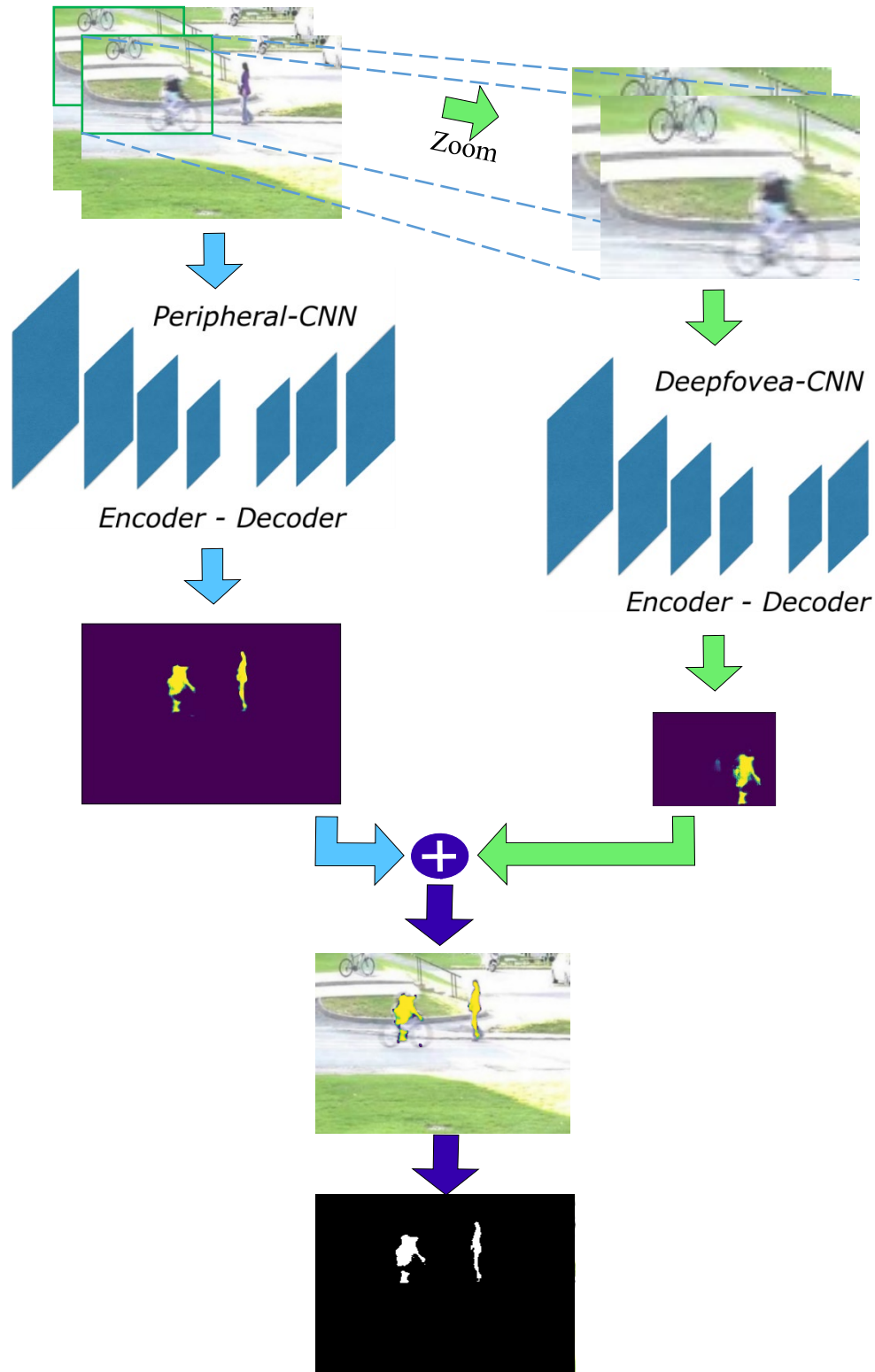


Figura 5.62. Combinación de las estimaciones de ambas redes (*Peripheral-CNN* y *Deepfovea-CNN*) para obtener la detección final del movimiento.

5.4.1. Discusión y conclusiones acerca de los filtros aprendidos de las redes *Peripheral-CNN* y *Deepfovea-CNN* del algoritmo *DeepFoveaNet*

Uno de los principales puntos que aclarar sobre los filtros aquí expuestos es que estos no fueron elegidos ni diseñados para obtener algún comportamiento específico. Se definió el número y tamaño de los filtros en el diseño de la arquitectura, pero el resultado final y su comportamiento depende totalmente del ajuste interno de sus parámetros mediante el algoritmo del entrenamiento. Los parámetros internos de los filtros inicialmente son aleatorios y luego son actualizados automáticamente a través del proceso de entrenamiento hasta que el error en la clasificación es lo suficientemente pequeño para lograr realizar la tarea deseada. Una vez entrenados, los filtros se vuelven sensibles a algún tipo de información (color, forma, textura, etc.) y gracias a esto se les conoce como extractores de características.

Debido a que los filtros de una red convolucional en su mayoría son arreglos que varían en profundidad su visualización se vuelve complicada, y aun visualizándolos, su forma o estructura no manifiesta de qué tipo de filtro se trata ni a qué características es sensible. Por lo que uno de los métodos que ayuda a identificar a qué características son sensibles los filtros es realizar la propagación hacia enfrente de la red e ir obteniendo los mapas de características generados por los filtros. Los mapas de características poseen un solo canal en profundidad y pueden ser visualizados fácilmente, de tal manera que se puede intuir de qué tipo de filtro se trata al observar los rasgos que se encuentran resaltados en el mapa de características.

Del análisis realizado en los mapas de características generados mediante los filtros aprendidos por las redes *Peripheral-CNN* y *Deepfovea-CNN* del algoritmo *DeepFoveaNet*, se observó lo siguiente. En la primera capa convolucional CONV1, existen filtros que son sensibles a características de alta frecuencia y refuerzan los bordes en la imagen (Figura 5.17 y Figura 5.18). Hay filtros de color y pasa bajas que son sensibles a texturas del escenario (Figura 5.19 y Figura 5.20). Además, existen filtros que resaltan la forma de los objetos de interés o alguna característica del escenario (Figura 5.21 y Figura 5.22). En las siguientes capas convolucionales CONV2, CONV3 y CONV4 (Figuras 99-104) siguen existiendo algunos filtros de texturas y bordes, pero conforme se avanza a capas más profundas (CONV5, CONV6) aumenta el número de filtros enfocados a detectar los objetos de interés y características del escenario (Figuras 105-108). Finalmente, en las últimas capas del *Encoder* (CONV7, CONV8) se observan mapas de

características de filtros que realzan rasgos o detalles que poseen los objetos de interés como en la Figura 5.38 y Figura 5.41, en las cuales se pueden observar los mapas de características generados por filtros que realzan la forma de las personas o incluso algún aspecto de éstas como su playera o piernas. El comportamiento observado en los mapas de características de las redes *Peripheral-CNN* y *Deepfovea-CNN* coincide con algunos esfuerzos encontrados en la literatura enfocados a visualizar las características aprendidas de las redes convolucionales [179][180][181][182][183]. En todos estos trabajos se ha encontrado que los filtros de las primeras capas convolucionales principalmente extraen características de bajo nivel como bordes, colores y texturas. Los filtros de capas intermedias son más específicos a formas de objetos o texturas y en las últimas capas existe un mayor número de filtros sensibles a objetos, partes de éstos y texturas. La Figura 5.63 fue tomada de [183] y muestra los conceptos semánticos que se observan en las capas de la red AlexNet entrenada para clasificación de imágenes en la base de datos ImageNet.

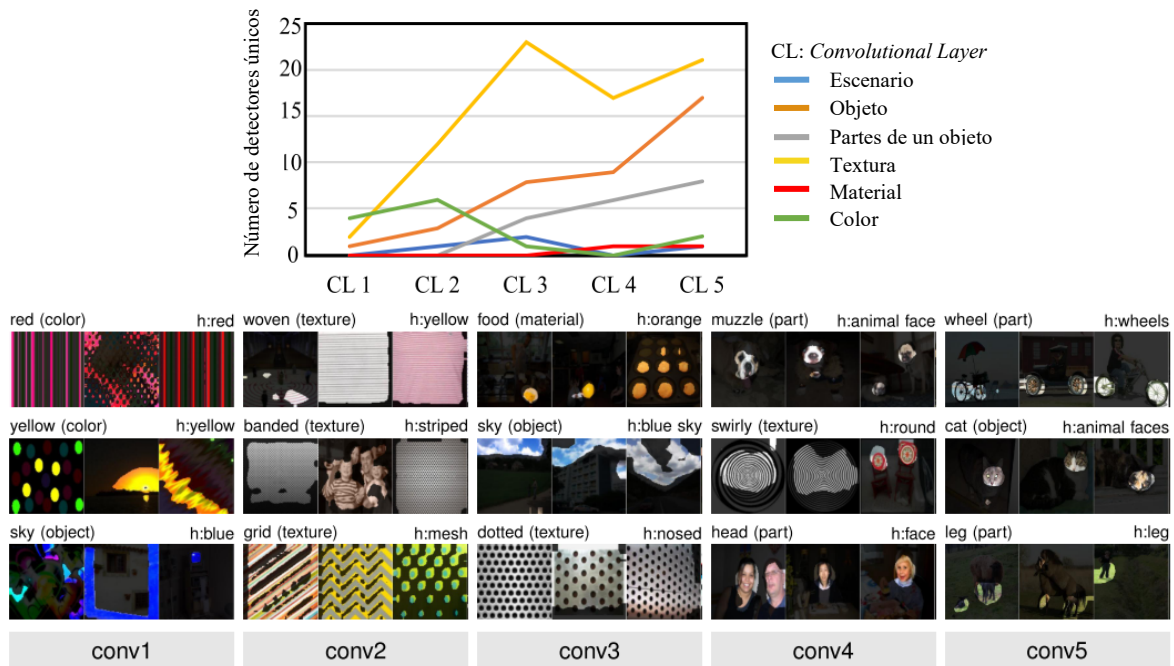


Figura 5.63. Interpretación de las características aprendidas por la red AlexNet, imagen tomada de [183].

Los trabajos mencionados previamente [179][180][181][182][183] ofrecen un método cualitativo para mejorar la interpretabilidad de las capas convolucionales y ayudan a adquirir alguna intuición para optimizar y depurar las arquitecturas. Sin embargo, la mayoría de estas técnicas de visualización han sido aplicadas en clasificación de imágenes u objetos y la visualización de los filtros de redes CNN en otras áreas como: estimación de profundidad, flujo

óptico, detección de movimiento, entre otras, continua sin ser explorada. Por lo que, el análisis de filtros mostrado en esta sección plantea una ruta para seguir realizando esfuerzos para lograr el entendimiento de las características aprendidas por redes tipo convolucionales aplicadas en detección de movimiento.

Un punto a destacar de la visualización de filtros realizada en este trabajo, es el hecho de que en las primeras capas de la red CONV1, CONV2 y CONV3 se observaron filtros que extraían la información de los objetos dinámicos en un tiempo t específico (Figura 5.23 y Figura 5.24). Observando este comportamiento, se infirió que podría tratarse de filtros espaciotemporales de Gabor (ETG) utilizados para detección de movimiento, también conocidos como filtros de Gabor 3D [161] y descritos en la sección 1.5.3. Por lo que, se realizó un experimento para comparar los resultados que se obtienen al utilizar filtros de espacio temporales de Gabor (ETG) contra los resultados obtenidos por filtros aprendidos de la red. Al visualizar los resultados obtenidos por ambos filtros se encontró que el comportamiento de los filtros aprendidos de la red era muy similar a los filtros ETG para detección de movimiento, pero además los filtros aprendidos por la red resaltaban bordes y algunas partes del escenario haciendo más fácil la identificación de los cambios de texturas, como fue el caso del filtro 27 de la capa CONV1 (Figura 5.26). Aunque resulta sorprendente que los filtros aprendidos por la red de alguna manera se ajustaran para otorgar resultados similares a los filtros ETG, se requiere un estudio y esfuerzo mayor para comprender como algunos de estos filtros temporales también detectan otras características como texturas y bordes en la imagen. Un posible trabajo futuro sería ampliar el análisis de este tipo de filtros en el dominio de la frecuencia y así poder identificar que frecuencias filtran e identificar si su rasgo distintivo es el de un pasa bajas o un pasa altas y como se relaciona esto con los filtros espaciotemporales de Gabor para detectar movimiento, los cuales poseen la particularidad de estar localizados tanto en el dominio espacial como en el de la frecuencia.

Continuando con el análisis de las siguientes capas de las redes *Deepfovea-CNN* y *Peripheral-CNN*, para la parte *Decoder* se observó que las primeras estimaciones no son muy acertadas ESTIMATION1 (Figura 5.42) y ESTIMATION2 (Figura 5.50), pero para la ESTIMATION3 (Figura 5.56) ya se obtiene una estimación aceptable de la ubicación de los objetos dinámicos. Las capas de ESTIMATION se generan de la convolución de las capas de CONCAT (Figura 5.43) y como se observó éstas cuentan con información de bajo y alto nivel y que sirve para obtener

estimaciones del movimiento cada vez más finas y con mejor resolución. Por lo tanto, la combinación y agregación de la información de diferentes filtros con diferentes características hace que la estimación final sea lo suficientemente robusta para identificar los objetos dinámicos de interés en el video (Figura 5.60).

5.5. Conclusiones del método DeepFoveaNet

En este capítulo se propuso una idea novedosa y original para realizar un algoritmo de detección de movimiento mediante redes convolucionales inspirado en la visión biológica de las aves rapaces llamado DeepFoveaNet. El esquema modular del algoritmo propuesto emula la visión monocular de estas aves mediante dos redes convolucionales; de periferia y de enfoque. La red de periferia se encarga de detectar el movimiento general de la escena y la red de fovea profunda emula el efecto de enfoque y magnificación producido en la fovea profunda para captar características de los objetos a mayor detalle.

DeepFoveaNet, en comparación a esquemas de redes convolucionales del estado del arte, al igual que 2FraCNN, no depende de modelos de red previamente entrenados, ni de algoritmos generadores de fondo a la entrada de la red. Además, su arquitectura utiliza dos cuadros consecutivos a la entrada de la red, lo que facilita que la red aprenda representaciones espaciotemporales de la escena, contrario a los primeros lugares de la base de datos *CDnet14* que solo consideran información espacial.

Los resultados cuantitativos y cualitativos mostraron que el uso de la red de fovea profunda ayuda en la correcta detección de objetos dinámicos pequeños y evita que estos se pierdan durante los cuadros del video. DeepFoveaNet logró situarse dentro los primeros diez lugares en el ranking de desempeños sobre todas las categorías de video de la base de datos *CDnet14*, demostrando su robustez ante diferentes escenarios de video con altos índices de desempeño. Aunado a esto, DeepFoveaNet mostró consistencia con los resultados obtenidos al ser evaluado mediante los videos de la base de datos LASIESTA, aun y cuando en la gran mayoría de videos solo se usaron 100 cuadros para su entrenamiento.

CAPÍTULO VI. RESULTADOS Y CONCLUSIONES GENERALES

En el capítulo I se resumió información a manera general del estado del arte de algoritmos enfocados a detección de movimiento. En total se encontraron 79 algoritmos, los cuales se clasificaron y resumieron por tipo de técnica en doce categorías, véase Tabla 1.1. Además, se hizo un análisis de las tendencias en cuanto a algoritmos de detección de movimiento y se encontró que los modelos estadísticos avanzados y las técnicas de aprendizaje profundo eran las más utilizadas por los autores, Figura 1.1. Aunado a esto, el capítulo I resume las principales métricas de evaluación, bases de datos y describe a detalle los principales elementos involucrados en el área de aprendizaje profundo y la percepción de movimiento mediante la visión biológica.

En el capítulo II se presentó una recopilación de los trabajos más representativos encontrados en la literatura respecto a algoritmos de detección de movimiento mediante aprendizaje profundo. Cada uno de los algoritmos se explicó de manera general y resumió su información más relevante en la Tabla 2.1. la Tabla 2.1 permitió identificar los problemas abiertos de cada uno de los algoritmos analizados. Dentro de los principales desafíos que enfrentan estos algoritmos son: altos tiempos de procesamiento, requieren un gran número de cuadros de entrenamiento, algunos enfoques no consideran información temporal del video, dependen de redes preentrenadas e imágenes de fondo generadas por otros algoritmos de sustracción de fondo. Además, otras técnicas requieren del uso de regiones o parches para obtener la estimación de movimiento por cada pixel de la imagen, lo cual es computacionalmente demandante.

Por su parte, el capítulo III se describieron y analizaron algunos de los algoritmos que se han desarrollado en los últimos años, inspirados en la percepción biológica del movimiento. Dentro del análisis se encontró que la mayoría de los trabajos utilizaban redes SOM, se limitaban a proponer un modelo matemático y se evaluaban de una manera escasa en videos muy simples. Sin embargo, una excepción fueron los trabajos [33] y [173], el primero se evaluó parcialmente en la base de datos *CDnet14* y el segundo completamente en la misma base de datos. Además, la técnica [173] exploró el uso de redes convolucionales para el diseño de una arquitectura bioinspirada y demostró poseer un buen desempeño. Por lo que, a raíz de este análisis surgió el interés en el capítulo V por buscar otros mecanismos, adaptaciones o estructuras de la visión biológica en animales.

En el capítulo IV, se propuso un nuevo algoritmo de detección de movimiento llamado 2FraCNN. El algoritmo propuesto se inspiró en la arquitectura de red FlowNet para la estimación de flujo óptico. 2FraCNN incorpora información temporal a la entrada de la red, un esquema de balanceo de clases y un esquema novedoso de aumento de datos en el entrenamiento. Gracias al esquema de entrenamiento de aumento de datos y balanceo de clases 2FraCNN no depende del uso de redes preentrenadas para lograr un aprendizaje correcto y se entrena desde cero con sólo 200 cuadros del video. Los resultados experimentales se mostraron en la Tabla 4.3, siendo los mejores desempeños las categorías de vídeo *Baseline*, *CameraJitter*, *DynamicBackground*, *Shadow* y *Thermal* con una F-Measure superior a 0.926. Además, 2FraCNN demostró su robustez al clasificarse dentro de los primeros lugares de la base de datos *CDnet14* y en la posición 12 en el instrumento de evaluación PVADN que considera aspectos tanto prácticos como teóricos y utiliza 119 algoritmos para realizar comparaciones de desempeño.

En el capítulo V, se propuso una idea novedosa y original para realizar un algoritmo de detección de movimiento mediante redes convolucionales inspirado en la visión biológica de las aves rapaces llamado DeepFoveaNet. El esquema modular del algoritmo propuesto emuló la visión monocular de estas aves mediante dos redes convolucionales; de periferia y de enfoque. La red de periferia se encargó de detectar el movimiento general de la escena y la red de fovea profunda simula el efecto de enfoque producido en la fovea profunda en las aves rapaces. En la Tabla 5.3 se demostró que el uso de la red de fovea profunda ayuda en la correcta detección de objetos dinámicos pequeños y evita que estos se pierdan durante los cuadros del video.

DeepFoveaNet, posee las mismas ventajas de 2FraCNN ya que lo utiliza como base para su arquitectura, por lo tanto, no depende de modelos de red previamente entrenados, ni de algoritmos generadores de fondo a la entrada de la red y su arquitectura facilita que la red aprenda representaciones espaciotemporales de la escena. El esquema modular propuesto de DeepFoveaNet emula la visión monocular de las aves rapaces mediante una red de periferia (*Peripheral-CNN*) que se encarga de detectar el movimiento general de la escena y una red de fovea profunda (*DeepFovea-CNN*) que imita el efecto de enfoque y magnificación producido en la fovea profunda de estas aves, para captar características de los objetos a mayor detalle. Las evaluaciones de DeepFoveaNet se realizaron en las bases de datos *CDnet14* y *LASIESTA* y se presentaron en las secciones 5.3.1 y 5.3.2. DeepFoveaNet logró situarse en el lugar siete del ranking de desempeños sobre todas las

categorías de video de la base de datos *CDnet14* con un valor general en la métrica F-Measure de 0.842, Tabla 5.4. Además, obtuvo un desempeño general de 0.87 en la métrica F-Measure en los 48 videos de LASIESTA (Tabla 5.5), demostrando su robustez ante diferentes escenarios de video con altos índices de desempeño.

Por otro lado, dentro del mismo capítulo V se realizó un análisis de los mapas de características que indican la información aprendida de los filtros de las redes de periferia (*Peripheral-CNN*) y fovea profunda (*Deepfovea-CNN*). De este análisis se observó que los primeros filtros se encargan de extraer algunas características del escenario como colores, texturas o bordes y se observan algunos filtros temporales. Estos filtros temporales aprendidos por la red se compararon con filtros espaciotemporales de Gabor y se encontró que ambos poseen comportamientos similares. En la parte *Encoder*, conforme se avanza a capas más profundas va aumentando considerablemente el número de filtros enfocados a detectar los objetos de interés y parte de sus características. Respecto a la parte *Decoder* de la red, se observó que las primeras estimaciones no son muy acertadas, pero la combinación de información de capas de bajo y alto nivel hace que para la ESTIMATION3 se obtenga una estimación aceptable de la ubicación de los objetos dinámicos de interés en el video.

REFERENCIAS

- [1] M. I. Chacon-Murguia, R. Sandoval-Rodriguez, and O. Arias-Enriquez, “Human gait feature extraction including a kinematic analysis toward robotic power assistance,” *Int. J. Adv. Robot. Syst.*, vol. 9, pp. 1–9, 2012.
- [2] T. Kimura, M. Ohashi, K. Crailsheim, T. Schmickl, R. Odaka, and H. Ikeno, “Tracking of Multiple Honey Bees on a Flat Surface,” in *Fifth International Conference on Emerging Trends in Engineering and Technology*, 2012, pp. 36–39.
- [3] M. Shakeri and H. Zhang, “Real-time bird detection based on background subtraction,” *Proc. World Congr. Intell. Control Autom.*, pp. 4507–4510, 2012.
- [4] M. Baillieul and P. Scheunders, “On-line determination of the velocity of simultaneously moving organisms by image analysis for the detection of sublethal toxicity,” *Water Res.*, vol. 32, no. 4, pp. 1027–1034, 1998.
- [5] J. García, A. Gardel, I. Bravo, J. Luis Lázaro, M. Martínez, and D. Rodríguez, “Detección y Seguimiento de Personas Basado en Estereovisión y Filtro de Kalman,” *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 9, no. 4, pp. 453–461, 2012.
- [6] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, “Wallflower: principles and practice of background maintenance,” *Comput. Vision, 1999. Proc. Seventh IEEE Int. Conf.*, pp. 255–261, 1999.
- [7] T. Bouwmans, “Traditional and recent approaches in background modeling for foreground detection: An overview,” *Comput. Sci. Rev.*, vol. 11–12, no. 1, pp. 31–66, 2014.
- [8] Y. Wang, Z. Luo, and P. M. Jodoin, “Interactive deep learning method for segmenting moving objects,” *Pattern Recognit. Lett.*, vol. 96, no. 1, pp. 66–75, 2016.
- [9] D. Sakkos, H. Liu, J. Han, L. Shao, and L. S. Org, “End-to-end video background subtraction with 3d convolutional neural networks,” *Multimed. Tools Appl.*, vol. 77, no. 17, pp. 23023–23041, 2017.
- [10] L. A. Lim and H. Y. Keles, “Foreground Segmentation Using a Triplet Convolutional Neural Network for Multiscale Feature Encoding,” *arXiv Pre-print*, vol. 1801.02225, no. v1, pp. 1–14, 2018.
- [11] A. Guzmán Pando, “Análisis y comparación de algoritmos de modelado de fondo de secuencias de video,” Master Thesis, Chihuahua Institute of Technology, Mexico, 2018.
- [12] M. I. Chacon-Murguia, A. Guzman-Pando, G. Ramirez-Alonso, and J. A. Ramirez-

- Quintana, “A novel instrument to compare dynamic object detection algorithms,” *Image Vis. Comput.*, vol. 88, pp. 19–28, 2019.
- [13] A. Dosovitskiy *et al.*, “FlowNet: Learning Optical Flow with Convolutional Networks,” in *IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.
- [14] J. Heikkilä and O. Silven, “A real-time system for monitoring of cyclists and pedestrians,” *Proc. - 2nd IEEE Int. Work. Vis. Surveillance, VS 1999*, pp. 74–81, 1999.
- [15] N. J. B. McFarlane and C. P. Schofield, “Segmentation and tracking of piglets in images,” *Mach. Vis. Appl.*, vol. 8, no. 3, pp. 187–193, 1995.
- [16] H. Al-Khateeb and M. Petrou, “Automatic Change Detection Method for an Indoor Environment,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, 2010, pp. 53–58.
- [17] B. Nikolov and N. Kostov, “Motion Detection Using Adaptive Temporal Averaging Method,” *Radioengineering*, vol. 23, no. 2, pp. 652–658, 2014.
- [18] S. Mahmoudpour and M. Kim, “Robust foreground detection in sudden illumination change,” *Electron. Lett.*, vol. 52, no. 6, pp. 441–443, 2016.
- [19] R. Kalsotra, “Morphological based Moving Object Detection with Background Subtraction Method,” in *4th IEEE International Conference on Signal Processing, Computing and Control (ISPCC)*, 2017, pp. 305–310.
- [20] M.-H. Hung, J.-S. Pan, and C.-H. Hsieh, “A Fast Algorithm of Temporal Median Filter for Background Subtraction,” *J. Inf. Hiding Multimed. Signal Process.*, vol. 5, no. 1, pp. 33–40, 2014.
- [21] G. M. Rao, “Object Tracking System Using Approximate Median Filter , Kalman Filter and Dynamic Template Matching,” *Int. J. Intell. Syst. Appl.*, vol. 6, no. 5, pp. 83–89, 2014.
- [22] F. A. Díaz González and D. A. Arévalo Suárez, “Recursive median filter for background estimation and foreground segmentation in surveillance videos ,” *Comput. y Sist.*, vol. 19, no. 2, pp. 283–293, 2015.
- [23] O. El Harrouss, D. Moujahid, and H. Tairi, “Motion detection based on the combining of the background subtraction and spatial color information,” in *2015 Intelligent Systems and Computer Vision, ISCV 2015*, 2015, pp. 2–5.
- [24] W. Hossain and M. N. Das, “Moving Object Detection in Dynamic Backgrounds for Surveillance Systems,” in *IEEE International Conference on Advanced Communication*

- Control and Computing Technologies (ICACCCT)*, 2014, no. 978, pp. 1476–1479.
- [25] K. K. Delibasis, T. Goudas, and I. Maglogiannis, “A novel robust approach for handling illumination changes in video segmentation,” *Eng. Appl. Artif. Intell.*, vol. 49, pp. 43–60, 2016.
 - [26] K. Wang, Y. Liu, C. Gou, and F. Wang, “A Multi-view Learning Approach to Foreground Detection for Traffic Surveillance Applications,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 4144–4158, 2016.
 - [27] F. Feiran, F. Ming, and Y. Huamin, “Temporal difference method based on positive and negative energy distribution in moving objects detection,” in *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2017, pp. 1–5.
 - [28] Y. T. Chan, S. J. Wang, and C. H. Tsai, “Real-time foreground detection approach based on adaptive ensemble learning with arbitrary algorithms for changing environments,” *Inf. Fusion*, vol. 39, pp. 154–167, 2018.
 - [29] Z. Chen and T. Ellis, “A self-adaptive Gaussian mixture model,” *Comput. Vis. Image Underst.*, vol. 122, no. 1, pp. 35–46, 2014.
 - [30] C. Spampinato, S. Palazzo, and I. Kavasidis, “A texon-based kernel density estimation approach for background modeling under extreme conditions,” *Comput. Vis. Image Underst.*, vol. 122, pp. 74–83, 2014.
 - [31] M. I. Chacon-Murguia and D. Urias-Zavala, “A DTCNN Approach on Video Analysis: Dynamic and Static Object Segmentation,” *Stud. Comput. Intell.*, vol. 551, pp. 315–336, 2014.
 - [32] G. Ramírez-Alonso and M. I. Chacón-Murguía, “Auto-Adaptive Parallel SOM Architecture with a modular analysis for dynamic object segmentation in videos,” *Neurocomputing*, vol. 175, no. a, pp. 990–1000, 2016.
 - [33] J. A. Ramirez-Quintana and M. I. Chacon-Murguia, “An Adaptive Unsupervised Neural Network Based on Perceptual Mechanism for Dynamic Object Detection in Videos with Real Scenarios,” *Neural Process. Lett.*, vol. 42, no. 3, pp. 665–689, 2015.
 - [34] J. A. Ramirez-Quintana and M. I. Chacon-Murguia, “Self-adaptive SOM-CNN neural system for dynamic object detection in normal and complex scenarios,” *Pattern Recognit.*, vol. 48, no. 4, pp. 1133–1145, 2015.
 - [35] R. Guo and H. Qi, “Partially-sparse restricted boltzmann machine for background modeling

- and subtraction,” in *2013 12th International Conference on Machine Learning and Applications, ICMLA 2013*, 2013, pp. 209–214.
- [36] M. Giordano and M. De Gregorio, “Background Modeling by Weightless Neural Networks,” *New Trends Image Anal. Process. -- ICIAP 2015 Work. Lect. Notes Comput. Sci.*, vol. 9281, no. 1, pp. 493–501, 2015.
- [37] B. N. Subudhi, S. Ghosh, and A. Ghosh, “Application of Gibbs–Markov random field and Hopfield-type neural networks for detecting moving objects from video sequences captured by static camera,” *Soft Comput.*, vol. 19, no. 10, pp. 2769–2781, 2015.
- [38] S.-C. Huang and B.-H. Do, “Radial Basis Function Based Neural Network for Motion Detection in Dynamic Scenes,” *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 114–125, 2013.
- [39] S. Lee, N. Kim, K. Jeong, I. Paek, H. Hong, and J. Paik, “Multiple moving object segmentation using motion orientation histogram in adaptively partitioned blocks for high-resolution video surveillance systems,” *Optik (Stuttg.)*, vol. 126, no. 19, pp. 2063–2069, 2015.
- [40] L. Gervasoni and R. Barbuzza, “Application of a background subtraction method using surveillance video images, In Spanish,” in *15th Argentine Symposium on Technology*, 2014, pp. 25–36.
- [41] J. Dou and J. Li, “Moving object detection based on improved VIBE and graph cut optimization,” *Opt. - Int. J. Light Electron Opt.*, vol. 124, no. 23, pp. 6081–6088, 2013.
- [42] O. Elharrouss, A. Abbad, D. Moujahid, and H. Tairi, “Moving object detection zone using a blockbased background model,” *IET Comput. Vis.*, vol. 12, no. 1, pp. 86–94, 2017.
- [43] D. Yang, C. Zhao, X. Zhang, and S. Huang, “Background Modeling by Stability of Adaptive Features in Complex Scenes,” *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1112–1125, 2018.
- [44] A. Shahbaz and K. Jo, “Exploiting Color Spaces for the Task of Foreground Detection,” in *17th International Conference on Control, Automation and Systems (ICCAS2017)*, 2017, no. 3, pp. 1343–1345.
- [45] M. Chen, Q. Yang, Q. Li, G. Wang, and M. Yang, “Spatiotemporal Background Subtraction Using Minimum Spanning Tree and Optical Flow,” in *Computer Vision–ECCV 2014*, 2014, pp. 521–534.
- [46] S. Jiang and X. Lu, “WeSamBE: A Weight-Sample-Based Method for Background

- Subtraction,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. 99, pp. 1–11, 2017.
- [47] H. Sajid and S.-C. S. Cheung, “Background Subtraction for Static & Moving Camera,” in *International Conference on Image Processing (ICIP)*, 2015, pp. 4530–4534.
- [48] H. Sajid and S.-C. S. Cheung, “Universal Multimode Background Subtraction,” *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3249–3260, 2017.
- [49] S. Wangsiripitak and W. Rattanapitak, “CDoTS: Change Detection on Time Series Background for Video Foreground Segmentation,” in *14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) CDoTS:*, 2017, pp. 400–403.
- [50] B. W. and Piotr, “AMBER: adapting multi-resolution background extractor,” in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, 2013, pp. 3417–3421.
- [51] Z. Zhong, B. Zhang, G. Lu, Y. Zhao, and Y. Xu, “An Adaptive Background Modeling Method for Foreground Segmentation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1109–1121, 2017.
- [52] J. D. Romero, M. J. Lado, and A. J. Mendez, “A Background Modeling and Foreground Detection Algorithm Using Scaling Coefficients Defined with a Color Model Called Lightness-Red-Green-Blue,” *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1243–1258, 2017.
- [53] J. Pan, W. Chen, and W. Peng, “A new moving objects detection method based on improved SURF algorithm,” in *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013, pp. 901–906.
- [54] J. Dou and J. Li, “Modeling the background and detecting moving objects based on Sift flow,” *Optik (Stuttg)*, vol. 125, no. 1, pp. 435–440, 2014.
- [55] F. J. López-Rubio and E. López-Rubio, “Foreground detection for moving cameras with stochastic approximation,” *Pattern Recognit. Lett.*, vol. 68, pp. 161–168, 2015.
- [56] H. Lu, Yingying Chen, Jinqiao Wang, “Learning Sharable Models For Robust Background Subtraction,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2015, pp. 1–6.
- [57] R. Azzam, M. S. Kemouche, N. Aouf, and M. Richardson, “Efficient visual object detection with spatially global Gaussian mixture models and uncertainties,” *J. Vis. Commun. Image Represent.*, vol. 36, no. C, pp. 90–106, 2016.

- [58] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, 2015.
- [59] M. Radolko, F. Farhadifard, E. Gutzeit, and U. F. Von Lukas, "Real time video segmentation optimization with a modified Normalized Cut," in *9th International Symposium on Image and Signal Processing and Analysis, ISPA 2015*, 2015, pp. 31–36.
- [60] P. Chiranjeevi and S. Sengupta, "Detection of moving objects using multi-channel kernel fuzzy correlogram based background subtraction," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 870–881, 2014.
- [61] D. K. Panda and S. Meher, "Detection of Moving Objects Using Fuzzy Color Difference Histogram Based Background Subtraction," *IEEE Signal Process. Lett.*, vol. 23, no. 1, pp. 45–49, 2016.
- [62] B. Yang and L. Zou, "Robust foreground detection using block-based RPCA," *Optik (Stuttg.)*, vol. 126, no. 23, pp. 4586–4590, 2015.
- [63] C. Guyon, T. Bouwmans, and E. H. Zahzah, "Foreground detection via robust low rank matrix decomposition including spatio-temporal constraint," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7728 LNCS, no. PART 1, pp. 315–320, 2013.
- [64] Y. Mo, H. Xu, and X. Ding, "Improved OR-PCA for Robust Foreground Detection," in *Chinese Automation Congress (CAC)*, 2017, pp. 5566–5571.
- [65] J. Feng and H. Xu, "Online Robust PCA via Stochastic Optimization," *Adv. Neural Inf. Process. Syst.*, vol. 26, pp. 1–9, 2013.
- [66] M. Shakeri and H. Zhang, "COROLA: A sequential solution to moving object detection using low-rank approximation," *Comput. Vis. Image Underst.*, vol. 146, pp. 27–39, 2015.
- [67] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 3, pp. 597–610, 2013.
- [68] X. Cao, L. Yang, and X. Guo, "Total Variation Regularized RPCA for Irregularly Moving Object Detection Under Dynamic Background," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 1014–1027, 2016.
- [69] S. Javed, S. H. Oh, A. Sobral, T. Bouwmans, and S. K. Jung, "Background Subtraction via

- Supapixel-Based Online Matrix Decomposition with Structured Foreground Constraints,” in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015, pp. 930–938.
- [70] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, “A self-adjusting approach to change detection based on background word consensus,” in *IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, 2015, pp. 990–997.
- [71] B. Zhang, D. Jiao, H. Pei, Y. Gu, and Y. Liu, “Infrared moving object detection based on local saliency and sparse representation,” *Infrared Phys. Technol.*, vol. 86, pp. 187–193, 2017.
- [72] S. Fan and X. Zeng, “A superpixel-based codebook model for real time foreground detection,” *I2MTC 2017 - 2017 IEEE Int. Instrum. Meas. Technol. Conf. Proc.*, 2017.
- [73] A. Staglianò, N. Noceti, A. Verri, and F. Odone, “Online space-variant background modeling with sparse coding,” *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2415–2428, 2015.
- [74] P. Dong, S. Wang, Y. Xia, D. Liang, and D. D. Feng, “Foreground detection with simultaneous dictionary learning and historical pixel maintenance,” *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5035–5049, 2016.
- [75] C. David and V. Gui, “Sparse Coding and Gaussian Modeling of Coefficients Average for Background Subtraction,” in *2013 8th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2013, pp. 223–228.
- [76] J. Šoda, I. Kuzmanić, and I. Vujović, “Stabilising illumination variations in motion detection for surveillance applications,” *IET Image Process.*, vol. 7, no. 7, pp. 671–678, 2013.
- [77] M. N. A. F, M. A. S. M, H. M. Ebeid, and M. F. Tolba, “Wavelet-Enhanced Detection of Small / Slow Object Movement in Complex Scenes,” in *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, 2016, pp. 172–180.
- [78] S. Li, D. Florencio, Y. Zhao, C. Cook, and W. Li, “Foreground Detection in Camouflaged Scenes,” in *IEEE International Conference on Image Processing*, 2017, pp. 4247–4251.
- [79] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, “Static and moving object detection using flux tensor with split gaussian models,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 420–424.
- [80] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S. I. Amari, “Bayesian Robust Tensor

- Factorization for Incomplete Multiway Data,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 4, pp. 736–748, 2016.
- [81] S. Bianco, G. Ciocca, and R. Schettini, “How Far Can You Get By Combining Change Detection Algorithms?,” *IEEE Trans. Image Process.*, vol. 2, pp. 1–10, 2015.
- [82] P. Tiefenbacher, M. Hofmann, D. Merget, and G. Rigoll, “PID-based regulation of background dynamics for foreground segmentation,” in *IEEE International Conference on Image Processing, ICIP*, 2014, pp. 3282–3286.
- [83] B. Dey and M. K. Kundu, “Efficient Foreground Extraction from HEVC Compressed Video for Application to Real-Time Analysis of Surveillance ‘Big’ Data,” *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3574–3585, 2015.
- [84] Z. Zhao, X. Zhang, Y. Fang, and S. Member, “Stacked Multi-layer Self-Organizing Map for Background Modeling,” *IEEE Trans. IMAGE Process.*, vol. 7149, no. c, pp. 1–10, 2015.
- [85] M. J. Shafiee, P. Siva, P. Fieguth, and A. Wong, “Embedded Motion Detection via Neural Response Mixture Background Modeling,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 19–26.
- [86] M. Braham and M. Van Droogenbroeck, “Deep background subtraction with scene-specific convolutional neural networks,” in *International Conference on Systems, Signals, and Image Processing*, 2016, pp. 3–6.
- [87] B. Heo, K. Yun, and J. Y. Choi, “Appearance and motion based deep learning architecture for moving object detection in moving camera,” in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 1827–1831.
- [88] M. Babaei, D. T. Dinh, and G. Rigoll, “A Deep Convolutional Neural Network for Background Subtraction,” *Comput. Res. Repos.*, vol. abs/1702.0, pp. 1–28, 2017.
- [89] L. Yang, J. Li, S. Member, Y. Luo, and S. Member, “Deep Background Modeling Using Fully Convolutional Network,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 1–9, 2017.
- [90] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks,” in *IEEE International Conference on Computer Vision (ICCV)*, 2016, pp. 1–9.
- [91] Z. Zivkovic and F. Van Der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780,

- 2006.
- [92] Z. Chen and T. Ellis, “Self-adaptive Gaussian mixture model for urban traffic monitoring system,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 1769–1776.
 - [93] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.
 - [94] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *European Conference on Computer Vision- ECCV*, 2006, pp. 404–417.
 - [95] C. Liu, J. Yuen, and A. Torralba, “Sift flow: Dense correspondence across scenes and its applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 978–994, 2011.
 - [96] C. Pojala and S. Sengupta, “Detection of moving objects using fuzzy correlogram based background subtraction,” in *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2011, pp. 255–259.
 - [97] G.-H. Liu and J.-Y. Yang, “Content-based image retrieval using color difference histogram,” *Pattern Recognit.*, vol. 46, no. 1, pp. 188–198, 2013.
 - [98] M. Aharon, M. Elad, and A. M. Bruckstein, “K-Svd: Design of Dictionaries for Sparse Representation,” *Spars ’05*, vol. 1, pp. 9–12, 2005.
 - [99] T. G. Kolda and B. W. Bader, “Tensor Decompositions and Applications,” *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
 - [100] M. De Gregorio, I. Cibernetica, E. C. I. Cnr, M. Giordano, P. Icar, and V. P. Castellino, “Change Detection with Weightless Neural Networks,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 409–413.
 - [101] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, “CDnet 2014: An Expanded Change Detection Benchmark Dataset,” *2014 IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 393–400, 2014.
 - [102] T. Bouwmans, L. Maddalena, and A. Petrosino, “Scene background initialization: A taxonomy,” *Pattern Recognit. Lett.*, vol. 0, pp. 1–9, 2016.
 - [103] L. Lacassagne, A. Manzanera, and A. Dupret, “Motion detection: Fast and robust algorithms for embedded systems,” *2009 16th IEEE Int. Conf. Image Process.*, pp. 3265–3268, 2009.
 - [104] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From

- error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.
- [105] C. Lallier, E. Reynaud, L. Robinault, and L. Tougne, “A testing framework for background subtraction algorithms comparison in intrusion detection context,” *2011 8th IEEE Int. Conf. Adv. Video Signal Based Surveillance, AVSS 2011*, pp. 314–319, 2011.
- [106] A. Vacavant, L. Tougne, and L. Robinault, “Evaluation of Background Models with Synthetic and Real Data,” in *Background Modeling and Foreground Detection for Video Surveillance*, 2015, pp. 1–14.
- [107] S. A. Macskassy and F. Provost, “A Brief Survey of Machine Learning Methods and their Sensor and IoT Applications,” in *International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2017, pp. 172–175.
- [108] S. Ben-David and S. Shalev-Shwartz, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
- [109] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [110] F. Chollet, *Deep Learning with Python*, vol. 80, no. 1. 2018.
- [111] A. Gulli and S. Pal, *Deep-Learning-with-Keras*. 2017.
- [112] R. Shanmugamani, *Deep Learning for Computer Vision - Expert techniques to train advanced neural networks using TensorFlow and Keras*. 2018.
- [113] G. Hinton, “RMSprop,” *Neural networks for machine learning. Coursera, video lectures*, 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [114] D. P. Kingma and J. L. Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,” *arXiv*, vol. 1412.6980v, no. preprint, pp. 1–15, 2015.
- [115] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*,” *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2012.
- [116] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv*, vol. 1212.5701v, no. Preprint, pp. 1–6, 2012.
- [117] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *25th International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

- [118] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” pp. 1–14, 2014.
- [119] C. Szegedy, “Going Deeper with Convolutions,” in *Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- [120] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image Recognition,” in *Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [121] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.
- [122] R. Girshick, “Fast R-CNN,” in *IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.
- [123] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 1137–1149, 2017.
- [124] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv1704.04861v1*, 2017.
- [125] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko, and M. Tuba, “Designing convolutional neural network architecture by the firefly algorithm,” *Proc. - 2019 Int. Young Eng. Forum, YEF-ECE 2019*, pp. 59–65, 2019.
- [126] D. Mishkin, N. Sergievskiy, and J. Matas, “Systematic evaluation of CNN advances on the ImageNet,” *arXiv Pre-print*, vol. 1606.02228, pp. 1–21, 2016.
- [127] H. H. Aghdam and E. J. Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. 2017.
- [128] Francois Chollet, *Deep Learning with Python*, 1st ed. Greenwich: Manning, 2017.
- [129] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *ECCV*, 2014, pp. 818–833.
- [130] R. E. Neapolitan and R. E. Neapolitan, *Neural Networks and Deep Learning*. 2018.
- [131] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*. 1996.
- [132] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1237, no. 2, 2019.
- [133] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A Comprehensive Survey of Loss Functions in

- Machine Learning,” *Ann. Data Sci.*, no. 0123456789, 2020.
- [134] S. Jadon, “A survey of loss functions for semantic segmentation,” 2020.
- [135] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfittin,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
- [136] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, 2019.
- [137] A. Lydia and F. S. Francis, “A Survey of Optimization Techniques for Deep Learning Networks,” *Int. J. Res. Eng. Appl. Manag.*, vol. 05, no. 02, pp. 601–604, 2019.
- [138] G. Hinton, “RMSprop,” *Lecture 6e*, coursera, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012. .
- [139] H. Saito, M. Yukie, K. Tanaka, K. Hikosaka, Y. Fukada, and E. Iwai, “Integration of direction signals of image motion in the superior temporal sulcus of the macaque monkey,” *J. Neurosci.*, vol. 6, no. 1, pp. 145–157, 1986.
- [140] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, vol. 160, no. 1, pp. 106–154.2, 1962.
- [141] H. B. Barlow and W. R. Levick, “The mechanisms of directionally selective units in the rabbit retina,” *J. Physiol. - London*, vol. 178, pp. 477–504, 1965.
- [142] F. Rind, “A Directionally Selective Motion-Detecting Neurone in the Brain of the Locust: Physiological and Morphological Characterization,” *J Exp Biol*, vol. 19, pp. 1–19, 1990.
- [143] F. C. Rind, “Identification of directionally selective motion-detecting neurones in the locust lobula and their synaptic connections with an identified descending neurone,” *J. Exp. Biol.*, vol. 149, pp. 21–43, 1990.
- [144] A. Borst and J. Haag, “Neural networks in the cockpit of the fly,” *J. Comp. Physiol. A Neuroethol. Sensory, Neural, Behav. Physiol.*, vol. 188, no. 6, pp. 419–437, 2002.
- [145] S. I. Fried, T. A. Münch, and F. S. Werblin, “Mechanisms and circuitry underlying directional selectivity in the retina,” *Nature*, vol. 420, no. 6914, pp. 411–414, 2002.
- [146] N. J. Priebe and D. Ferster, “Direction selectivity of excitation and inhibition in simple cells of the cat primary visual cortex,” *Neuron*, vol. 45, no. 1, pp. 133–145, 2005.
- [147] J. P. Gabriel, C. A. Trivedi, C. M. Maurer, S. Ryu, and J. H. Bollmann, “Layer-Specific Targeting of Direction-Selective Neurons in the Zebrafish Optic Tectum,” *Neuron*, vol. 76,

- p. 6, pp. 1147–1160, 2012.
- [148] J. H. Martin, *Neuroanatomy Text and ATLAS*. 2012.
 - [149] C. Enroth-cugell and J. G. Robson, “The contrast sensitivity of retinal ganglion cells of the cat,” *J. Physiol.*, vol. 187, pp. 517–552, 1966.
 - [150] E. Kaplan, “M, P, and K Pathways in the Primate Visual System,” in *The Visual Neurosciences*, no. January, 2003, pp. 887–887.
 - [151] S. H. C. Hendry and R. Clay Reid, “The koniocellular pathway in primate vision,” *Annu. Rev. Neurosci.*, vol. 23, no. May, pp. 127–153, 2000.
 - [152] J. Martinovic, “Magno-, Parvo-, Koniocellular Pathways,” *Encyclopedia of Color Science and Technology*, no. L. pp. 1–5, 2020.
 - [153] L. M. Martinez and J. M. Alonso, “Complex receptive fields in primary visual cortex,” *Neuroscientist*, vol. 9, no. 5, pp. 317–331, 2003.
 - [154] J. Ng, A. A. Bharath, and L. Zhaoping, “A survey of architecture and function of the primary visual cortex (V1),” *EURASIP J. Adv. Signal Process.*, vol. 2007, no. June 2014, pp. 1–17, 2007.
 - [155] R. A. Andersen, “Neural mechanisms of visual motion perception in primates,” *Neuron*, vol. 18, no. 6, pp. 865–872, 1997.
 - [156] D. Gabor, “Theory of communication. Part 1: The analysis of information,” *J. Inst. Electr. Eng. - Part III Radio Commun. Eng.*, vol. 93, no. 26, pp. 429–441, 1946.
 - [157] W. Chao, “Gabor wavelet transform and its application,” *Mech. Syst. Signal Process.*, vol. 24, no. 3, pp. 823–834, 2010.
 - [158] J. G. Daugman, “Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters,” *J. Opt. Soc. Am.*, vol. 2, no. 7, pp. 1160–1169, 1985.
 - [159] J. P. Jones and L. A. Palmer, “An Evaluation of the Two-Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex,” *J. Neurophysiol.*, vol. 58, no. 6, pp. 1187–1211, 1987.
 - [160] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb, “CloudID: Trustworthy cloud-based and cross-enterprise biometric identification,” *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7905–7916, 2015.
 - [161] E. H. Adelson and J. R. Bergen, “Spatiotemporal energy models for the perception of

- motion,” *J. Opt. Soc. Am. A*, vol. 2, no. 2, p. 284, 1985.
- [162] N. Petkov and E. Subramanian, “Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal Gabor filters with surround inhibition,” *Biol. Cybern.*, vol. 97, no. 5–6, pp. 423–439, 2007.
- [163] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.
- [164] K. M. Yi, K. Yun, S. W. Kim, H. J. Chang, H. Jeong, and J. Y. Choi, “Detection of moving objects with non-stationary cameras in 5.8ms: Bringing motion detection to your mobile device,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 27–34.
- [165] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2016.
- [166] J. J. Gibson, *The Perception of the Visual World*. 1950.
- [167] J. A. Marshall, “Self-organizing neural networks for perception of visual motion,” *Neural Networks*, vol. 3, no. 1, pp. 45–74, 1989.
- [168] T. Tversky and R. Miikkulainen, “Modeling Directional Selectivity Using Self-Organizing Delay-Adaptation Maps,” *Computational Neurosci. Trends Res.*, vol. 1, pp. 1–6, 2002.
- [169] B. Hu, S. Yue, and Z. Zhang, “A Rotational Motion Perception Neural Network Based on Asymmetric Spatiotemporal Visual Information Processing,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 1, pp. 1–19, 2016.
- [170] S. Yue and F. C. Rind, “Collision detection in complex dynamic scenes using an LGMD-based visual neural network with feature enhancement,” *IEEE Trans. Neural Networks*, vol. 17, no. 3, pp. 705–716, 2006.
- [171] Z. Zhang, S. Yue, and G. Zhang, “Fly visual system inspired artificial neural network for collision detection,” *Neurocomputing*, vol. 153, pp. 221–234, 2015.
- [172] H. Wang, J. Peng, and S. Yue, “A Directionally Selective Small Target Motion Detecting Visual Neural Network in Cluttered Backgrounds,” *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1541–1555, 2018.
- [173] J. Zhang *et al.*, “X-Net: A Binocular Summation Network for Foreground Segmentation,” *IEEE Access*, vol. 7, pp. 71412–71422, 2019.

- [174] L. A. Lim and H. Yalim Keles, “Foreground segmentation using convolutional neural networks for multiscale feature encoding,” *Pattern Recognit. Lett.*, vol. 112, pp. 256–262, 2018.
- [175] S. Wu, S. Zhong, and Y. Liu, “Deep residual learning for image steganalysis,” *arXiv:1512.03385*, vol. v1, pp. 1–12, 2017.
- [176] D. Eigen, C. Puhrsch, and R. Fergus, “Depth Map Prediction from a Single Image using a Multi-Scale Deep Network,” in *Neural Information Processing Systems Conference 2014*, 2014, pp. 1–9.
- [177] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2528–2535, 2010.
- [178] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *In ICCV*, 2011, pp. 2018–2025.
- [179] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *ECCV 2014*, 2014, pp. 818–833.
- [180] Z. Qin, F. Yu, C. Liu, and X. Chen, “How convolutional neural networks see the world --- A survey of convolutional neural network visualization methods,” *Math. Found. Comput.*, vol. 1, no. 2, pp. 149–180, 2018.
- [181] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding Neural Networks Through Deep Visualization,” *arXiv Pre-print*, vol. 1506.06579, pp. 1–12, 2015.
- [182] R. Fong and A. Vedaldi, “Net2Vec: Quantifying and Explaining How Concepts are Encoded by Filters in Deep Neural Networks,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8730–8738.
- [183] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, “Network dissection: Quantifying interpretability of deep visual representations,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3319–3327, 2017.
- [184] W. S. Ahmed and A. A. A. Karim, “The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN,” in *Proceedings of the 2020 International Conference on Computer Science and Software Engineering, CSASE 2020*, 2020, pp. 88–93.
- [185] S. Han *et al.*, “Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5070–5078.

- [186] L. Perez and J. Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning,” *arXiv:1712.04621*, vol. 1, pp. 1–8, 2017.
- [187] L. Taylor and G. Nitschke, “Improving Deep Learning with Generic Data Augmentation,” in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1542–1547.
- [188] J. González-Martín-Moro, J. L. Hernández-Verdejo, and A. Clement-Corral, “The visual system of diurnal raptors: updated review,” *Arch. Soc. Esp. Oftalmol.*, vol. 92, no. 5, pp. 225–232, 2017.
- [189] L. Reymond, “Spatial Visual Acuity of The Eagle Aquila Audax: A Behavioural, Optical and Anatomical Investigation,” *Vision Res.*, vol. 25, no. 10, pp. 1477–1491, 1985.
- [190] G. L. Walls, *The vertebrate eye and its adaptive radiation*. 1942.
- [191] M. P. Jones, K. E. Pierce, and D. Ward, “Avian Vision: A Review of Form and Function with Special Consideration to Birds of Prey,” *J. Exot. Pet Med.*, vol. 16, no. 2, pp. 69–87, 2007.
- [192] A. Bringmann, “Structure and function of the bird fovea,” *J. Vet. Med. Ser. C Anat. Histol. Embryol.*, vol. 48, no. 3, pp. 177–200, 2019.
- [193] M. F. Land and D.-E. Nilsson, *Animal eyes*, 2nd ed. New York: Oxford University Press Inc., 2012.
- [194] A. Glasser, M. T. Pardue, M. E. Andison, and J. G. Sivak, “A behavioral study of refraction, corneal curvature, and accommodation in raptor eyes,” *Can. J. Zool.*, vol. 75, no. 12, pp. 2010–2020, 2008.
- [195] M. Ruggeri, J. C. Major, C. McKeown, R. W. Knighton, C. A. Puliafito, and S. Jiao, “Retinal structure of birds of prey revealed by ultra-high resolution spectral-domain optical coherence tomography,” *Investig. Ophthalmol. Vis. Sci.*, vol. 51, no. 11, pp. 5789–5795, 2010.
- [196] M. Mitkus, S. Potier, G. R. Martin, O. Duriez, and A. Kelber, “Raptor Vision,” in *Oxford Research Encyclopedia of Neuroscience*, no. February, 2018, pp. 1–40.
- [197] A. W. Snyder and W. H. Miller, “Telephoto lens system of falconiform eyes,” *Nature*, vol. 275, no. 5676, pp. 127–129, 1978.
- [198] D. L. Williams and D. L. Williams, “The avian eye,” *Ophthalmol. Exot. Pets*, pp. 119–158, 2013.

- [199] N. Kruger *et al.*, “Deep Hierarchies in the Primate Visual Cortex: What Can We Learn for Computer Vision?,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1870, 2013.
- [200] L. A. Lim and H. Y. Keles, “Learning Multi-scale Features for Foreground Segmentation,” *Pattern Anal. Appl.*, vol. 23, pp. 1369–1380, 2020.
- [201] W. Zheng, K. Wang, and F. Y. Wang, “A novel background subtraction algorithm based on parallel vision and Bayesian GANs,” *Neurocomputing*, vol. 394, pp. 178–200, 2020.
- [202] W.-B. Zhen, K.-F. Wang, and F.-Y. Wang, “Background Subtraction Algorithm With Bayesian Generative Adversarial Networks,” *Acta Autom. Sin.*, vol. 44, no. 5, pp. 878–890, 2018.
- [203] M. Braham, S. Pierard, and M. Van Droogenbroeck, “Semantic background subtraction,” in *Proceedings - International Conference on Image Processing, ICIP*, 2017, pp. 4552–4556.
- [204] M. O. Tezcan, J. Konrad, and P. Ishwar, “A Fully-Convolutional Neural Network for Background Subtraction of Unseen Videos,” *arXiv Pre-print*, vol. 1907.11371, pp. 1–10, 2019.
- [205] D. Berjón, C. Cuevas, F. Morán, and N. García, “Real-time nonparametric background subtraction with tracking-based foreground update,” *Pattern Recognit.*, vol. 74, pp. 156–170, 2018.
- [206] L. Maddalena and A. Petrosino, “The SOBS algorithm: What are the limits?,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 21–26.
- [207] T. S. F. Haines and T. Xiang, “Background subtraction with dirichletprocess mixture models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, 2014.
- [208] L. Maddalena and A. Petrosino, “A Self-organizing approach to background subtraction for visual surveillance applications,” *IEEE Trans. IMAGE Process.*, vol. 17, no. 7, pp. 1168–1177, 2008.
- [209] C. Cuevas and N. García, “Improved background modeling for real-time spatio-temporal non-parametric moving object detection strategies,” *Image Vis. Comput.*, vol. 31, no. 9, pp. 616–630, 2013.
- [210] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “Pfinder: real-time tracking of the human body,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785,

1997.

- [211] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.