Tecnológico de Estudios Superiores de Tianguistenco

TÍTULO

Reconocimiento de emociones a través de las expresiones faciales de los alumnos del TEST, basado en la red neuronal convolucional YOLO V5x

Tesis

Para obtener el título de :

Ingeniero Mecatrónico

Presentada por:

Aldair Brayam Rivera Castro

Director:

Dr. Hugo Yáñez Badillo

Asesora:

Dra. Laura Cleofas Sánchez

Santiago Tianguistenco México, 2023

Dedicatoria

Dedico este trabajo de tesis principalmente a mis padres, Ofelia Castro Samano y Valentin Rivera Morales, quienes siempre han sido mi motor e impulso para seguir superándome y apoyarme en cada una de las decisiones que he tomado.

Agradecimientos

El presente trabajo lo dedico principalmente a Dios, por brindarme la oportunidad de vivir, permitirme disfrutar cada momento de mi vida y guiarme por el camino que ha trazado para mí.

A mis padres, Ofelia Castro Samano y Valentin Rivera Morales, por darme la vida, gracias por su amor, paciencia y esfuerzo, que me han permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo del esfuerzo y valentía, de no temer a las adversidades porque Dios esta conmigo siempre.

Este trabajo fue realizado bajo la supervisión de la Dra. Laura Cleofas Sánchez y del Dr. Hugo Yáñez Badillo, a quienes me gustaría expresar mi más profundo agradecimiento, por hacer posible la realización de este trabajo. Además, de agradecer su paciencia, tiempo y dedicación que tuvieron para que esto saliera de manera exitosa.

A mis amigos, por ser parte de mi vida, de mis momentos tristes y alegres, por apoyarme, por nunca dejarme caer, por estar siempre allí; María Teresa Alcantara Ortíz, Wendy Díaz Díaz, Laura Itzel López Hernández.

A mis compañeros de la división de Ingeniería Mecatrónica, quienes autorizaron el uso de sus fotografías para la generación de la base de datos que permitió comprobar el funcionamiento del modelo entrenado.

A mis maestros, que compartieron conmigo sus conocimientos para convertirme en un profesionista, por su tiempo, dedicación y por su pasión por la actividad docente.

Resumen

Las emociones ayudan a la mente a concentrarse y a establecer prioridades. Por ejemplo, nuestro lado lógico dice: "Fíjate una meta". Pero son las emociones que nos impulsan a actuar hacia este objetivo (Jensen, 2003). El estado de ánimo de los estudiantes en el momento de atender sus clases, es importante para tener un buen desarrollo académico. Actualmente en el Tecnológico de Estudios Superiores de Tianguistenco (*TEST*), se realizan evaluaciones diagnósticas que permiten canalizar a los estudiantes a las áreas correspondientes p. ej. psicología, médica, académica, entre otros, con la finalidad de mejorar los procesos de aprendizaje, identificando, y en dado caso, reportar situaciones de violencia intrafamiliar, enfermedades psicológicas, entre otros aspectos. Sin embargo, este proceso se realiza de manera tradicional, es decir, las tutorías hacia los estudiantes toman un tiempo prolongado debido a que se realizan tres evaluaciones por cada semestre, y a veces es difícil detectar al mismo tiempo expresiones faciales de un grupo de más de treinta estudiantes. Por esta razón, el presente trabajo describe el proceso del reconocimiento de expresiones faciales para la identificación de las emociones básicas propuestas por Ekman y Friesen (1971) (*Afraid*, *Angry*, *Disgust*, *Happy*, *Sad and Surprise*), que presentan los estudiantes al momento de realizar sus actividades académicas, esto se realiza de manera grupal e individual. Para ello se considera implementar el entrenamiento de la red neuronal convolucional *You Only Look Once v5x* (*YOLO*), y a través de una *webcam* se identificarán las expresiones faciales que representen el estado de ánimo en los estudiantes del *TEST*, estas emociones se verán reflejadas a través de un pictograma formado en una interfaz electrónica conectada a una Raspberry pi 4B de 4G de *RAM*.

Abstract

Emotions help the mind to focus and prioritize. For example, our logical side says: "Set a goal". But it is the emotions that drive us to act toward this goal (Jensen, 2003). The state of mind of the students at the time of attending their classes is important to have a good academic development. Currently at the Tecnológico de Estudios Superiores de Tianguistenco (*TEST*), diagnostic evaluations are carried out to channel students to the corresponding areas, e.g. psychology, medical, academic, among others, with the purpose of improving learning processes, identifying and, if necessary, reporting situations of domestic violence, psychological illnesses, among other aspects. However, this process is done in a traditional way, that is, tutoring students takes a long time due to the fact that there are three evaluations per semester, and sometimes it is difficult to detect facial expressions of a group of more than thirty students at the same time. For this reason, the present work describes the process of facial expression recognition for the identification of the basic emotions proposed by Ekman y Friesen, 1971 (*Afraid*, *Angry*, *Disgust*, *Happy*, *Sad* y *Surprise*), presented by the students when performing their academic activities. *and Surprise*), that students present when performing their academic activities, this is done in a group and individual way. For this purpose, it is considered to implement the training of the convolutional neural network *You Only Look Once v5x* (*YOLO*), and through a *webcam* the facial expressions that represent the mood of the students in the *TEST* will be identified, these emotions will be reflected through a pictogram formed in an electronic interface connected to a Raspberry pi 4B of 4G *RAM*.

Índice

Índice de figuras

Índice de tablas

Introducción

Las tecnologías de la información y la comunicación han ido evolucionando a lo largo de los años, dando lugar al desarrollo de la inteligencia artificial (*Artificial Intelligence*, *AI*). Definida por Coppin (2004) como la capacidad que tienen las máquinas para desarrollar desafíos que impliquen la reducción de tiempos, adaptarse a nuevas situaciones, resolver problemas, responder preguntas y realizar otras funciones que requieran un cierto nivel de inteligencia autónoma.

Los diferentes cambios emocionales, las intenciones o las comunicaciones sociales de una persona son reflejados a través de las expresiones faciales. Un estudio realizado por científicos del comportamiento, (Jones & Jonsson, 2005), ha demostrado que la comunicación verbal (lenguaje hablado) de un mensaje solamente contribuye en un 7 % al efecto del mensaje, la parte vocal (la entonación) aporta un 38 %, y las expresiones faciales un 55 % al efecto del mensaje hablado. Demostrando la importancia del análisis de la expresión facial en el campo de la psicología y las ciencias del comportamiento.

El presente trabajo se enfoca en la elaboración de un prototipo para identificar el estado de ánimo de los alumnos del *TEST*, a través de sus expresiones faciales, de manera individual y grupal en el momento de atender sus actividades de aprendizaje. Basado en el entrenamiento de la red neuronal convolucional *YOLO v5x*. Utilizando una *Raspberry Pi 4B* de 4G (RPi-4G) y una *webcam* para realizar la inferencia con los alumnos del *TEST*. Los resultados son visualizados a través de una interfaz de diodos emisores de luz (*Light Emitting Diode*, *LED*), para representar la emoción detectada a través de un pictograma [1]. Con la finalidad de complementar las evaluaciones diagnósticas realizadas actualmente en el *TEST*. "Tanto los estudiantes como los maestros pueden estar aburridos, felices, enojados, tristes y estas emociones influyen en sus percepciones, comportamientos, en el proceso de enseñanza y aprendizaje" (Schutz et al., 2007).

La organización de este trabajo es de la siguiente manera: primero se presenta el marco histórico, donde se abordan investigaciones previas relacionadas con este trabajo, posteriormente se muestra la metodología, en la cual se describe la manera en como se obtuvo y preparo la base de datos para entrenar la red neuronal convolucional (*Convolutional Neural Network, CNN*) para la detección de emociones, posteriormente se realiza el entrenamiento a través de los pesos obtenidos del repositorio https://github.com/ultralytics/yolov5, seguido del proceso de validación considerando mil ocho imágenes divididas de la siguiente manera: quinientas cuatro imágenes descargadas de internet y quinientas cuatro fotografías pertenecientes a expresiones faciales de alumnos del *TEST*, pertenecientes a las divisiones de Ingeniería Mecatrónica (*IMT*) y Sistemas Computacionales (*ISC*), posteriormente se muestran los resultados. Por último, se presentan las principales conclusiones del trabajo.

[1] Signo visual que representa figurativamente un objeto real, y a través de este, un significado.

Justificación

De acuerdo con Fernández et al. (2010) se ha evidenciado que el aprendizaje se consolida de mejor manera en nuestro cerebro cuando se involucran las emociones. Es un proceso psicológico que nos prepara para adaptarnos y responder al entorno de aprendizaje. Generalmente, los estudiantes con problemas emocionales presentan un déficit en habilidades como; socialización, autoaprendizaje, concentración, entre otras, que afectan su rendimiento académico.

Es por ello, que en el presente proyecto se pretende desarrollar un prototipo de reconocimiento de expresiones faciales automático, mediante el entrenamiento de la red neuronal *YOLO v5x*, identificando el estado de ánimo a través de sus expresiones faciales de los alumnos del *TEST* de manera grupal e individual, debido a que el aprendizaje es un proceso complejo y va más allá de almacenar información e involucra tener un buen estado emocional (Smith, 2019). Algunas emociones específicas como, por ejemplo, la felicidad, pueden mejorar el desempeño de aprendizaje, mientras que otras, como la tristeza, pueden afectar el rendimiento en la escuela haciendo que el índice de deserción aumente.

La importancia de implementar la inteligencia artificial en el proceso de enseñanza-aprendizaje, radica en que se puede mejorar el diagnóstico cuando se presenten resultados de emociones con impacto negativo, y con ello, incrementar las alternativas de solución, por ejemplo, canalizar a los estudiantes a las áreas de apoyo correspondientes o cambiar la metodología de aprendizaje con la finalidad de mantener la atención, disposición y motivación.

Planteamiento del problema

¿De qué manera se puede complementar el proceso de evaluaciones diagnósticas a los alumnos del *TEST* dependiendo de su estado emocional?

Actualmente en el *TEST* se realizan tres evaluaciones diagnósticas en un periodo de seis meses. Estas evaluaciones permiten llevar un control estadístico y de canalización de los alumnos a las principales áreas que brindan apoyo: académico, psicológico, económico y médico. Sin embargo, este proceso hacia los estudiantes es complejo, debido a factores como: tiempo prolongado en realizar estas evaluaciones y a los grupos de más de treinta alumnos. Teniendo en cuenta que el estado emocional influye en el rendimiento académico de los alumnos se considera fundamental identificar estos problemas y canalizarlos a las áreas correspondientes para su pronta atención y así tener un mejor desempeño en sus actividades académicas.

Hipótesis

Implementando un sistema para identificar el estado de ánimo de los alumnos a través de sus expresiones en el momento que se encuentren realizando sus actividades académicas, para visualizar estos resultados a través de una interfaz de *LED's* rojo, verde y azul (*Red Green Blue*, *RGB*). Canalizando a los estudiantes al momento de la detección de alguna emoción desfavorable que provoque un déficit en su desarrollo académico, complementando las tutorías de evaluación, debido a que el estudiante no tendrá que esperar a los tiempos de evaluación diagnostica estipulados por el *TEST* para poder solicitar la atención correspondiente.

Objetivos

General

- Desarrollar un prototipo de reconocimiento de expresiones faciales automático, mediante el entrenamiento de la red neuronal *YOLO v5x*, para identificar y visualizar en tiempo real el estado de ánimo de los estudiantes del Tecnológico de Estudios Superiores de Tianguistenco.

Específicos

- Crear una base de datos considerando 3,180 imágenes para el aprendizaje de la red neuronal, dividida en un 80 % para entrenamiento y 20 % para validación.
- Clasificar la información de la base de datos de acuerdo a las expresiones faciales asociadas con la alegría, tristeza, miedo, enojo, sorpresa y desagrado mediante la herramienta *LabelImg* para el etiquetado de las imágenes.
- Generar diez bases de datos diferentes para seleccionar la de mayor desempeño en el entrenamiento.
- Emplear el repositorio: https://github.com/ultralytics/yolov5 para configurar el entrenamiento de la red *YOLO v5x*.
- Probar el modelo entrenado considerando 500 imágenes descargadas de internet y 500 fotografías de los alumnos del *TEST*, por parte de la división de mecatrónica y sistemas computacionales, con el propósito de evaluar la eficiencia en la detección de emociones.
- Construir un prototipo mediante una *Raspberry Pi 4B* y una *webcam*, para la visualización de los resultados de predicción con un pictograma en una interfaz electrónica de *LED's RGB*.
- Desarrollar un código de programación a través del lenguaje *Python*, para identificar la emoción que predomina de manera individual o grupal de los alumnos de *TEST*.

Capítulo 1

Generalidades

1.1. Marco histórico

Estado del arte

A continuación se presentan diferentes investigaciones relacionadas con el presente proyecto. En la investigación presentada por Kirkvik (2022), se desarrolla un modelo capaz de distinguir diversas emociones humanas usando el aprendizaje automático (*Machine Learning*, *ML*), el programa es capaz de distinguir entre las emociones humanas y clasificar cada sentimiento en una de siete categorías, así como también se optimiza el modelo encontrando y combinando algoritmos que dan un porcentaje de predicción satisfactorio. A través de esta investigación se identifica que las precisiones de entrenamiento mejorarán si se ejecutan más *epochs* [1] en un modelo pre-entrenado. Con cincuenta *epochs* se obtuvo un *accuracy* del 89.7 %, mientras que en un modelo pre-entrenado se obtuvo un *accuracy* del 86.5 % en tan solo 20 *epochs*. Una de las recomendaciones sugeridas son; investigar si otros modelos pre-entrenados identificarán características en un grado aún mayor, y con un menor uso de recursos computacionales.

De acuerdo a la investigación realizada por Minaee et al. (2021), en donde se propone una estructura basada en el aprendizaje tipo *End to End* para el reconocimiento de expresiones faciales, que utiliza un mecanismo de atención para centrarse en áreas ricas en características del rostro, aplicado en tres bases de datos diferentes: imágenes en escala de grises,imágenes con filtro azul e imágenes de animación en donde se obtuvieron los siguientes resultados de *accuracy* 70.02 %, 92.8 % y 98 % respectivamente. Teniendo como principal conclusión que al utilizar técnicas de visualización para resaltar las regiones más destacadas de una imagen facial se pueden obtener mejores resultados de *precision* debido a que el modelo solo se concentra en la extracción de características principales de acuerdo a la emoción correspondiente.

En el trabajo de Nguyen (2021), se proponen como objetivos; estudiar, evaluar, experimentar y mejorar los modelos de *DL* en los métodos modernos. Teniendo como metodología implementar y volver a entrenar algunos modelos modernos de redes en *DL* para comparar los resultados con algunos modelos propuestos. Se obtuvieron los siguientes resultados durante la implementación de este trabajo; se investigó, probando y mejorando una red residual de *DL*, obteniendo una exactitud del 66.8 %. Se descubrió que muchas redes dan resultados mucho mejores que la red neuronal residual *v4* (*Residual Network*, *ResNet*) implementada en este trabajo, pero son de una arquitectura compleja, por tal motivo requieren una gran cantidad de parámetros para entrenar. Algunas limitaciones que se tuvieron para el desarrollo de la investigación fue la limitación de *hardware* y tiempo, el proceso experimental también tiene deficiencias, ya que no puede volver a entrenar otras grandes redes modernas para una mejor comparación objetiva. Dado que es una red de clasificación, *Resnet v4* probablemente funcionará bien en algunos otros problemas de clasificación que no forman parte de la clasificación emocional.

El desarrollo de un semáforo inteligente basado en redes neuronales para la optimización de la afluencia vehicular desarrollado por Gutiérrez y Jiménez (2021), el cual tuvo como objetivo general, desarrollar y entrenar un sistema inteligente mediante un detector de objetos y

[1] Se refiere al número de iteraciones que se realizan en el entrenamiento.

CNN para el control de encendido de las luces de un semáforo para el aforo vehicular de dos avenidas. La propuesta se basa en la utilización de los detectores de objetos *CNN YOLO v3* y *Faster RCNN* los cuales fueron entrenados con una base de datos que contenía imágenes de automóviles particulares, con la finalidad de realizar el entrenamiento de los modelos y estos fueran capaces de identificar y cuantificar automáticamente el número de vehículos que hay en cada vialidad de tránsito, enviando decisiones del estado de encendido de luces a un prototipo de semáforo inteligente, de acuerdo a la afluencia vehicular de cada vialidad. Como resultados de la investigación se recomienda tener una buena calidad de imagen, y un buen enfoque de cámara para una buena detección. También se realizó la comparación de los detectores de objetos *CNN Faster RCNN 101 COCO* y el modelo *YOLO v3* debido a la gran cantidad de datos computacionales requeridos para llevar a cabo la detección de objetos, sé observo que la *YOLO v3* es más eficiente, rápida y con un grado mayor de optimización, debido a que se basa en figuras, formas, bordes y líneas para detectar y determinar a que clase pertenece el objeto detectado.

En el trabajo de Guevara et al. (2021) se proponen nueve funciones de activación basadas en combinaciones como *ReLU* y *sigmoide*. Además, se presenta un estudio sobre los efectos causados por las funciones de activación en el rendimiento de una *CNN*. Primeramente se comienza describiendo cada una de las funciones de activación, presentando sus gráficas, formas analíticas y derivadas. Se utilizó una *laptop HP*, con un procesador *Intel Core i5* y memoria *RAM* de 8 *GB*, para la realización de todos los experimentos. Los algoritmos de aprendizaje fueron desarrollados en el *software MATLAB*, y el *DL toolboxs*. Las funciones de activación fueron implementadas en los conjuntos de entrenamiento y validación de las tres bases de datos. Como resultado se demuestra el aumento de la precisión en un 1.18 %, iniciando con un 0.91 %, esto aplicando nuevas funciones de activación como lo son la activación lineal rectificada con fugas (*Leaky Rectified Linear Activation*, *LReLU*), unidad lineal rectificada paramétrica (*Parametric Rectified Linear Unit*, *PReLU*), *tanhx*, *sigmoid*. De acuerdo con los resultados del clasificador, se concluyó que es posible realizar combinaciones de funciones de activación comúnmente utilizadas con el fin de crear una función distinta y posteriormente emplearla como función de activación en una *CNN*, ya que se lograron obtener resultados semejantes que los generados por funciones conocidas, tales como la función *ReLU* (y sus variaciones) y la función *sigmoid*. Por lo anterior, se puede concluir que las funciones de activación propuestas tienen un efecto positivo cuando son añadidas al conjunto de funciones clásicas utilizadas en un problema de clasificación.

Teniendo en cuenta el trabajo de Badrulhisham y Mangshor (2021), el cual propone realizar el reconocimiento de emociones utilizando un dispositivo móvil, utilizando una red neuronal convolucional. La metodología que fue utilizada para el desarrollo de este estudio comienza con la introducción de las imágenes en tiempo real, seguida de la implementación de la *CNN* para reconocer la emoción. Posteriormente se muestra la emoción a través de un *display*. Como conclusión, muestra la implementación con éxito de una aplicación para el reconocimiento de emociones utilizando la *CNN*, la cual es capaz de reconocer cuatro tipos de emociones que son feliz, triste, sorprendido y desagrado. Utilizando el algoritmo *MobileNet* con un conjunto de datos personalizado y evaluado. La aplicación desarrollada alcanzó una precisión media del 92,50 %. En cuanto a la sensibilidad y la especificidad, fue capaz de alcanzar el 85,00 % y el 95,00 % respectivamente. Por lo tanto, la aplicación de la *CNN* en el

reconocimiento de la emoción logró resultados prometedores y podría ser capaz de contribuir al trabajo de sucesión en la *CNN*. En el futuro, se cree que la integración de una *CNN* con cualquier otro método de *IA* mejorará el rendimiento de la aplicación.

Por otra parte se encuentra la aportación de Zhihao (2020), donde se contribuyó a resumir los modelos clásicos de redes neuronales que han surgido. La red de clasificación, así como el reconocimiento de detección de objetos. Se estudia la red y el algoritmo de detección de rostros, la estructura, ventajas y desventajas de los modelos de clasificación tradicionales, como *Alex-net, Google-net, Res-net*. Este trabajo estudia y resume los sistemas existentes, incluidos *R-CNN, Fast-RCNN, Faster-RCNN, YOLO*, también el modelo de la red convolucional en cascada multitarea (*Multi Task Cascaded Convolutional Networks*, *MTCNN*) específicamente utilizado para detectar rostros. La segunda aportación es introducir los problemas que aparecen en el entrenamiento de modelos y el concepto de transferencia de aprendizaje. Propone un nuevo método de procesamiento de vídeo y utiliza este método para construir un nuevo conjunto de modelos de reconocimiento de expresiones faciales en vídeo. La conexión entre los cuadros de vídeo nunca ha sido efectivo en el *DL*. Propone que hay un patrón de movimientos faciales cuando las personas generan expresiones, y este patrón puede ser utilizado para la construcción de características de reconocimiento de expresión humana. Sin embargo, se cuenta con algunas limitaciones debido a que el algoritmo de reconocimiento de expresiones propuesto en este trabajo ha demostrado ser eficaz, la precisión aún debe mejorarse. El sistema de reconocimiento de expresiones todavía tiene algunas dificultades en la vida real. Además, todavía hay otros métodos de procesamiento para encontrar las características en el análisis de vídeos.

La implementación por parte de Chaganti et al. (2020), de una máquina de vectores de soporte (*Support Vector Machine*, *SVM*) en la clasificación de imágenes, es una técnica muy potente, por tal motivo se decidió realizar una comparación con una *CNN*, ya que los resultados se debían a la falta de un conjunto de datos lo suficientemente grande. Así que, se realizó un aumento de datos y, al volver a implementar la *SVM*, se consiguió una precisión del 82 %, una disminución significativa. Sin embargo, se buscaron resultados mucho más eficientes por lo cual se implementaron otras técnicas de *DL*. Esta búsqueda hizo que se implementaran redes como; las redes neuronales artificiales (*Artificial Neural Network*, *ANN*) y a la *CNN*. Al implementar con éxito la *CNN*, se logró una precisión de un 93,57 %, en el mismo conjunto de datos. Teniendo como conclusión que las *CNN* tienen un mayor potencial de las técnicas de aprendizaje tradicionales.

De acuerdo con el artículo de Moon et al. (2020), tiene como principal objetivo presentar un nuevo sistema de clasificación que utiliza el cerebro para la conectividad con una nueva *CNN*, esto con la finalidad de validar su efectividad a través de la clasificación de vídeos emocionales, además proponen dos métodos basados en datos para construir la matriz de conectividad y así maximizar el rendimiento de clasificación. Tras desarrollar este trabajo se pudo observar, la eficacia que se tiene al aplicar una matriz de conectividad y los métodos de ordenación basados en datos que validarán diferentes tareas, por lo cual se recomienda aplicar matrices de conectividad a métodos u otras modalidades de imágenes cerebrales como radiología.

De la investigación de Dao (2020), se estudió la aplicación del *DL* en la clasificación de

imágenes utilizando *CNN*. Para el desarrollo del trabajo se utilizó el lenguaje de programación *Python* implementado en el *framework TensorFlow* y el *software Google Colaboratory*. Como conclusión y recomendaciones para futuros trabajos se tiene que; la combinación de las funciones de activación en las diferentes capas ayuda a tener una normalización en el *overfitting* y mejora el rendimiento del modelo.

El desarrollo de la clasificación de imágenes utilizando el entrenamiento de un modelo de una *CNN* en donde se utilizó *Image-Net PASCAL VOC* 2007 para la clasificación de imágenes de un conjunto de datos. Como resultados se encuentran que el *DL*, es una solución simple y efectiva para una tarea de clasificación de imágenes multiclase. Este modelo de *CNN* tiene la capacidad de cambio flexible en su arquitectura de acuerdo a los requerimientos que se necesitan para tener un mejor resultado. Se recomienda para la elaboración de trabajos futuros tener una base de datos con una gran cantidad de imágenes para obtener mejores resultados de entrenamiento (Ezat et al., 2020).

Por su parte Sisquella (2019), presenta una comparación de diferentes tipos de modelos de *ML* y *DL* incluyendo; máquinas de soporte de vectores (*Support Vector Machine*, *SVM*), bosque aleatorio (*Random Forest*, *RF*), árbol de decisiones (*Decision Tree*, *DT*), perceptrón multicapa (*multilayer perceptron*, *MLP*) y *CNN*. La evaluación se centra en el reconocimiento de emociones a través de las expresiones faciales, utilizando archivos de vídeo y audio. Se utilizaron diferentes *dataset* para clasificación de las siguientes emociones: *afraid*, *angry*, *disgusted*, *happy*, *neutral*, *sad* y *suprised*. Este proyecto demuestra que con las dos técnicas (imágenes y audio) se ha podido predecir correctamente al menos el 80 % de las imágenes procesadas. Como trabajos futuros se recomienda el desarrollo de un modelo multimodal que combine las entradas de vídeo y audio, y por lo tanto mejorar significativamente los resultados de efectividad. Este modelo podría agregarse a un robot humanoide por ser capaz de detectar los sentimientos de las personas con quién interactúa y mejorar su rendimiento de interacción y tratar de hacerlos sentir mejor.

El trabajo de Wu y Chen (2019), muestra el desarrollo de un sistema capaz de capturar imágenes faciales del mundo real a través de una cámara frontal de un ordenador portátil. Este sistema es capaz de capturar y predecir los resultados en tiempo real. Se realizó a través de técnicas de *DL* para entrenar un modelo de reconocimiento de emociones faciales. Se observó que los modelos profundos como *DenseNeet o ResNet* pueden tener *overfitting* fácilmente. En el proyecto se utilizó una *CNN* con cuatro bloques, cada bloque esta formado por una normalización de lotes, una capa convolucional con (N×N) tamaño de *kernel* con una función de activación de unidad lineal rectificada (*Rectifier Linear Unit*, *ReLU*) y una capa *max pooling* con tamaño de (2×2).

De acuerdo a la investigación de Artola (2019), se estudiaron las *CNN* y su importancia que tienen en el análisis de imágenes para la detección de cáncer de mama. Esto a través de un entrenamiento que permita conseguir un elevado porcentaje de precisión, sensibilidad y especificidad, realizando una investigación de los principales lenguajes de programación utilizados en el campo de la *AI*. Se obtuvo una precisión del 92 %. El principal problema que se encontró para el desarrollo de este proyecto fue en el apartado de *hardware*, ya que se utilizó un equipo con un procesador *Core* i7 6500 y velocidad de 2.5 *GHz*. Debido a que no se contaba con una *GPU* se limitó a utilizar la gran mayoría de librerías realizadas por

Python. Otra de las dificultades fue que al utilizar una gran cantidad de imágenes para la base de datos para el entrenamiento de las redes, pues según los parámetros que se utilizaban crecían de manera exponencial y hacía que la memoria *RAM* detuviera el proceso por falta de espacio.

Teniendo en cuenta el trabajo de Farias (2019), el cual tiene como principal objetivo desarrollar un sistema que realice un conteo de los siguientes objetos; carros, motocicletas, y personas, con la finalidad de medir el flujo vehicular y peatonal de las calles de la ciudad de Machala a través de la *CNN YOLO v3*. Para la realización de este trabajo se recolectó una gran cantidad de imágenes de los objetos a identificar y posteriormente fueron etiquetadas de forma manual utilizando la herramienta *Labelimg* de acuerdo a las clases a identificar. Para la detección de las clases se utilizó el protocolo de transmisión en tiempo real (*Real Time Streaming Protocol*, *RTSP*). Finalmente se tienen las siguientes recomendaciones para el desarrollo de futuros trabajos; tener un equipo con gran capacidad computacional y así acelerar el entrenamiento de la red neuronal. Tener una gran cantidad de imágenes para el entrenamiento, ya que esto se verá reflejado en el resultado, así como también tener claro el número de imágenes que se destinaran para el entrenamiento.

En el artículo realizado por Gu et al. (2018), se detallan mejoras en las *CNN* en diferentes aspectos, incluyendo el diseño de las capas, la función de activación, la función de pérdida, la regularización, la optimización y el cálculo rápido. Además, también se proponen varias aplicaciones de las *CNN* en la visión por ordenador. Las diferentes áreas de aplicación de las *CNN* son mostradas como parte de las conclusiones, de las cuales se destacan; la clasificación de imágenes, la detección de objetos, el seguimiento de objetos, la estimación de la postura, la detección de texto, la detección de deficiencia visual, el reconocimiento de acciones, el etiquetado de escenas, el habla y el procesamiento del lenguaje natural. Sin embargo las *CNN*, requieren un conjunto de datos a gran escala y una enorme potencia de cálculo para el entrenamiento. La recopilación manual de conjuntos de datos etiquetados requiere una gran cantidad de esfuerzo humano, por lo que se desea explorar el aprendizaje no supervisado de las *CNN*, por lo cual se recomienda utilizar un buen *CPU* y *GPU*.

En el trabajo desarrollado por Sultana et al. (2018), se describe la arquitectura y los detalles de entrenamiento de diferentes modelos de *CNN* y finalmente se realiza una comparación entre varias *CNN*. Como conclusión a este trabajo se tiene que aunque la red *AlexNet* y la red de grupo de geometría visual (*Visual Geometry Group Networt, VGGN*), siguen la arquitectura de un modelo *CNN* convencional como *LeNet-5* sus redes son más grandes y profundas. Se experimentó al combinar el módulo de inicio y los bloques residuales con modelo *CNN* convencional, *LeNet-5*, *GoogLeNet* y *ResNet* obtuvieron mejor precisión que apilando los mismos bloques de construcción una y otra vez. El resultado de la *SENet* en el conjunto de datos *ImageNet* hace que se pueda implementar para otras tareas que requieran fuertes características de aprendizaje.

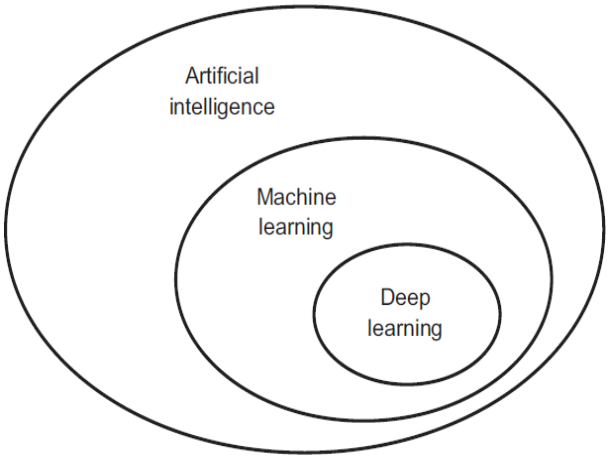
1.2. Marco Conceptual

A continuación se introducen los fundamentos teóricos para la comprensión de este trabajo.

1.2.1. Inteligencia artificial

Se considera a la *AI* como la habilidad de los elementos que posean un comportamiento inteligente automático (a través de algoritmos) para razonar como lo hacen los seres humanos, interactuando a través del lenguaje natural, aprendiendo de sus resultados y adaptándose a nuevos ambientes. El aprendizaje, la capacidad de adaptación a entornos cambiantes, la creatividad, entre otros, son facetas que usualmente se relacionan con el comportamiento inteligente. (Romero et al., 2007). "Es importante tener claramente de qué estamos hablando cuando mencionamos *AI*. ¿Qué es la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo (véase Figura 1.1)? ¿Cómo se relacionan entre sí?"(Chollet, 2021).

Figura 1.1:
Relación entre la AI, ML *y* DL (Chollet, 2021).



Aprendizaje automático (*Machine learnig*)

Este concepto surge de la incógnita de sí un ordenador podría ir más allá de lo que se le ha ordenado y aprender por sí mismo cómo realizar una tarea determinada, por tal motivo el principal objetivo del aprendizaje automático (*Machine Learnig*, *ML*) es estudiar el reconocimiento de patrones y el aprendizaje por parte de las computadoras, con la finalidad de que las máquinas puedan aprender sin ser explícitamente programadas. Esta pregunta hace referencia al estilo de programación convencional, en donde los humanos introducen instrucciones a través del código de programación y datos para procesar según esas instrucciones, y de ahí obtener un resultado (Saez De La Pascua, 2019).

Los filtros de *spam* de correo electrónico son un ejemplo del uso de algoritmos para aprender patrones, ya que utilizan este tipo de aprendizaje con la finalidad de detectar y clasificar el tipo de mensajes entrantes, haciendo la separación de correos basura y correos relevantes.

Aprendizaje supervisado

Este tipo de aprendizaje muestra la relación que existe entre los datos de entrada y de salida. Es aplicable cuando se requiere enseñar a los algoritmos cuál es el resultado que se desea obtener para un determinado valor de entrada, esto se consigue mostrándole varios ejemplos al algoritmo con la finalidad de aumentar el aprendizaje y garantizar que el resultado sea correcto incluso si le llegan valores que no se le han mostrado antes. Este tipo de algoritmos se van a apoyar de una base de datos los cuales previamente se deben etiquetar para que este algoritmo tenga la facilidad de como categorizar la nueva información recibida. Por tal motivo en este método es necesario la intervención humana, la cual es la encargada de proporcionar una etiqueta a los datos (Rouhiainen, 2018).

Un ejemplo de la aplicación del aprendizaje supervisado es al organizar 10000 fotografías en las cuales el algoritmo debe identificar las fotos en las que aparece un gato. En este tipo de aprendizaje se le muestran previamente al algoritmo fotos donde aparece un gato para que el algoritmo tenga la facilidad de como categorizar la nueva información recibida. Este tipo de aprendizaje muestra la relación que existe entre los datos de entrada y de salida. Es aplicable cuando se requiere enseñar a los algoritmos cuál es el resultado que se desea obtener para un determinado valor de entrada, esto se consigue mostrándole varios ejemplos al algoritmo con la finalidad de aumentar el aprendizaje y garantizar que el resultado sea correcto incluso si le llegan valores que no se le allá mostrado antes (Rouhiainen, 2018).

Aprendizaje no supervisado

Se define como el paradigma que consigue producir conocimientos únicamente de los datos que se proporcionan como entrada, sin necesidad en ningún momento explicarle al sistema que resultado se desea obtener. Los algoritmos no usan ningún dato etiquetado u organizado previamente para indicar cómo tendría que ser categorizada la nueva información, sino que tienen que encontrar la manera de clasificarlas ellos mismos. Por tanto, este método no requiere la intervención humana (Rouhiainen, 2018).

Aprendizaje de refuerzo

En esta modalidad de aprendizaje, los algoritmos van adquiriendo información de la experiencia de interacción con el entrono, por lo cual no le indicaremos que acciones tomar, sino que este debe tomar estas decisiones de manera secuencial, por si solo, para generar la máxima de recompensa. En otras palabras, tenemos que darles "un refuerzo positivo" cada vez que aciertan. La forma en que estos algoritmos aprenden se puede comparar con la de los perros cuando les damos "recompensas" al aprender a sentarse (Rouhiainen, 2018).

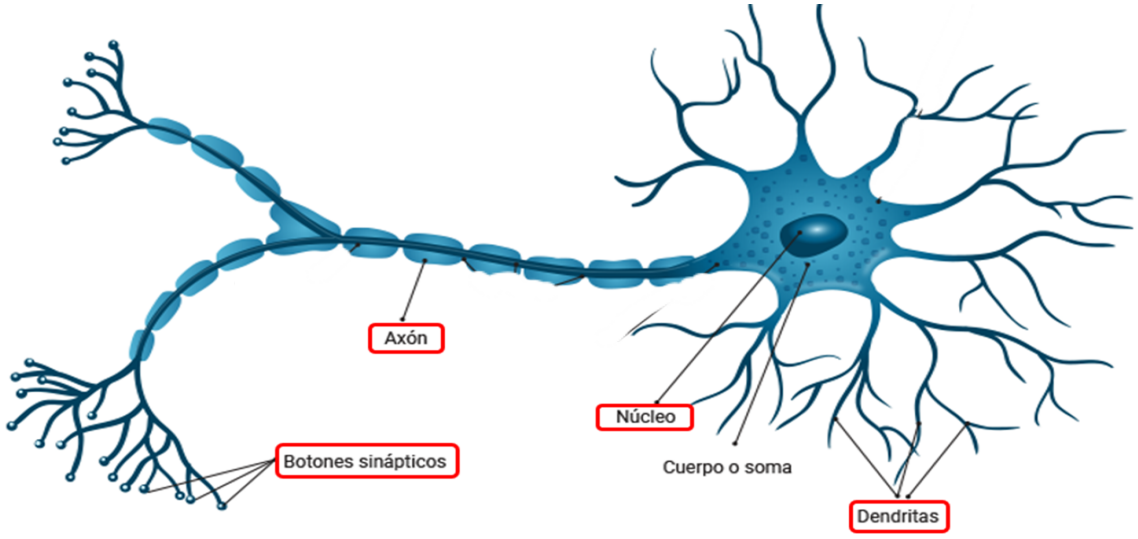
1.2.2. *Deep Learnig*

El *DL*, es la capacidad que puede tener una máquina en aprender sin necesidad de que haya un humano supervisando e interviniendo en ello. Esto se debe a que la máquina es capaz de aprender a partir de algoritmos y extraer conocimiento a partir de datos.

Redes neuronales artificiales

De acuerdo con (Basogain, 2008) las *ANN* están inspiradas en las redes neuronales biológicas del cerebro humano (véase Figura 1.2). Están constituidas por elementos que se comportan de forma similar a la neurona biológica en sus funciones más comunes. Estos elementos están organizados de una forma parecida a la que presenta el cerebro humano.

Figura 1.2:
Estructura general de una neurona biológica (Campos, 2018).



Una neurona es una célula del sistema nervioso (ver Figura 1.2), y sus principales componentes son:

- Núcleo: alberga y protege el material genético de la célula, que consiste en el *ADN* (ácido desoxirribonucleico), es la información genética de la célula, siendo esencial para la regulación y la síntesis de proteínas, que son fundamentales para el funcionamiento de la célula y la transmisión de señales en el sistema nervioso.
- Dendritas: membranas de una célula nerviosa (neurona), las cuales reciben información proveniente del Axón.
- Axón: prolongación de una neurona, por la que se transmite impulsos nerviosos hasta una o varias células musculares, glandulares, nerviosos, etc. Transmite información.
- Botones sinápticos: es la parte extrema del axón que se divide para producir una serie de terminales que forman sinapsis con otras neuronas o con células musculares o de glándulas.

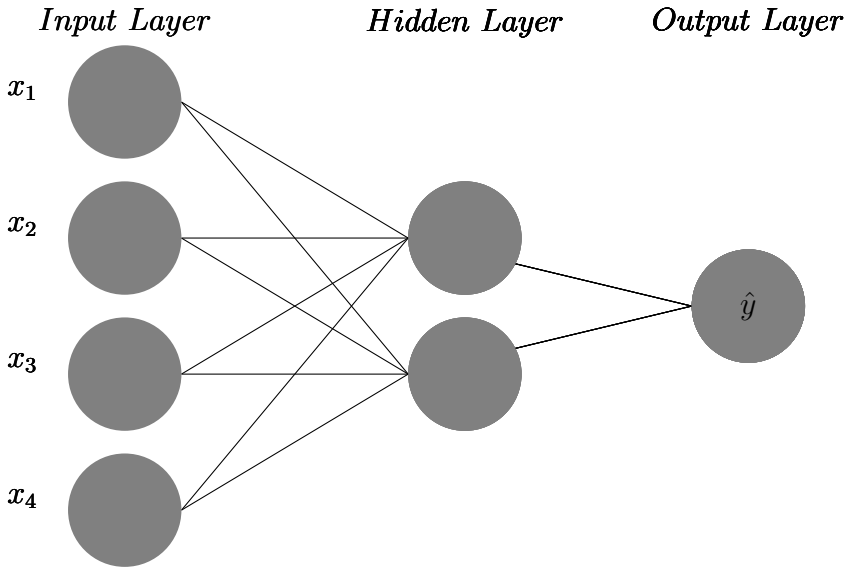
En la Figura 1.3, se muestra la estructura básica de una *ANN*: la entrada (*input layer*), normalmente en forma de vector multidimensional a la capa de entrada, que lo distribuirá a las capas ocultas (*hidden layer*). Las capas ocultas tomarán entonces las decisiones de la capa anterior y sopesarán cómo un cambio estocástico dentro de sí mismo que perjudica o mejora la salida final (*output layer*), y esto se conoce como el proceso de aprendizaje. Tener

múltiples capas ocultas apiladas unas sobre otras se denomina comúnmente aprendizaje profundo (O'Shea & Nash, 2015).

El funcionamiento de las *ANN* son similares al cerebro humano, por ejemplo, el aprendizaje ganado con la experiencia, la capacidad de interpretar nuevos datos y obtener características principales para entregar un resultado (Basogain, 2008).

- Aprender: las *ANN* pueden cambiar su comportamiento en función del entorno. Se les muestra un conjunto de entradas y ellas mismas se ajustan para producir unas salidas consistentes.
- Generalizar: extender o ampliar una cosa. La estructura que tienen las *ANN* permiten generalizar automáticamente, ofreciendo respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión.
- Abstraer: considerar por separado las cualidades de un objeto.

Figura 1.3:
Características principales de una ANN (S. Hernández, 2019).



La estructura de la Figura 1.3 es la base de una serie de arquitecturas de *ANN* como: las redes neuronales prealimentadas (*Feedforward Neural Network*, *FNN*), máquinas de boltzmann restringidas (Restricted Boltzmann Machine, RBMs) y las redes neuronales recurrentes (*Recurrent Neural Networks*, *RNN*).

Perceptrón

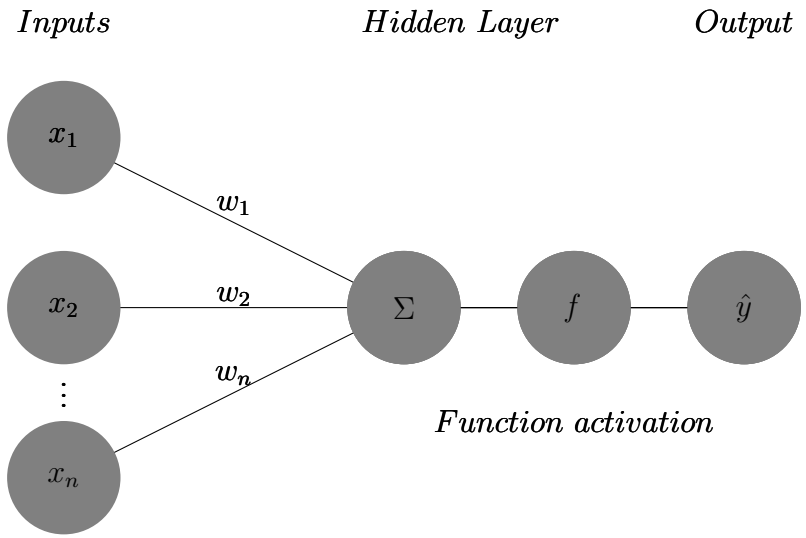
El procedimiento básico conocido dentro del aprendizaje de las neuronas artificiales es conocido como aprendizaje de perceptrón. El perceptrón tiene únicamente una neurona umbral binaria (también conocida como unidades de umbral binarias) y la regla de aprendizaje del perceptrón y, en conjunto, parece una regresión logística modificada (Skansi, 2018).

- Propuesto por Frank Rosenblatt en 1958.
- En 1969 fue refinado y analizado en detalle por Marvin MInisky y Seymour Papert (Marvin & Seymour, 1969).
- Mejora el planteamiento de la neurona de McCulloch y Pitts añadiendo el concepto de "peso" (parámetros de modelo) numérico a las entradas y planteando un mecanismo para ajustarlos (aprendizaje).
- No recibe únicamente valores de entrada binarios, permite valores de entrada reales.
- Se basa en un tipo de neurona artificial conocida como la unidad lógica de umbral (*Threhold Logic Unit*, *TLU*).

En la Figura 1.4 se observa los siguiente:

- Un conjunto de sinapsis o conectores, cada uno de los cuales se caracteriza por un peso o longitud propia. A diferencia de una sinapsis cerebral, el peso sináptico de una neurona artificial puede tomar un rango de valores que incluye tanto valores positivos como negativos.
- La función de la red de propagación es la encargada de realizar la suma de las señales de entrada, ponderado por las respectivas sinapsis de la neurona; las operaciones descritas constituyen un combinador linear.
- Una función de activación para limitar la amplitud de la salida de una neurona. También se denomina función de aplastamiento pues limita el rango de amplitud permitido en la salida a un valor finito.
- La salida se obtiene de la neurona en función de la activación de la misma. Normalmente únicamente se aplica la función identidad, y se toma como salida el valor de activación.

Figura 1.4:
Perceptrón (S. Hernández, 2019).



Es importante seleccionar la red neuronal de acuerdo a la aplicación que se requiera para obtener resultados óptimos, en la Tabla 1.1 se aprecian algunos tipos de redes neuronales y el desempeño que tienen en algunas áreas de aplicación (en donde el número de *checkmarks* indican la aplicabilidad de cada uno de los tipos de redes neuronales a ese problema en particular).

Tabla 1.1:
Tipos de redes neuronales y su dominio en distintas aplicaciones (Heaton, 2015)

.

	Clust	*Regis*	*Classif*	*Predict*	*Robot*	*Vision*	*Optim*
Self-organizing Map	✓✓✓				✓	✓	
Feedforward		✓✓✓	✓✓✓	✓✓	✓✓	✓✓	
Hopfield			✓			✓	✓
Boltzmann Machine			✓				✓✓
Deep Belief Network			✓✓✓		✓✓	✓✓	
Deep Feedforward		✓✓✓	✓✓✓	✓✓	✓✓✓	✓✓	
NEAT [2]		✓✓	✓✓		✓✓		
CPPN [3]					✓✓✓	✓✓	
HyperNEAT		✓✓	✓✓		✓✓✓	✓✓	
Convolutional Network		✓	✓✓✓		✓✓✓	✓✓✓	
Elman Network		✓✓	✓✓	✓✓✓			
Jordan Network		✓✓	✓✓	✓✓	✓✓✓		
Recurrent Network		✓✓	✓✓	✓✓✓	✓✓	✓	

Para este trabajo se utiliza una *CNN* debido a que se necesita un buen desempeño en la clasificación de seis clases que representan las emociones de *Angry*, *Afraid*, *Disgust*, *Happy*, *Sad* y *Surprice*. Así como también que tenga un buen desempeño en la visión artificial, ya que la inferencia se realizará a través de una *webcam*.

Funciones de activación

Una función de activación se utiliza para dar una no linealidad y esta se encuentra entre las capas de la red, determinando qué neurona debe activarse para que la información pase al la siguiente neurona (Kirkvik, 2022). La función de activación *f(x)* se elige de acuerdo a la tarea a realizar.

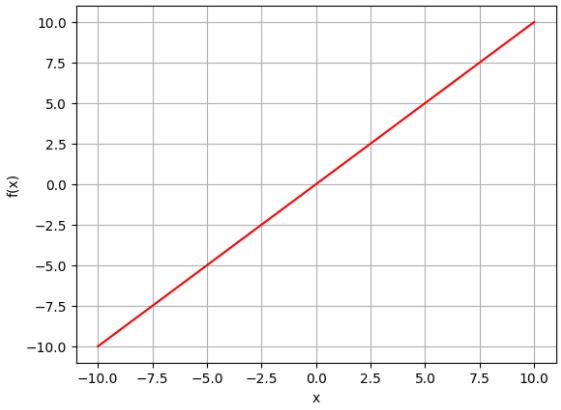
- ***Identy***

  También conocida como función lineal (ver Figura 1.5), permite que la entrada sea igual a la salida como se expresa en la función 1.1. Entonces, si la salida requiere una regresión lineal, se utiliza esta función, para generar valores únicos en la red neuronal que la usa (Ponce, 2010).

[2]Neuroevolución de topologías de aumento (*Neuro Evolution of Augmenting Topologies, NEAT*)
[3]Redes de producción de patrones composicionales (*Compositional Pattern-Producing Networks, CPPN*)

Figura 1.5:
Representación gráfica de la función Lineal.

$$f(x) = x \tag{1.1}$$

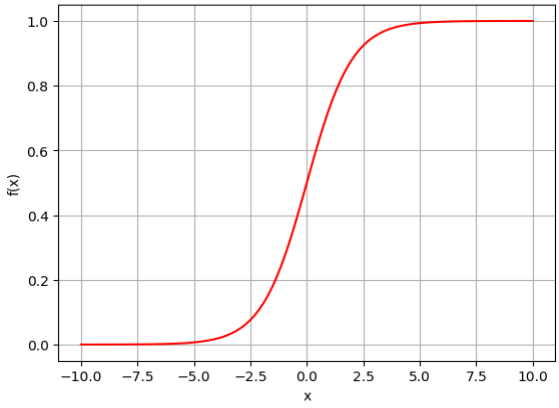


- ***Sigmoide***

  También es llamada función logística, toma cualquier rango de valores en la entrada y los mapea en un rango de 0 a 1 a la salida, como se muestra en la Figura 1.6 (Kim, 2017). Utilizada para problemas de clasificación binaria, la función que la representa es la (1.2):

Figura 1.6:
Representación gráfica de la función Sigmoide.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$



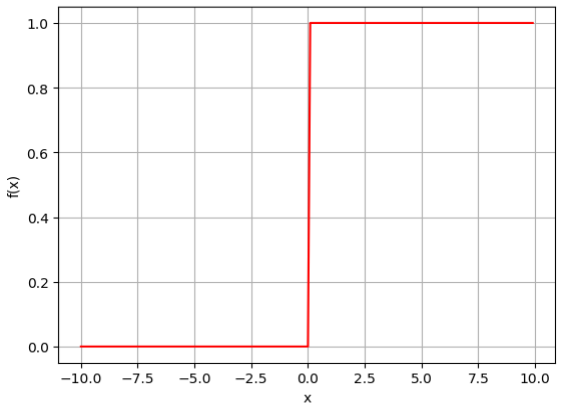
Propiedades matemáticas:

- Es continua.
- Es derivable.
- Va de 0 a 1.

- ***Heaviside step function*** Si la entrada es menor que cero la neurona va a ser cero, pero cuando es mayor o igual a cero dará como salida igual 1 (ver Figura 1.7). Esta función es aplicada cuando se quiere clasificar distintas clases (Ponce, 2010).

Figura 1.7:
Representación gráfica de la función Heaviside step function.

$$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases} \tag{1.3}$$

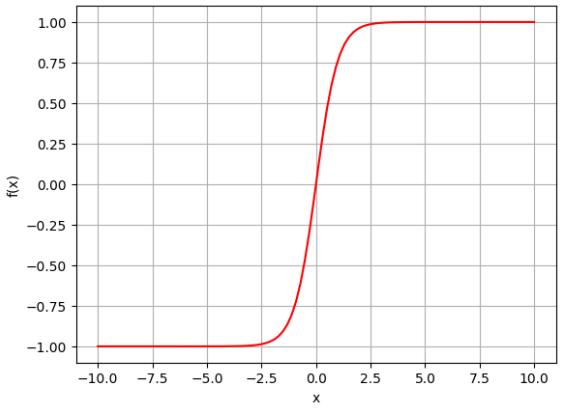


- ***Tanh function***

  Para un valor x, se transformará en un valor entre -1 y 1. A diferencia de la función *sigmoide*, esta función tiene media 0 y desviación típica uno, por lo que para conjuntos de datos en donde la media esté cercana cero suele funcionar muy bien, como se muestra la Figura 1.8 (Ponce, 2010). Representada por la función (1.4)

Figura 1.8:
Representación gráfica de la función tanh.

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.4}$$



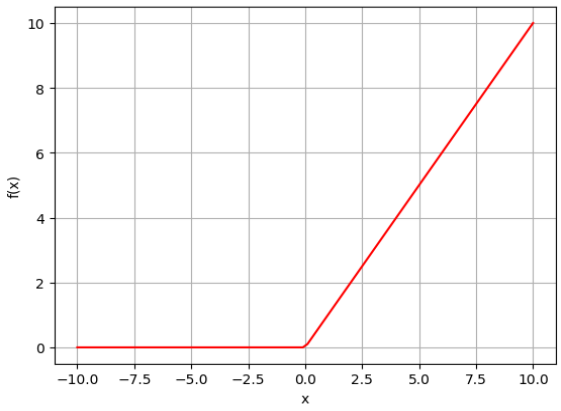
Propiedades matemáticas:

- Es continua.
- Es derivable.
- Va de -1 a 1.
- Media de 0 y desviación típica 1.

- **Unidad lineal rectificada (*Rectified Linear Unit*, *ReLU*)**

  Es comúnmente utilizada en *DL*, sobre todo en la clasificación de imágenes, dado un valor x, si dicho valor es menor o igual que 0, conviértelo a 0, y si es mayor, devuelve el mismo valor de entrada, como se muestra en la Figura 1.9 y en la función 1.5:

Figura 1.9:
Representación gráfica de la función ReLu.



$$f(x) = \begin{cases} x \text{ for } x \geq 0 \\ 0 \text{ for } x < 0 \end{cases} \tag{1.5}$$

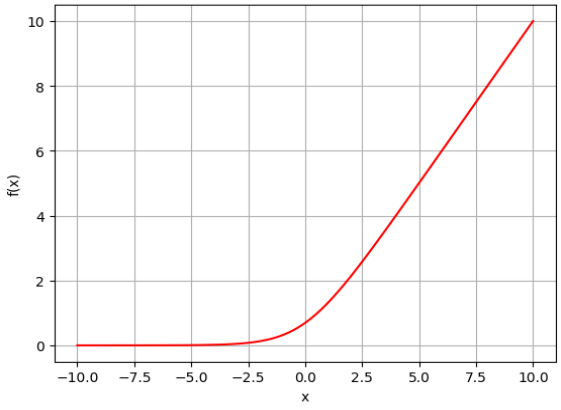
Propiedades matemáticas:

- Es continua.
- No lineal.
- No derivable.
- Va de 0 a ∞.

- ***Softmax***

  Es una variación de la función de activación *ReLU*. Su nombre proviene dado que la función es continua y derivable. Suaviza el codo asociado con la función *ReLU* (véase la Figura 1.10). Utilizada para la clasificación de clases debido a que transforma los datos de entrada en un porcentaje de probabilidad en la clasificación de cada clase.

Figura 1.10:
Representación gráfica de la función Softmax.



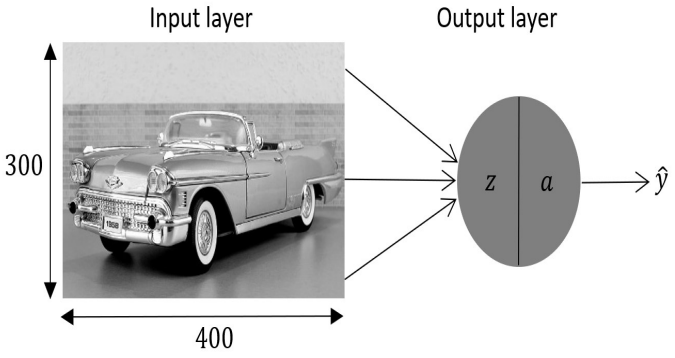
$$f(x) = log(1 + e^x) \tag{1.6}$$

Propagación hacia adelante (*Forward propagation*)

Es el proceso de la información que fluye desde las *inputs* hasta las *outputs* sin que haya ningún tipo de conexión de *feedback*. Se utiliza en el proceso de aprendizaje y se dejará de utilizar una vez que se tengan los valores óptimos de los pesos[4] (*weight*) (S. Hernández, 2019).

Ejemplo: en la Figura 1.11 se observa una imagen a la entrada de una neurona, la cual tiene una dimensión de 300×400 pixeles y corresponden al número de características (*features*).

Figura 1.11:
Forward propagation (S. Hernández, 2019).



El total de *features* es representado por n, como se muestra en la expresión 1.2.2

$$n = \#input\ features$$

Posteriormente los datos de la Figura 1.11, correspondientes al ancho y alto, son multiplicados para obtener el número total de *features*.

$$n = 300 \times 400 = 120000\ input\ features$$

El total de *features* son propagados hacia adelante, haciéndolos pasar por cada neurona, que están compuestas por 2 funciones: $f(z)$ (agregación) y $f(a)$ (activación).

Calculando la $f(z)$: es la suma de la neurona de sesgo (*bias*, b [5]) más el producto entre cada *feature* (x_n) y su respectivo *weight* (w_n) correspondiente, como se muestra en la ecuación 1.7.

$$z = x_1 w_1 + x_2 w_2 + ... + x_n w_n + b \tag{1.7}$$

Existen diferentes tipos de funciones de activación que se pueden utilizar de acuerdo a la aplicación que se requiera, para este ejemplo se aplica una función *sigmoidal* (ver 1.8) y el resultado de esta función es la salida de la neurona.

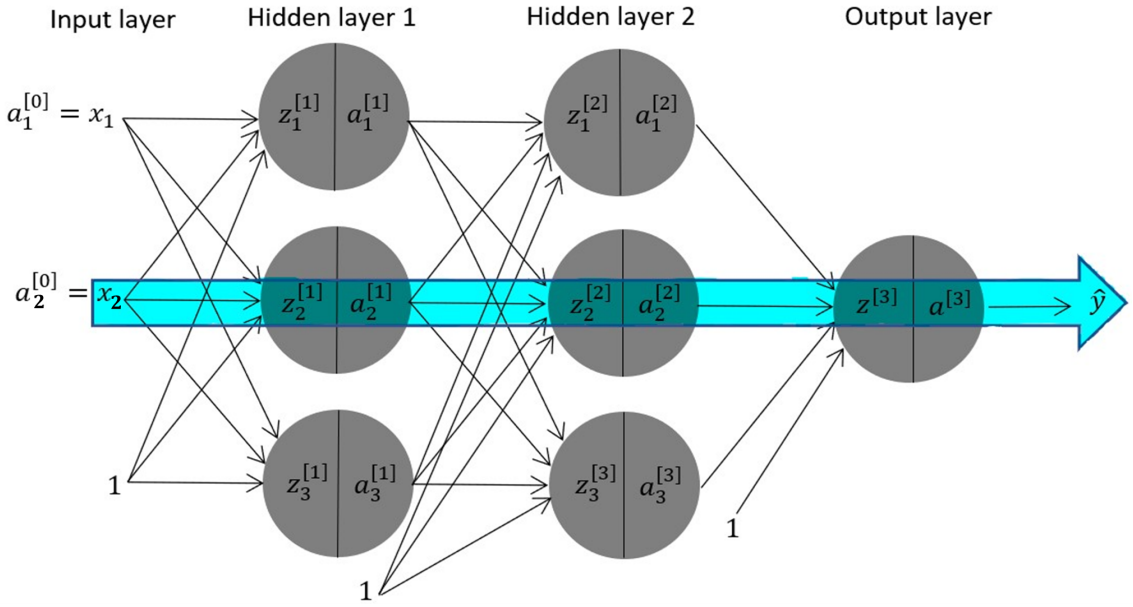
$$\hat{y} = a = sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{1.8}$$

[4]Representan la intensidad de interacción entre cada neurona, los pesos más altos tienen mayor influencia en las demás neuronas.

[5]Constante ajustable que permite que la neurona aprenda y se adapte mejor a los patrones de datos subyacentes durante el proceso de entrenamiento.

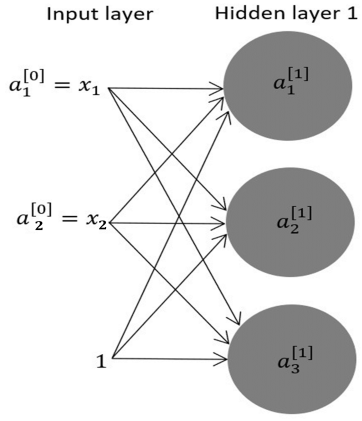
Ejemplo de *forward propagation* con el perceptrón multicapa (ver Figura 1.12): tiene dos *inputs* (x_1 y x_2), dos *hidden layers* con 3 neuronas cada capa y finalmente se tiene la salida $\hat{y}$. Al igual que en ejemplo anterior, los datos de entrada deben ser propagados desde de x_1, x_2 hacia la salida $\hat{y}$ y la neurona *bias*) a través de las capas ocultas (*hidden layers 1* y *hidden layers 2*), calculando las funciones correspondientes de agregación $f(z)$ y activación $f(a)$ por cada neurona que permitan la obtención de una salida $\hat{y}$ (S. Hernández, 2019).

Figura 1.12:
Forward propagation con el perceptrón multicapa (S. Hernández, 2019).



En la Figura 1.13 se muestra el procesamiento de los datos de entrada correspondientes a x_1, x_2 y a la neurona *bias*. Cada una de las entradas es conectada a las neuronas que componen a la primer *hidden layer*, en este caso se tienen tres neuronas con sus respectivas funciones de agregación y de activación.

Figura 1.13:
Forward propagation entre Input layer y hidden layer1 (S. Hernández, 2019).

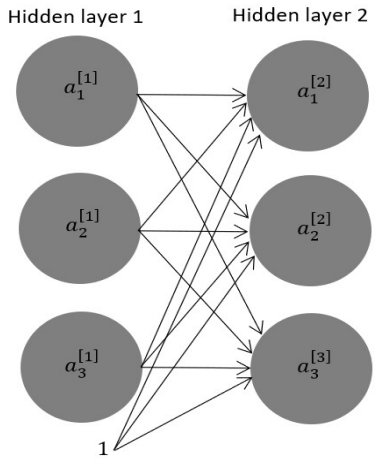


Ecuaciones para la primer *hidden layer*: de lado izquierdo se muestran las funciones de agregación $f(z)$ y del lado derecho se encuentran las funciones de activación $f(z)$ correspondientes a las tres neuronas que conforman la *hidden layer 1*:

$$z_1^{[1]} = a_1^{[0]} w_{11}^{[1]} + a_2^{[0]} w_{21}^{[1]} + b_1^{[1]} \qquad a_1^{[1]} = sigmoid(z_1^{[1]})$$
$$z_2^{[1]} = a_1^{[0]} w_{12}^{[1]} + a_2^{[0]} w_{22}^{[1]} + b_2^{[1]} \qquad a_2^{[1]} = sigmoid(z_2^{[1]})$$
$$z_3^{[1]} = a_1^{[0]} w_{13}^{[1]} + a_2^{[0]} w_{23}^{[1]} + b_3^{[1]} \qquad a_3^{[1]} = sigmoid(z_3^{[1]})$$

Las *outputs* de las neuronas anteriores pasan a ser los *inputs* de la *hidden layer 2*, como se muestra en la Figura 1.14.

Figura 1.14:
Forward propagation entre hidden layer 1 y hidden layer 2 (S. Hernández, 2019).

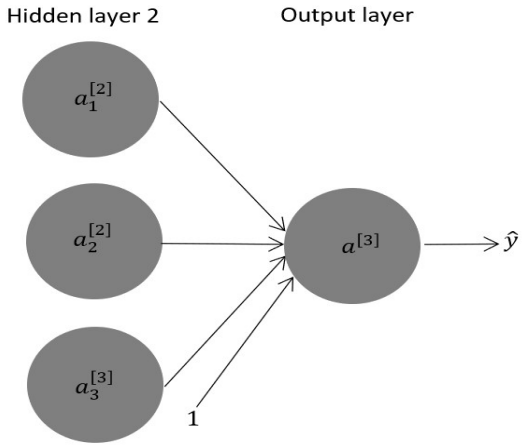


Ecuaciones para la *hidden layer 2*: de lado izquierdo se muestran las funciones de agregación $f(z)$ y del lado derecho se encuentran las funciones activación $f(z)$:

$$z_1^{[2]} = a_1^{[1]} w_{11}^{[2]} + a_2^{[1]} w_{21}^{[2]} + a_3^{[1]} w_{31}^{[2]} + b_1^{[2]} \qquad a_1^{[2]} = sigmoid(z_1^{[2]})$$
$$z_2^{[2]} = a_1^{[1]} w_{12}^{[2]} + a_2^{[1]} w_{22}^{[2]} + a_3^{[1]} w_{32}^{[2]} + b_2^{[2]} \qquad a_2^{[2]} = sigmoid(z_2^{[2]})$$
$$z_3^{[2]} = a_1^{[1]} w_{13}^{[2]} + a_2^{[1]} w_{23}^{[2]} + a_3^{[1]} w_{33}^{[2]} + b_3^{[2]} \qquad a_3^{[2]} = sigmoid(z_3^{[2]})$$

Finalmente en la Figura 1.15 se observan las conexiones entre la *hidden layer 2* y la única *output layer* que se tiene:

Figura 1.15:
Forward propagation entre hidden layer 2 y output layer (S. Hernández, 2019).



Ecuaciones de la *output layer*: de lado izquierdo se observa la función agregación y de lado derecho la función activación que permite la existencia de una salida.

$$z^{[3]} = a_1^{[2]}w_1^{[3]} + a_2^{[2]}w_2^{[3]} + a_3^{[2]}w_3^{[3]} + b^{[3]} \qquad \hat{y} = a^{[3]} = sigmoid(z^{[3]})$$

Todos estos cálculos realizados tienen la finalidad de obtener el resultado de la red neuronal, o en el caso de estar entrenando realizar una predicción que posteriormente se comparará con la etiqueta del conjunto de datos. El número resultante en $\hat{y}$ se encontrará entre cero y uno ($0 \leq \hat{y} \geq 1$), esto es un valor continuo que se puede interpretar como la probabilidad de que lo que se le haya proporcionado en las *input features* correspondan con una clase positiva, es decir igual a uno (S. Hernández, 2019).

Función de coste

También conocida como función de perdida, se encarga de calcular la diferencia que hay entre los datos de predicción $\hat{y}$ y los datos nuevos "y". El objetivo del entrenamiento es minimizar la función de coste, lo que implica ajustar los parámetros (*weight* y *bias*) de la red de manera que la diferencia entre las predicciones y los valores reales se reduzca, son calcificadas como se muestra a continuación (S. Hernández, 2019).

- Problemas de regresión
  - Error medio cuadrado (*Mean Squared Error*)
  - Error absoluto medio (*Mean Absolute Error*)
- Problemas de clasificación
  - Entropía cruzada binaria (*Binary Cross Entropy*)
  - Entropía cruzada categórica (*Categorical Cross Entropy*)

Propagación hacia atrás (*Back propagation*)

Es un proceso que sirve para modificar los valores de los parámetros del modelo, de tal manera que se vaya minimizando el resultado de la fusión de error. Comenzando desde el resultado en función de error respecto a la predicción de la red neuronal y a la etiqueta de base de datos, haciendo una propagación hacia atrás de todas las derivadas en cada neurona, de manera que se tenga una colusión sobre como modificar el parámetro del modelo $w_1 ... w_n$.

Al modificar los pesos en la *ANN* se garantiza la minimización en la función de error. En la Figura 1.17 se muestra la representación en diagrama de bloques de la Figura 1.16. En ella se se puede observar que tenemos dos características de entradas $x_1 = a_1^{[0]}$, $x_2 = a_2^{[0]}$ multiplicados por los pesos correspondientes $w_1^{[1]}$, $w_2^{[1]}$ y $b^{[1]}$. La función $z^{[1]}$ se muestra en el primer bloque, seguida de la función activación $a^{[1]}$(función tanh). Posteriormente el resultado de la función anterior llega a la entrada de la siguiente función de $z^{[2]}$, pasando a la función $a^{[2]}$ y finalmente se obtiene el error $L(a^{[1]}, y)$.

Figura 1.16:
RNN con una sola hidden layer *y* output layer (S. Hernández, 2019).

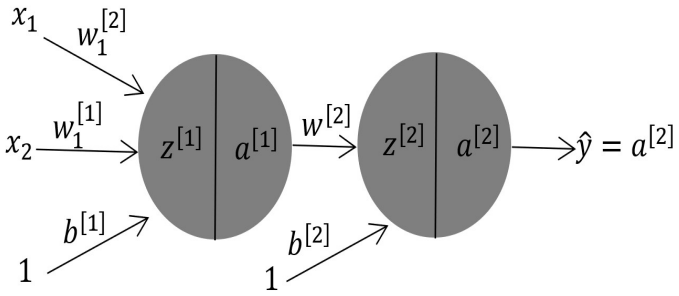
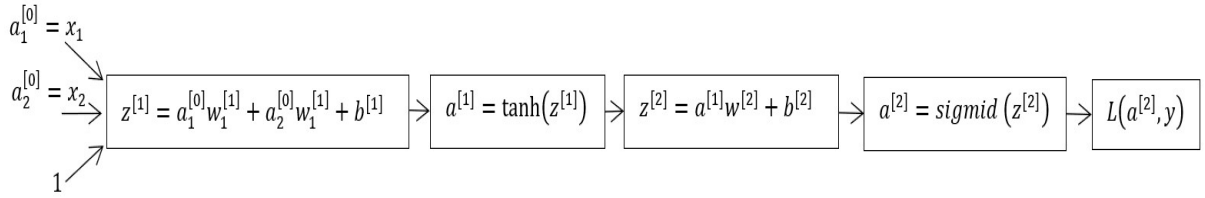


Figura 1.17:
Representación en diagrama de bloques de la Figura 1.16 (S. Hernández, 2019).



Se muestran las ecuaciones de las derivadas que corresponden a cada uno de los bloques en la Figura 1.17 y el flujo comenzando de derecha a izquierda.

$$\frac{\mathrm{d}z^{[1]}}{\mathrm{d}w_1^{[1]}} \leftarrow \frac{\mathrm{d}a^{[1]}}{\mathrm{d}z^{[1]}} \leftarrow \frac{\mathrm{d}z^{[2]}}{\mathrm{d}a^{[1]}} \leftarrow \frac{\mathrm{d}a^{[2]}}{\mathrm{d}z^{[2]}} \leftarrow \frac{\mathrm{d}L}{\mathrm{d}a^{[2]}}$$

En la ecuación (1.9) se muestran las derivadas que se realizan para la modificación del *weight* de entrada $w_1^{[1]}$.

$$\frac{\mathrm{d}L}{\mathrm{d}w_1^{[1]}} = \frac{\mathrm{d}L}{\mathrm{d}a^{[2]}} \cdot \frac{\mathrm{d}a^{[2]}}{\mathrm{d}z^{[2]}} \cdot \frac{\mathrm{d}z^{[2]}}{\mathrm{d}a^{[1]}} \cdot \frac{\mathrm{d}a^{[1]}}{\mathrm{d}z^{[1]}} \cdot \frac{\mathrm{d}z^{[1]}}{\mathrm{d}w^{[1]}} \tag{1.9}$$

En la ecuación (1.10) se muestran las derivadas que se realizan para la modificación del *weight* que se encuentra en la zona intermedia de la *ANN*, $w_1^{[1]}$.

$$\frac{\mathrm{d}L}{\mathrm{d}w^{[2]}} = \frac{\mathrm{d}L}{\mathrm{d}a^{[2]}} \cdot \frac{\mathrm{d}a^{[2]}}{\mathrm{d}z^{[2]}} \cdot \frac{\mathrm{d}z^{[2]}}{\mathrm{d}w^{[2]}} \tag{1.10}$$

El resultado de realizar la multiplicación de cada una de las derivadas de cada nodo respecto al anterior, obtenemos las siguientes ecuaciones, que son las derivadas de cada uno de los parámetros:

Backward propagation

$$dw^{[2]} = a^{[1]} \cdot (a^{[2]} - y)$$
$$db^{[2]} = a^{[2]} - y$$
$$dw_1^{[1]} = (a^{[2]} - y) \cdot w^{[2]} \cdot (1 - a^{[1]^2}) \cdot x_1$$
$$dw_2^{[1]} = (a^{[2]} - y) \cdot w^{[2]} \cdot (1 - a^{[1]^2}) \cdot x_2$$
$$db^{[1]} = (a^{[2]} - y) \cdot w^{[2]} \cdot (1 - a^{[1]^2})$$

Parameter update

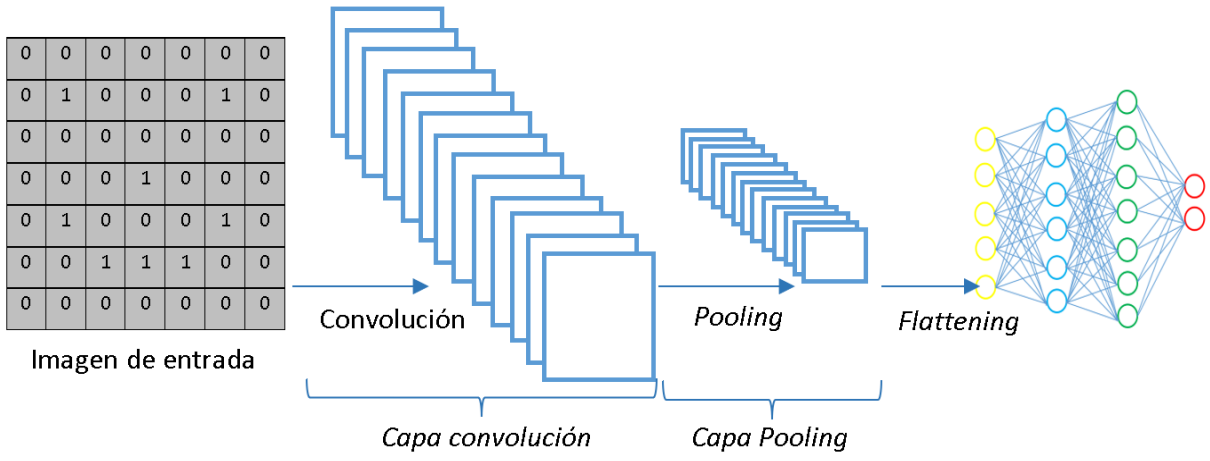
$$w^{[2]} = w^{[2]} - \eta \cdot dw^{[2]}$$
$$b^{[2]} = b^{[2]} - \eta \cdot db^{[2]}$$
$$w_1^{[1]} = w_1^{[1]} - \eta \cdot dw_1^{[1]}$$
$$w_2^{[1]} = w_2^{[1]} - \eta \cdot dw_2^{[1]}$$
$$b^{[1]} = b^{[1]} - \eta \cdot db^{[1]}$$

Redes neuronales convolucionales

Una *CNN* es una tecnología de red neuronal utilizada en el campo de la visión por computadora (*Compute Vision*, *CV*). (Fukushima, 1980) introdujo el concepto original de redes neuronales convolucionales. Las *CNN* están compuestas por múltiples capas convolucionales y de agrupación, seguidas de capas completamente conectadas que realizan la clasificación final (ver Figura 1.18). Al entrenar una *CNN* con un conjunto de datos etiquetado, la red aprende automáticamente a reconocer patrones y características relevantes.(Heaton, 2015).

Arquitectura

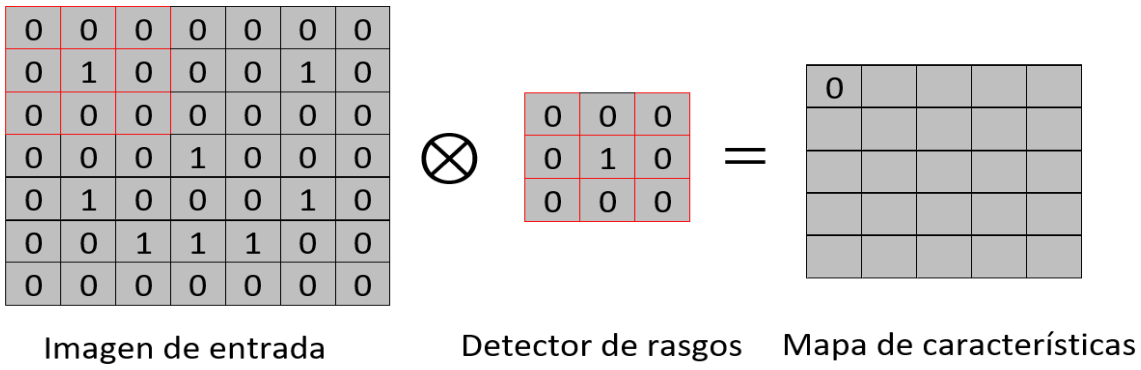
Figura 1.18:
Principales capas de una CNN (Team, 2022).



Capa convolucional (*Layer convolutional*)

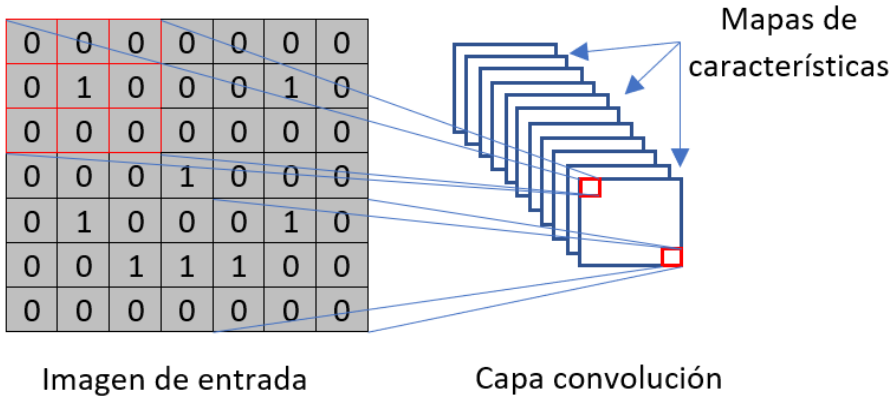
La convolución efectúa una serie de productos y sumas entre la matriz de entrada y una matriz *kernel* o filtro de tamaño n (ver Figura 1.19), reduciendo la dimensión de la matriz de entrada dividiéndola en subregiones y permitiendo generalizar las características.

Figura 1.19:
Proceso de convolución (Andueza, 2022).



Después de la convolución se obtiene una versión simplificada de la información de la imagen original, reduciendo también su tamaño a través de mapas de características como se muestran en la Figura 1.20. Teniendo en cuenta que se pierde información de la imagen original, sin embargo, el objetivo de realizar el filtrado es detectar ciertas formas, características, y rasgos de la imagen global.

Figura 1.20:
Formación de la capa convolución (Andueza, 2022).

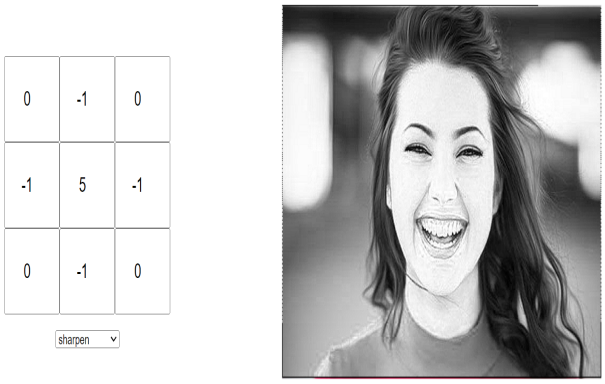


Filtro (*Kernel*)

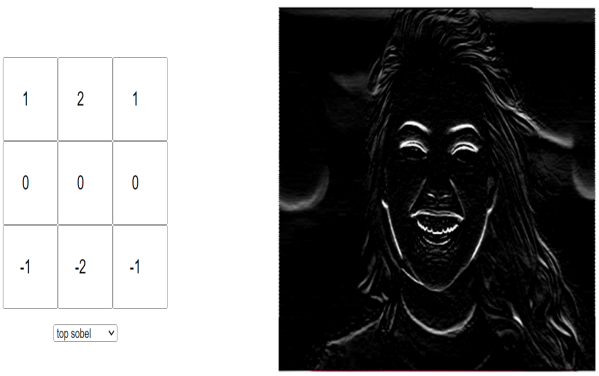
Es un filtro que se aplica a una imagen para extraer ciertas características importantes como por ejemplo; bordes, enfoque, desenfoque, entre otros, de acuerdo a su configuración, como se puede observar en la Figura 1.21 con algunos tipos de *Kernel*.

Figura 1.21:
Tipos de kernel (Andueza, 2022).

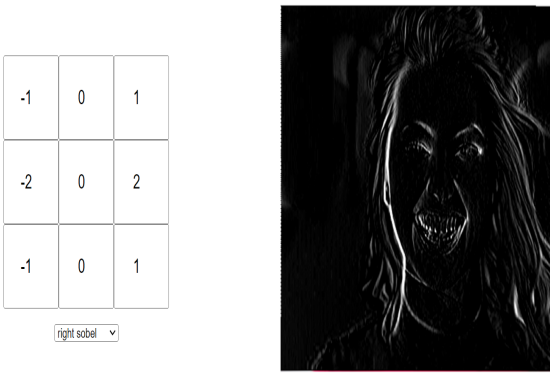
(a) *Sharpen*



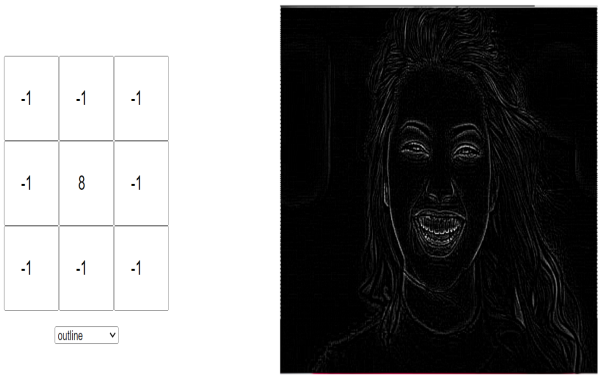
(b) *Top Sobel*



(c) *Right sobel*



(d) *0utline*



Capa agrupación (***Layer pooling***)

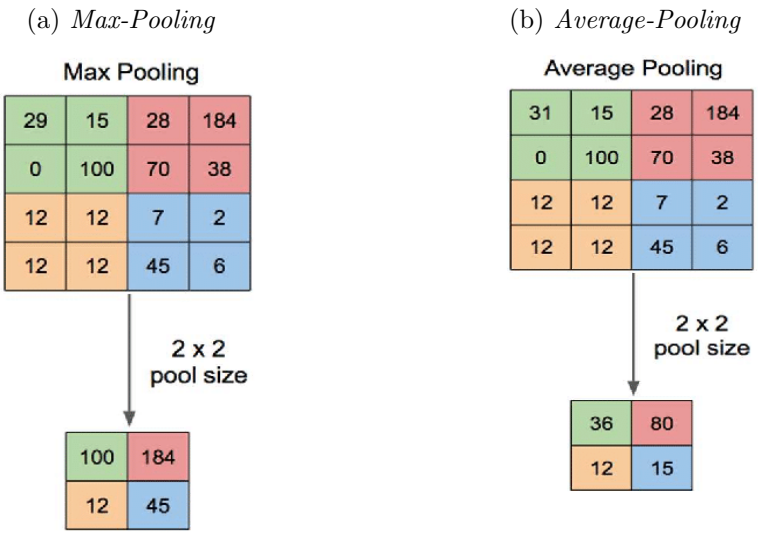
La operación que permite tener el análisis dentro de una imagen dividida en regiones, para obtener información relevante de las imágenes, es denominado *pooling* (Aggarwal, 2018).

Los dos tipos de *pooling* más comunes son:

- ***Max-Pooling*** Al utilizar el *max-Pooling* se reduce la cantidad de datos que se extraen entre capas, lo que ayuda a facilitar el procesamiento de imágenes y el entrenamiento de la red. El funcionamiento consiste en dividir la imagen en regiones del mismo tamaño, posteriormente de cada región se va a extraer el valor máximo que corresponderá a un píxel en la imagen resultante. Por ejemplo, a la imagen de 4×4 (véase Figura 1.22a), se aplica un filtro *max-pooling* de 2×2. Resultarán 4 diferentes regiones y por cada una de estas regiones se extrae el valor máximo asignando dicho valor al *pixel* correspondiente a la imagen de salida (Chollet, 2021).
- ***Average-Pooling*** Es una operación de agrupamiento que calcula el valor promedio de los bloques en un mapa de características y lo usa para producir un mapa de características reducido (ver Figura 1.22b). Las características de extracción son más

suaves que el *max-pooling*, obtiene características pronunciadas como los bordes (Yani et al., 2019). Esto no es preferible al *max-pooling*, ya que falla en la detección de bordes afilados y otras características complejas.

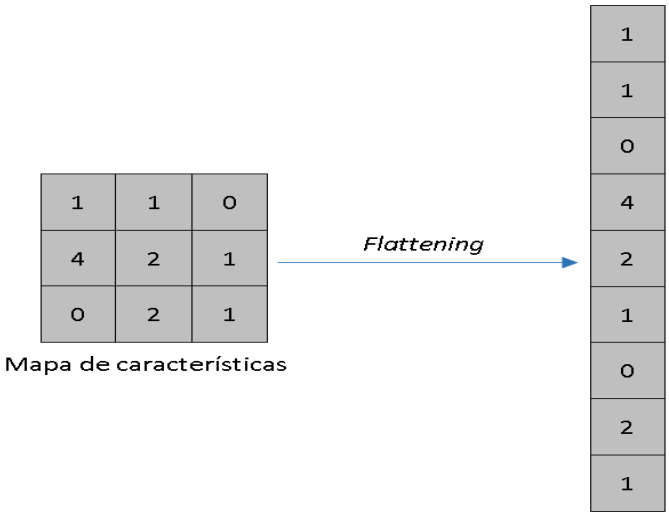
Figura 1.22:
Tipos de pooling (Yani et al., 2019).



Aplanamiento (*Flattening*)

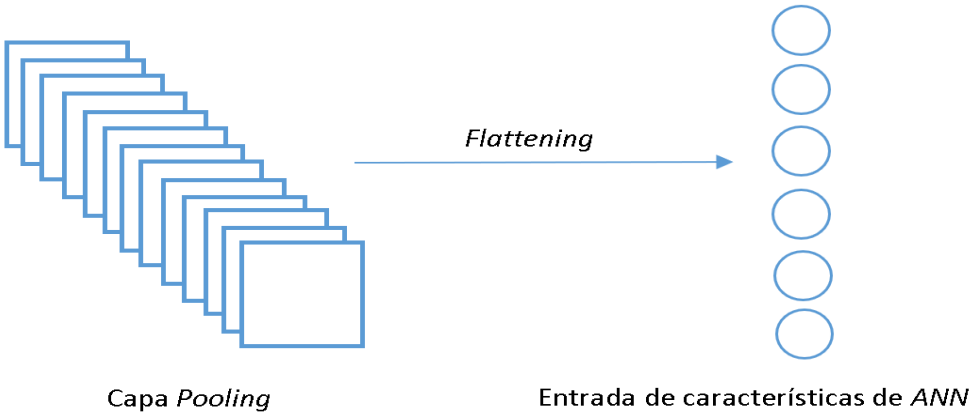
Después de que los datos han pasado a través de las diferentes capas; pooling, convolucional y *flattening* se obtiene un mapa de características agrupadas. Como el nombre de este paso indica, literalmente se aplana el mapa de características agrupadas en una columna, como se muestra en la Figura 1.23.

Figura 1.23:
Representación del flattening (Frogames, 2020).



En la Figura 1.24 se observan múltiples mapas de características que terminan como un vector de datos de entrada, que posteriormente pasan a través de la *ANN* para que sean procesados y así tomar la decisión final (Frogames, 2020).

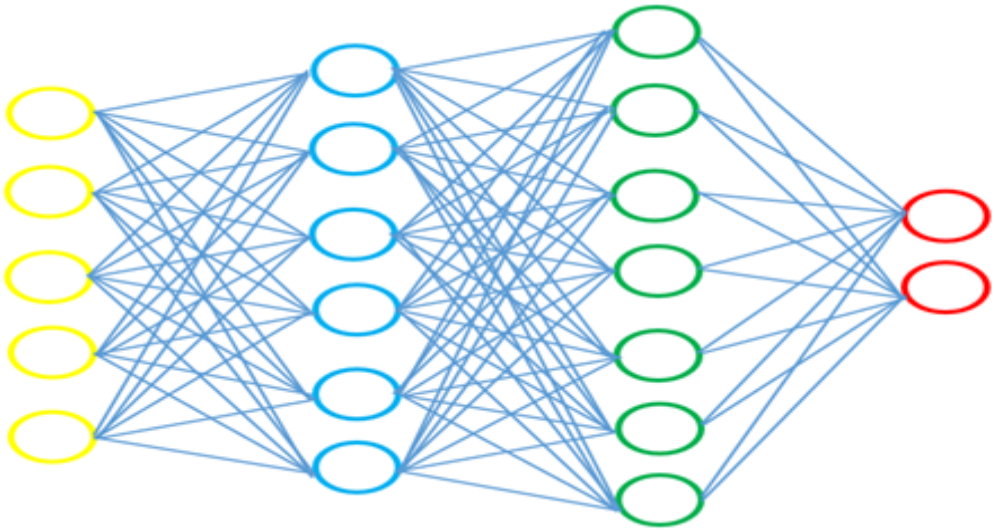
Figura 1.24:
Representación de flattening como CNN (Frogames, 2020).



Capa totalmente conectada (*Fully-connected*)

La capa *fully-connected* es una red multicapa del tipo *feed-forward* (ver Figura 1.25), que utiliza una función *softmax* y se encuentra después del *flattening*. El término "totalmente conectado" significa que cada neurona de una capa está conectada a todas las neuronas de la siguiente capa. La salida de las capas de convolución y agrupación representa las características de alto nivel en la imagen de entrada. El propósito de la capa completamente conectada es usar estas características para clasificar la imagen de entrada en diferentes categorías según el conjunto de datos de entrenamiento (Martínez, 2017).

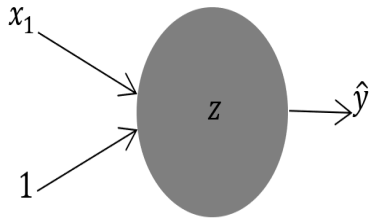
Figura 1.25:
Capa fully-connected (Martínez, 2017).



Desajustado (*Underfittign*)

Se representa con un problema en el cual se tiene un conjunto de datos de entrenamiento, formado por una única característica de entrada y las características de salida que representan un problema de regresión, la perdición del costo y de una casa basándose en el número de habitaciones x_1. En la Figura 1.26 se muestra una neurona, la cual tiene una característica de entrada x_1 y una neurona *bias*, las cuales entran a una función de relación z. Al ser un problema de regresión no se cuenta con la función de activación, de esta manera se obtendría la función de salida (S. Hernández, 2019).

Figura 1.26:
Arquitectura básica de una red neuronal profunda.

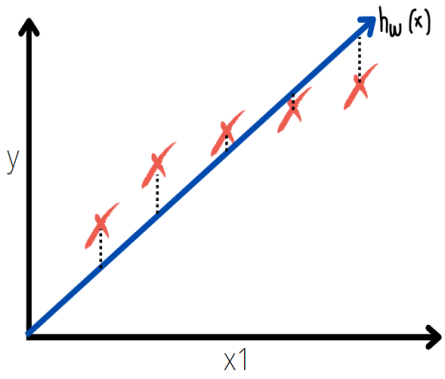


La Figura 1.26 es representada matemáticamente de la siguiente forma:

$$h_w(x) = w_0 + w_1 x_1$$

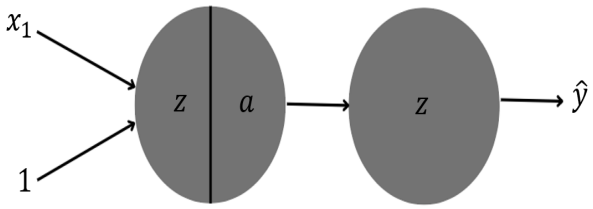
Al no contar con una función de activación solo se genera una linea recta como se muestra en la Figura 1.27 en la cual se observa que hay un error respecto al conjunto de datos de entrenamiento y por lo tanto, las predicciones que realice no serán las adecuadas (S. Hernández, 2019).

Figura 1.27:
Representación gráfica de la generación de underfitting (S. Hernández, 2019).



Se aumenta la complejidad de arquitectura en la red neuronal (ver Figura 1.28), añadiendo una primer neurona la cual tendrá función de agregación y de activación, posteriormente se agrega otra neurona que solo tendrá función de agregación, finalmente se tiene la salida obteniendo la función hipótesis (S. Hernández, 2019).

Figura 1.28:
Ejemplo de una arquitectura para un entrenamiento óptimo.

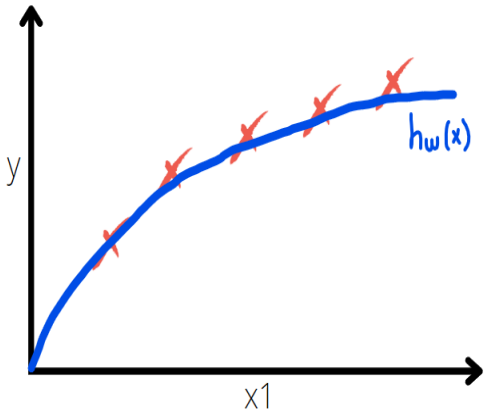


La Figura 1.28 es representada matemáticamente de la siguiente forma:

$$h_w(x) = w_0 + w_1 x_1 + w_2$$

Esta función hipótesis construye un límite de decisión como se muestra en la Figura 1.29 que a pesar de que se genera un error para los ejemplos de entrenamiento, este es muy cercano a 0, además para ejemplos nuevos se observaría que la predicción también se encuentra alineada con la tendencia del conjunto de datos. Por lo cual siempre se debe de buscar este equilibrio entre un modelo demasiado sencillo para la resolución de un problema y un modelo complejo que provoca *overfitting* en el conjunto de datos de entrenamiento (S. Hernández, 2019).

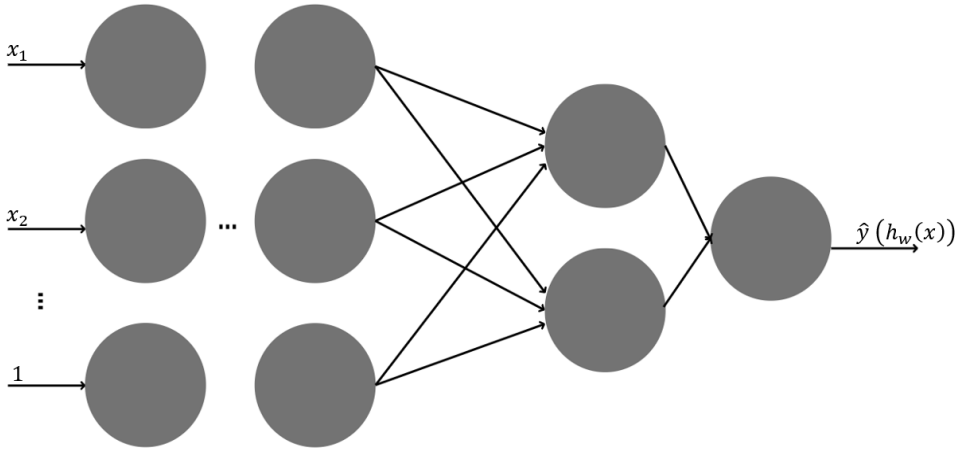
Figura 1.29:
Nivel óptimo de un entrenamiento (S. Hernández, 2019).



Sobreajuste (*Overfitting*)

Si la red neuronal tiene muchas capas y neuronas, es posible que la función hipótesis generada por el algoritmo se adapte muy bien al conjunto de entrenamiento ($J(W) = 0$), pero falle al generalizar con nuevos ejemplos. Esto se conoce como *overfitting*. Se esta ajustando muy bien al los ejemplos de entrenamiento del conjunto de datos, que si se le mostraran ejemplos nuevos que no se encuentren en el conjunto de datos y se realizará la predicción sería errónea (S. Hernández, 2019).

Figura 1.30:
Ejemplo de una arquitectura para generar el overfitting.

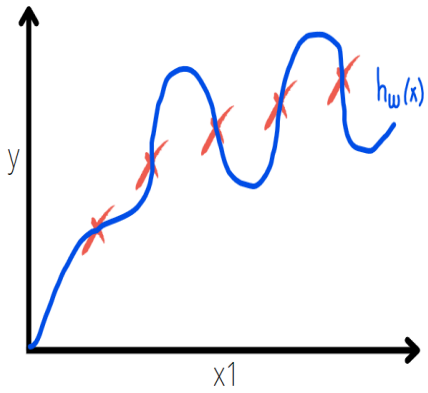


La Figura 1.30 es representada matemáticamente de la siguiente forma:

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + ... + w_n x_1^n$$

El resultado gráfico de este entrenamiento se muestra en la Figura 1.31, observando que la función se está ajustando demasiado al conjunto de datos de entrenamiento, viendo la red como un perfecto entrenamiento, sin embargo, funciona solo para los ejemplos que se tiene en ese conjunto de datos y no para ejemplos nuevos.

Figura 1.31:
Representación gráfica de la generación de overfitting (S. Hernández, 2019).



Las técnicas para resolver el *overfitting*:

- Aumentar el número de datos para entrenamiento. Entre más grande sea nuestra base de datos de entrenamiento se puede resolver el problema del *overfitting* porque se contaría con más ejemplos diferentes con lo cual generaría un menor error de predicción.
- Reducción del número de capas.

- Reducción del número de neuronas por capa.
- Regularización.
- *Dropout*.
- *Data Augmentation*.

Transfer learnig

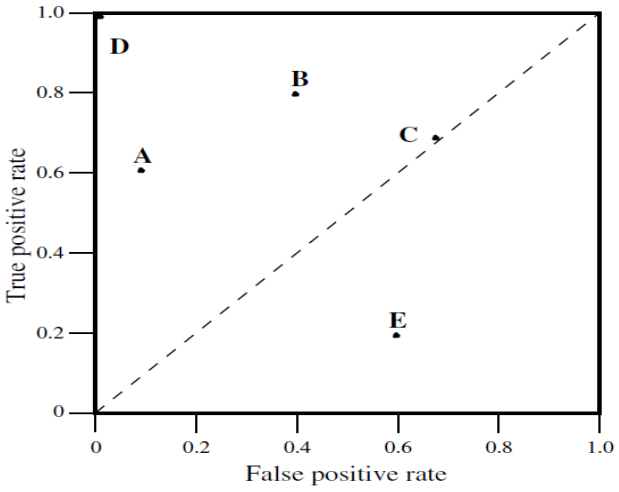
Una de las técnicas para realizar *DL* es el *Transfer learnig*. Esta técnica se puede usar cuando se desea entrenar una red neuronal profunda muy grande, por lo que no se recomienda comenzar desde cero, ya que puede requerir una gran cantidad de datos y cálculos que requieren mucho tiempo (días o semanas) para entrenar, esta técnica proporciona la capacidad de transferir el conocimiento de una red pre-entrenada con características similares a las necesarias para resolver el problema. El uso de esta técnica acelera el proceso de entrenamiento porque no necesita un conjunto de datos tan grande como lo haría desde cero, y no necesita tantos recursos computacionales (Goodfellow et al., 2016).

Métricas de evaluación

La curva *ROC*

Para métodos de *ML*, se utiliza la métrica de las características de funcionamiento del receptor (*Receiver Operating Characteristic*, *ROC*) para visualizar el rendimiento de un clasificador. Se considera un modelo óptimo cuando el valor de la tasa de verdaderos positivos (*True Positive Rate*, *TPR*) aumenta rápidamente y se mantiene bajo la tasa de falsos positivos (*False Positive Rate* (*FPR*). En la Figura 1.32 se muestra un ejemplo en donde el punto inferior izquierdo (0,0) representa que el modelo no tiene predicciones positivas ni negativas, para el punto superior derecho (1, 1) solo realiza predicciones positivas y no negativas, finalmente en el punto (0, 1) representa la clasificación perfecta (Fawcett, 2006).

Figura 1.32:
Curva ROC (Fawcett, 2006).

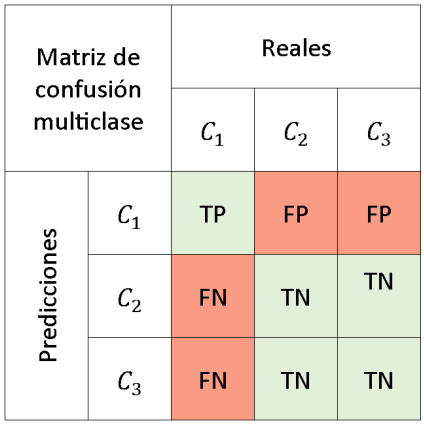


Matriz de confusión

Permite analizar los resultados de cómo trabaja un algoritmo de aprendizaje supervisado, midiendo que tan bueno es un modelo de clasificación construido sobre un sistema de aprendizaje automático. La matriz de confusión evalúa los resultados obtenidos, dividiéndose en verdaderos positivos, falso positivos, falsos negativos y verdaderos negativos. Con estos valores se puede calcular las métricas de *Precision* y *Recall* (Fawcett, 2006).

- Verdaderos positivos (*True Positive, TP*): instancias correctas respecto a los datos reales.
- Falsos positivos (*False Positive, FP*): son los casos que en realidad son negativos, pero que el modelo predice como positivos.
- Falsos negativos (*False Negative, FN*): instancias predichas incorrectamente como negativas cuando en realidad son positivas.
- Verdaderos negativos (*True Negative, TN*): son las instancias que el modelo predice correctamente como negativas (Sánchez, 2016).

Figura 1.33:
Matriz de confusión multiclase (Ameyaw et al., 2022).



A partir de estos 4 valores surgen las métricas de la matriz de confusión: la precisión (*Precision*) es la proporción de predicciones correctas en relación con todas las instancias que el modelo clasifica como positivas (tanto las que son verdaderamente positivas como las falsas positivas). Para obtener este valor se aplica la fórmula (1.11) (Goodfellow et al., 2016).

$$Precision = \frac{TP}{TP + FP} \tag{1.11}$$

La puntuación *F1-score* es una métrica que combina las puntuaciones de *Precision* y *Recall*, como se muestra en la fórmula (1.12). La puntuación *F1-score* es especialmente útil en casos en los que el desequilibrio entre clases puede conducir a una evaluación engañosa del rendimiento (Aurelien, 2019).

$$F1 - score = \frac{(2)(precision)(recall)}{precision + recall} \tag{1.12}$$

La sensibilidad (*recall* o *sensitivity*) también se conoce como tasa de verdaderos positivos (*True Positive Rate, TPR*), se refiere a la proporción de instancias positivas que el modelo clasifica correctamente como positivas, en relación con todas las instancias que realmente son positivas (incluyendo tanto las clasificadas correctamente como las que se clasifican incorrectamente como negativas). Se calcula dividiendo el número de predicciones verdaderas positivas entre la suma de las predicciones verdaderas positivas y las predicciones falsas negativas, como se muestra en la fórmula (1.13) (Aurelien, 2019).

$$Recall = \frac{TP}{TP + FN} \tag{1.13}$$

La tasa de falsos positivos (*False Positive Rate, FPR*), representa la proporción de instancias negativas que el modelo predice incorrectamente como positivas, en comparación con el número total de instancias negativas reales (ver fórmula (1.14)). Es una métrica importante, especialmente utilizada para problemas de desequilibrio de clases (Raschka, 2015).

$$FPR = \frac{FP}{FP + TN} \tag{1.14}$$

La métrica de exactitud (*Accuracy*, ACC) se refiere a la aproximación del resultado de una medición con el valor verdadero, da el porcentaje total de las predicciones positivas que fueron correctas, para obtener este valor se aplica la fórmula (1.15) (Müller & Guido, 2016).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.15}$$

La métrica de especificidad (*specificity*), también conocida como la tasa de verdaderos negativos, (*True Negative Rate, TNR*). Se trata de los casos negativos que el algoritmo ha clasificado correctamente. Expresa cuan bien puede el modelo detectar esa clase (ver fórmula (1.16)) (Berzal, 2018).

$$Specificity = \frac{TN}{TN + FP} \tag{1.16}$$

Existen cuatro casos posibles al relacionar las métricas de *precision* y *recall*, y se muestran a continuación:

- Alta *Precision* y alto *Recall*: el modelo de *ML* seleccionado maneja perfectamente esa clase.
- Alta *Precision* y bajo *Recall*: el modelo de *ML* seleccionado no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- Baja *Precision* y alto *Recall*: el modelo de *ML* seleccionado detecta bien la clase, pero también incluye muestras de la otra clase.
- Baja *Precision* y bajo *Recall*: el modelo de *ML* seleccionado no logra clasificar la clase correctamente.

1.2.3. Herramientas computacionales

Dentro de la presente sección se abordarán las herramientas tecnologías que se utilizaron durante el desarrollo de este trabajo.

Python

A principios de los años 90, el informático Guido Van Rossuma creo el lenguaje de programación *Python*, cuyo nombre está inspirado en el grupo de cómicos ingleses *Monty Python*. Es un lenguaje de programación de alto nivel, lo que lo ha convertido en uno de los más populares a nivel mundial por su versatilidad y facilidad de programación similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de *script*, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos (González, 2011).

Python aplicado en la *IA*

Debido a la baja complejidad de operación en el código de *Python*, lo posiciona como uno de los lenguajes de mayor popularidad en el mundo, esto lo ha convertido en el medio más aceptable para la creación de sistemas de *IA*. Las bibliotecas de *Python* abarcan áreas de computación científica avanzada y el aprendizaje automático como *Numpy*, *Scipy* y *Pybrain*. Por tal motivo, su amplio uso en los lenguajes para *IA* (Joshi, 2017).

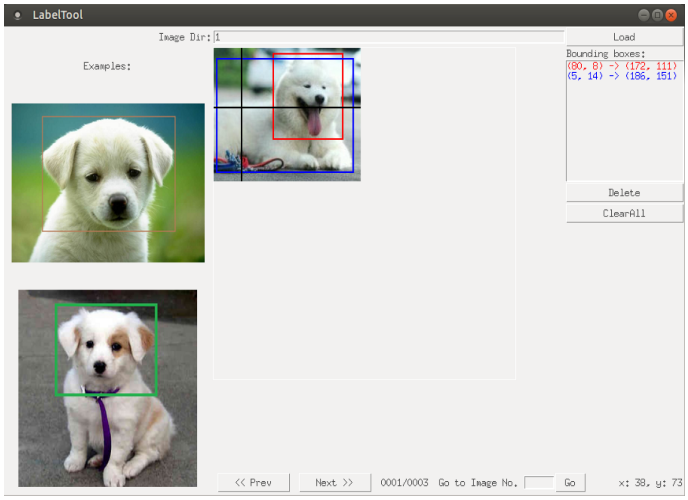
Bibliotecas de *Python* implementadas en *IA* y *DL*

- **TensorFlow**: esta biblioteca fue desarrollada por *Google*, permite a través de diagramas de flujo de datos realizar cálculos numéricos para codificar un gráfico. Los nodos de este gráfico pasan a ser operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales) (Pattanayak, 2017).
- **Keras**: es una interfaz de alto nivel, pero de fácil manejo, el cual permite trabajar con las redes neuronales de una manera sencilla. Utilizada para verificar que los desarrollos obtengan los resultados esperados rápidamente, gracias a su eficacia. **Keras** utiliza otras librerías de *DL* (*TensorFlow*, *CNTK* o *Theano*) de forma transparente para hacer el trabajo que se indique (Kirkvik, 2022).
- **PyTorch**: es una librería desarrollada por *Facebook*, que permite de forma eficiente realizar cálculo numérico. Posee una cualidad de procesamiento mejorado en *GPUs*, por eso se considera en desarrollo de *DL* (Pattanayak, 2017).

LabelImg

LabelImg es una de las herramientas que se pueden utilizar para crear las anotaciones de las imágenes (véase Figura 1.34) que van a ser parte del *set* de entrenamiento. A través de *bounding boxes* que contienen a cada uno de los objetos a identificar (Noman et al., 2019). La finalidad de generar etiquetas, es para ayudar a identificar las características principales de los datos que desea entrenar, así mismo se identifican los conjuntos de datos que no están etiquetados y por lo tanto no son relevantes para el entrenamiento. La calidad de los datos etiquetados son esenciales para la visión por ordenador y la construcción de un modelo de alto rendimiento.

Figura 1.34:
Ejemplo del uso de la herramienta LabelImg (Noman et al., 2019).



Visión por computadora

La visión por computadora es un campo interdisciplinario en la intersección de la informática, la óptica y la inteligencia artificial. Su principal objetivo es desarrollar algoritmos y métodos que permitan a las computadoras interpretar y comprender contenidos visuales del mundo real, es decir, imágenes y vídeos. Estos algoritmos permiten a las computadoras realizar tareas como reconocimiento de objetos, seguimiento de movimiento, análisis de imágenes médicas, reconocimiento facial, visión artificial y muchas otras aplicaciones relacionadas con la percepción visual (Alegre et al., 2019).

YOLO v5

YOLO es un algoritmo para detección de objetos desarrollado por Joseph Redmon, Santosh Divvala, Ross Girshick y Ali Farhadi en 2015, predice simultáneamente múltiples cuadros delimitadores y probabilidades de clase para esos cuadros, usando características de toda la imagen para predecir cada cuadro limitador utiliza únicamente capas convolucionales, lo que la convierte en una red totalmente convolucional, por lo que puede evitar el uso de capas de agrupación. En su lugar, se utiliza una capa convolucional con un paso de dos para reducir la muestra de los mapas de características (PyLessons, 2019).

YOLOv5 desarrollado por el grupo *Ultralytics* en el 2020, tiene cinco versiones: nano (*nano, n*), pequeña (*small, s*), mediana (*medium, m*), grande (*large, l*) y extra grande (*extra-large, x*) como puede observar en la Figura 1.35, cada una de las cuales ofrece tasas de precisión progresivamente más altas (ver 1.36). Cada variante también requiere una cantidad diferente de tiempo para entrenar.

Figura 1.35:
Comparación de las versiones de YOLO v5 (Jocher, 2020).

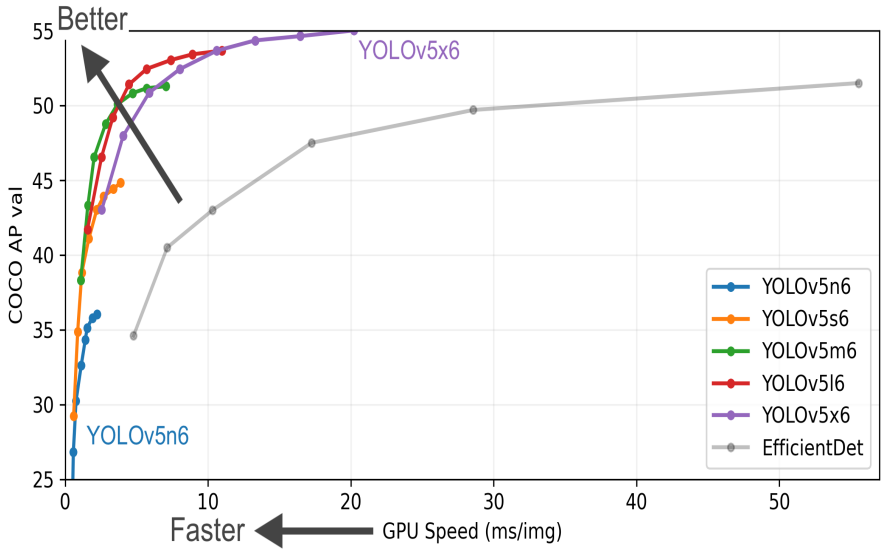
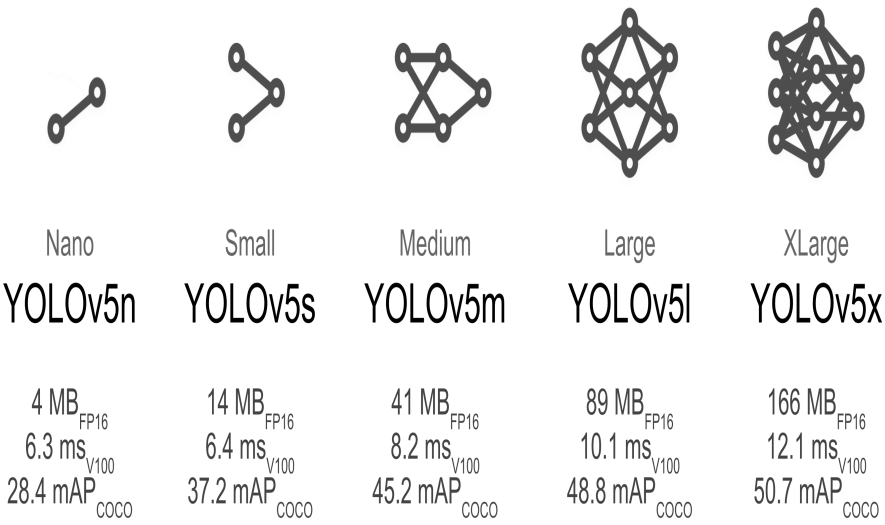


Figura 1.36:
Descripción de los modelos para las versiones de YOLO v5 (Jocher, 2020).



En comparación con los algoritmos de reconocimiento, un algoritmo de detección predice las etiquetas de clase y detecta la ubicación de los objetos. Por lo tanto, no solo clasifica la imagen en una categoría, sino que también puede detectar múltiples objetos dentro de una imagen (Redmon, 2012). Las principales características de operación para las diferentes versiones de *YOLOv5* se muestran en la Tabla 1.2, donde una de las características para poder realizar una inferencia sin problemas de latencia en algún dispositivo son las operaciones de punto flotante por segundo (Floating Point Operations per Second, *FLOPS*).

Tabla 1.2:
Comparación de las características de los modelos de YOLO v5 (Jocher, 2020)

	YOLOv5n	*YOLOv5s*	*YOLOv5m*	*YOLOv5l*	*YOLOv5x*
Size (pixels)	640	640	640	640	640
mAP 50-95	28.0	37.4	45.4	49.0	50.7
mAP 50	45.7	56.8	64.1	67.3	68.9
Speed ms	45	98	224	430	776
FLOPS (G)	4.5	16.5	49.0	109.1	205.7
Model depth multiple	0.33	0.33	0.67	1.0	1.33
Layer channel multiple	0.25	0.50	0.75	1.0	1.25

.

Existen dos hiperparámetros que ayudan a configurar la arquitectura, escalar el tamaño y la complejidad del modelo:

- Múltiplo de profundidad del modelo (*model depth multiple*): utilizado para determinar la profundidad o el número de capas de un modelo, determinado cuantas capas deben añadirse o eliminarse a la arquitectura base.
- Canal de capas múltiples *(layer channel multiple)*: escala el número de canales en una capa concreta o en toda la red

Procesamiento

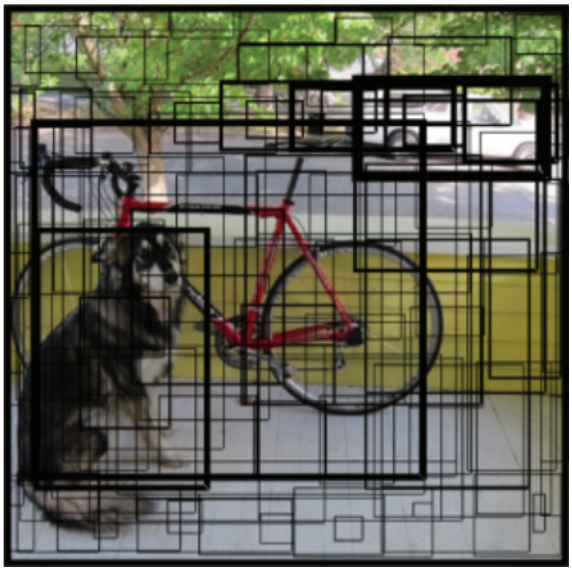
Para la detección de objetos, *YOLO* comienza dividiendo la imagen en una cuadricula S×S (véase Figura 1.37a). Si la celda de la cuadrícula contiene el centro de un objeto, es responsable de detectar ese objeto. Tras haber dividido la imagen en cuadriculas, cada una de las celdas se encargará de predecir B *bounding boxes* (cuadros delimitadores) y cuadros de confianza (véase Figura 1.37b). Son 5 las predicciones que conforman cada caja delimitadora B: x, y, w, h y confianza. Las coordenadas (x, y) representan el centro de la caja en relación con los límites de la celda de la cuadrícula. La anchura (w) y la altura (h) son predicciones relativas a toda la imagen. La predicción de confianza representa la intersección sobre la unión entre la caja predicha y cualquier caja de valor real. De no existir ningún objeto en una celda, se obtiene como resultado cero en la puntuación de confianza, debido a que P(*Objectness*) será cero. En caso contrario, la puntuación de confianza es igual a la *IOU* entre la casilla predicha y la verdad (Redmon et al., 2016).

Cada celda de la cuadrícula también predice C probabilidades de clase condicionales, P(Clase-i Objeto) (véase Figura 1.37c). Estas probabilidades están condicionadas a que la celda de la cuadrícula contenga un objeto. Independientemente del número de cuadros delimitadores asociados, solo se predice un conjunto de probabilidades de clase por celda de la cuadrícula (Wang et al., 2021). Para terminar el proceso se realiza el producto de las probabilidades de clase condicional y las predicciones de confianza de cada celda. Obteniendo como resultado el valor de confianza de la clase para cada celda. Estos valores codifican tanto la probabilidad de que esa clase aparezca en el cuadro como qué tan bien el cuadro predicho se ajusta al objeto, como se muestra en la Figura 1.37d (Redmon et al., 2016).

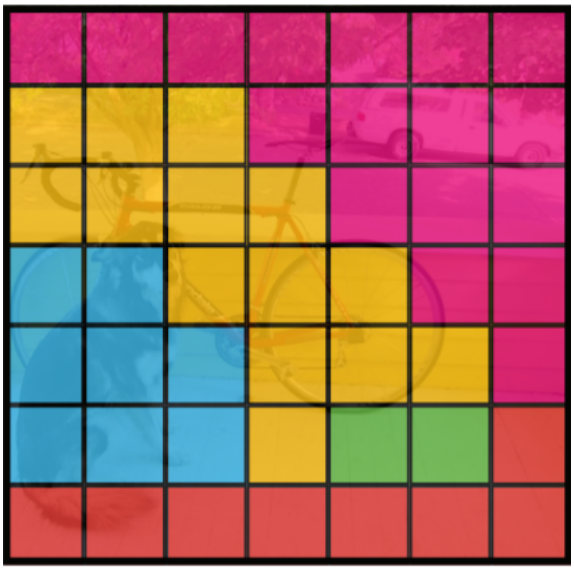
Figura 1.37:
Procesamiento de la detección de objetos con la red YOLO (Wang et al., 2021).

(a) *División de imagen en cuadricula de SxS*

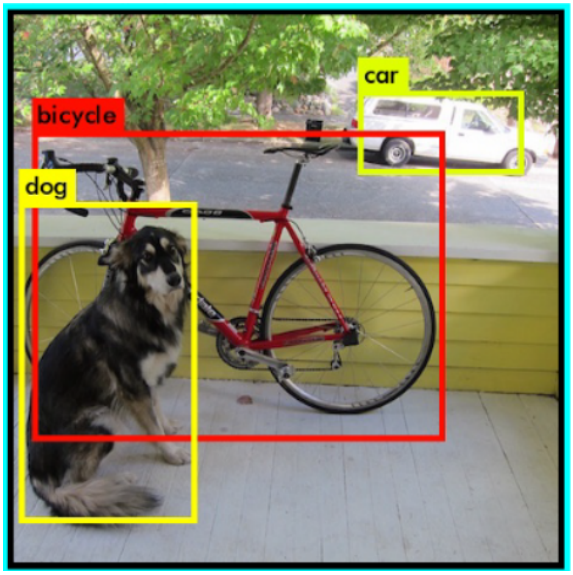
(b) *Cuadros delimitadores*



(c) *Probabilidades de clase*



(d) *Detecciones finales*

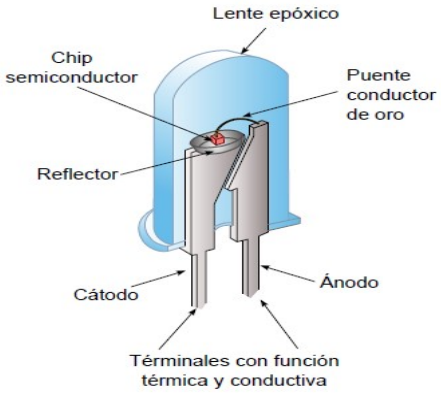


1.2.4. Componentes

Tecnología *LED*

Un *LED*, es un chip de material semiconductor (ver Figura 1.38), que cuando es atravesado por una corriente eléctrica, emite luz monocromática sin producir calor (Barolet, 2008). El chip es conectado con un cable fino a un contacto que actúa como cátodo, al ser conectado eléctricamente produce una luz. Está enlazado con un disipador de calor térmico y una base de cerámica. Este chip está dentro de una lente que protege y modula el haz de luz hacia el ángulo deseado. Para la producción de luz blanca, los chips se revisten con fósforo.

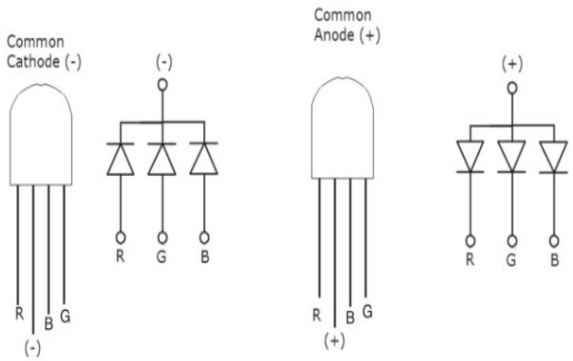
Figura 1.38:
Estructura interna de un LED (Cortéz, 2009).



LED RGB

Es la combinación de tres diodos *LED's* (ver Figura 1.39) normales de colores rojo, verde y azul, comprimidos en uno solo. Los productos *LED RGB* combinan estos tres colores para producir más de dieciséis millones de tonos de luz dentro del espectro electromagnético visible. Existen dos tipos, el ánodo común y cátodo común, donde la única diferencia es la forma de conexión de alimentación (voltaje positivo) y tierra (voltaje cero), por supuesto existe variedad de encapsulados, intensidad y tamaño (Narváez & Cuaspa, 2018).

Figura 1.39:
Esquema de un LED RGB (M. Hernández et al., s.f.).



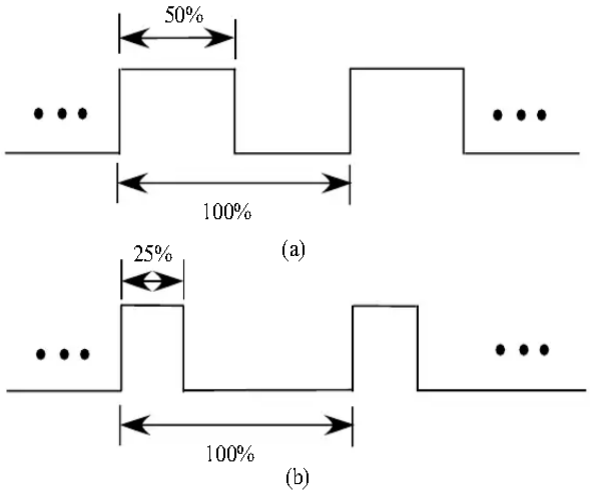
Modulación por ancho de pulso (*Pulse Width Modulation*, *PWM*)

Consiste en generar trenes de pulsos con frecuencia y magnitud fija. Sin embargo, la anchura de los pulsos cambia según una señal moduladora (Yu et al., 1997). Los rápidos flancos ascendentes y descendentes garantizan que los dispositivos semiconductores de potencia se enciendan o apaguen lo más rápido posible para minimizar el tiempo de transición y conmutación (Siew-Chong et al., 2012).

Ciclo de trabajo (*Duty Cycle*)

Es el porcentaje de tiempo que la señal está encendida. Por lo tanto, llamamos a la señal que se muestra en la Figura 1.40 (a) una señal de pulso periódico con un ciclo de trabajo del 50 %, y a la señal correspondiente en la Figura 1.40 (b) una señal de pulso periódico con un ciclo de trabajo del 25 % (Barrett & Pack, 2006).

Figura 1.40:
Comparación de dos señales con el mismo período pero con diferentes ciclos de trabajo (Barrett & Pack, 2006).



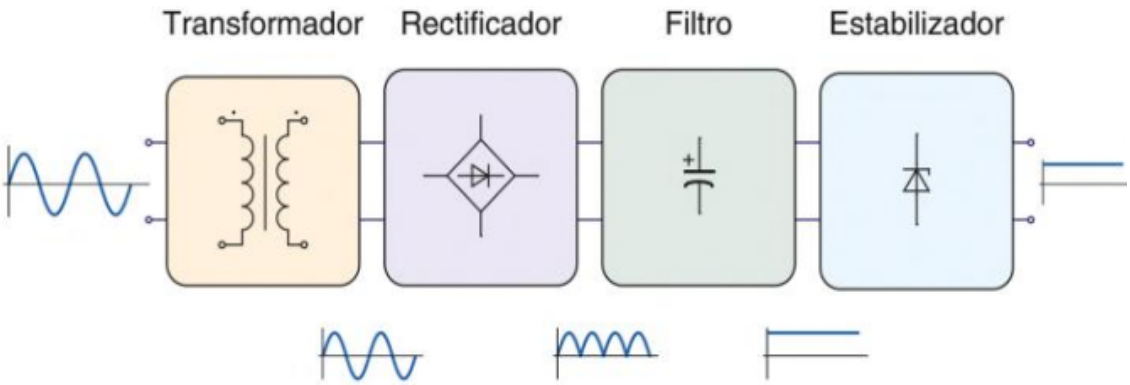
Fuentes de alimentación

Dispositivo electrónico el cual permite la conversión de la tensión alterna de entrada en una tensión continua que es utilizada en distintas áreas de trabajo, en la industria, o en el hogar (Martín, 2017).

Existen dos tipos de fuentes de alimentación, las cuales se describen a continuación:

- Fuentes de alimentación lineales. Este tipo de fuentes se caracteriza principalmente por su sencillo diseño, sin embargo, son de gran tamaño y tienen una eficiencia moderada debido a las perdidas de energía. Están conformadas de varias etapas: transformación, rectificación, filtrado y regulación como se muestra en la Figura 1.41.

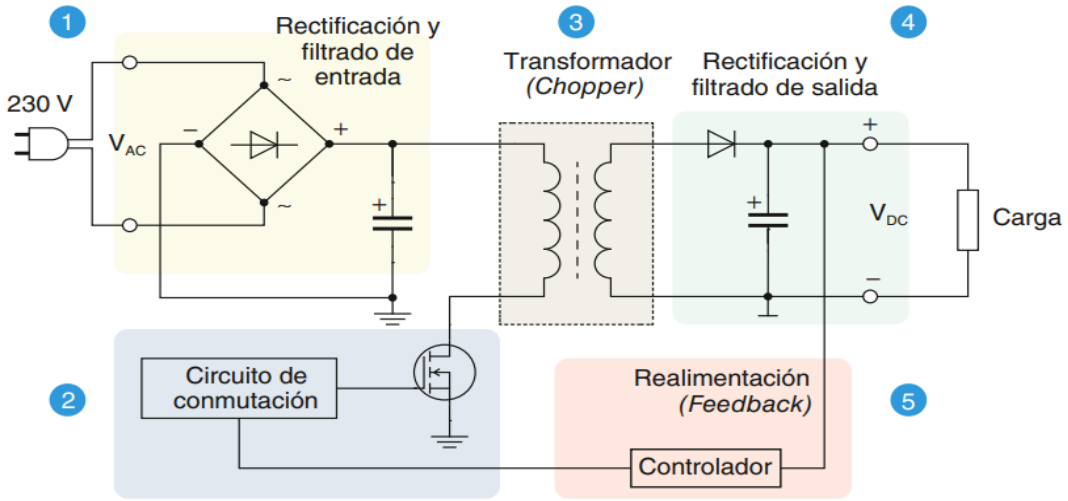
Figura 1.41:
Diagrama de bloques de una fuente de alimentación lineal (Martín, 2017).



- Fuentes de alimentación conmutadas. Este tipo de fuentes trabaja con frecuencias al rededor de los 100 *KHz*, teniendo como principal ventaja, en comparación las fuentes lineales, la reducción de tamaño en el transformador y de la fuente en su totalidad, por tal motivo, hay una disminución en las perdidas de energía al no generar tanto calor como las de tipo lineal, sin embargo , son de mayor complejidad para construir y de mantener debido a los competentes utilizados. En la figura 1.42 se muestran las diferentes etapas de este tipo de fuentes:

  1. Rectificación y filtrado de entrada
  2. Circuito de conmutación
  3. Transformador.
  4. Rectificación y filtrado de salida.
  5. Controlador.

Figura 1.42:
Diagrama de bloques de una fuente de alimentación lineal (Martín, 2017).



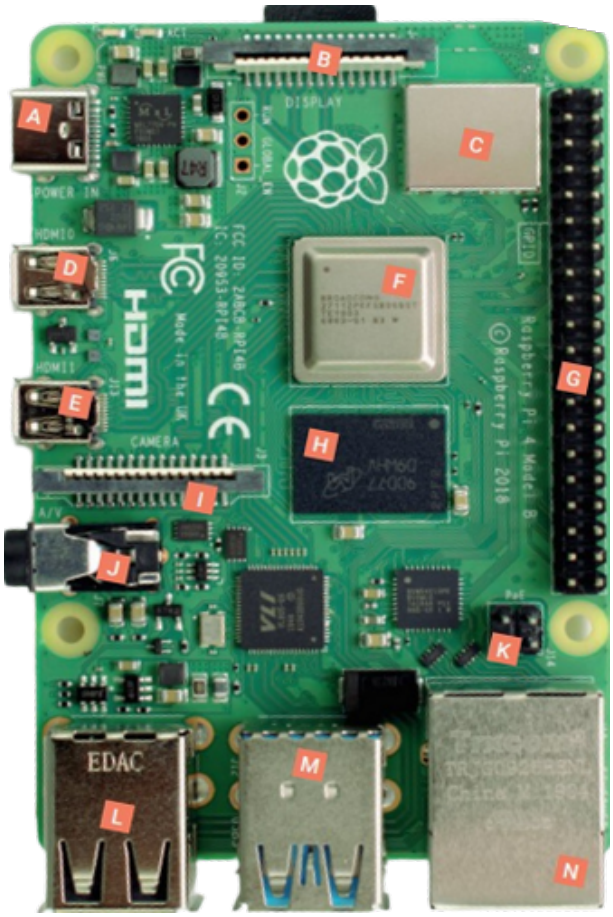
Raspberry pi

Es una computadora de placa única, de tamaño reducido, desarrollada en el Reino Unido por la *Raspberry pi Foundation* en febrero del 2012, con el objetivo de poner en manos de las personas de todo el mundo el poder de la informática y la creación digital.(Halfacree, 2020). El software que utiliza es de código abierto, siendo su sistema operativo oficial una versión adaptada de *Debian*, denominada *Raspberry Pi OS*, aunque permite usar otros sistemas operativos, incluido una versión de *Windows* 10. Para este trabajo se utiliza la *Raspberry pi 4B* con las características que se muestran a continuación.

Componentes de *Rasberry pi 4B*

En todas sus versiones, incluye un procesador *Broadcom*, memoria *RAM*, *GPU*, puertos *USB*, *HDMI*, *Ethernet*, 40 pines *GPIO* (desde la RPi-2G) y un conector para cámara. En la Figura 1.43 se muestra la estructura de una RPi-4G B.

Figura 1.43:
Características de la RPi-4G (Halfacree, 2020).



A. Puerto *USB* tipo-C.

B. Conector de display (Display Serial Interface, DIS).

C. Wireless/ *Bluetooth*.

D. Micro-HDMI 0.

E. Micro-HDMI 1.

F. Procesador: *Broadcom BCM2711*, *quad-core Cortex-A72* (*ARM V8*) 1.5 *GHZ*

G. Puertos *GPIO*.

H. *RAM*.

I. Interfaz serie de cámara (Camera Serial Interface, *CSI*) para cámara.

J. Puerto audio-visual 3.5 mm.

K. Puertos PoE.

L. *USB* 2.0 (480 Mb/s).

M. *USB* 3.0 (5 Gb/s).

N. Puerto Ethernet.

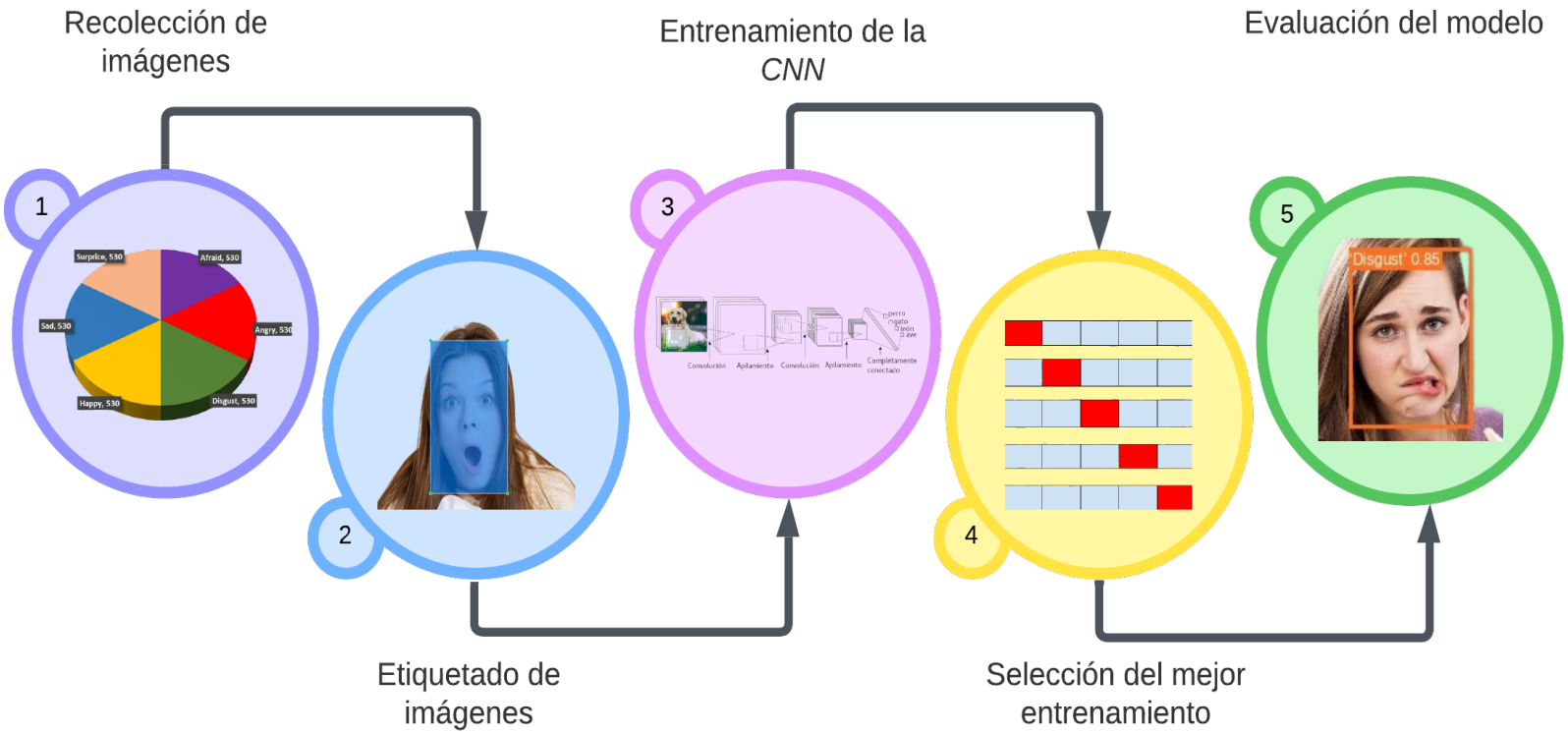
Capítulo 2

Metodología

2.1. Entrenamiento de la *CNN YOLO v5x*

A continuación, se muestra la representación gráfica del desarrollo del proyecto, el cual se encuentra dividido en 3 etapas, entrenamiento, interfaz electrónica e inferencia:

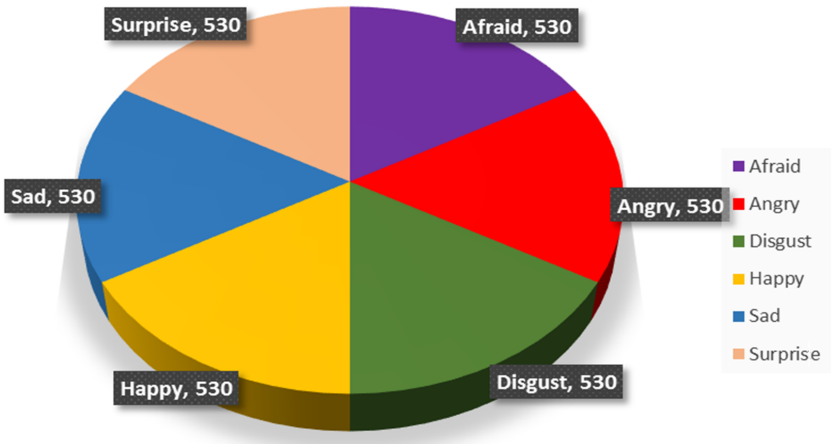
Figura 2.1:
Metodología para el entrenamiento de la CNN.



2.1.1. Recolección de imágenes

La base de datos está formada por 3,180 imágenes provenientes del internet que reflejan las emociones de; *Afraid*, *Angry*, *Happy*, *Disgust*, *Sad* y *Surprise* (ver Figura 2.2). Se encuentra dividida en un 80 % para el entrenamiento y el 20 % para validar el entrenamiento.

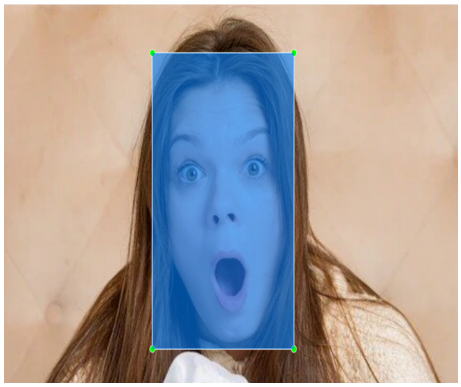
Figura 2.2:
Base de datos para el entrenamiento y validación de la CNN.



2.1.2. Etiquetado de imágenes

Existen varios programas y herramientas de etiquetado, como por ejemplo: *LabelImg*, *Labelbox*, *Vgg Image Annotator*, *RectLabel*, que permiten realizar el etiquetado de imágenes para el entrenamiento de redes neuronales, facilitando el proceso de anotación de datos, lo que ahorra tiempo y mejora la precisión del modelo. Para este trabajo se utilizó la herramienta *LabelImg*, para etiquetar la base de datos como se muestra en la Figura 2.3.

Figura 2.3:
Ejemplo del etiquetado de imágenes en LabelImg.

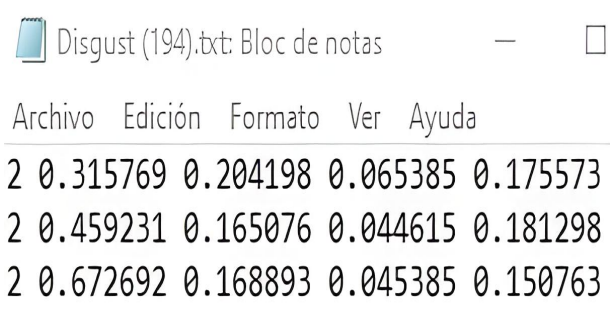


El etiquetado de las imágenes consiste en ir encerrando en un recuadro el área de interés donde se encuentre la clase y colocar el nombre de la emoción que se aprecia. Automáticamente son generados dos archivos con extensión *.txt. Uno contiene características como:

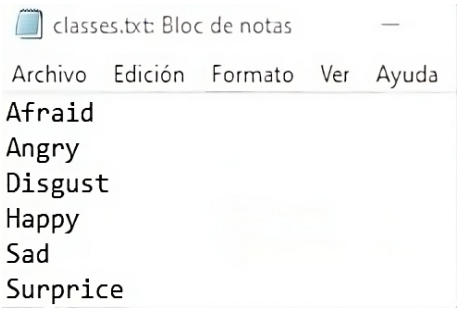
el número de clase, coordenadas de ubicación, ancho y alto del recuadro (ver Figura 2.4a) del etiquetado por cada imagen, y el otro que corresponde a un listado de las clases que se etiquetaron (ver Figura 2.4b).

Figura 2.4:
*Archivos *.txt generaros por LabelImg.*

(a) Características de etiquetado.



(b) Nombre de las clases.



Las imágenes y los documentos *.txt son guardados en una carpeta como se describe a continuación, siento **train_data** la carpeta principal y dentro de ella se generan dos carpetas; *images* y *labels*, dentro de ellas se generan dos carpetas más con nombre *train* y *val*. Posteriormente la carpeta principal es comprimida en un archivo *.zip y subida a *Google drive* para realizar el entrenamiento:

- train_data
  - images
    - train (Contiene 2,544 imágenes)
    - val (Contiene 636 imágenes)
  - labels
    - train (Contiene 2,544 documentos *.txt)
    - val (Contiene 636 documentos *.txt)

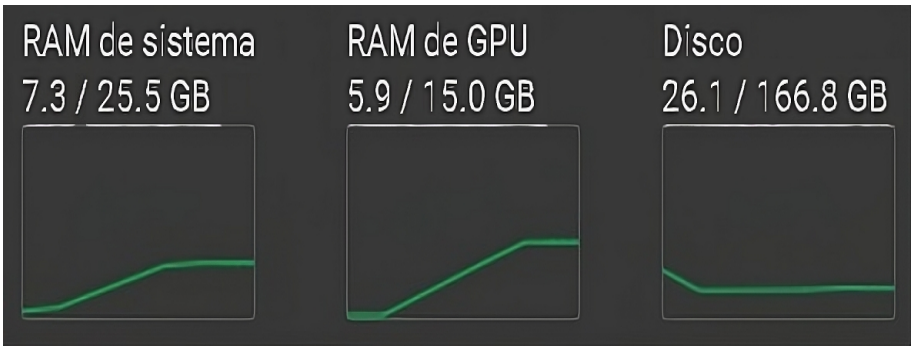
2.1.3. Entrenamiento de la *CNN*

Se implemento en *Google Colab*, el cual permite tener acceso a recursos informáticos, como se aprecia en la Figura 2.5. Debido a que no se cuenta con un equipo informático físico capaz de procesar el entrenamiento de la red. Se siguen los siguientes pasos para desarrollar el entrenamiento:

- Se verifica que se tenga conexión con el entrono de *Google Colab*, ejecutando la siguiente linea: "**!nvidia-smi**".
- Se clona el repositorio de la red *YOLO v5* ejecutando las siguiente linea de código: https://github.com/ultralytics/yolov5.

- Se carga el documento en formato ∗.zip el cual contiene la base de datos de entrenamiento y validación, como se describe en la pág. 49.
- Se configura el documento, customdata.yaml, el cual se especifican el número de clases con sus respectivos nombres, para que posteriormente sea cargado en la carpeta principal del repositorio que se clono anteriormente.
- Finalmente se ejecuta la siguiente linea de código para realizar el entrenamiento: "**!python train.py –img 640 –batch 4 –epochs 100 –data customdata.yaml –weights yolov5x.pt –cache**", en donde se especifica el tamaño de la imagen de 640 por 640 pixeles, el número de *epchs* el cual fue de 100, la ubicación del documento customdata.yaml y finalmente se especifica la versión del modelo la cual fue yolov5x.pt.

Figura 2.5:
Recursos proporcionados por Google Colab.



2.1.4. Generación de diferentes *dataset* para el entrenamiento

Para este trabajo se realizaron diez *dataset* diferentes que fueron distribuidos como se muestra en la Tabla 2.1 80 % (zona de color verde) para el entrenamiento y el 20 % (zona de color rojo) para validación, por tal motivo se hicieron diez entrenamientos (los resultados se muestran en la Tabla 3.1 de la sección 3.1) y se seleccionó el de mejor rendimiento, tomando en cuenta las siguientes métricas; *Precision*, *Recall* y *F1-score* (Azis et al., 2020).

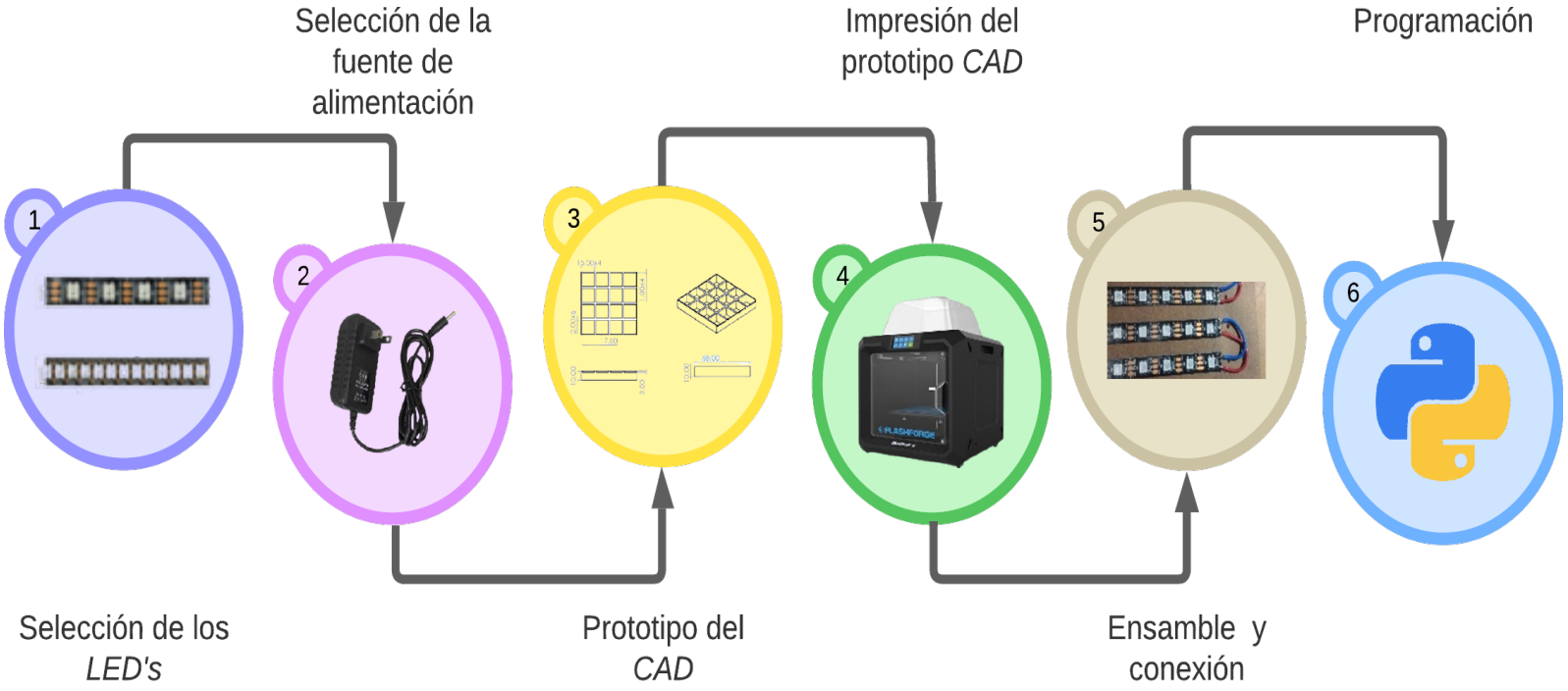
Tabla 2.1:
Diferentes particiones del 80 % - 20 % para los diez dataset (Azis et al., 2020).

0	Val
1	Val
2	Val
3	Val
4	Val
5	Val
6	Val
7	Val
8	Val
9	Val

2.2. Interfaz electrónica

La segunda etapa es para la elaboración de un prototipo de una matriz de 16 × 16 *LED's RGB*.

Figura 2.6:
Metodología para la elaboración de la interfaz electrónica.



2.2.1. Selección de los *LED's RGB*

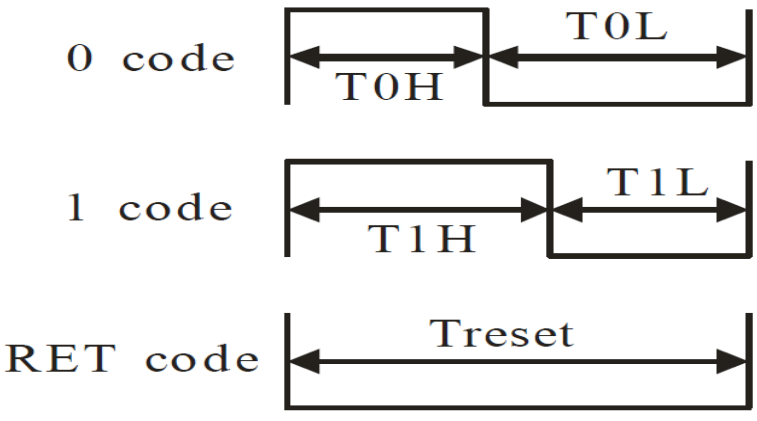
La cinta *LED WS2812B*, que es una fuente de luz *LED* en la que el circuito de control y el chip *RGB* están integrados en un paquete de 50×50 mm, tomando en cuenta las siguientes características para su selección:

- **Comunicación en cadena**: la característica clave de la cinta *LED WS2812B* es que cada *LED* es controlable individualmente debido a que cada tiene su propio chip controlador y se comunica con otros *LED* de la cinta a través de una conexión en serie sin la necesidad de cables o conexiones externas.
- **Protocolo de comunicación**: utiliza un protocolo de comunicación unidireccional y síncrono. Esto significa que el controlador, en este caso *RPi*, envía los datos en serie a través de un solo cable al primer *LED* de la cadena. Cada *LED* en la cinta recibe estos datos, retiene la información que le corresponde y pasa los datos restantes al siguiente permitiendo accionamiento de amplificación de remodelación de señal.
- **Control de colores**: para controlar el color de cada *LED*, se utiliza un control *PWM* para representar diferentes niveles de intensidad en cada color (rojo, verde y azul). Combinando diferentes niveles de intensidad en estos tres colores básicos, se pueden crear aproximadamente 16 millones de colores distintos. En la Tabla 2.2 se observa la duración del estado alto y bajo (ver Figura 2.7) que deben tener los pulsos para la generación de un 0, 1 o un *reset* para definir la intensidad de color.

Tabla 2.2:
Tiempo de transferencia de datos (TH + TL = 1.25µs ±600 ns.) (Worldsemi, 2020).

T0H	0 *code, high voltage time*	0.4 µs	± *150 ns*
T1H	1 *code, high voltage time*	0.8 µs	± *150 ns*
T0L	0 *code, low voltage time*	0.85 µs	± *150 ns*
T1L	1 *code, low voltage time*	0.45 µs	± *150 ns*
Rest	*low voltage time*	50 µs	

Figura 2.7:
Diagrama de secuencia (Worldsemi, 2020).



- **Alimentación**: la cinta *LED WS2812B* necesita una fuente de alimentación adecuada, ya que los *LED's* pueden consumir una cantidad significativa de energía. Dependiendo de la longitud de la cinta y la cantidad de *LED's* que haya en ella, es posible que necesites una fuente de alimentación externa capaz de proporcionar la potencia necesaria.
- **Controladores**: para programar y controlar la cinta *LED WS2812B*, se pueden utilizar microcontroladores como *Arduino*, *RPi* y otros dispositivos con salidas digitales. También hay bibliotecas de *software* específicas disponibles que facilitan el control de los *LED's* y la creación de efectos y animaciones complejas.

2.2.2. Selección de la fuente de alimentación

La selección de la fuente, es de acuerdo a la limitación de brillo que se establezca a través de la programación en cada *LED* o *píxel*. De acuerdo a la hoja de especificaciones de fabricante, (Worldsemi, 2020), se establece que el consumo por cada *LED* cuando se encuentra trabajando en su brillo máximo es de 60 mA. Considerando que se tiene un total 256 *LED's* se tiene un amperaje total 15.36 *Ampers* (A), por lo tanto para satisfacer la demanda de corriente se debe seleccionar una fuente que soporte la carga mayor a 15 A, si se desea trabajar con el brillo en su máxima capacidad.

Para este trabajo se utiliza una fuente con un máximo de 3 A para la alimentación de la matriz de *LED's*. Para esto, dentro del código de programación se coloca un factor de ***brightness=0.15***, el cual garantiza que el amperaje máximo sea de 2.5 A, y no sobrepase los 3 A de la fuente, evitando problemas de sobrecarga. Este factor se calculó utilizando la ecuación 2.1.

$$\left.\begin{array}{ll} A & \rightarrow B \\ Y & \rightarrow X \end{array}\right\} \rightarrow X = \frac{(B)(Y)}{A} \tag{2.1}$$

Donde: A = amperaje máximo de la matriz, B = El amperaje deseado para trabajar, Y = el valor máximo para configuración de brillo. Por lo tanto, sustituyendo Y, A y B en la ecuación 2.1, se obtiene el valor del factor que se debe colocar dentro del código.

$$\left.\begin{array}{ll} 15{,}36 & \rightarrow 2{,}5 \\ 1 & \rightarrow X \end{array}\right\} \rightarrow X = \frac{(2{,}5)(1)}{15{,}36}$$

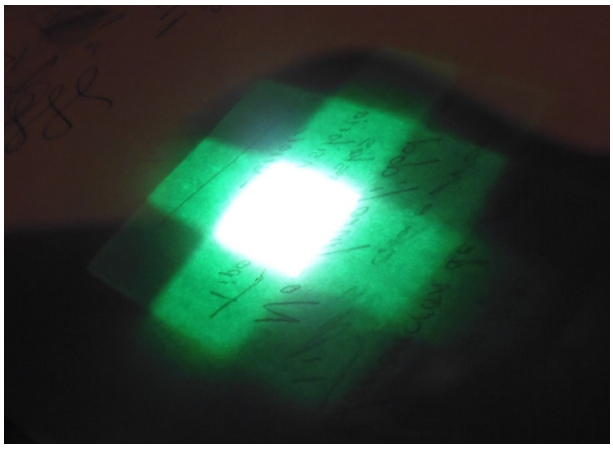
$$X \approx 0{,}15$$

2.2.3. Prototipo *CAD* en *SolidWorks*

Rejilla

Se utiliza una rejilla para cada *pixel*, con la finalidad de que el brillo de cada uno de ellos no se mezcle con los demás como se puede apreciar en la Figura 2.8 y así se podrá observar de una manera más clara el color que se genera.

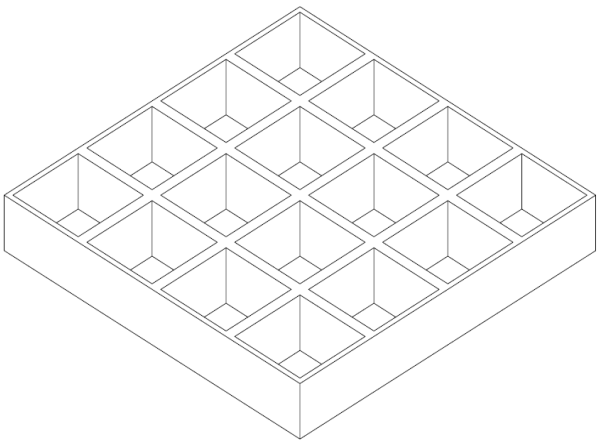
Figura 2.8:
Píxel sin la rejilla.



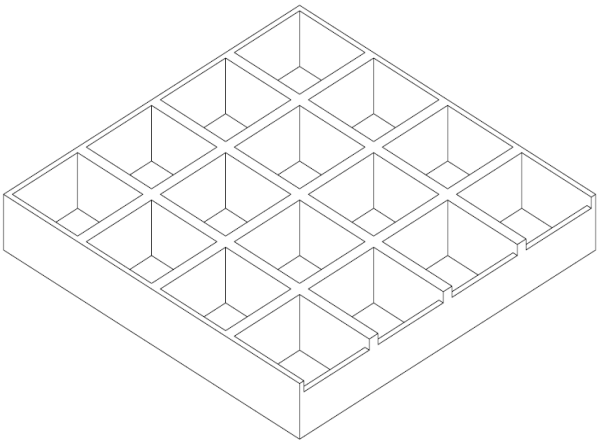
Se realizan dos diseños de matriz 4×4, 8 para posicionarlas en el interior (ver Figura 2.9b) y 8 para los laterales (ver Figura 2.9).

Figura 2.9:
Tipos de rejillas.

(a) *Rejilla para el interior de la matriz.*



(b) *Rejillas para los laterales de la matriz .*

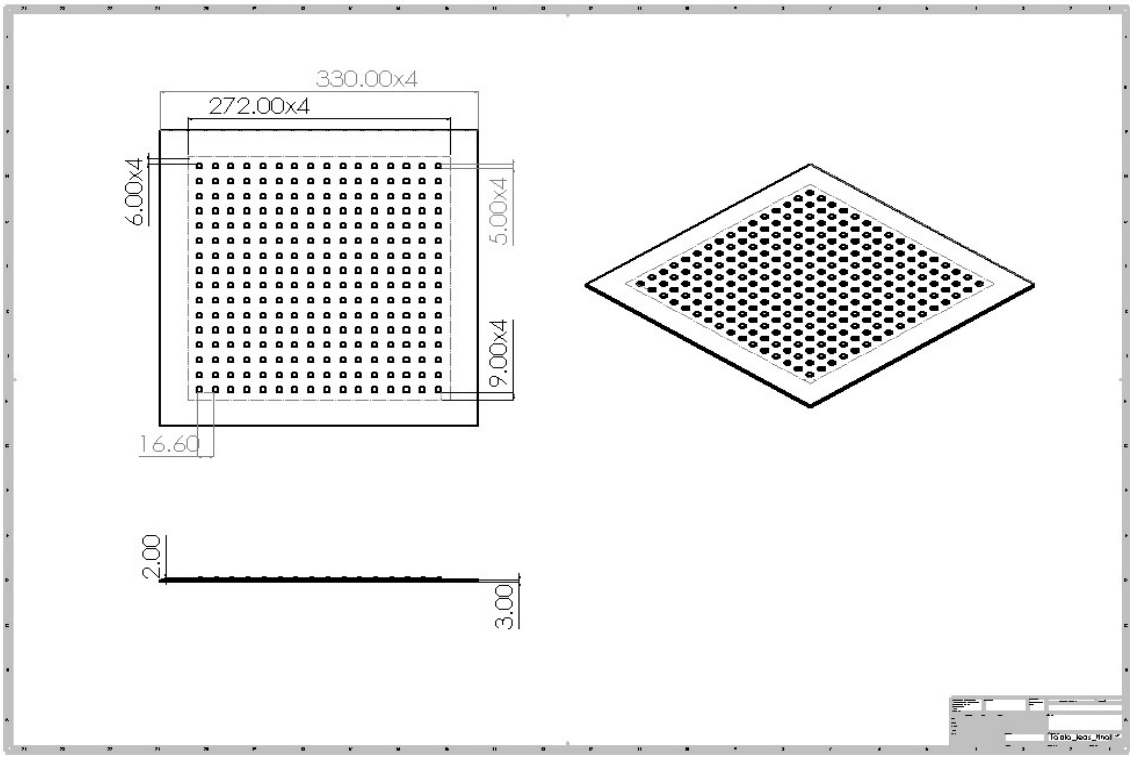


Al hacer uso de las rejillas se resuelve el problema del brillo que se menciona anteriormente, y ahora el color se concentra en el *pixel* correspondiente, como se puede observar en la Figura 2.10.

Figura 2.10:
Píxel utilizando la rejilla (evita la dispersión del brillo).

Posteriormente se realiza la distribución de cada cinta *LED* en el espacio de trabajo, dimensionando de acuerdo al espacio que hay entre cada *LED* de la cinta *WS2812B*, como se observa en la Figura 2.11.

Figura 2.11:
Vista de la matriz de LED's en SolidWorks.

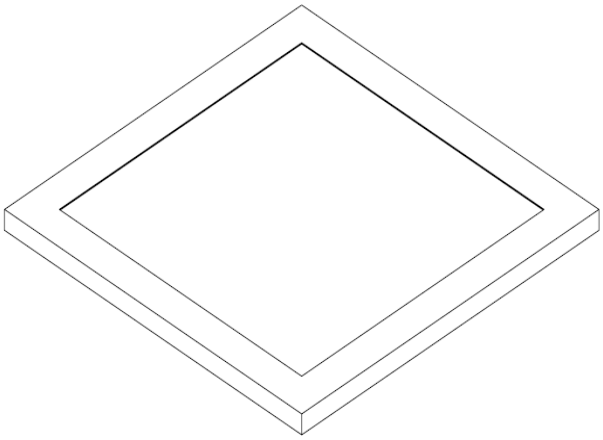


Marco

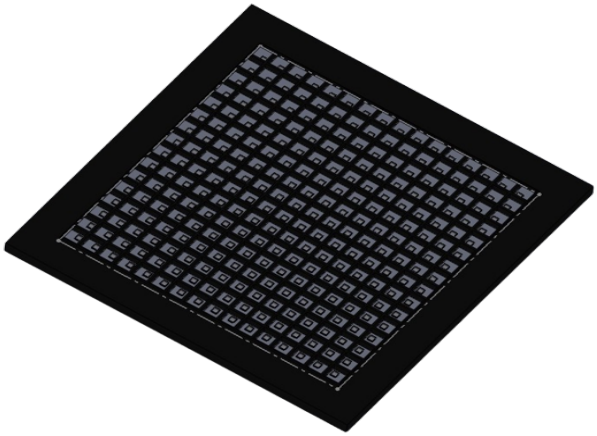
Finalmente se realiza un marco (ver Figura 2.12a) para dar un mejor acabado y poder realizar el ensamble de los *LED's*, las rejillas y la cubierta (ver Figura 2.12b).

Figura 2.12:
Ensamble de la interfaz electrónica.

(a) *Vista superior del marco.*



(b) *Ensamble de LED's, rejillas y marco.*

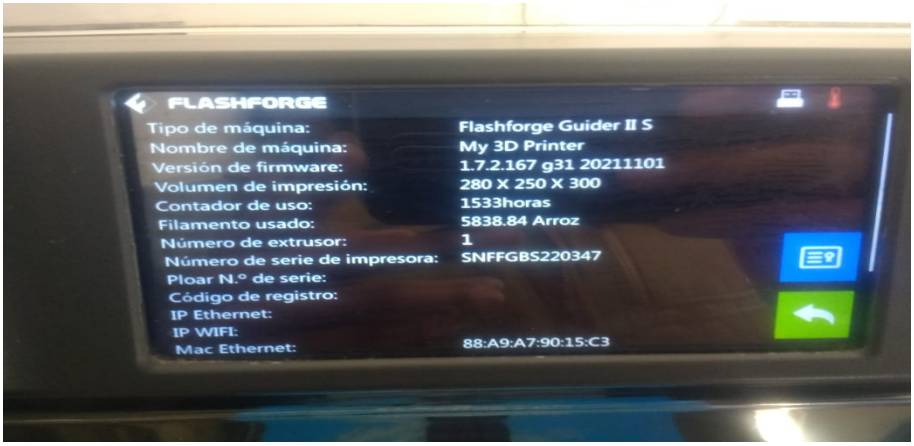


2.2.4. Impresión 3D del *CAD*

Rejilla

Se aplicó la manufactura aditiva, en específico la impresión 3D para realizar las rejillas y el marco. Utilizando la impresora 3D *Guider II S* con las características que se muestran en la Figura 2.13, utilizando como filamento, ácido poliláctico para impresión (*Polylactic Acid*, *PLA*).

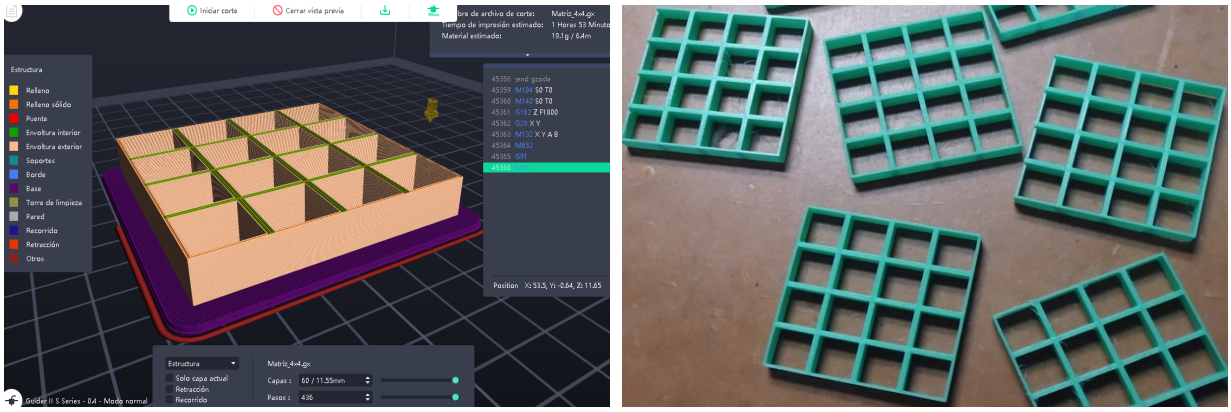
Figura 2.13:
Características de la impresora Guider II S.



Las características por las cuales se utilizó el filamento *PLA* son: su flexibilidad, resistencia y la baja inflamabilidad. La temperatura de fundición es de 250°c con una cama calentada entre 30°-40°C. Se utilizó el *software FlashPrint* en donde se establecieron los parámetros de impresión, por ejemplo; relleno, velocidad, acabado, soportes, etc. (ver Figura 2.14a), estos parámetros ayudan a mejorar los resultados y a aumentar o disminuir el tiempo de impresión, en este caso el tiempo fue de aproximadamente dos horas por cada matriz, dando un total treinta y seis horas de impresión por las dieciséis rejillas que se utilizaron. En la Figura 2.14b se muestran algunas de las rejillas impresas.

Figura 2.14:
Proceso de impresión de las rejillas.

(a) *Configuración del software FlashPrint.* (b) *Rejillas impresas.*

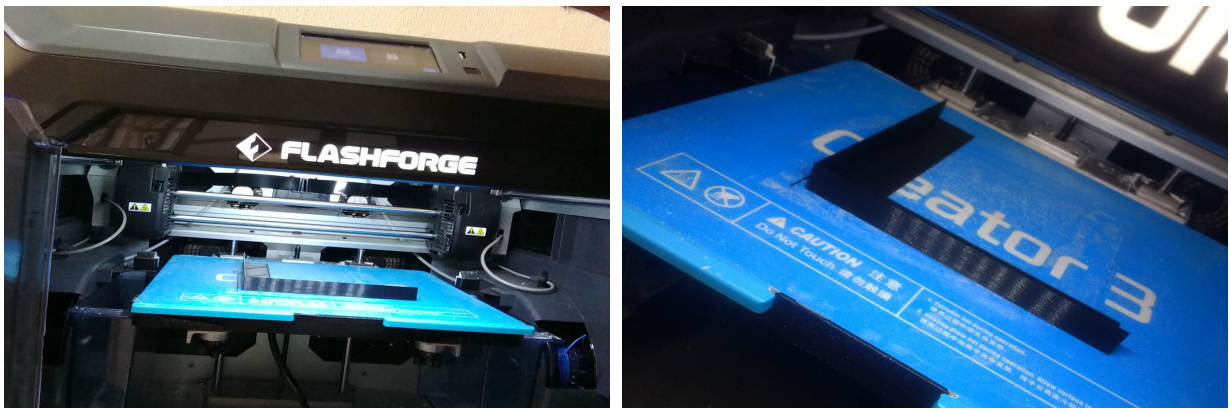


Marco

El marco se imprimió en 4 piezas, debido a que sobrepasaba las dimensiones de área de la impresora *Flashforge Creator 3 PRO*, utilizando *PLA* como filamento de impresión. En la Figura 2.15 se muestran la manera en como se imprimió el marco.

Figura 2.15:
Impresión del marco para la interfaz electrónica.

(a) (b)



2.2.5. Ensamble y conexiones

Conexión de la cinta *LED*

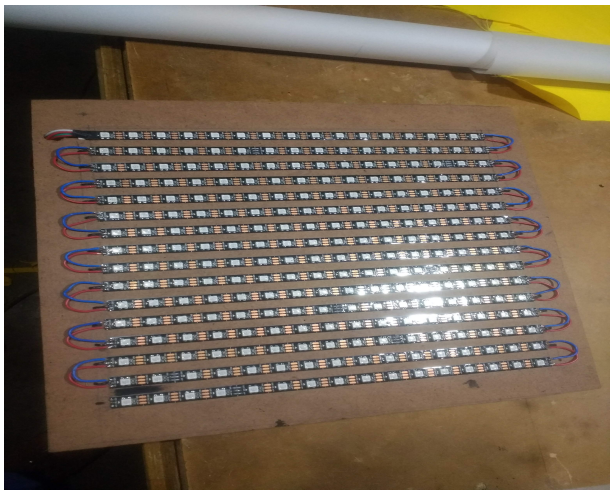
Se realizan las conexiones eléctricas de acuerdo a la hoja de especificaciones del fabricante (ver Tabla 2.4).

Tabla 2.3:
Vista de las conexiones realizadas de cada cinta LED (Worldsemi, 2020).

NO.	*Symbol*	*Function description*
1	*VDD*	*Power suppply LED*
2	*DOUT*	*Control data signal output*
3	*VSS*	*Ground*
4	*DIN*	*Control data signal input*

Tabla 2.4: *PINOUT* del *LED WS2812B*.

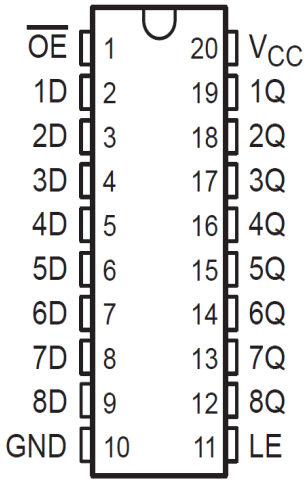
Figura 2.16:
Vista de las conexiones realizadas de cada cinta LED.



Conexión de la interfaz electrónica a la *RPi-4G*

Derivado a que los pines de la *RPi-4G* están limitados a un voltaje de salida de 3.3 *vcd* y la alimentación de voltaje requerida por interfaz electrónica es de 5 *vcd* se utilizó un *latch* tipo D con número SN54HC573A, para poder operar con los 5 *vcd* a través de una fuente externa. La selección de este *latch* se determino tomando en cuenta las características que se muestran en la Figura 2.17 debido a que la frecuencia demandada por la matriz de *LED's* es de 800 *KHz* y este *latch* trabaja a una frecuencia de conmutación de 28 *MHz*, mayor a la que demanda la matriz de *LED's*.

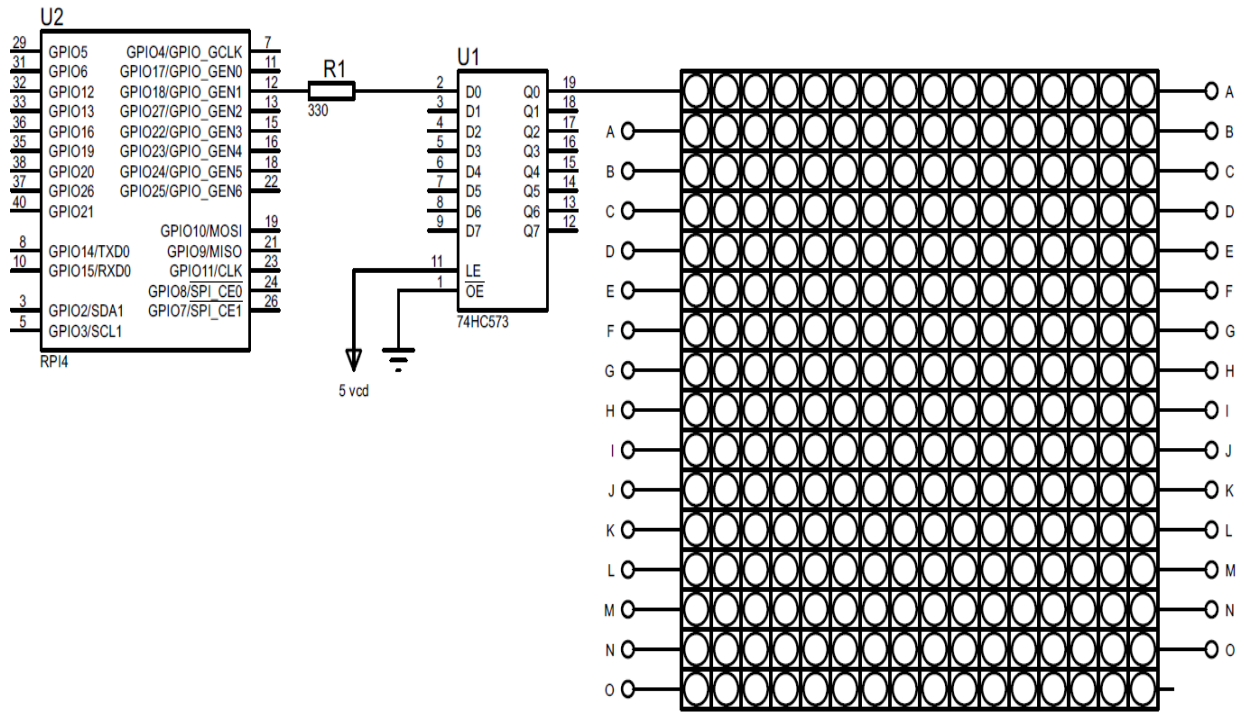
Figura 2.17:
PINOUT y características del latch SN54HC573A (Instruments, 2023).



- Número de canales: 8.
- Voltaje (min) (V): 2.
- Voltaje $(máx)$ (V): 6.
- IOL $(máx)$ (mA): 7.8.
- IOH $(máx)$ (mA): -7.8.
- Frecuencia $(máx)$ (MHz): 28.

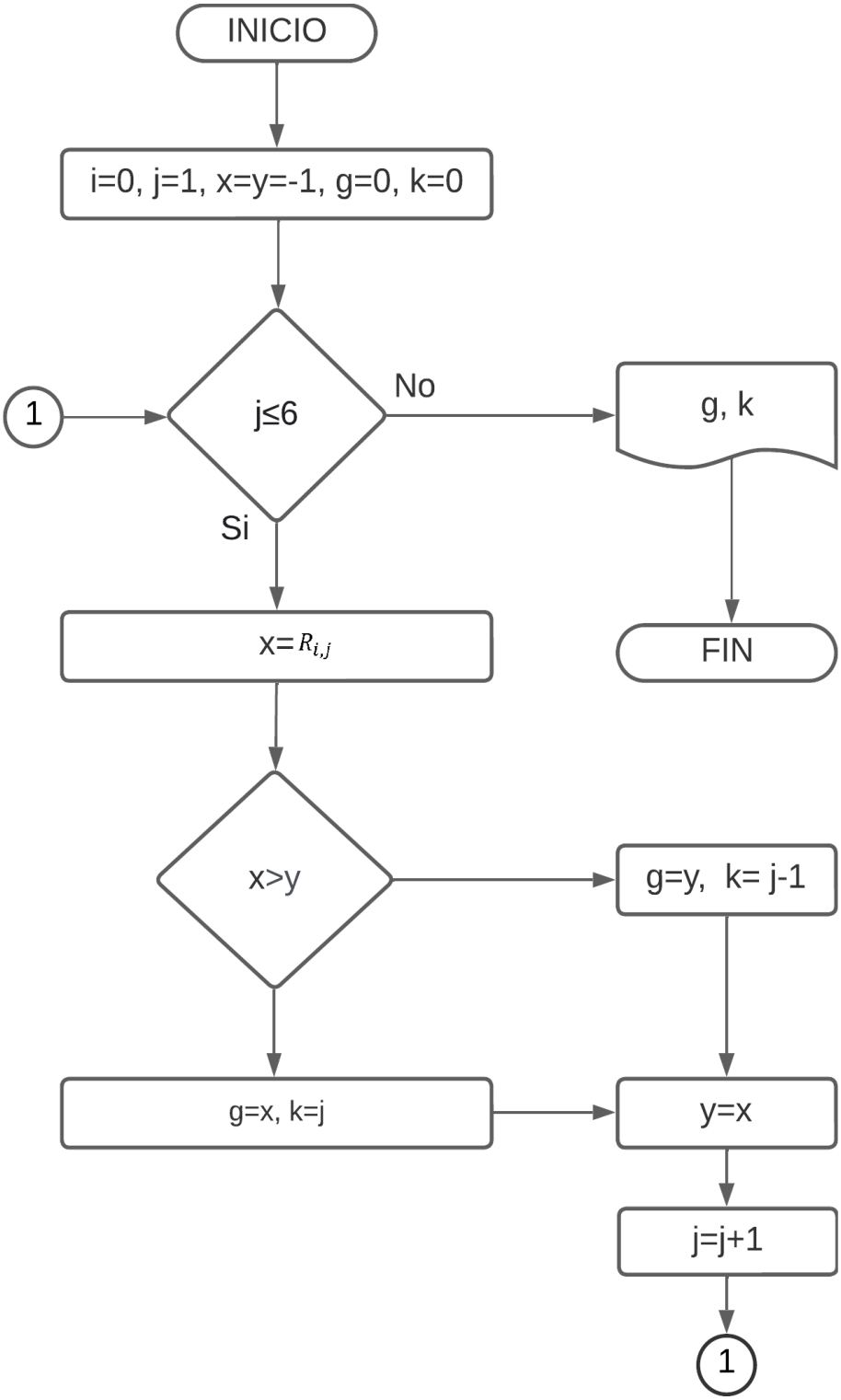
Finalmente en la Figura 2.18 se muestran las conexiones realizadas de la *RPi* al *latch* y del *latch* a la matriz de *LED´s*. La fuente de alimentación se conecta en el *latch* que separa la etapa de control y de potencia.

Figura 2.18:
Conexión de los dispositivos.



2.2.6. Programación

Se muestra el diagrama de flujo de programación que permite seleccionar la clase con el mayor nivel de confianza para mostrarse como salida de predicción del modelo.



2.3. Inferencia

En la Figura 2.21 se muestran las diferentes etapas realizadas por una *CNN*, las cuales son descritas a continuación:

- La captura de la imagen a través de la *webcam USB* con las especificaciones que se muestran en la Figura 2.19, la cual se encuentra conectada a uno de los puertos *USB* 3.0 que posee la *RPi*.

Figura 2.19:
Webcam USB Full HD.

- *webcam USB Full HD.*
- Modelo: COM122.
- Marca: Steren.
- Alimentación: 5 vcd.
- *Frame rate*: 15⁄30 *FPS*.
- Resolución: 10810p (1920×1080).

- La imagen es dividida en píxeles, de acuerdo a su resolución, y son representados en su formato *RGB*.
- Las imágenes tienen una resolución de 640×640×3, dando como resultado un total de 1,228,800 neuronas de entrada a la *CNN*.
- Posteriormente pasan a la capa de convolución, en la cual se realiza una multiplicación con un *kernel*, con la finalidad de extraer características significativas de la imagen de entrada.
- La capa de *pooling* se encarga de agrupar los resultados de las características de la capa de convolución para obtener formas más definidas y compactas.
- En la capa de aplanamiento se pasan los datos de forma matricial a forma vectorial, y estos datos son las neuronas de entrada a una *ANN* denominada como capas totalmente conectadas.
- Finalmente a la salida de la *ANN* se encuentran las capas que determinaran de acuerdo al porcentaje predicción que emoción se esta presentando en la entrada a través de la *webcam*.

Figura 2.20:
Conexión del Hardware para la realización de la inferencia.

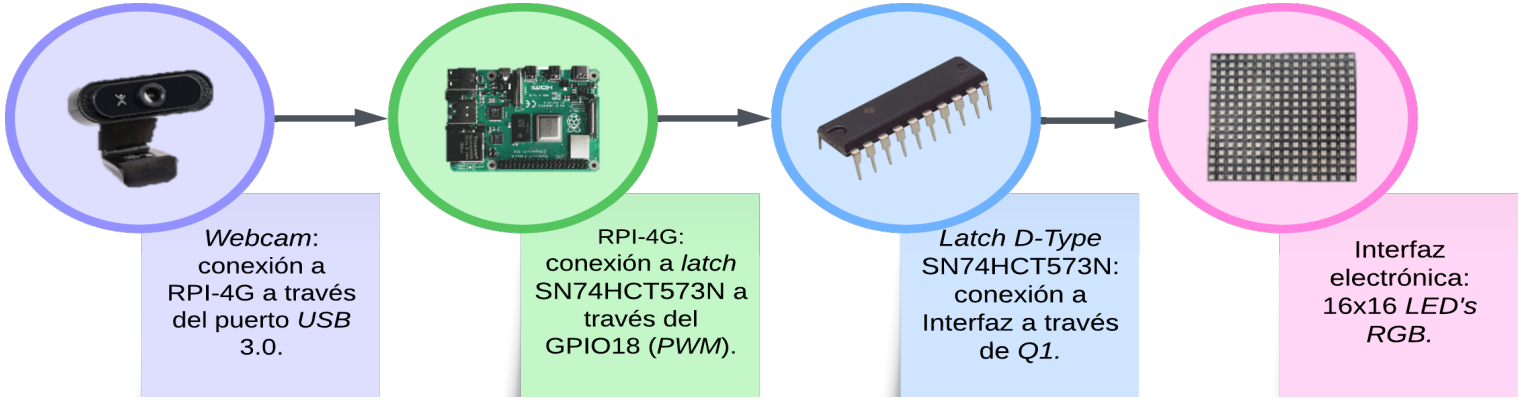
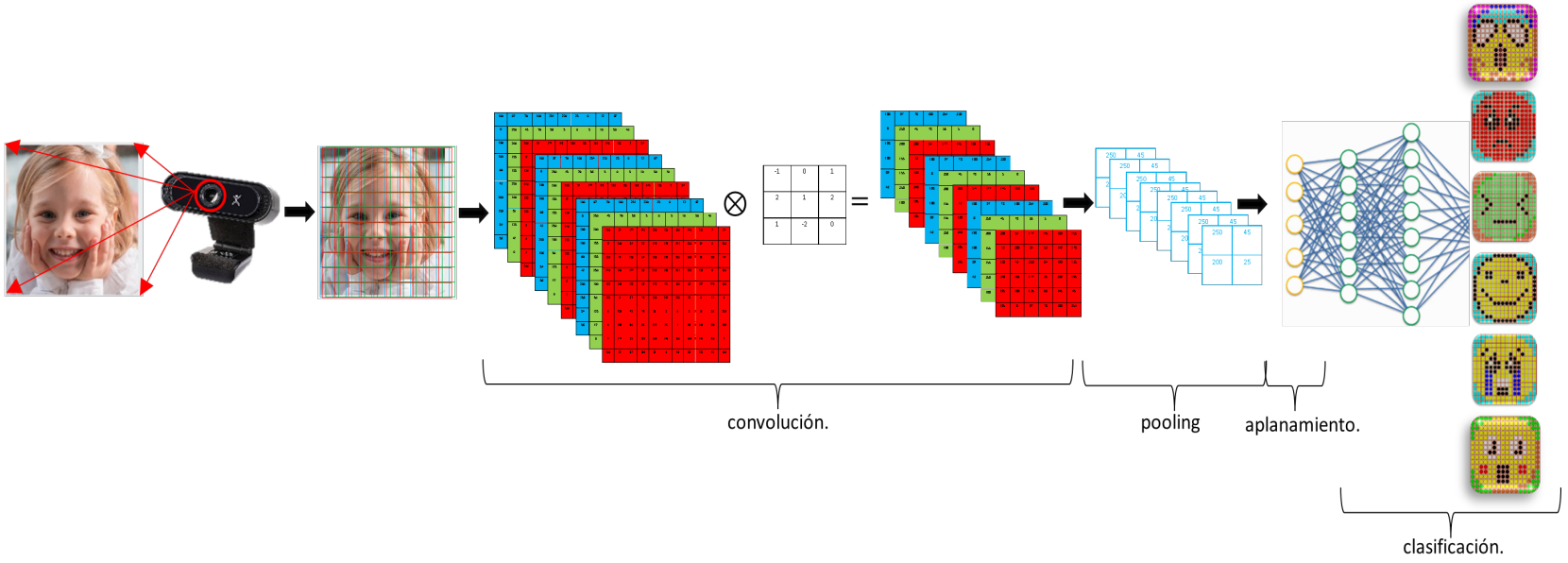


Figura 2.21:
Proceso de inferencia a través de la webcam.



Capítulo 3

Resultados y discusión

3.1. Resultados del entrenamiento con la versión *x*

Los resultados de los diez entrenamientos se muestran en la Tabla 3.1, a través de las métricas de *Precision*, *Recall*, *mAp50* y *mAp50-95* para evaluar su desempeño. Se observa que el entrenamiento (*training*) tres (en color amarillo) tiene una *Precision* de 0.80 y un 0.79 de *Recall*, indicando que es el mejor *training* de los diez que fueron realizados, por tal motivo es utilizado para el desarrollo de este proyecto.

Tabla 3.1:
Métricas de los diez entrenamientos realizados con la versión x.

Training	*Precision*	*Recall*	*mAP50*	*mAP50-95*
0	0.73	0.74	0.80	0.55
1	0.67	0.72	0.74	0.48
2	0.63	0.70	0.71	0.47
3	0.80	0.79	0.85	0.58
4	0.75	0.76	0.81	0.55
5	0.71	0.78	0.80	0.54
6	0.77	0.74	0.81	0.55
7	0.73	0.78	0.82	0.56
8	0.63	0.75	0.76	0.50
9	0.60	0.69	0.67	0.45

- *mAp50*: umbral en el 50 %
- *mAp50-95*: umbral entre el 50 % y el 95 %

3.1.1. Validación

Matriz de Confusión

En la Figura 3.1 se muestra la matriz de confusión a través de un mapa de temperatura (*heatmap*) en donde la intensidad del color cambia de acuerdo al valor de cada celda, siendo 1 el valor máximo y 0 el valor mínimo. En el eje de las abscisas se encuentran los valores reales (*true*) representando a los *FP* y en el eje de las ordenadas a los valores predichos (*predicted*) que representan a las *FN*. A través de esta matriz se observa el porcentaje que tiene cada clase al confundirse con alguna otra, así como también el porcentaje de *TP* que tiene cada una de ellas (a través de la diagonal principal en donde convergen las predicciones con los valores reales), la clase *Happy* tiene 0.91 en la clasificación de *TP*, siendo la clase con el mejor desempeño mientras que la clase *Disgust* tiene 0.69 siendo el porcentaje de más bajo en comparación con las demás clases.

En la Tabla 3.2 se muestran las métricas de *Precision*, *Recall* y *mAP50* para la evaluación de seiscientas treinta y seis imágenes que representan el 20 % del *dataset* implementado. Cada clase tiene sus propios resultados y hay una que representa el promedio de las seis clases (*class all*). *Happy* tiene una *Precision* de 0.86 y un *Recall* de 0.92, siendo la clase con el mejor desempeño en comparación con el resto de las clases.

Figura 3.1:
Matriz de confusión del training tres con la versión x.

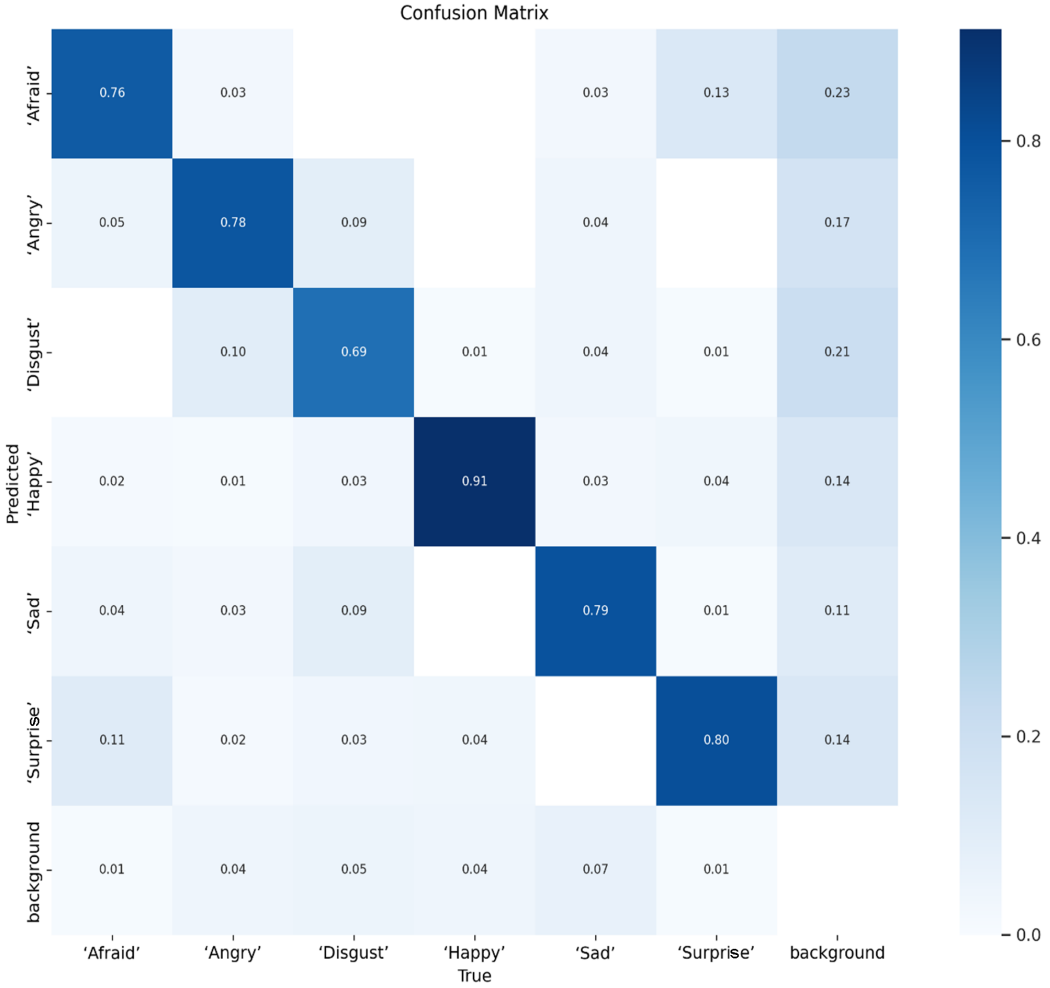


Tabla 3.2:
Métricas para cada clase del training tres con la versión x.

Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
All	636	714	0.80	0.79	0.85	0.58
Afraid	636	114	0.77	0.77	0.82	0.55
Angry	636	118	0.79	0.77	0.86	0.55
Disgust	636	116	0.77	0.70	0.80	0.55
Happy	636	138	0.86	0.92	0.95	0.65
Sad	636	116	0.81	0.77	0.84	0.60
Surprise	636	112	0.78	0.83	0.86	0.58

- *Class*: nombre de la clase.
- *Images*: 20 % del *dataset* para validación.
- *Instances*: número de veces que se detecto la clase.

Curva *Precision-Recall*

El cambio de la *Precision* y el *Recall* a medida que se ajusta el umbral de decisión para las predicciones se muestra en la Figura 3.2. En donde el punto ideal es tener una alta *Precision* y un alto *Recall*. Esto significaría que el modelo hace predicciones correctas (alta *Precision*) y también es capaz de identificar la mayoría de los *FP* (alto *Recall*). En la Figura 3.3 se muestra los gráficos de *box_loss*, *obj_loss*, *Precision*, *Recall*, *mAP*, durante las 100 *epoch* para el conjunto de entrenamiento y validación (fila superior e inferior respectivamente).

Figura 3.2:
Curva Precision-Recall del training tres con la versión x.

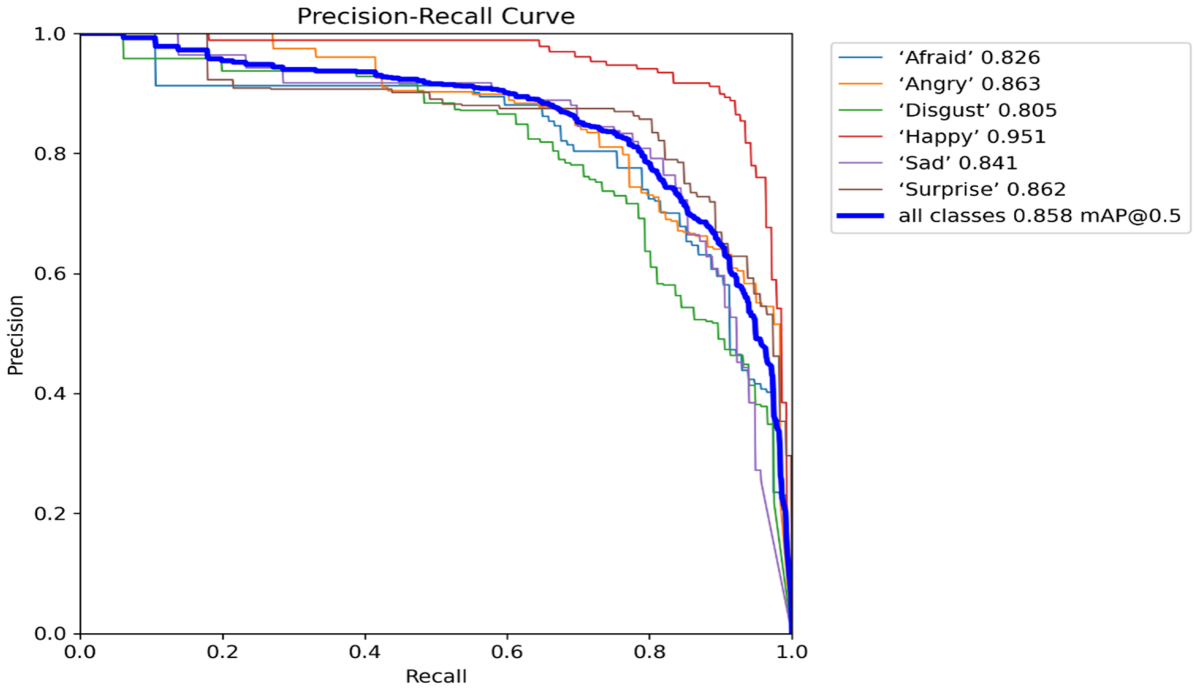
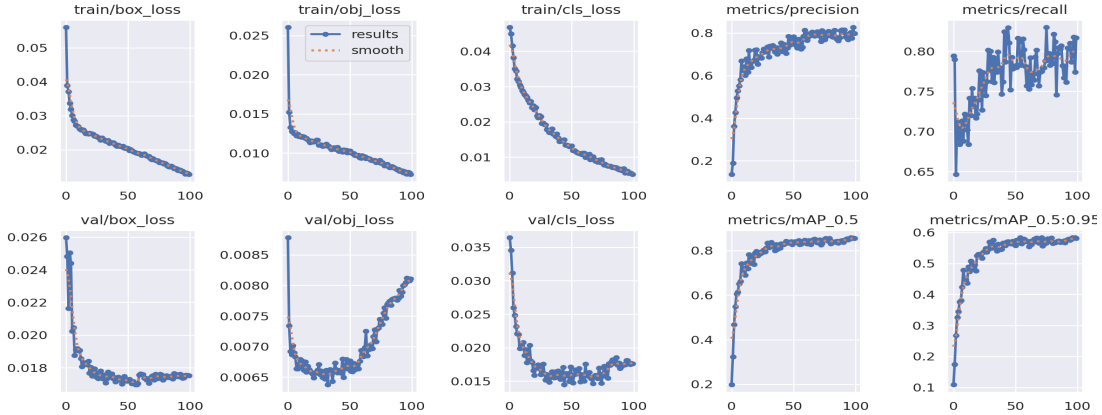


Figura 3.3:
Resultados del entrenamiento y validación del training tres con la versión x.



3.2. Resultados de la efectividad

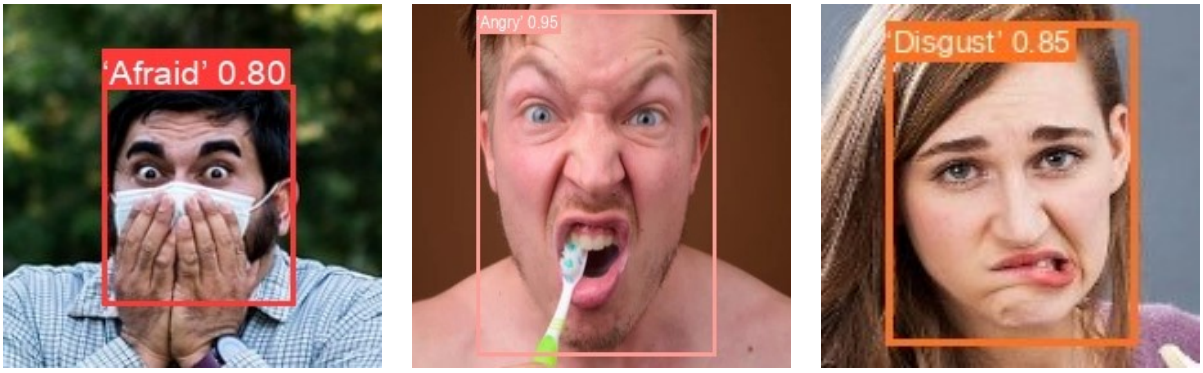
En esta sección se muestran los resultados de la efectividad que tiene el modelo entrenado, al aplicar la inferencia en imágenes descargadas del internet y en fotografías.

3.2.1. Pruebas en las imágenes descargadas del internet

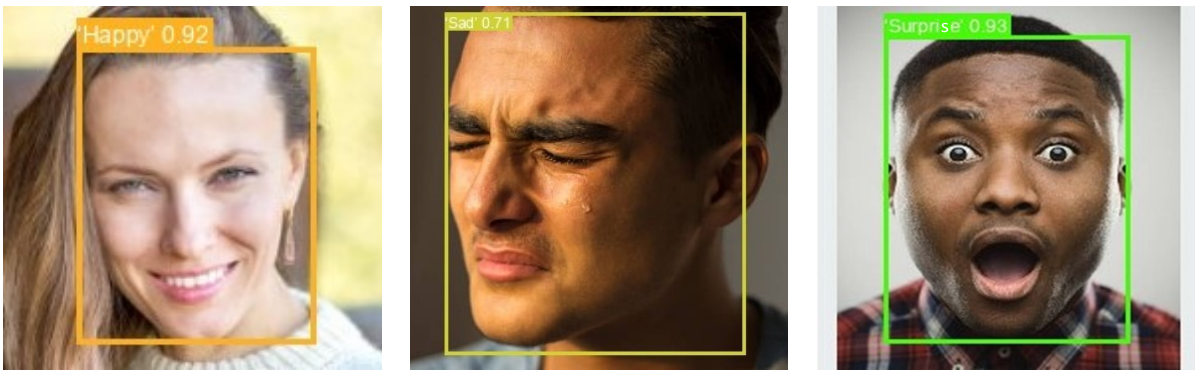
Se consideran quinientas cuatro imágenes que no fueron etiquetadas como se hizo con la carpeta de entrenamiento (como se describe en la sección 2.1.2 de la pág. 49). Estas imágenes son procesadas por el modelo, el cual coloca un cuadro delimitador (*bounding boxes*), con el nombre de la clase y el nivel de confianza que indica la aproximación que tiene la predicción respecto al valor real (1), como se muestra en la Figura 3.4.

Figura 3.4:
Inferencia en las imágenes descargadas del internet con el training tres de la versión x.

(a) *Afraid* (b) *Angry* (c) *Disgust*



(d) *Happy* (e) *Sad* (f) *Surprise*



Los resultados de la evaluación en las imágenes descargadas del internet se muestran en la matriz de confusión de la Figura 3.5, en donde la diagonal principal representa la convergencia de los valores reales con los predichos, siendo la clase *Happy* la que tiene el mejor desempeño, debido a que se predijeron 81 imágenes correctamente de 84, mientras que las clases *Angry* y *Sad* solo detectaron correctamente 56 imágenes. En la Tabla 3.5 se muestran los *TP*, *FN*, *FP* y *TN* que se obtuvieron de la matriz de confusión de la Figura 3.5.

Figura 3.5:
Matriz de confusión para las imágenes descargadas del internet con el training tres de la versión x.

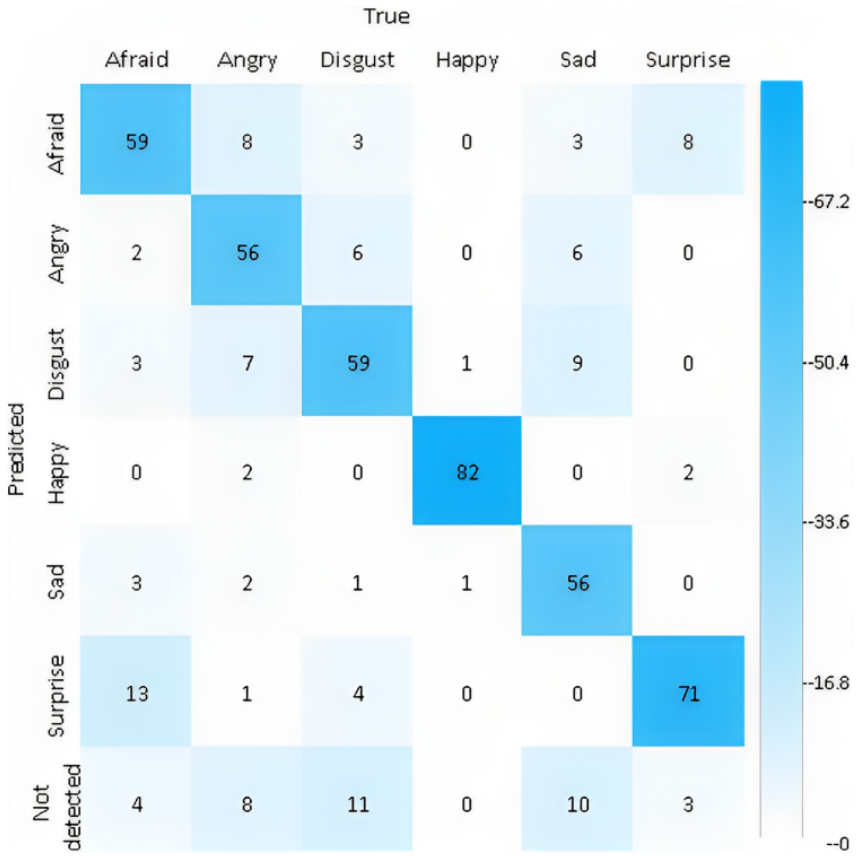


Tabla 3.3:
Resultados de la matriz de confusión de la Tabla 3.5.

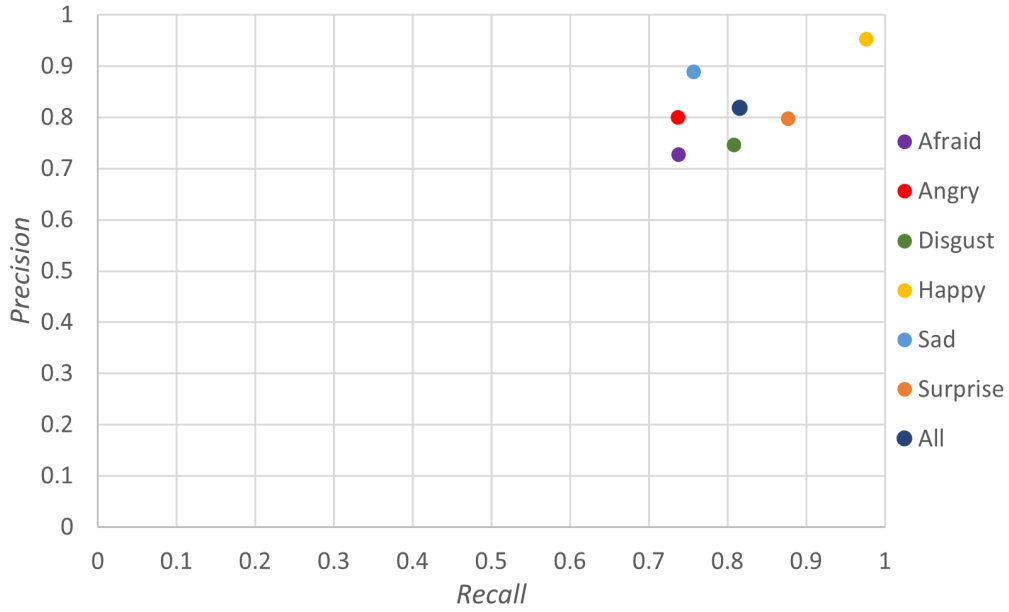
Class	*TP*	*FP*	*FN*	*TN*
Afraid	59	22	21	366
Angry	56	14	20	378
Disgut	59	20	14	375
Happy	82	4	2	380
Sad	56	7	18	387
Surprise	71	18	10	369

Se aplican las fórmulas (1.11) - (1.14) que se encuentran en la pág. 35 y 36, para obtener las métricas de; *Precision*, *Recall*, *FPR*, y el *F1-score* que se muestran en la Tabla 3.4 en donde se observa que la clase *Happy* es la que tiene un mejor desempeño con un 0.95 en *Precision* y 0.97 en *Recall*. Mientras que para la clase *Afraid* tiene 0.72 en *Recall* y una *Precision* de 0.38, por tal motivo esta clase tiene problemas en predecir casos nuevos, pero cuando lo realiza es altamente confiable. En la Figura 3.6 se observa la gráfica de la relación entre la *Precision* y el *Recall* de los datos mostrados en la Tabla 3.4.

Tabla 3.4:
Métricas de las imágenes con el training tres de la versión x.

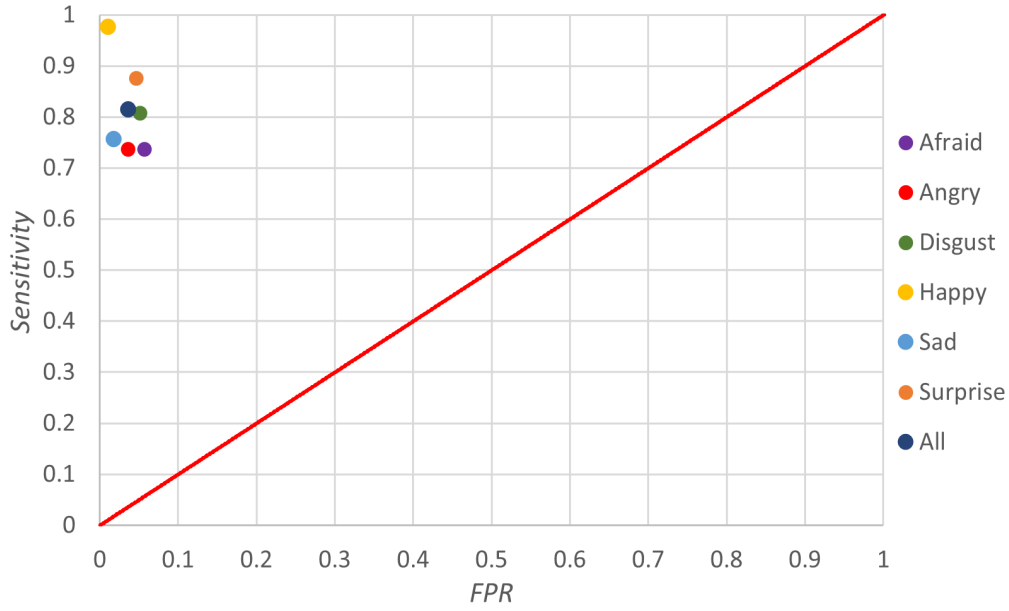
Class	*Precision*	*Recall*	*FPR*	*F1-score*
Afraid	0.72	0.73	0.05	0.73
Angry	0.80	0.73	0.03	0.76
Disgut	0.74	0.80	0.05	0.77
Happy	0.95	0.97	0.01	0.96
Sad	0.88	0.75	0.01	0.81
Surprise	0.79	0.87	0.04	0.83
All	0.81	0.81	0.03	0.81

Figura 3.6:
Gráfica Precision-Recall de las imágenes con el training tres de la versión x.



La métrica de la curva *ROC* se muestra en la Figura 3.7, en donde cualquier clase del modelo que se aproxime a la esquina superior izquierda del gráfico, se le considera que tiene un buen rendimiento debido a que indica un alto valor de *Sensitivity* mientras mantiene bajo el valor de *FPR*. En este caso se observa que la clase con mayor *Sensitivity* es *Happy* con 0.95 y un *FPR* del 0.01. Mientras que la clase *Afraid* es la que se encuentra alejada del punto ideal, con un valor de *Sensitivity* de 0.73 y un valor de *FPR* de 0.05.

Figura 3.7:
Curva ROC de las imágenes con el training tres de la versión x.

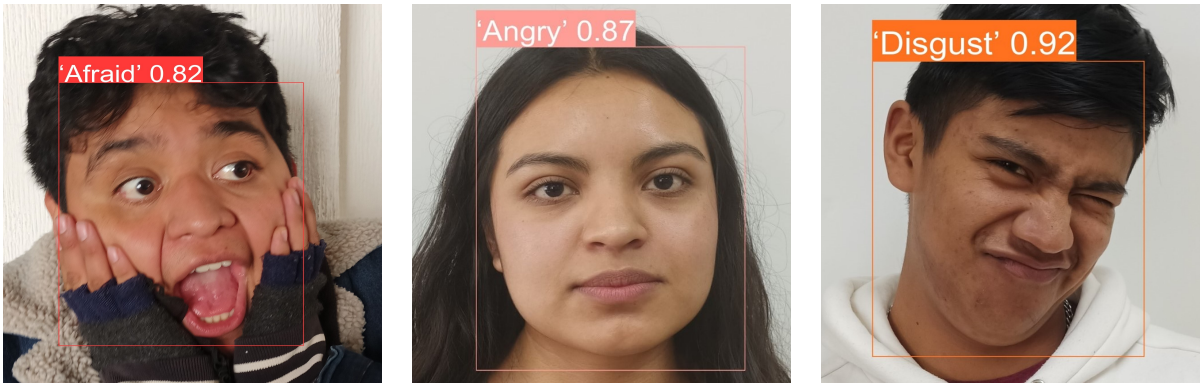


3.2.2. Pruebas en las fotografías

Estas pruebas se realizaron con quinientas cuatro fotografías que fueron tomadas a los alumnos del *TEST*, doscientas cincuenta y dos pertenecen a la división *IMT* y el resto a la división de *ISC*, los resultados de la inferencia se muestran en la Figura 3.8. Las fotografías de la división de ingeniería en sistemas computacionales fueron capturadas sin tomar en cuenta la iluminación y el desenfoque en la cámara, con la finalidad de evaluar el desempeño del modelo en este tipo de entornos.

Figura 3.8:
Inferencia en las fotografías de los alumnos del TEST con el training tres de la versión x.

(a) *Afraid* (b) *Angry* (c) *Disgust*



(d) *Happy* (e) *Sad* (f) *Surprise*

Fotografías de los alumnos de *IMT*

Al evaluar las fotografías se obtiene la matriz de confusión que se muestra en la Figura 3.9, en donde a través de la intensidad del color azul se observa el desempeño que tuvo el modelo para la predicción de cada clase, una mayor intensidad de color en cada cuadrante indica que el valor se encuentra próximo al valor máximo de 42 fotografías. La clase *Happy* tiene el mejor desempeño, debido a que se predijeron 31 imágenes correctamente de las 42, mientras que en la clase *Angry* se predijeron correctamente 8 imágenes, siendo la de menor valor en comparación con el resto de las clases. En la Tabla 3.5 se muestran los *TP*,*FN*,*FP* y *TN* que se obtuvieron de la matriz de confusión de la Figura 3.9.

Figura 3.9:
Matriz de confusión para las fotografías de IMT con la versión x.

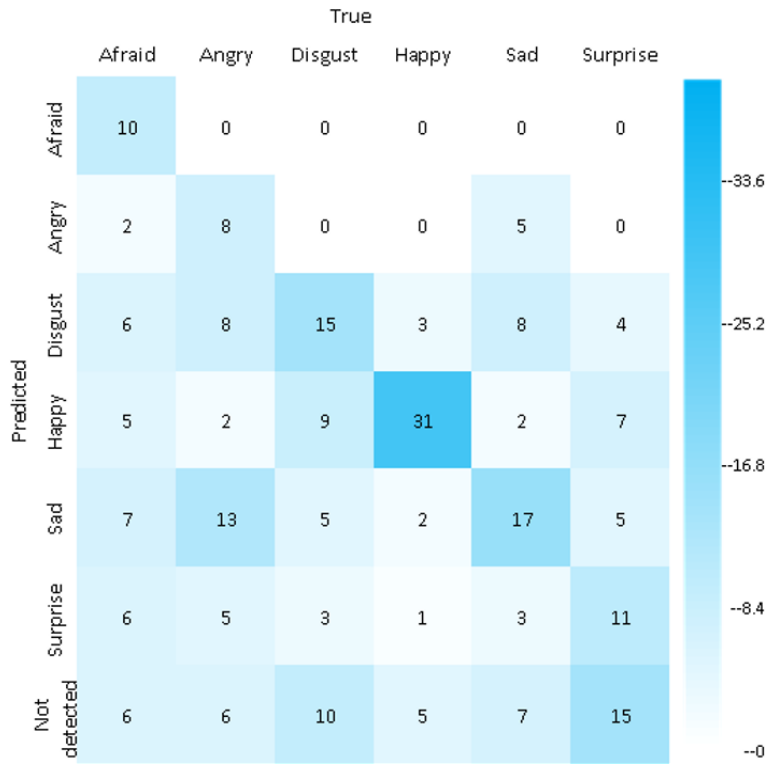


Tabla 3.5:
Resultados de la matriz de confusión de la Figura 3.9.

Class	*TP*	*FP*	*FN*	*TN*
Afraid	10	0	26	167
Angry	8	7	28	160
Disgut	15	29	17	142
Happy	31	25	6	141
Sad	17	32	18	136
Surprise	11	18	16	158

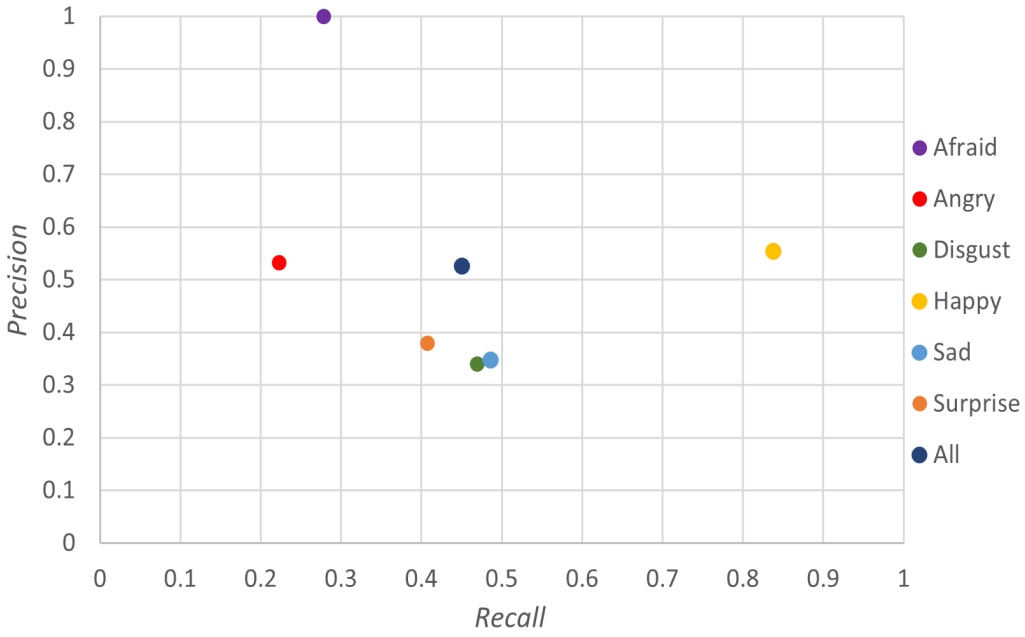
Aplicando las ecuaciones (1.11) - (1.14), que se encuentran en la pág. 35 y 36, pertenecientes a *Precision*, *F1*, *Recall* y *FPR* respectivamente, se obtienen los datos que se muestran en la Tabla 3.6, en donde las clases *Disgust*, *Sad* y *Surprise* tienen un valor de *Recall* y una *Precision* menor al 0.50, indicando que el modelo tuvo problemas para la predicción en las fotografías. En las clases *Afraid* y *Angry* se tiene valor de *Precision* arriba del 0.50, sin embargo, en el valor del *Recall* están por debajo del 0.50 indicando que el modelo tiene problemas para reconocer estas clases, pero cuando lo hacen son altamente confiables.

Tabla 3.6:
Métricas de las fotografías de IMT con el training tres de la versión x.

Class	*Precision*	*Recall*	*FPR*	*F1-score*
Afraid	1	0.27	0	0.43
Angry	0.53	0.22	0.04	0.31
Disgut	0.34	0.46	0.1	0.39
Happy	0.54	0.83	0.15	0.66
Sad	0.34	0.48	0.19	0.40
Surprise	0.37	0.40	0.10	0.39
All	0.52	0.45	0.10	0.48

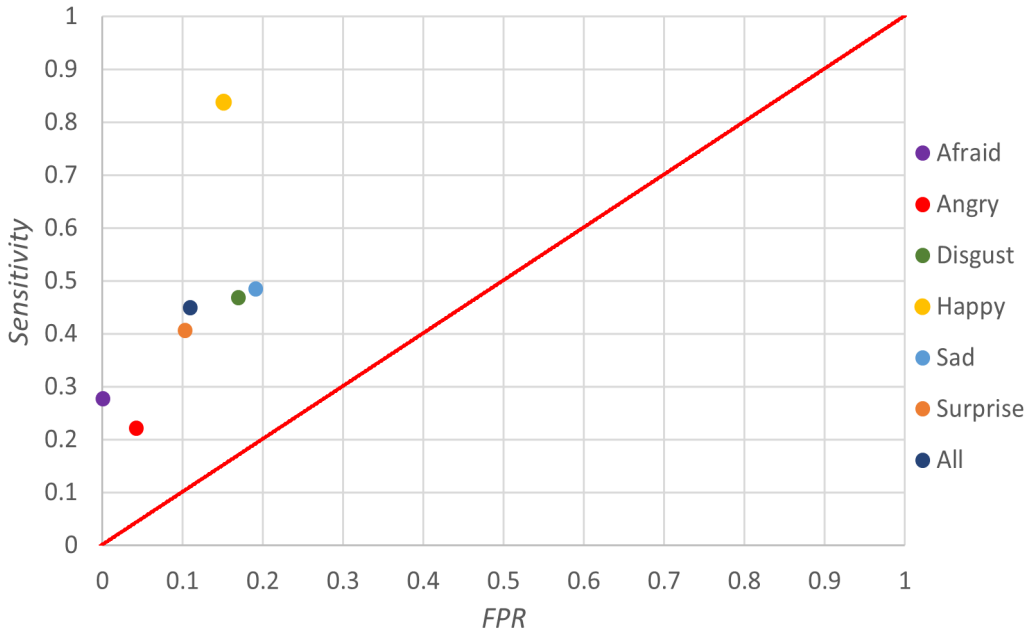
En la Figura 3.10 se observa una gran dispersión entre todas las clases, debido a los resultados tan variados de *Precision* y el *Recall*, la clase *Happy* se aproxima al punto ideal con un valor de 0.83 en *Recall*, sin embargo, en la *Precision* tiene un valor de 0.54, lo cual indica que tendrá problemas de confusión con algunas de las demás clases. *Angry* se encuentra alejado del punto ideal, ya que el modelo no logra predecir correctamente esta clase.

Figura 3.10:
Gráfica Precision-Recall de las fotografías de IMT con el training tres de la versión x.



La curva *ROC* (ver Figura 3.11), en donde todas las clases a excepción de *Happy* tienen una *Sensitivity* menor a 0.50, indicando que el modelo no identificó correctamente las clase en las fotografías.

Figura 3.11:
Curva ROC de las fotografías de IMT con el training tres de la versión x.



Fotografías de los alumnos de *ISC*

Los resultados de evaluar el modelo con las fotografías de los alumnos de *ISC* se muestran en la Figura 3.9 de la cual se obtuvieron los datos para la Tabla 3.5. La clase *Happy* tiene el mejor desempeño, debido a que se clasificaron 30 imágenes correctamente de las 42, mientras que la clase *Afraid* solo detecto correctamente 1 imagen.

Figura 3.12:
Matriz de confusión de las fotografías de ISC con el training tres de la versión x.

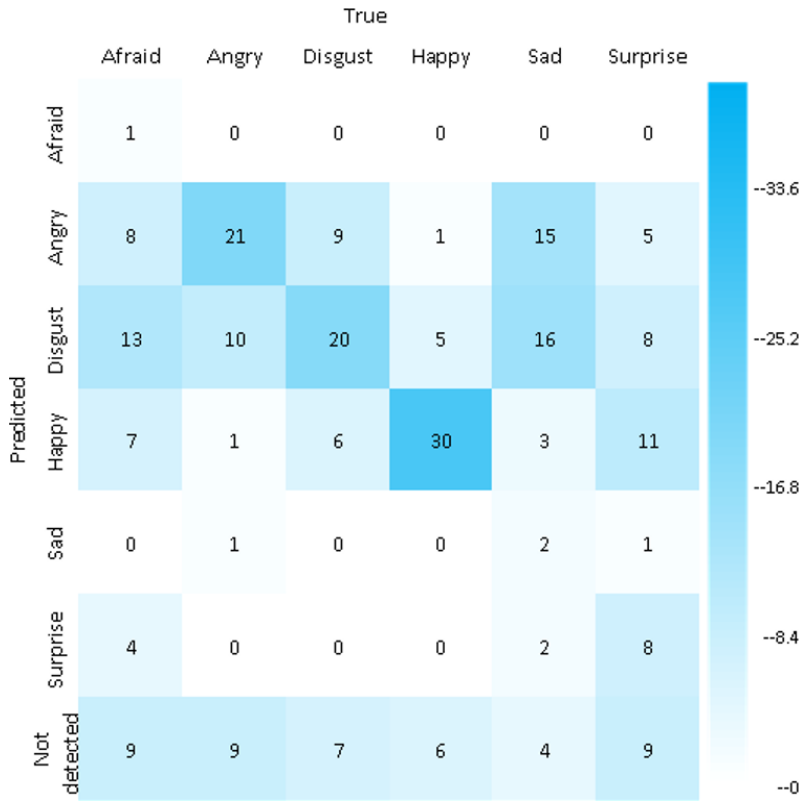


Tabla 3.7:
Resultados de matriz de confusión 3.12.

Class	*TP*	*FP*	*FN*	*TN*
Afraid	1	0	32	175
Angry	21	38	12	137
Disgut	20	52	15	121
Happy	30	28	6	144
Sad	2	2	36	168
surprise	8	6	25	169

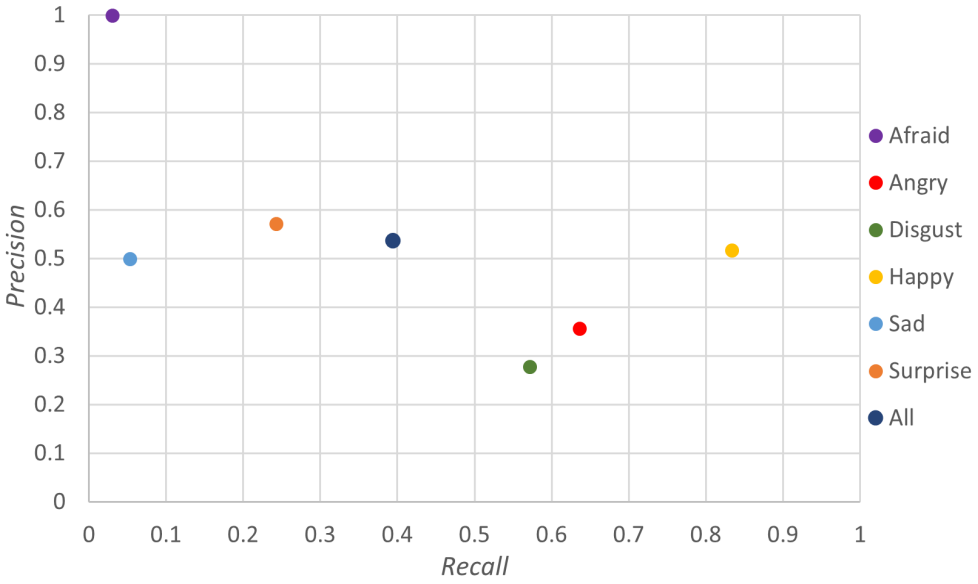
En la Tabla 3.8 se observa que las clases *Afraid*, *Sad* y *Surprise* tienen un valor de *Recall* menor al 0.50, indicando que el modelo tuvo problemas para la predicción de la clase en las fotografías. La clase *Happy* tiene un valor de *Recall* de 0.83, sin embargo, la *Precision* es de 0.51 indicando que el modelo tiene problemas de confusión con las demás clases.

Tabla 3.8:
Métricas de las fotografías de ISC con el training tres de la versión x.

Class	*Precision*	*Recall*	*FPR*	*F1-score*
Afraid	1	0.03	0	0.05
Angry	0.35	0.63	0.21	0.45
Disgut	0.27	0.57	0.30	0.37
Happy	0.51	0.83	0.16	0.63
Sad	0.5	0.05	0.01	0.09
Surprise	0.57	0.24	0.03	0.34
All	0.53	0.39	0.12	0.45

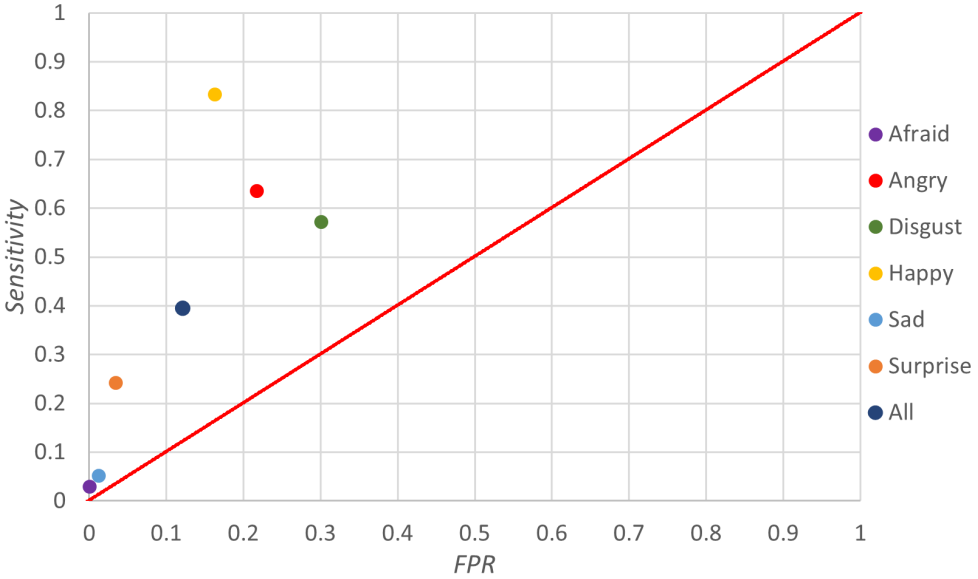
En la Figura 3.13 se observa que la clase *Happy* se aproxima al punto ideal con un valor de 0.83 en *Recall*, sin embargo, en *Precision* se tiene 0.51, lo cual indica que el modelo tendrá problemas de confusión con algunas de las demás clases. *Angry* y *Afraid* se encuentran alejados del punto ideal, ya que el modelo no logra detectar correctamente estas clases.

Figura 3.13:
Gráfica Precision-Recall de las fotografías de ISC con el training tres de la versión x.



La curva *ROC* (ver Figura 3.14), en donde las clases *Happy*, *Angry*, *Disgust* tienen una *Sensitivity* mayor a 0.50, indicando que el modelo identificó correctamente las clase en las fotografías, sin embargo tiende a confundirse entre las otras clases.

Figura 3.14:
Curva ROC de las fotografías de ISC con el training tres de la versión x.



En la Figura 3.15 se muestra la aplicación de la inferencia de manera grupal a cada una de las clases que representan las emociones.

Figura 3.15:
Inferencia de manera grupal con la versión x.

(a) *Afraid*

(b) *Angry*

(c) *Disgust*

(d) *Happy*

(e) *Sad*

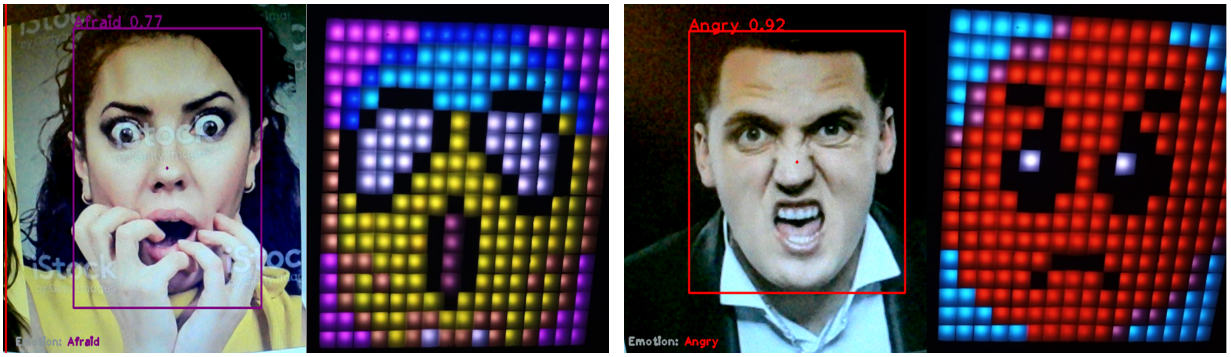
(f) *Surprise*

3.3. Inferencia con la *webcam* y la interfaz electrónica

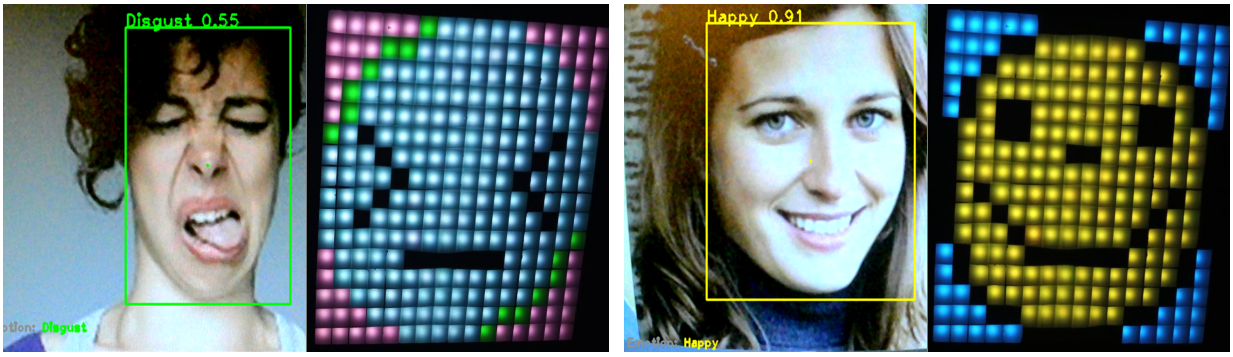
Los resultados de aplicar la inferencia utilizando la RPi-4G se muestran en la Figura 3.16, capturando los datos a través de la *webcam* (lado izquierdo de cada imagen) y visualizado los resultados a través de la interfaz electrónica (lado derecho de cada imagen).

Figura 3.16:
Inferencia con webcam de manera individual con la versión x.

(a) *Afraid* (b) *Angry*



(c) *Disgust* (d) *Sad*



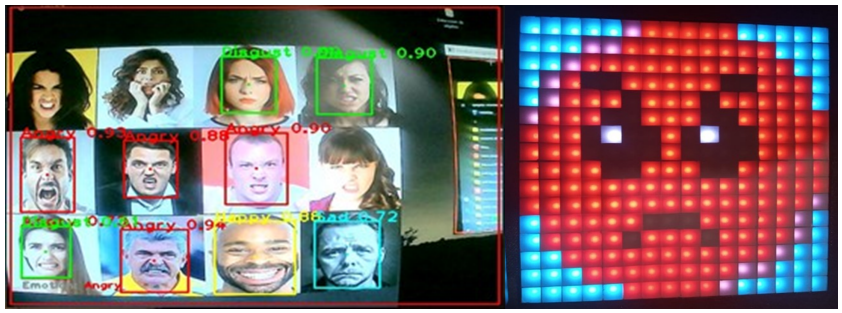
(e) *Sad* (f) *Surprise*

En la Figura 3.17 se muestra la aplicación de la inferencia de manera grupal, en donde el resultado en la interfaz electrónica es la clase con el mayor número de predicciones.

Figura 3.17:
Inferencia con la webcam de manera grupal con la versión x.

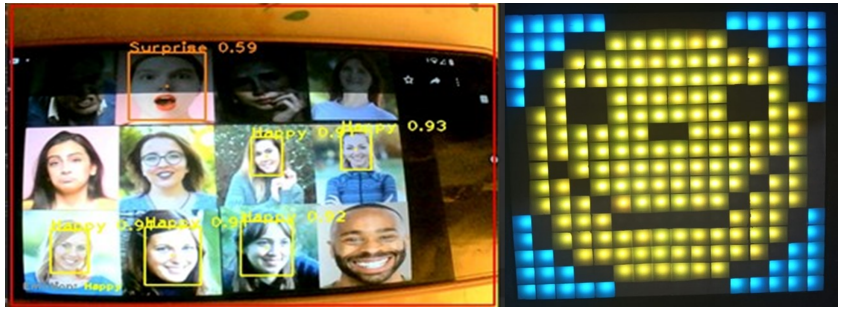
(a) *Afraid*

(b) *Angry*

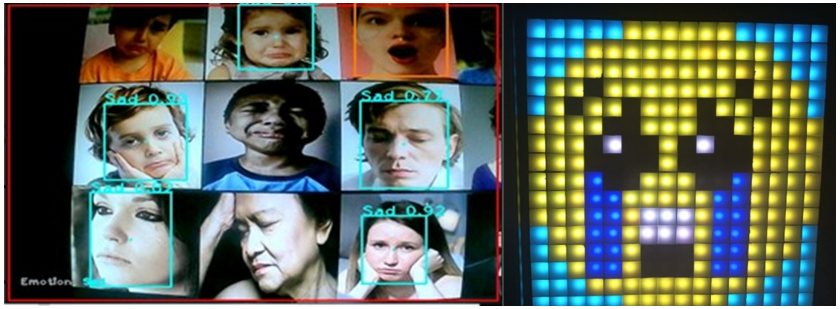


(c) *Disgust*

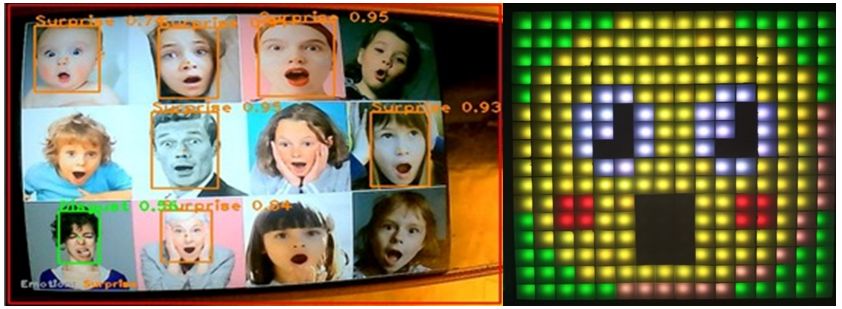
(d) *Happy*



(e) *Sad*



(f) *Surprise*



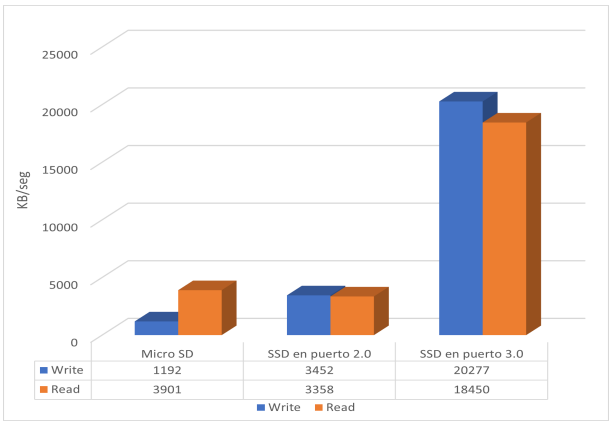
La inferencia se ejecuto utilizando la versión x de *YOLO* (ver Figura 1.35 de la pág. 39), sin embargo, al implementar el modelo en la RPi-4G se tuvo problemas de latencia de un minuto con veintidós segundos, por lo cual se implementaron alternativas para reducir el tiempo de latencia y se describen a continuación.

- *Overclocking*: es trabajar a una velocidad de procesamiento mayor a la certificada por el fabricante original (Ashwin, 2017). La velocidad a la que funciona la RPi-4G es de 1.5 *GHz* y se aumentó a 2 *GHz*.
- Memoria externa: se utilizó un disco de estado sólido (*Solid State Disk, SSD*) con las características que se muestran en la Figura 3.18 para aumentar la velocidad de lectura y escritura de los datos, como se muestra en la comparación de la Figura 3.19a, con la tarjeta micro *SD*, el *SSD* conectado al puerto *USB* 2.0 y 3.0. Mientras que en la Figura 3.19b se muestra de una manera general la velocidad a la que se transmiten los datos, demostrando que hay una diferencia de 279558 *KB/s* entre la tarjeta micro *SD* y el *SSD*.

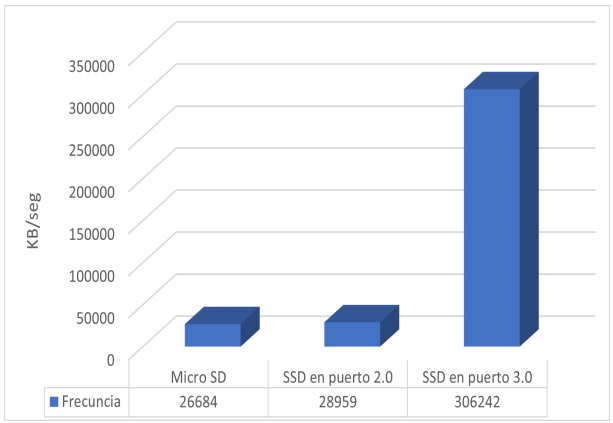
Figura 3.18:
Características del SSD.

Figura 3.19:
Comparación de velocidades de lectura read y write entre micro SD y SSD.

(a) *Read y write en IOPS.*



(b) *Velocidad de transferencia de datos.*



- Aumento de memoria *RAM*.

  Se ejecuto el modelo en la RPi-8G, adicionalmente se realizó *overclocking* (los resultados se muestran en la Figura 3.24 de la pág. 83).

- Implementar las versiones de la red *YOLO* (*n*, *s*, *m* y *l*).

  En la Figura 3.20 se muestra la matriz de confusión con de la versión *n*, en donde la clase *Happy* tuvo un valor de 0.89 en la identificación de *TP*, siendo la de mejor desempeño, mientras que *Disgust* tuvo un valor de 0.68 siendo de menor desempeño en el entrenamiento. En la Tabla 3.9 se muestran las métricas de: *Precision*, *Recall*, *mAp50* y *mAp50-59* para cada clase.

Figura 3.20:
Matriz de confusión del training tres con la versión n.

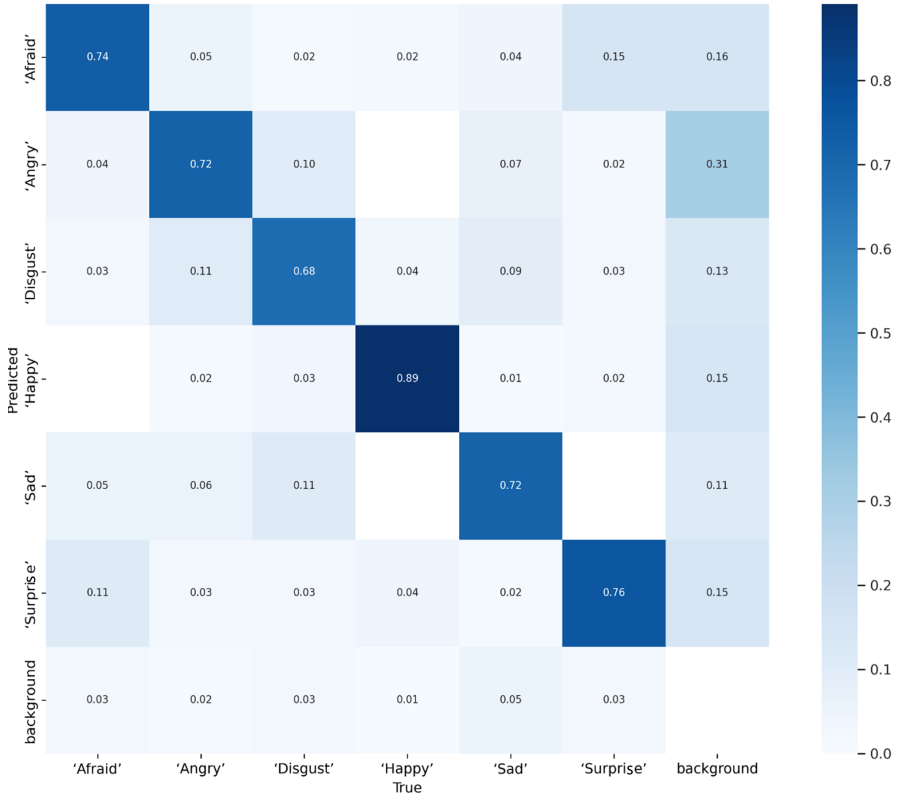


Tabla 3.9:
Métricas de cada clase del training tres con la versión n.

Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
All	636	714	0.75	0.76	0.82	0.56
Afraid	636	114	0.70	0.73	0.78	0.52
Angry	636	118	0.72	0.78	0.82	0.53
Disgust	636	116	0.68	0.66	0.75	0.52
Happy	636	138	0.87	0.90	0.96	0.67
Sad	636	116	0.74	0.69	0.77	0.56
Surprise	636	112	0.80	0.80	0.85	0.57

En la Figura 3.21 se muestra la matriz de confusión correspondiente a la versión *s*

en donde las clases *Sad* y *Disgut* obtuvieron un valor de 0.71 en la identificación de verdaderos positivos, mientras que *Happy* obtuvo un valor de 0.87 siendo la de mayor efectividad. En la Tabla 3.10 se muestran los resultados de las métricas obtenidas a través de la matriz de confusión de la Figura (3.21).

Figura 3.21:
Matriz de confusión del training tres con la versión s.

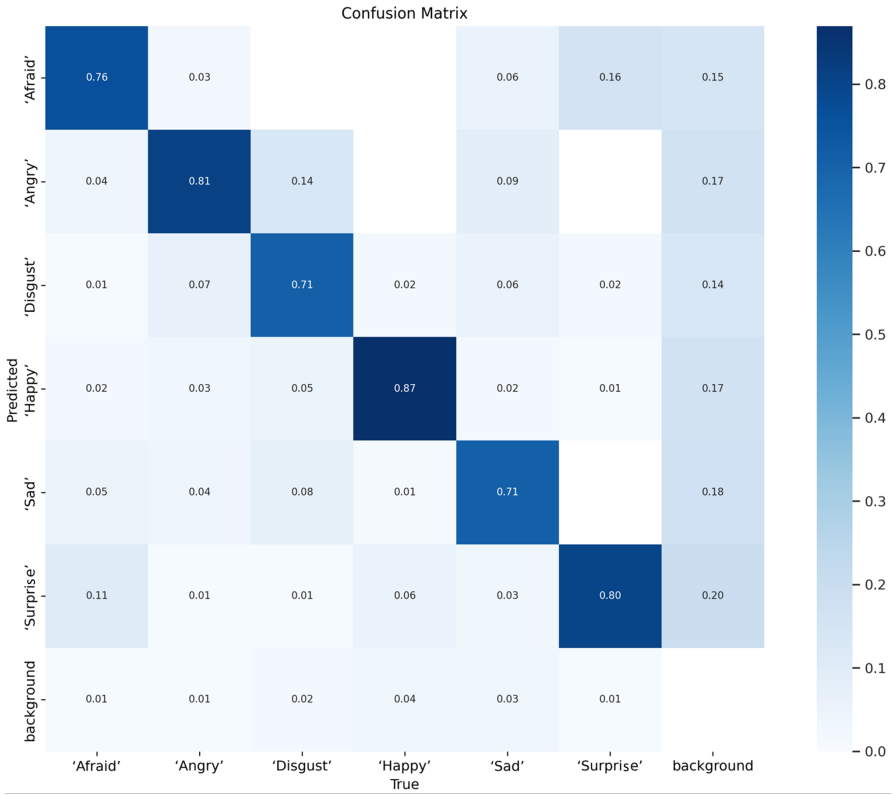


Tabla 3.10:
Métricas de cada clase con training tres y la versión s.

Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
All	636	714	0.78	0.78	0.83	0.57
Afraid	636	114	0.8	0.78	0.80	0.55
Angry	636	118	0.77	0.82	0.83	0.53
Disgust	636	116	0.80	0.69	0.8	0.56
Happy	636	138	0.83	0.87	0.93	0.64
Sad	636	116	0.77	0.71	0.79	0.57
Surprise	636	112	0.74	0.80	0.82	0.56

Para la versión *m* se obtuvo la siguiente matriz de confusión mostrada en la Figura 3.22 en donde las clases con mayor y menor porcentaje en identificación de verdaderos

positivos fueron; *Happy* y *Sad* con valores de 0.89 y 0.71 respectivamente. En la Tabla 3.11 se observan las métricas para evaluar el rendimiento de cada clase.

Figura 3.22:
Matriz de confusión del training tres con la versión m.

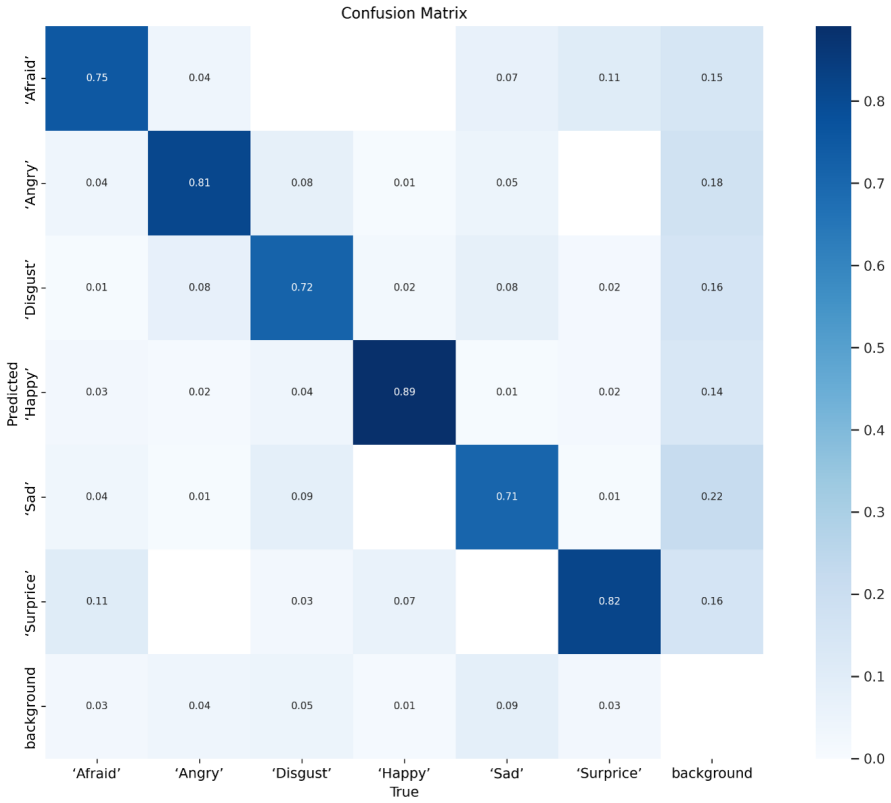


Tabla 3.11:
Matriz de confusión del training tres con la versión m.

Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
All	636	714	0.77	0.78	0.84	0.56
Afraid	636	114	0.74	0.71	0.79	0.52
Angry	636	118	0.77	0.82	0.84	0.52
Disgust	636	116	0.78	0.71	0.79	0.53
Happy	636	138	0.86	0.89	0.96	0.65
Sad	636	116	0.77	0.72	0.81	0.57
Surprise	636	112	0.73	0.82	0.84	0.56

Finalmente en la Figura 3.23 se muestra la matriz de confusión de la versión *l*, en donde la clase *Happy* tuvo un valor de 0.95 en la identificación de verdaderos positivos siendo la de mejor desempeño, mientras que *Disgust* tuvo un valor de 0.70, siendo de menor desempeño en el entrenamiento. En la Tabla 3.12 se muestran las métricas de: *Precision*, *Recall*, *mAp50* y *mAp50-59* para cada clase.

Figura 3.23:
Matriz de confusión del training tres con la versión l.

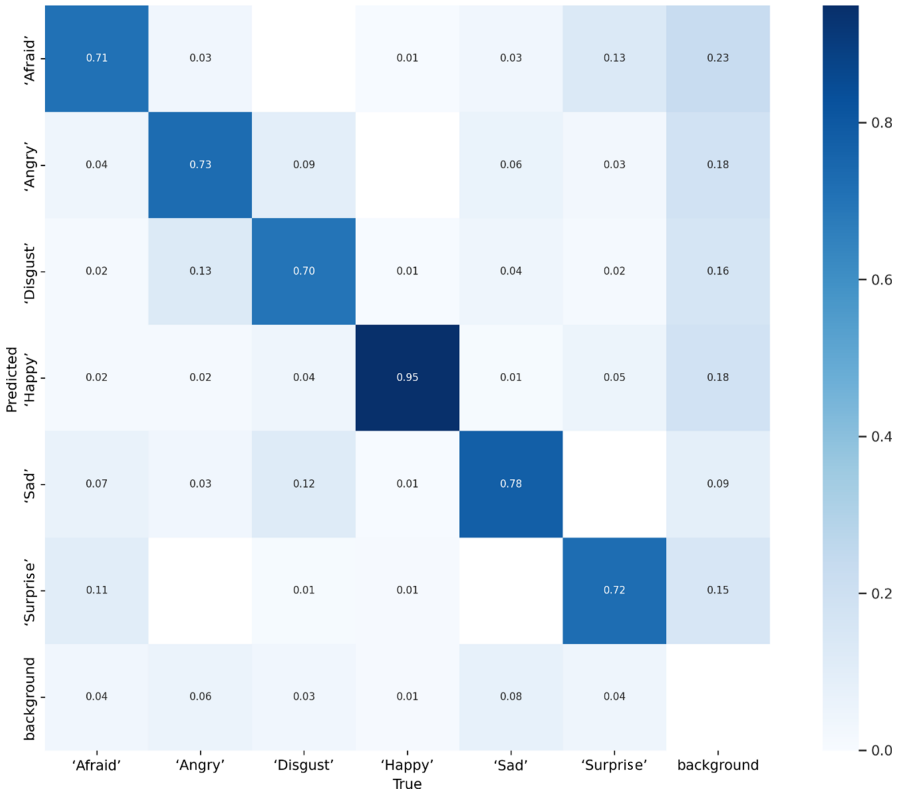


Tabla 3.12:
Métricas para cada clase del training tres con la versión l.

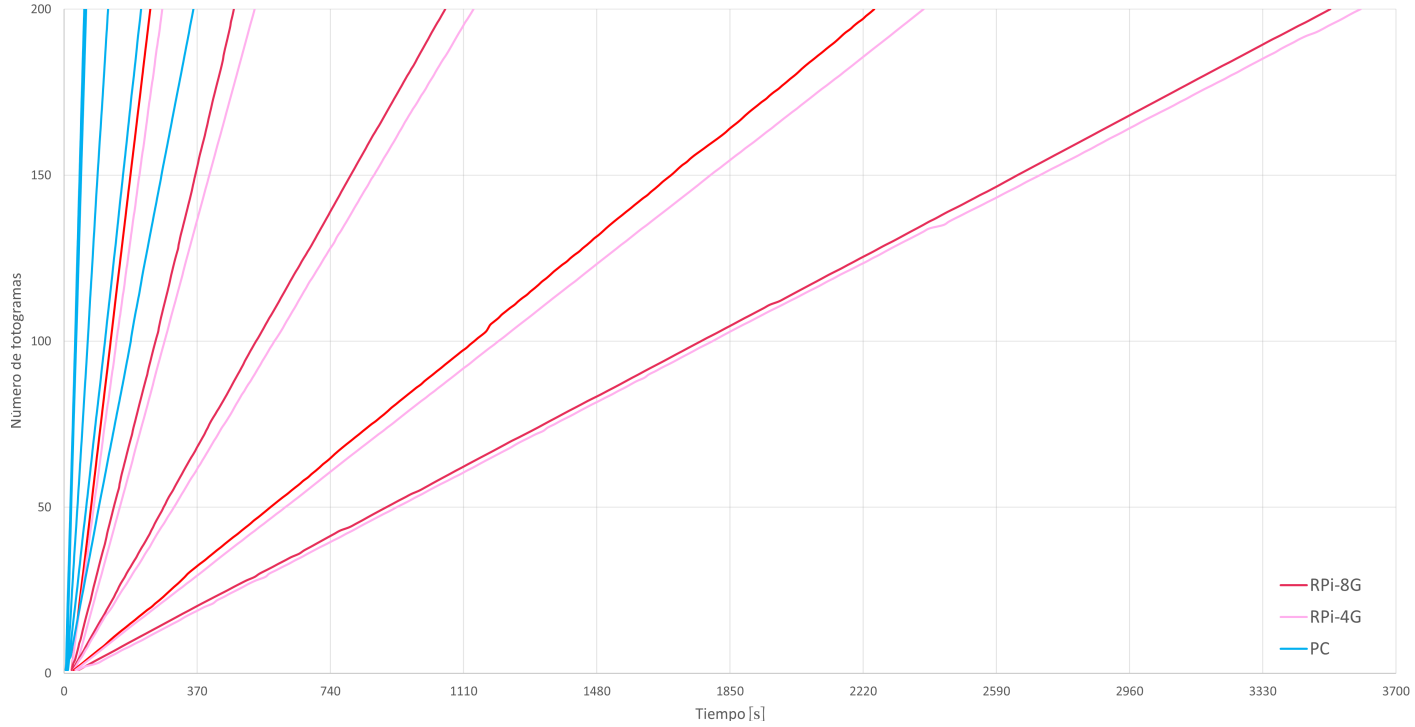
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
All	636	714	0.74	0.78	0.83	0.56
Afraid	636	114	0.68	0.73	0.78	0.53
Angry	636	118	0.73	0.75	0.83	0.53
Disgust	636	116	0.69	0.71	0.77	0.53
Happy	636	138	0.81	0.95	0.54	0.65
Sad	636	116	0.72	0.76	0.78	0.56
Surprise	636	112	0.81	0.77	0.84	0.58

Los modelos entrenados fueron implementados en la RPi-4G, RPi-8G y en una PC (con un procesador *Rizen 3 3200G*) para observar la latencia en cada versión. En la Tabla 3.13 se muestra la comparativa de latencia para cada versión de *YOLO v5*, al inicio de la ejecucion del modelo y al termono de realizar doscientos fotogramas de actualización en la cámara, su representación gráfica se muestra en la Figura 3.24, en donde las líneas de color azul representan la ejecución de los modelos en la PC, las líneas de color rojo representan la ejecución de los modelos en la Rpi-8G, finalmente las líneas de color rosa representan a la Rpi-4G en donde la versión x de *YOLO* realizo los doscientos fotogramas en un tiempo de 3601.47 [s] (61 [min]).

Tabla 3.13:
Comparación de tiempo [s] *entre la* RPi-4G, RPi-8G *y la* PC *para la realización de 200 fotogramas.*

A	RPi-4G		RPi-8G		*PC*	
Versión	Inicio	Final	Inicio	Final	Inicio	Final
n	20.41	272.76	20.77	239.96	4.50	56
s	31.68	529.54	21.16	472.01	7.69	61.18
m	26.62	1137.50	23.04	1058.64	8.43	112.68
l	31.60	2387.64	27.65	2250.29	10.99	214.33
x	39.20	3601.47	40.27	3517.79	4.99	359.36

Figura 3.24:
Comparación del rendimiento [s], *entre la* RPi-4G, RPi-8G *y la* PC *para la realización de 200 fotogramas.*



El cálculo de los *FLOPS* de una computadora es variable de acuerdo a las diferentes arquitecturas de los procesadores y la variabilidad en las capacidades de procesamiento. Sin embargo, a través de la ecuación (3.1) se pueden estimar los *FLOPS*:

$$FLOPS = (FLOPS \text{ por núcleo})(\text{Ciclos por segundo})(\text{Número de núcleos}) \quad (3.1)$$

- *GLOPS* para RPi. La RPi-4G está equipada con un procesador *Broadcom BCM2711*, que contiene cuatro núcleos *ARM Cortex-A72* funcionando a 1.5 *GHz*. Cada núcleo tiene una capacidad de punto flotante de 2 *FLOPS* por núcleo y cada uno funciona a 1.5 GHz, calculando el rendimiento de la siguiente manera:

  $$FLOPS_{RPi} = (2)(1.5\ GHz)(4) = 12\ GFLOPS$$

  Aproximadamente, la RPi-4G puede procesar alrededor de 12 *GFLOPS*. Teniendo en cuenta que esta es una estimación simplificada y que el rendimiento real puede variar de acuerdo a la carga de trabajo.

- *GFLOPS* para *PC* (*AMD Ryzen 3 3200G*).

  El procesador *AMD Ryzen 3 3200G* es una *CPU* de la serie *Ryzen 3000* basada en la arquitectura *Zen+* de *AMD*. Tiene 4 núcleos, 4 subprocesadores, funciona a una frecuencia base de 3.6 *GHz* e integra una unidad de procesamiento de gráficos *Radeon Vega*. La arquitectura *Zen+* tiene una capacidad de 4 operaciones de punto flotante por núcleo, por lo que el rendimiento en *GFLOPS* sería:

  $$FLOPS_{CPU} = (4)(3.6\ GHz)(4) = 57.6\ GFLOPS$$

  Por lo tanto, aproximadamente, el *AMD Ryzen 3 3200G* puede procesar alrededor de 57.6 *GFLOPS* considerando solo los núcleos de *CPU*, debido a que la unidad de gráficos tiene su propio rendimiento independiente. La *GPU* integrada en el procesador *AMD Ryzen 3 3200G* es una unidad de procesamiento gráfico *Radeon Vega*. La capacidad de procesamiento en *GFLOPS* de una *GPU* se calcula de manera diferente a la de una *CPU*, ya que las *GPU's* están diseñadas para ejecutar simultáneamente múltiples subprocesos en paralelo y tienen arquitecturas especializadas para cargas de trabajo altamente paralelas. La *GPU Radeon Vega* tiene 8 unidades de cómputo y funciona a una frecuencia base de 1250 *MHz* (1.25 *GHz*). Cada unidad de cómputo es capaz de realizar 64 operaciones de punto flotante por ciclo, por lo que el rendimiento en *GFLOPS* sería:

  $$FLOPS_{GPU} = (64)(1.25\ GHz)(8) = 640\ GFLOPS$$

  Por lo tanto, la *GPU* integrada en el *AMD Ryzen 3 3200G* puede procesar alrededor de 640 *GFLOPS*. Este cálculo se basa en estimaciones y el rendimiento real puede variar de acuerdo a la carga de trabajo. El rendimiento total en *GFLOPS* considerando tanto la *GPU* como la *CPU* en el procesador *AMD Ryzen 3 3200G*, se obtienen al sumar los rendimientos en *GFLOPS* de ambas partes:

  $$Rendimiento\ total = 57.6\ GFLOPS + 640\ GFLOPS = 697.6\ GFLOPS$$

  Por lo tanto el *AMD Ryzen 3 3200G* puede procesar aproximadamente 697.6 *GFLOPS* en total.

Capítulo 4

Conclusiones

Conclusiones

- Se realizó la recolección y etiquetado de la base de datos con 3,180 imágenes descargadas del internet. En la Tabla 3.2 de la pág. 65 se observa la columna perteneciente al número de instancias, el cual indica que existe una diferencia en los resultados entre cada clase, debido a que no se tiene un *dataset* balanceado y de acuerdo con Pulgar et al., 2018, el desequilibrio de las clases es uno de los principales problemas en la clasificación de datos reales, lo que influye negativamente en el rendimiento predictivo del modelo.
- Con la intención de evaluar el desempeño de la versión x de *YOLO*, en la inferencia en las imágenes descargadas del internet y en las fotografías de los alumnos pertenecientes a las divisiones de *IMT* e *ISC*, se considera la métrica *F1-score*. De acuerdo con Aurelien, 2019, el *F1-score* combina las métricas de *Precision* y *Recall* teniendo un solo resultado que permite la comparación con diferentes clases y modelos. Para las imágenes descargadas del internet el resultado fue 0.81, sin embargo, para las pruebas realizadas en las fotografías de las divisiones de *IMC* e *ISC* se obtuvieron 0.48 y 0.45 respectivamente, existiendo una diferencia de 0.33 aproximadamente entre las imágenes descargadas y las fotografías, esto debido a factores como la calidad de la cámara, la iluminación y el bajo número de imágenes que se consideraron para realizar el entrenamiento.
- Aplicando técnicas de visión artificial se consiguió controlar la interfaz electrónica al ejecutar el modelo en la Rpi-4G, ya que al predecir la emoción con la *webcam* de manera grupal o individual se observaba el pictograma en la interfaz electrónica correspondiente a la emoción predicha por el modelo. Cabe mencionar que se puede optimizar el código para tener mejores resultados.
- Al realizar las pruebas de inferencia con la versión x de *YOLO* en la Rpi-4G y con la *webcam* se tuvo una de latencia de un minuto aproximadamente, haciendo que la Rpi-4G no sea adecuada para la ejecución del modelo debido a la alta demanda de recurso computacional de 204 *GFLOPS*.
- Los resultados de la inferencia con la *webcam*, evaluados a través de la matriz de confusión que se muestra en la pág. 65, demuestran que el modelo necesita un *dataset* mayor al que se utilizó de 3,180 imágenes, con la finalidad de aumentar la *Precision* al momento de realizar la inferencia en tiempo real y reducir los problemas de confusión que se tienen entre ellas.

Recomendaciones.

- Aumentar el tamaño de la base de datos para entrenamiento, para este trabajo se utilizaron un total de 3180 imágenes, sin embargo, fueron divididas entre las 6 clases, dejando un total de 530 imágenes por clase, además de incluir en el entrenamiento fotografías de las expresiones faciales de los alumnos.
- Tener en cuenta la calidad de imágenes que son utilizadas para la evaluación del modelo, debido a que una mala iluminación en ellas, afecta la predicción en la clase correspondiente. Así como también utilizar una cámara superior a los 5 *megapixeles* que fue la que se utilizó en este trabajo, de esta manera se pueden obtener mejores resultados en la captura de los fotogramas.
- Implementar una metodología diferente para el reconocimiento de emociones en los alumnos, por ejemplo, tomar diferentes muestras en un determinado tiempo de cada alumno y guardar los datos para tener un control estadístico que pudiesen ayudar en el desarrollo de futuros trabajos.
- Al aplicar el modelo *YOLO* en cualquiera de sus versiones, verificar la demanda de *GFLOPS* del modelo a utilizar para seleccionar adecuadamente el equipo correspondiente que pueda ejecutar correctamente.

Anexos

Productos del trabajo de investigación

- 13A Semana BEIFI y proyectos de investigación, Instituto Politécnico Nacional Unidad ESIME-Culhuacan, Culhuacan Ciudad de México, 16 y 17 de junio.

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MÉCANICA Y ELÉCTRICA
UNIDAD CULHUACAN
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
CAPÍTULO DE CIRCUITOS Y SISTEMAS DE LA IEEE SECCIÓN MÉXICO
IEEE RAMA ESTUDIANTIL-ESIME CULHUACAN

OTORGA LA PRESENTE **CONSTANCIA** POR SU PARTICIPACIÓN EN LA ***13A SEMANA BEIFI Y PROYECTOS DE INVESTIGACIÓN*** QUE SE REALIZÓ EL 16 Y 17 DE JUNIO DE 2022 EN MODALIDAD PRESENCIAL A:

JUAN CARLOS LARA GORDILLO, ALDAIR BRAYAM RIVERA CASTRO, LAURA CLEOFAS SÁNCHEZ

CON EL **CARTEL:**

RECONOCIMIENTO DE EMOCIONES DESFAVORABLES EN LA EFICIENCIA TERMINAL DE LOS ALUMNOS DEL TEST

"La Técnica al Servicio de la Patria"

M. en I. Lidia Prudente Tixteco
Jefa del Departamento de Investigación

Dr. Juan Carlos Sánchez García
Presidente CAS IEEE Sección México

M. en C. Osvaldo López García
Jefe Interino de la Sección de Estudios de Posgrado e Investigación

M. en E. Dalia Ruiz Domínguez
Directora

DIRECCIÓN

- Exposición en la semana de ciencia y tecnología edición 2022, Tecnológico de estudios Superiores de Tianguistenco, Santiago Tilapa Tianguistenco Méx, 27 de octubre.

- Inova TEST en el Tercer Torneo de Robótica “ROBOTEST”, Tecnológico de estudios Superiores de Tianguistenco, Santiago Tilapa Tianguistenco Méx, 30 de marzo.

- IX Congreso Internacional de Investigación Tijuana, Tijuana B. C. México, Abril 2023.

Referencias

Aggarwal, C. (2018). *Neural networks and deep learning* (Vol. 10). Springer.

Alegre, E., Campos, G., & De la Escalera, A. (2019). *Conceptos y Métodos en visión por computadorgráfica de visualizaciones* (1st). Comité Español de Automática.

Ameyaw, D., Deng, Q., & Söffker, D. (2022). Evaluating machine learning-based classification approaches: a new method for comparing classifiers applied to human driver prediction intentions. *IEEE Access*, *10*, 62429-62439.

Andueza, A. (2022). Aplicación de filtros de concurrencias en redes neuronales convolucionales para mejorar la transferencia de estilos en imágenes. *UPNA*.

Artola, M. (2019). *Clasificación de imágenes usando redes neuronales convolucionales en Python* [Trabajo Fin de Grado]. Unidad Académica de Ingeniería de Civil.

Ashwin, P. (2017). *Raspberry Pi Supercomputing and Scientific Programming. MPI4PY, NumPy, and SciPy for Enthusiasts*. Apress.

Aurelien, G. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2.ª ed.). O'reilly.

Azis, H., Purnawansyah, P., Fattah, F., & Putri, I. (2020). Performa Klasifikasi K-NN dan Cross Validation Pada Data Pasien Pengidap Penyakit Jantung. *ILKOM Jurnal Ilmiah*, *12*(2), 81-86.

Badrulhisham, N., & Mangshor, N. N. A. (2021). Emotion Recognition Using Convolutional Neural Network (CNN). *Journal of Physics: Conference Series*.

Barolet, D. (2008). Light-emitting diodes (LEDs) in dermatology. *Seminars in cutaneous medicine and surgery*, *27*(4), 227-238.

Barrett, S., & Pack, D. (2006). *Microcontrollers Fundamentals for Engineers And Scientists*. Morgan; Claypool Publishers.

Basogain, X. (2008). Redes neuronales artificiales y sus aplicaciones. *Dpto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingeniería Bilbao*.

Berzal, F. (2018). *Redes Neuronales & Deep Learning* (1st). EUG, Editorial Universidad de Granada.

Campos, M. (2018). Inspiración biológica de las redes neuronales artificiales. https://medium.com/soldai/inspiraci%C3%B3n-biol%C3%B3gica-de-las-redes-neuronales-artificiales-9af7d7b906a

Chaganti, S. Y., Nanda, I., Pandi, K. R., Prudhvith, T. G., & Kumar, N. (2020). Image Classification using SVM and CNN. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 1-5.

Chollet, F. (2021). *Deep learning with Python*. Simon; Schuster.

Coppin, B. (2004). *Artificial intelligence illuminated*. Jones & Bartlett Learning.

Cortéz, J. (2009). Diseño y construcción de un sistema de alimentación para un arreglo RGB de tres led de potencia. *Universidad Tecnológica de la Mixteca). Tesis para optar al Título de ingeniero Electrónico México*.

Dao, H. (2020). *Image classification using convolutional neural networks* [Trabajo Final de Máster]. Oulu University of Applied Sciences.

Ekman, P., & Friesen, W. (1971). Constants across cultures in the face and emotion. *Journal of personality and social psychology*, *17*(2), 124.

Ezat, W., Dessouky, M., & Ismail, N. (2020). Multi-class image classification using deep learning algorithm. *Journal of Physics: Conference Series*, 012021.

Farias, F. (2019). *Implementación de un sistema para el conteo volumétrico de objetos mediante Redes Neuronales Convoluciones* [Trabajo Final de Máster]. Unidad Académica de Ingeniería de Civil.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874.

Fernández, E., Martín, M. D., & Sánchez. (2010). Capitulo 1: Psicología de la emoción. En *Psicología de la emoción* (1.ª ed., pp. 17-74). Editorial Universitaria Ramón Areces.

Frogames. (2020). Redes Neuronales Convolucionales, la Guía Definitiva - Frogames. https://frogames.es/la-guia-definitiva-%20de-las-redes-neuronales-convolucionales/

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193-202.

González, R. (2011). Python para todos.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354-377.

Guevara, C., Cruz, V., Vergara, O., Nandayapa, M., Dominguez, H., & Sossa, J. (2021). Study of the Effect of Combining Activation Functions in a Convolutional Neural Network. *IEEE Latin America Transactions*, *19*(5), 844-852.

Gutiérrez, L., & Jiménez, J. (2021). *Desarrollo de un semáforo inteligente basado en redes neuronales para la optimización de la afluencia vehicular* [Tesis]. Ingeniería en comunicaciones y electrónica, Instituto Politécnico Nacional.

Halfacree, G. (2020). *The Official Raspberry Pi Beginner's Guide: How to Use Your New Computer*. Raspberry Pi Trading Limited.

Heaton, J. (2015). *Artificial intelligence for humans*. Heaton Research, Inc.

Hernández, M., Flores, R. O., Morales, R., Hernández, J. C., & Saldaña, J. M. (s.f.). Diseño, construcción y control de un cubo de leds RGB con un Microcontrolador PIC18F25J11 y manejado con Visual C.

Hernández, S. (2019). Inteligencia Artificial y Deep Learning desde cero en Python. https://www.udemy.com/course/deep-learning-desde-cero-en-python/

Instruments, T. (2023). SN74HC573N Datasheet. https://www.ti.com/product/es-mx/SN54HC573A

Jensen, E. (2003). *Cerebro y aprendizaje: competencias e implicaciones educativas* (Vol. 96). Narcea Ediciones.

Jocher, G. (2020). YOLOv5 by Ultralytics. https://doi.org/10.5281/zenodo.3908559

Jones, C., & Jonsson, M. (2005). Detecting emotions in conversations between driver and in-car information systems. *International Conference on Affective Computing and Intelligent Interaction*, 780-787.

Joshi, P. (2017). *Artificial intelligence with python*. Packt Publishing Ltd.

Kim, P. (2017). *Matlab deep learning with machine learning, neural networks and artificial intelligence*. Springer.

Kirkvik, E. B. (2022). *Facial emotion recognition using deep learning* [Trabajo Final de Máster]. Metropolitan University.

Martín, J. (2017). *Fuentes de alimentación (Electrónica)*. Editex.

Martínez, A. (2017). *Reconocimiento de imágenes con redes convolucionales en C* [Trabajo de Fin de Grado]. Escuela Técnica Superior de Ingeniería Universidad de Sevilla.

Marvin, M., & Seymour, P. (1969). Perceptrons. *Cambridge, MA: MIT Press*, *6*, 318-362.

Minaee, S., Minaei, M., & Abdolrashidi, A. (2021). Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network. *Sensors*, *21*(9). https://doi.org/10.3390/s21093046

Moon, S.-E., Chen, C.-J., Hsieh, C.-J., Wang, J.-L., & Lee, J.-S. (2020). Emotional EEG classification using connectivity features and convolutional neural networks. *Neural Networks*, *132*, 96-107.

Müller, A., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists* (1.ª ed.). O'Reilly Media.

Narváez, S., Luis nd Arciniegas, & Cuaspa, C. (2018). Caracterización de imágenes de baja resolución en matrices led rgb. un modelado electrónico bajo herramientas libres. *AXIOMA*, (19), 69-85.

Nguyen, H. (2021). *Facial Emotional Recognition Experiment by applying R-CNN* [Master thesis]. University of Eastern Finland, Faculty of Science y Forestry, Joensuu. School of Computing.

Noman, M., Stankovic, V., & Tawfik, A. (2019). Object detection techniques: Overview and performance comparison. *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 1-5.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Pattanayak, S. (2017). *Pro deep learning with TensorFlow A mathematical approach to advanced artificial intelligence in Python*. Springer.

Ponce, P. (2010). *Inteligencia artificial: con aplicaciones a la ingeniería*. Alpha Editorial.

Pulgar, F., Rivera, A., Charte, F., & del Jesus, M. J. (2018). Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales. *XVIII Conferencia de la Asociacion Espanola para la Inteligencia Artificial*, 1213-1218.

PyLessons. (2019). YOLOv3 theory explained. https://pylessons.com/YOLOv3-introduction

Raschka, S. (2015). *Python Machine Learning*. Packt Publishing. http://gen.lib.rus.ec/book/index.php?md5=76c1629372ef61f1a600117a71016573

Redmon, J. (2012). YOLO: Real-Time Object Detection. https://pjreddie.com/darknet/yolo/

Redmon, J., Divala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

Romero, J. J., Dafonte, C., Gómez, A., & Penousal, F. J. (2007). Inteligencia artificial y computación avanzada. *Santiago de Compostela: Fundación Alfredo Brañas*, 10-15.

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*.

Saez De La Pascua, A. (2019). *Deep learning para el reconocimiento facial de emociones básicas* [B.S. thesis]. Universitat Politècnica de Catalunya.

Sánchez, J. M. (2016). Análisis de Calidad Cartográfica mediante el estudio de la Matriz de Confusión. *Pensamiento matemático*, *6*(2), 9-26.

Schutz, P., Pekrun, R., & Phye, G. (2007). *Emotion in Education (Educational Psychology)* (1.ª ed.). Academic Press.

Siew-Chong, T., Chi Kong, T., Vasca, F., & Iannelli, L. (2012). *Dynamics and Control of Switched Electronic Systems: Advanced Perspectives for Modeling, Simulation and Control of Power Converters* (1.ª ed.). Springer.

Sisquella, J. (2019). *Machine learning and deep learning for emotion recognition* [Master thesis]. Universidad Politécnica de Catalunya.

Skansi, S. (2018). *Introduction to Deep Learning: from logical calculus to artificial intelligence*. Springer.

Smith, M. (2019). *Las emociones de los estudiantes y su impacto en el aprendizaje: Aulas emocionalmente positivas* (Vol. 157). Narcea Ediciones.

Sultana, F., Sufian, A., & Dutta, P. (2018). Advancements in image classification using convolutional neural network. *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 122-129.

Team, L. I. (2022). Deep Learning de A a Z:redes neuronales en Python desde cero. https://www.udemy.com/course/deep-learning-a-z/

Wang, C., Huang, Y., & Shi, B. (2021). Front obstacle detection system based on YOLOv3. *Journal of Physics: Conference Series*, *1732*(1), 012112.

Worldsemi. (2020). WS2812B Datasheet. https://www.alldatasheet.com/datasheet-pdf/pdf/1179113/WORLDSEMI/WS2812B.html

Wu, C.-D., & Chen, L.-H. (2019). Facial emotion recognition using deep learning. *arXiv preprint arXiv:1910.11113*.

Yani, M., Irawan, S., & Setianingsih, C. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *Journal of Physics: Conference Series*, *1201*. https://doi.org/10.1088/1742-6596/1201/1/012052

Yu, Z., Mohammed, A., & Panahi, I. (1997). A review of three PWM techniques. *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, *1*, 257-261.

Zhihao, H. (2020). *Emotion recognition in video using Deep learning method with subtract pre-processing* [Master thesis]. University of Strathclyde.