INSTITUTO TECNOLÓGICO SUPERIOR DEL SUR DE GUANAJUATO

Detección automática de densidad de pasajeros del transporte público mediante técnicas de aprendizaje máquina

OPCIÓN 2: TITULACIÓN INTEGRAL – TESIS PROFESIONAL

Elaborada por:

BERENICE CALDERÓN DAMIÁN

ASESOR:

DR. ROGELIO HASIMOTO BELTRÁN

CENTRO DE INVESTIGACIÓN DE MATEMATICAS, CIMAT

CO-ASESORA:

M.C. MARIA TRINIDAD PIMENTEL VILLEGAS

Guanajuato, Gto. Octubre, 2021

Detección automática de densidad de pasajeros del transporte público mediante técnicas de aprendizaje máquina

Elaborada por:

BERENICE CALDERÓN DAMIÁN

Aprobado por. . . . . .

M.C. Maria Trinidad Pimentel Villegas
Docente de la carrera de Ingeniería en Sistemas Computacionales
Asesor de la opción de titulación 2. Tesis Profesional

Revisado por. . . . . .

M.C. Hugo Armando Aguilera García
Docente de la carrera de Ingeniería en Sistemas Computacionales
Revisor de la opción de titulación 2. Tesis Profesional

Revisado por. . . . . .

M.C. Jeziel Vázquez Nava
Docente de la carrera de Ingeniería en Sistemas Computacionales
Revisor de la opción de titulación 2. Tesis Profesional

Instituto Tecnológico Superior del Sur de Guanajuato

TECNOLOGÍA Y CALIDAD PARA LA VIDA

LIBERACIÓN DE PROYECTO PARA LA TITULACIÓN INTEGRAL

Uriangato,Gto., 23/septiembre/2021

Asunto: Liberación de proyecto para la titulación integral

Ing. J. Trinidad Tapia Cruz
Director Académico y de Estudios Profesionales
ITSUR
PRESENTE

Por este medio informo que ha sido liberado el siguiente proyecto para la titulación integral:

Nombre de estudiante y/o egresado(a): Berenice Calderón Damián	
Carrera: Ingeniería en Sistemas Computacionales	Núm. de control: S16120253
Nombre del proyecto: Detección automática de densidad de pasajeros del transporte público mediante técnicas de aprendizaje máquina	
Producto: Opción 2: Titulación Integral – Tesis profesional	

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestras y nuestros egresados.

ATENTAMENTE

Mca. Miguel Cruz Pineda
Coordinador de Ingeniería en Sistemas Computacionales
ITSUR

Instituto Tecnológico Superior del Sur de Guanajuato
ESTADOS UNIDOS MEXICANOS
COORDINACIÓN INGENIERÍA SISTEMAS COMPUTACIONALES

La comisión revisora ha tenido a bien aprobar la reproducción de este trabajo.

MC. MARIA TRINIDAD PIMENTEL VILLEGAS	LIC. HUGO ARMANDO AGUILER GARCIA	MC. JEZIEL VAZQUEZ NAVA

c.c.p.- Expediente

Resumen

El sistema de transporte público es considerado como el medio de traslado más importante en México; el 80% de los viajes a nivel nacional se realizan por este medio de transporte, según lo demuestra un estudio realizado por el Instituto de Geografía de la UNAM (Lastra, 2017). Su importancia en el desarrollo de las ciudades es cada vez mayor, ya que al trasladar un gran número de personas a la vez reduce la contaminación ambiental, el congestionamiento vehicular y el tiempo de traslado en general de la población. De esta manera, la gestión y administración óptima del transporte público municipal repercute directamente en la calidad de servicio del mismo. El presente trabajo de tesis, propone el uso de la Inteligencia Artificial, en particular redes neuronales profundas para la detección de rostros dentro de unidades del transporte público. La finalidad es determinar el número de pasajeros que usa el sistema de transporte público para en un futuro gestionar, el número de unidades requeridas para ofrecer un servicio de calidad (bajos tiempos de espera y unidades no sobresaturadas). Adicionalmente, el sistema desarrollado tiene la capacidad de detectar rostros con cubrebocas, para cuantificar el porcentaje de personas dentro del transporte urbano que cumplen con la normatividad establecida por el gobierno federal. Se describen las actividades teórico-prácticas realizadas durante el desarrollo de este trabajo, asi como una descripción detallada de los conceptos nuevos de inteligencia artificial, tales como aprendizaje máquina, procesamiento de imágenes digitales, redes neuronales y su implementación mediante el lenguaje de programación Python.

Abstract

Public transportation network is considered as one of the most important commuting system in México, nearly 80% of commuters nationwide use public transport, as reported by the Instituto de Geografía de la UNAM (Lastra, 2017). Its relevance for city development is crucial, it moves thousands or even millions of people a day reducing environmental pollution, traffic congestion, and population commuting time.

Optimal managing and control of public transportation systems directly improves the quality of service and people wellness. In this work, we propose an artificial intelligent scheme (Deep Neural Networks) for face detection in public transport units. Our goal is to measure the density and number of public transport users as a function of time, to improve the commuting quality of service such as low traveling times, waiting time of the bus, and not oversaturated units. Additionally, in order to control the spread of Covid-19, our algorithm is capable of detecting people wearing protection mask. We provide an in-depth description of the theoretical basis of artificial intelligence, image processing, neural networks and their implementation in Python programming language.

Palabras clave

Aprendizaje máquina, redes neuronales, procesamiento de imágenes digitales, Deep Learning, detección de rostros.

Keywords

Machine learning, neural networks, digital image processing, Deep Learning, face detection.

Agradecimientos

Primero que nada, agradezco el apoyo otorgado mediante el proyecto fordecyt 296737 “consorcio en inteligencia artificial” para el desarrollo de la presente tesis.

También agradecerle a mis padres y hermanos por el apoyo brindado durante toda mi carrera profesional, mi asesora y equipo de trabajo del proyecto por guiarme y enseñarme durante todo este proceso correctamente, mis amigos que estuvieron conmigo durante este viaje, su apoyo y la ayuda que siempre me brindaron. A mis maestros de escuela secundaria, sin ellos no estaría escribiendo esto ni hubiera tomado el reto de seguir a delante. Y a todas aquellas personas que he conocido a lo largo del camino que me impulsaron a seguir adelante, en especial a Verónica Cimadoro, una amiga y consejera que me guío en la recta final de mi formación profesional.

Tabla de contenido

Índice de Figuras

Índice de Gráficas

Índice de Tablas

Índice de Ecuaciones

Capítulo 1
Introducción

En este documento se presenta la investigación de la cual se desarrolló el proyecto de la presente la tesis y fue realizada en el Centro de Investigación de Matemáticas (CIMAT) de Guanajuato.

El objetivo del proyecto es resolver la problemática de la densidad de pasajeros en el transporte público mediante herramientas y técnicas de Inteligencia Artificial, dado que en la actualidad la demanda de transporte público ha incrementado por ello, la ocupación del espacio del vehículo escasea, dañando las unidades y creando desorden y caos en tiempos de espera, llegada y calidad del servicio, para ello se describe el proceso que fue llevado a cabo para la resolución de dicha problemática, lo cual forma parte de un proyecto de mayor alcance que consiste en la automatización del servicio de transporte público. Como primera instancia es importante entrar en el entorno del estado del arte, el aprendizaje de nuevas herramientas como lenguajes de programación en este caso Python junto con el conocimiento básico sobre los temas de: aprendizaje máquina y redes neuronales. Estos son relevantes para la investigación, de manera que al menos las actividades más destacables y puntos clave para entender el proceso son descritas en el documento. La metodología consta del uso de un conjunto de materiales (conjunto de datos, máquinas virtuales, etc.), y una fase de experimentación, de la que se esperaban buenos resultados. La experimentación es la etapa más importante de la investigación creando así posibilidades de aprovechamiento y posibles soluciones e implementaciones, que son descritas y detallas una a una. Se espera que, en este documento, quede plasmado el trabajo y dedicación puesta en el proyecto, asi como el compromiso de la obtención de resultados positivos. Se pretende que este proyecto impacte positivamente a la sociedad, tomando problemáticas de lo cotidiano como oportunidades de aprovechamiento y resolviéndolas con herramientas como la Inteligencia Artificial que tiene

implementaciones ilimitadas. Aprovechando así todo el potencial que la tecnología y el conocimiento ofrece, convirtiendo de actividades cotidianas en avances tecnológicos.

Planteamiento del problema

1.1. Identificación.

El servicio de transporte urbano es uno de los más utilizados en todas las poblaciones del país, sin embargo, sigue presentando algunas deficiencias para los usuarios, siendo las principales, largos tiempos de espera y saturación de las unidades de transporte. Ante esta situación que trasciende a todos los usuarios, cabe destacar que los adultos mayores y personas con incapacidad física son las más afectadas. Por otro lado, puede afectar en la rutina de trabajo de algunas personas o estudiantes.

No solo hay déficit en el servicio como tal, sino que también en la parte administrativa existen algunos problemas como: mal cobro, robo de dinero, unidades que pueden ser dañadas por el excesivo peso, debido a que se somete a una carga excesiva por la falta de control que se tiene al subir cierta cantidad de pasajeros. Todo en su conjunto hace que el transporte público carezca de calidad y eficiencia, lo cual no solo afecta a las personas que administran tal servicio si no a la población que hacen uso de él.

1.2. Justificación.

El proyecto en general pretende administrar de manera eficiente el servicio de transporte tanto para usuarios como para los concesionarios de mismo, la parte que corresponde a esta tesis se enfocará principalmente a que los usuarios conozcan en tiempo real, la hora de llegada de las unidades a paradas oficiales y su porcentaje de ocupación; de esta manera el usuario podrá administrar mejor su tiempo de traslado y abordar la unidad que más le convenga, con la seguridad de que estará

el camión a tiempo y con disponibilidad de asientos. El problema mencionado ya ha sido el foco de atención en algunos países de primer mundo, como Japón, Suiza, Estados Unidos y Alemania, en donde el sistema de transporte se encuentra lo suficientemente automatizado como para ofrecer a los usuarios una precisión de segundos de sus llegadas, tiempos de espera y tiempos de recorrido. Sin embargo, ese no es el caso en la gran mayoría de las poblaciones mexicanas.

La meta principal del presente proyecto es obtener un balance entre saturación de las unidades y tiempos de espera por parte de los usuarios para que los coordinadores del servicio público reaccionen en tiempo real a las necesidades de la población, ya sea aumentando cuando se requiera el número de unidades en circulación, o bien disminuyendo el número sin afectar severamente el tiempo de espera por la unidad.

1.3. Alcance.

El alcance esperado es tener una aplicación tanto móvil como web a futuro, así como todo el conjunto de sus funciones que se espera sea utilizado primeramente en el estado de Guanajuato, aunque dada la identificación de varias ciudades a lo largo del país donde existe la misma problemática, se espera que su alcance sea nacional.

Objetivos

1.4. Objetivo General

Implementar un esquema basado en aprendizaje máquina para detectar y cuantificar los usuarios en las unidades de transporte urbano a través de imágenes digitales.

1.4.1 Objetivos específicos

- Integración e introducción al Aprendizaje Máquina
- Obtención de datos de entrenamiento
- Entrenar en redes neuronales enfocadas al problema
- Interpretación de datos para la elección de la mejor solución al problema

Hipótesis

1.5 Hipótesis

Mediante el empleo de redes neuronales y procesamientos de imágenes digitales es posible hacer un conteo de densidad de pasajeros en el transporte público.

Capítulo 2
Fundamento Teórico

2.1 Antecedentes

El proyecto de la automatización del servicio de transporte público se ha desarrollado con el pasar de los años, siendo este una propuesta en respuesta a una problemática previamente observada de la cual se determinó un punto de aprovechamiento. Actualmente cuenta con una aplicación móvil desarrollada en Java mediante el IDE Android Studio, la cual satisface la parte del usuario, hasta el momento cuenta con la funcionalidad de mostrar en tiempo real la unidad que se desea visualizar, la ruta que sigue, los tiempos de espera e incluso poder poner alarmas o recibir notificaciones del transporte deseado, así como algunos detalles de evaluación de app.

En cuanto a la parte que corresponde a los concesionarios, quienes son los proveedores del servicio, se contaba al menos con un modelo o mejor dicho algunos diseños previos, los cuales simulaban lo que sería la vista del concesionario y lo que podría visualizar, tales como: la ruta que sigue el camión, el chofer, las estadísticas generales (velocidad, accidentes, horario, cambios de ruta, entre otras). Además de asignar un chofer a una unidad, una ruta y un horario.

El funcionamiento estaba planteado de la siguiente forma: se tenía que un dispositivo enviaba las señales de GPS y demás sensores mediante señales móviles de LTE, y el chofer registraba sesión e inicio de actividades mediante otra aplicación móvil (exclusiva para el chofer), en este momento se planean mejores formas de realizar esta actividad, mientras que la parte del dispositivo no cambio se realizó una versión más completa y estable.

Inicialmente se tenía planeado que el sistema estuviera presente en diferentes ciudades del estado de Guanajuato, pero se espera que mediante su desarrollo pueda crecer la idea y llevarlo a toda la república.

2.2 Marco Teórico

2.2.1. Aprendizaje Máquina

"El Aprendizaje Máquina (Machine Learning) es un marco algorítmico unificado para identificar modelos computacionales que actualmente describen datos empíricos y los fenómenos subyacentes" (Rosebrock, 2014). Suele utilizarse para enseñar a las computadoras a realizar una alta gama de tareas útiles como es: detección de objetos (en nuestro caso), pero también reconocimiento de voz, descubrimientos en medicina, predecir análisis, entre otros.

2.2.1.1. Fases

El aprendizaje maquina consiste en enseñarle a las computadoras, así como a los humanos a realizar ciertas tareas específicas que se logran a base de un tipo de entrenamiento. Esto se puede distinguir en las siguientes fases:

1. Recopilación de datos: Se recopilan datos de entrenamiento en lotes, estos contienen imágenes de las cosas que se desean reconocer, por ejemplo, podemos diferenciar o identificar patos de gallinas (Watt, Borhani, & Katsaggelos, 2020). Entre más grande sea el conjunto de pruebas (training set) la computadora hará mejor la tarea de aprendizaje.

Imagen 1 Ejemplo de conjunto de datos de imágenes

2. Diseño de características: Elegir las características que permitan describir lo que se desea conocer de manera que sean precisas, como por ejemplo reconocer la gallina del pato por la cresta o el tamaño del pico. Estas características ahora representarán un dato específico en este caso un punto o un número, los cuales se pueden ver representados en un plano gráfico, donde cada eje es la representación de la medida de la característica. En la *Imagen 2*, podemos apreciar cómo se han agrupado nuestros datos (imágenes) en base del diseño de las características que vamos a evaluar, agrupando a los patos separado de las gallinas.

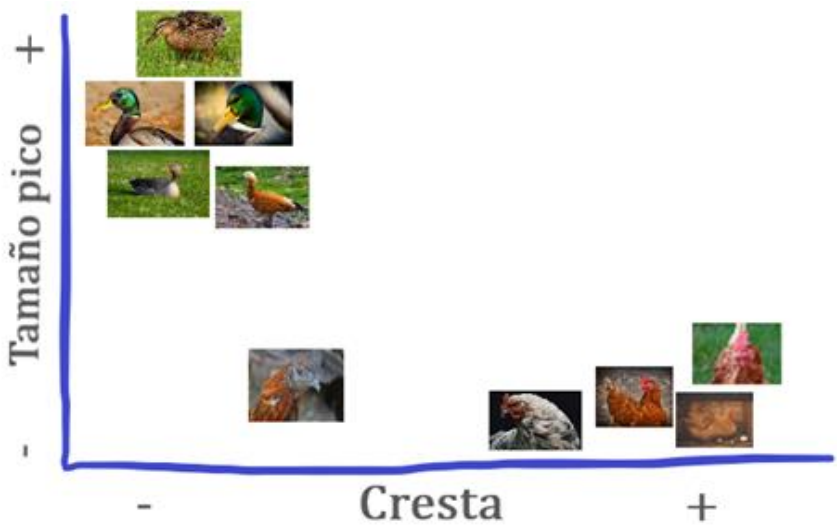


Imagen 2. Representación de agrupación por características

3. Modelo de entrenamiento: El paso siguiente consiste en separar estas características ya sea con una línea o una curva. En caso de usar la línea, los parámetros que se deben encontrar son la pendiente y la intersección vertical. Estos parámetros se obtienen mediante la optimización matemática y el ajuste de este conjunto de datos con el conjunto de datos de entrenamiento, se le conoce como modelo de entrenamiento. La separación del espacio de características mediante la línea o curva, le brinda al sistema una regla la cual le hace posible a la computadora determinar que es un pato o una gallina. En la *Imagen 3*, se representa al modelo puesto en marcha, la recta es el punto de ajuste que se convertirá en la nueva regla, para la cual todo aquel dato en la parte rosada es gallina y en la parte verde es pato.

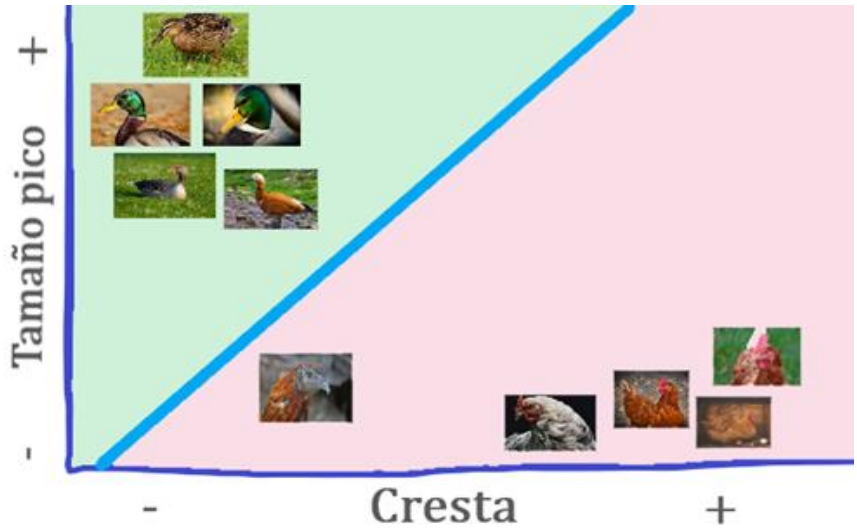


Imagen 3. Representación de modelo de entrenamiento.

4. Validar el modelo: Consiste en poner a prueba nuestro modelo base de nuevas imágenes o datos, los cuales pueden permitirnos ver el margen de error, en caso de existir algún margen de error muy evidente es conveniente crear un mejor diseño de características o agregar más, de manera que sea más estricto y describa cada vez mejor lo que se quiere distinguir, en el caso de patos y gallinas puede ser el tamaño de la cola, el color de los ojos, etc. Aunque esto haría que nuestro espacio de características crezca de manera que sea multidimensional.

2.2.1.2. *Tipos de aprendizaje*

Como se mencionó anteriormente, al crear el modelo y delimitar con una recta o curva, creamos una regla que ayuda al aprendizaje, del cual se divide en dos tipos:

1. Aprendizaje supervisado: Es el aprendizaje automático de reglas que contienen entradas y salidas (input/output). Este a su vez puede ser de dos maneras:

a) *Regresión*: Emplea los mismos pasos antes descritos, pero a cada entrada le corresponde una salida, por ejemplo: en inversiones, la entrada

es el dinero aportado y la salida el dinero resultante, en este caso sus características pueden ser tiempo, precio del dólar, banco, etc. Y para conectarlos podemos entrenar un modelo lineal simple o una línea de regresión usando nuestros datos de entrenamiento. Un ejemplo de esto se muestra en la *Imagen 4* donde la gráfica representa un punto sobre un modelo de regresión lineal, es decir una predicción, lo cual nos indica una posible ganancia neta del dinero que inviertas con respecto a la ganancia por inversión.

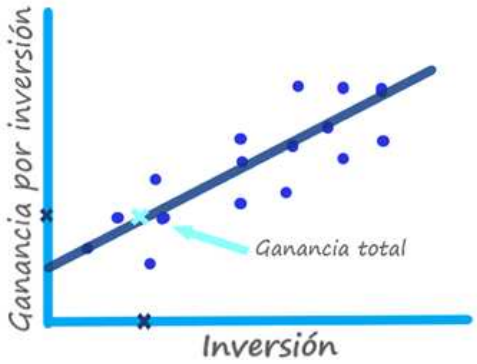


Imagen 4. Gráfica de regresión.

En conclusión, la regresión consiste en el ajuste de un modelo a un conjunto de datos de entrenamiento para que las predicciones de un proceso de valor continuo sean más exactas.

b) *Clasificación*: Similar a la regresión en las etapas iniciales, solo que, en lugar de predecir una salida de un valor continuo, la predicción toma clases o valores discretos. Si lo llevamos a un plano de dos dimensiones, lo que la clasificación intenta es separar los datos mediante una línea o más comúnmente una curva, como el ejemplo de las gallinas y los patos. Esto es gracias a que tenemos un diseño de características muy específico. En la *Imagen 5*, representa un caso de agrupación donde tenemos dos grupos que representan datos específicos y mediante la línea se pretende separarlos, creando así una regla computacional.

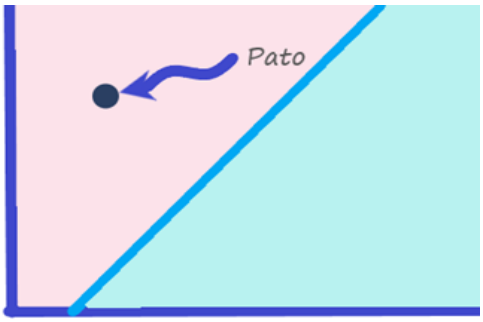


Imagen 5. Representación de agrupación.

2. Aprendizaje no supervisado: A diferencia del lenguaje supervisado este únicamente describe los datos de entrada. Algunas veces se usa para simplificar los datos que permiten el aprendizaje supervisado. Sus dos principales funciones son las siguientes:

a) *Reducción de dimensión*: Algunos tipos de datos como las imágenes a color o videos, poseen una gran cantidad de dimensiones lo cual puede afectar el funcionamiento del modelo. “Esto se logra aplastando o proyectando hacia abajo en una línea o curva de dimensión inferior adecuada (o más generalmente un hiperplano lineal o una variedad no lineal), preferiblemente uno que retiene tantas de las características definitorias de los datos originales como sea posible” (Watt, Borhani, & Katsaggelos, 2020). La *Imagen 6* muestra que los puntos celestes son una representación de datos en un plano de dos dimensiones, al hacer la reducción de dimensión se crea una sola línea y ahora los datos de las dos dimensiones forman una sola recta formada por sus “proyecciones” (puntos azules), creando ahora un conjunto de datos de una sola dimensión, una recta.

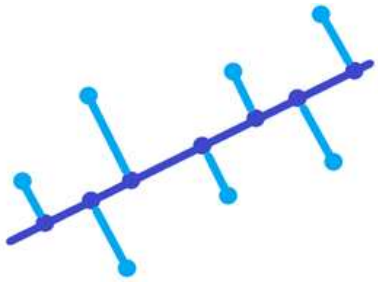


Imagen 6. Datos de dos dimensiones reducidos.

b) Agrupación: identifica la estructura subyacente bruta en un conjunto de datos de entrada mediante agrupar puntos que comparten alguna característica estructural de acuerdo a (Watt, Borhani, & Katsaggelos, 2020). Como por ejemplo la proximidad, si un grupo de datos tiene una cercanía marcada se pueden agrupar. En la *Imagen 7*, podemos ver cómo agrupar aquellos datos que poseen características similares, lo cual nos facilita el procesamiento.

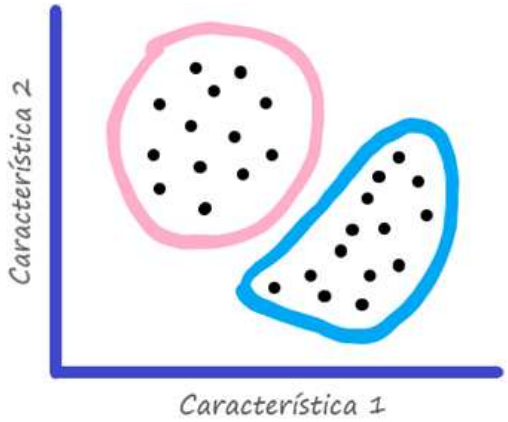


Imagen 7. Agrupación de datos/objetos

2.2.1.3. Optimización matemática

Se puede formalizar la búsqueda de los parámetros de un modelo de aprendizaje mediante funciones matemáticas bien definidas. Estas funciones son comúnmente conocidas como *costo* y *función de pérdida*, que tomando un conjunto de parámetros de un modelo y retornando valores nos da una noción de que tan bien

realiza la tarea de aprendizaje usando los valores seleccionados. Esto es, si el valor es alto tenemos rendimiento bajo, mientras que su opuesto es un valor bajo. En una predicción lineal los parámetros que se deben ajustar son la pendiente y su intersección vertical, dicho en otras palabras, este ajuste se logra encontrando el conjunto de datos que nos den un mínimo.

Conceptualizando, un bajo valor corresponde a un modelo de alto rendimiento en ambos casos de la regresión y la clasificación siempre apuntamos en minimizar el costo de las funciones en orden de encontrar los parámetros ideales para nuestros modelos de aprendizaje. Todo esto se logra usando las herramientas de la optimización matemática. Incluso la optimización aplica para modelos no lineales. Por tanto, la optimización matemática tiene un rol importante en los modelos de aprendizaje.

2.2.2. Procesamiento de imágenes

El procesamiento de imágenes se forma por una serie de conceptos, es importante conocerlos todos para poder comprender todo su desarrollo. Debemos conocer la estructura, forma y demás conceptos que ayuden a comprender lo que es la imagen, los cuales se explicaremos a continuación.

2.2.2.1. Estructura de la imagen

Una imagen está conformada por un grupo de pixeles. Estos son el crudo o la unidad de bloque de una imagen. Normalmente la resolución de nuestra imagen no es más que el indicativo de cuantas columnas por cuantas filas de píxeles es nuestra imagen. Pueden estar representadas de dos maneras: escala de grises y color.

a) *Escala de grises*: En esta escala cada pixel contiene un valor de 0 o 255, donde 0 es negro y 255 es blanco.

b) Color: Normalmente representado en RGB, (Red, Green, Blue / Rojo, Verde, Azul), cada color es representado en un rango de 0 a 255 que indica que tanto de ese color contiene, estos se representan en una tupla.

Es también descrita como una representación gráfica de los valores de intensidad, obtenidos a través de un sensor por ejemplo cámaras, celulares, entre otros.

Los valores de intensidad se almacenan en una matriz con base a su resolución y tipo de imagen, es decir, imagen a escala de grises, a color o con transparencia. Si tenemos una imagen en escala de grises con una resolución de 300 x 500 píxeles, tendremos una matriz de 300 x 500 x 1, en cambio sí tenemos una imagen a color de la misma resolución, obtendremos una matriz de 300 x 500 x 3, y por último si tenemos una imagen con transparencia tendremos una matriz de 300 x 500 x 4.

Por ejemplo, si tenemos la *Imagen 8* de 8 x 8 píxeles en escala de grises tendremos una matriz como la siguiente:

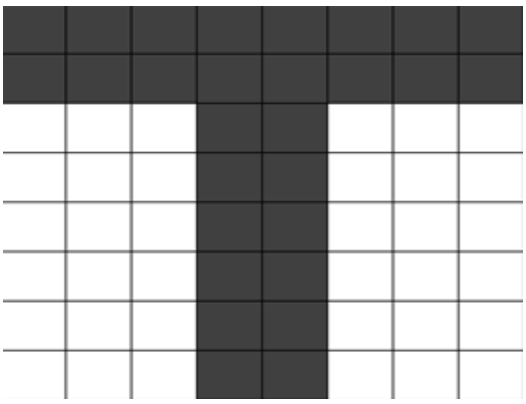


Imagen 8. Representación de una imagen en pixeles.

Resumiendo, tenemos que las imágenes independientemente de su tipo es un sistema de coordenadas, representado como una matriz (*Imagen 9*), que puede ser

multidimensional, donde cada campo contiene un valor dependiendo del tipo de imagen, que representa cada píxel.

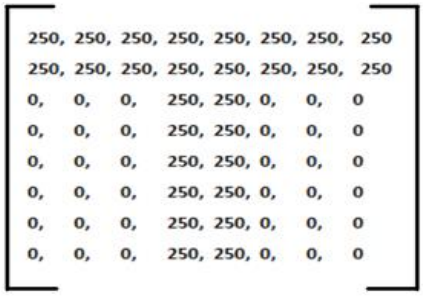


Imagen 9. Representación en matriz.

2.2.2.2. Espacios de color

Estas son unas herramientas muy útiles en el procesamiento digital de imágenes ya que permiten analizar desde diferentes puntos de vista un pixel y aprovechar de la mejor manera la información que nos da. Existe una gran variedad de modelos, pero en este caso sólo mencionaremos aquellos que nos son útiles o bien podemos aprovechar en la resolución del problema.

Modelo RGB: Como ya se mencionó anteriormente las imágenes con este modelo contiene 3 planos de imagen, uno por color, el cual puede ser representado por un cubo donde cada arista es un color primario, secundario (Cian, magenta, etc.) y blanco y negro.

Modelo HSV (HSB): Las siglas representan tono (Hue), saturación (saturation) y valor/brillo (value/brightness). A diferencia del RGB este es representado por un cono o pirámide hexagonal, donde el valor depende de la altura, la saturación del vértice y el tono del área (de área hexagonal).

2.2.2.3. Conceptos básicos de procesamiento

El procesamiento de imágenes se encarga de extraer datos o mediciones que una imagen pueda tener. Todo esto incluye los métodos que facilitan su interpretación o su extracción de datos que las imágenes puedan tener. La entrada siempre será una imagen mientras que la salida será un valor numérico que contenga toda la información que contiene la imagen. Para poder lograr este resultado la imagen primero debe pasar por algunas etapas de filtrado y procesamiento. Existen tres métodos del procesamiento de imágenes como son: de dominio espacial, de características y de frecuencia. Ahora solo nos enfocaremos en el dominio espacial ya que esta se encarga de analizar pixel por pixel, y como sabemos una imagen puede ser presentada como una matriz de "x" dimensiones, de la cuales cada campo contiene la información de un píxel, esto en machine learning puede ser nuestro conjunto de datos procesar en lote.

2.2.2.4. Filtros

Los filtros no son otra cosa más que una especie de capa que combina las características de un pixel con su píxel vecino, mediante algunos procesos matemáticos, este en base a una máscara y determina un kernel. La subimagen que resulta de la combinación se denomina filtro. Los "pixeles" en una subimagen de un filtro se denominan coeficientes, estos coeficientes no hacen más que un promedio del valor del píxel procesado de la imagen a aplicar el filtro, por los coeficientes del este. "El proceso de aplicar un filtro a una imagen consiste simplemente en mover el filtro de un punto a otro en una imagen. En cada punto (x, y), la respuesta del filtro en ese punto se calcula utilizando una relación predefinida" (Woods & González, 2002), es decir y como se explica antes, el punto es el pixel y este se modifica usando como base los coeficientes (pixeles) que usará el filtro. El kernel se determina por la suma de productos de los coeficientes del filtro y los píxeles de imagen original que están en el área abarcada por la máscara, entiéndase como mascara a aquella área de pixeles de donde el filtro se basa, del filtro. En la *Imagen 10* podemos ver como a un segmento de una imagen se le aplica

un filtro y este a su vez le aplica una operación con sus coeficientes para crear una versión o matriz nueva de esa imagen.

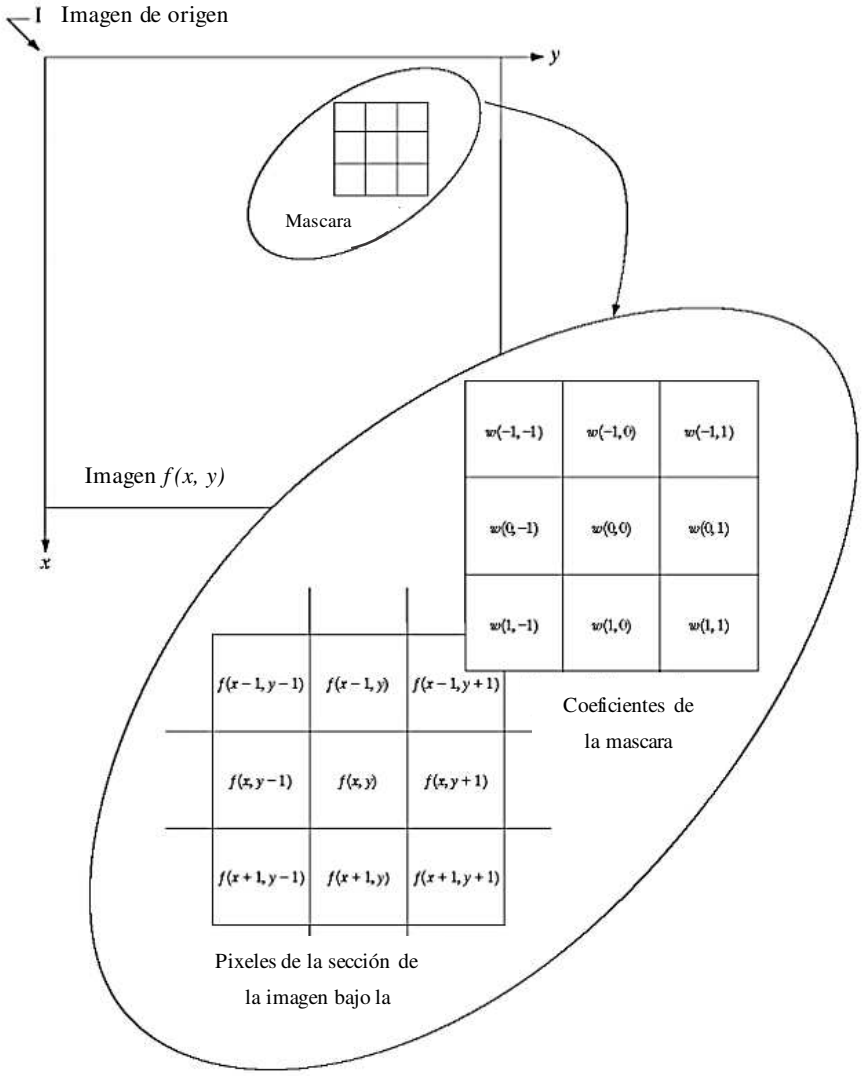


Imagen 10. Representación de un filtro. Imagen obtenida de (Woods & Gonzalez, 2002)

Entrando en contexto a los filtros, existen o se pueden clasificar de dos maneras:

Suavizado (Smoothing): Desenfocan y reducen el ruido de una imagen, se entiende por ruido. Esto es que se suavizan los bordes de una imagen o bien hacer menos notables los cambios de intensidad. Existen algunos filtros de suavizado como

- Filtro Media: La salida es un kernel con coeficientes que son el resultado de realizar la media aritmética de la sección de máscara elegida.
- Filtro Gaussiano: Es similar al anterior, pero a diferencia que en este los píxeles vecinos al pixel de referencia tienen mayor peso que los exteriores. Esta se calcula mediante la campana de Gauss.

Desenfoque (Blurring): Este se usa especialmente antes del procesamiento, elimina los pequeños detalles que estén mal antes de la extracción de cualquier objeto. Y al contrario de la anterior esta destaca los bordes y áreas con mayor intensidad. El resultado se logra sumando el Laplaciano a la imagen original. Este está definido por la ecuación 1, donde I representa la intensidad:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Ecuación 1. Formula del Laplaciano.

Siendo los filtros herramientas muy útiles en el procesamiento de imágenes, actualmente existen muchos más creados por diferentes personas, asi como con las herramientas creadas como Python y sus librerías especiales para esto como es TensorFlow, al momento de querer procesar una imagen o extraer sus características, esta paquetería mediante sus métodos son capaces de aplicar filtros de manera rápida y simultánea, e incluso en una gran cantidad y solo aquellos que crea más aptos para obtener las mayores características posible, es decir, ya poseemos una herramienta que lo haga automáticamente sin necesidad de aplicar cada uno de ellos y determinar cuál es el mejor. La cantidad de filtro depende del resultado que se quieran obtener o de la herramienta que se use, por ejemplo, en una sola neurona se pueden aplicar 5 filtros diferentes para un conjunto que necesite varias correcciones o que estén muy ambiguas las imágenes, o puede no aplicar ninguno, esto claro al criterio de la herramienta que se use.

2.2.2.5 Reconocimiento de objetos

El reconocimiento de objetos mediante procesamiento de imagen posee muchos métodos y muy variados, algunos de ellos te ofrecen valores cuantitativos y otros cualitativos, todo depende de las necesidades o el tipo de información que se desea obtener. Dicho esto, para nuestro proyecto se usarán las redes neuronales, como un método de reconocimiento de objetos basados en métodos de decisiones. Los cuales se basan en los patrones y clases de patrones, donde los patrones es un vector, cadena o árbol que contiene características y una clase es un conjunto de patrones que comparten características. En el apartado de redes neuronales se explicará de mejor manera el cómo es el proceso de reconocimiento de objetos.

2.2.3 Redes Neuronales (Deep Learning)

Una neurona a nivel biológico, es una estructura nerviosa de nuestro cerebro que, mediante impulsos eléctricos, procesa, guarda y aprende información. En redes neuronales, la neurona es la unidad básica de la estructura siendo esta una formación similar a la estructura biológica, ya que cuenta con terminales de entrada y terminales de salida con la información procesada, siendo también capaz de propagarse, o mejor dicho pasarla a otra neurona. En la *Imagen 11* vemos una neurona donde $\mathbf{x_n}$ son los datos de entrada, se procesan dentro de la neurona y retorna una salida **y.**

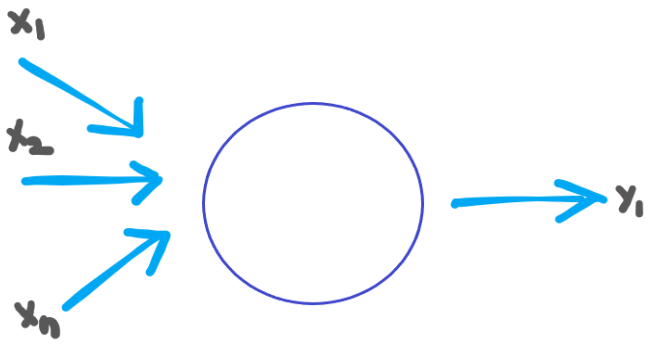


Imagen 11. Estructura de una neurona.

Tanto en la ciencia como en el cuerpo humano, una neurona necesita ser activada para ellos existe la siguiente formula de activación (*Ecuación 3*), donde w es el peso (pendiente), T es la traspuesta, δ (sigmoide) es la función de activación compuesta por el valor de la *Ecuación 2* y b es la constante de la fórmula de la recta o en este caso llamado bias:

$$\delta = \frac{1}{1 + e^{-x}}$$

Ecuación 2 Valor de la sigmoide.

$$y_i = \sigma(w^T x_i + b)$$

Ecuación 3 Fórmula de activación.

"Las redes Neuronales son modelos computacionales que surgieron como un intento de conseguir formalizaciones matemáticas acerca de la estructura del cerebro" (Nielsen, 2019). En otras palabras, las redes neuronales son la imitación de un cerebro humano que consigue o extrae el conocimiento en base a sus experiencias o aprendizajes. Dicho esto, tenemos entonces que una red neuronal es la agrupación de muchas neuronas conectadas que tienen entradas y salidas. Sí las entradas y salidas están en la misma dirección se considera que esas neuronas están en la misma capa.

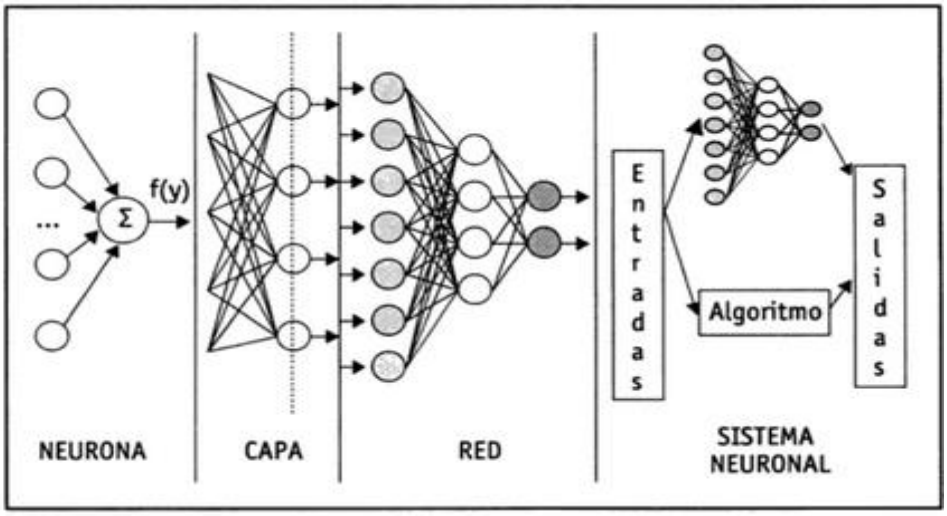


Imagen 12. Representación de una red neuronal.

Con respecto a las redes neuronales "Pueden considerarse modelos de cálculo caracterizados por algoritmos muy eficientes que operan de manera masiva paralela y permiten desarrollar tareas cognitivas como el aprendizaje de patrones, la clasificación o la optimización" (Flórez López & Fernández Fernández, 2008). Teniendo el conocimiento básico, se tiene que las redes neuronales son un conjunto de neuronas que se comunican entre sí, donde:

1. La misma entrada puede alimentar a varias neuronas por capa
2. La salida de una neurona puede ser la entrada de otras neuronas en otras capas.

Dicho esto, tenemos que una red neuronal aprende a mejorar las salidas al actualizar los pesos de las conexiones que tiene internamente. La salida de una neurona puede funcionar de entrada para la siguiente capa de neuronas ajustar el valor y su salida nuevamente servir de entrada para la siguiente capa.

Como se ha mencionado la red neuronal aprende conforme obtiene experiencia, para ello existen la fase:

Fase de aprendizaje: Este parte de un modelo determinado de una neurona que va otorgando pesos aleatorios o que también pueden ser nulos. Este modelo es el que

se entrena para poder resolver la problemática que se desea, este ajuste se puede llevar a cabo mediante una fórmula de pesos. Esta fórmula o función tiene como objetivo determinar el error que tiene nuestro modelo, buscando siempre un mínimo.

Para la determinación de estos pesos existen dos tipos de reglas de las cuales ya se habló que es el aprendizaje supervisado y no supervisado, la diferencia radica en que al primero un agente externo lo "controla" el aprendizaje de la red mientras que el segundo es autónomo capaz de tomar daciones el solo y seguir el mejor camino posible.

Dentro de las redes neuronales existen dos tipos, las de regresión lineal y las de regresión logística, ambas con diferencias muy marcadas como sus funciones de peso y costo. Además, es importante conocer el proceso matemático, durante este proceso es primordial aprender al menos los conceptos básicos como el hecho de que la regresión lineal intenta ajustar la pendiente y la intersección vertical (*w(T) b* en nuestra función de activación de la neurona) para tener una predicción más acertada de lo que queremos obtener, teniendo una función de costo que nos permite saber el margen de error, mientras más bajo es el costo más confiable es nuestro modelo. En la *Imagen 13* está la representación de una neurona de regresión lineal, está compuesta por un conjunto de datos (x_i) y la respectiva función de activación ($\hat{y}$) es igual sumatoria de la transpuesta (w^T) de los pesos (x_i), por lo que esto permite mantener una línea recta al momento de graficar los resultados.

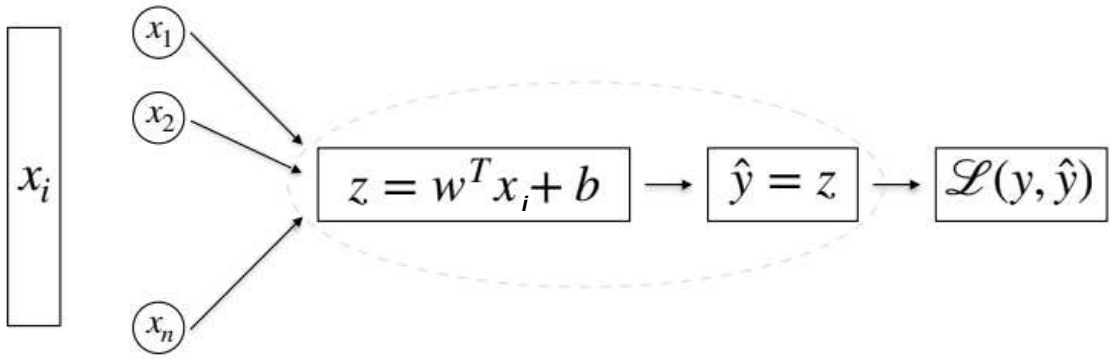


Imagen 13. Representación de una neurona de regresión lineal

Fuente: Elaborado por el Dr. Odín Eufrasio, instructor del curso de redes neuronales.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}_i, y_i) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}(\hat{y}_i - y_i)^2$$

Ecuación 4 Función de costo de neurona de regresión lineal

Por otro lado, en regresión logística tenemos una curva en lugar de una recta, de igual manera funciona para predecir eventos. Al igual que la regresión lineal cuenta con una función de activación y una función de costo. En la *Imagen 15* está la representación de una neurona de regresión logística, conteniendo los mismos componentes que en la *Imagen 13* a diferencia que tenemos a sigmoide (δ) como parte de la función de activación, donde tenemos que sigmoide está definido por la representación de la *Imagen 14*.

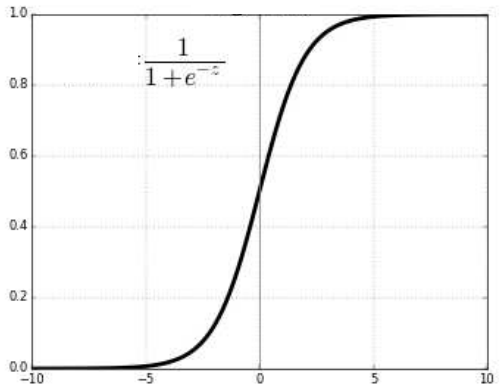


Imagen 14 Representación gráfica de la sigmoide

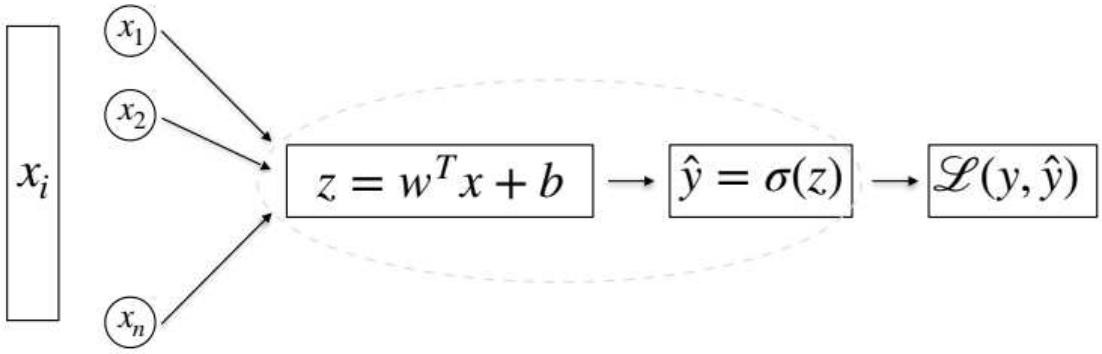


Imagen 15. Representación de una neurona de regresión logística.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}_i, y_i) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

Ecuación 5 Función de costo donde δ es 1/1+ez

Existen diferentes tipas de funciones de activación, tal como la ReLu (función en base a máximos), sigmoide y tanh (función en base a la tangente hiperbólica), cada una con sus características particulares, el uso de cada una depende de las necesidades de cada problema.

2.2.3.1. Redes Neuronales Convolucionales

Las Redes neuronales comunes están todas interrelacionadas entre neuronas, lo que provoca que su aprendizaje sea más lento y menos optimizado, por lo que basado en la corteza visual de un animal se propusieron las Redes Neuronales Convolucionales (CNN) se propuso que utilizaran la estrategia de reparto de peso para explotar estructuras similares ocurridas en diferentes ubicaciones de una imagen. Al compartir los pesos convolucionales localmente para una imagen completa, esto reduce drásticamente la cantidad de parámetros que deben aprenderse y hace que la red sea equivalente con respecto a las traducciones de la entrada (Xiaoyue , Abdenour , Yanwei , & Granger, 2019).

En la *Imagen 16* está la estructura de las Redes Neuronales Convolucionales que consta de capas convolucionales que son los componentes básicos de una CNN.

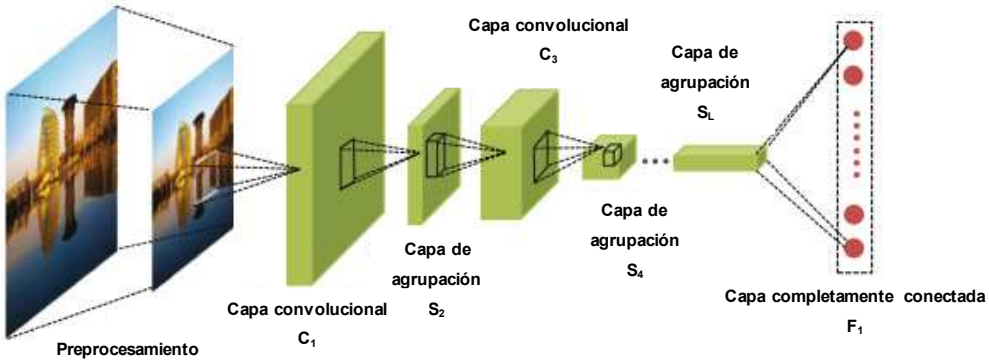


Imagen 16 Estructura de una CNN

Fuente: (Xiaoyue , Abdenour , Yanwei , & Granger, 2019)

En cada capa convolucional, los datos de entrada se convolucionan con un conjunto de K núcleos $W = \{W_1, W_2, \ldots, W_k\}$ sumando por sesgos $b = \{b_1, b_2, \ldots, b_k\}$. Luego, se genera un nuevo mapa de características X_k ingresando los resultados de la convolución en una función no lineal de elementos σ. (Xiaoyue , Abdenour , Yanwei , & Granger, 2019) Dado el vector de salida de la x_z capa, el y_z mapa de características se calcula mediante la *Ecuación 6*.

$$X_k^{l+1} = \sigma\left(W_k^l X^l + b_k^l\right)$$

Ecuación 6 Cálculo del vector de salida

En cuanto al modelo que usaremos tenemos dos de ellos:

- Faster R-CNN ResNet50 V1 640x640: una arquitectura dedicada a la detección en múltiples escalas, mediante dos generadores de regiones de interés dedicados a peatones a corta y larga distancia.

- EfficientDet D1 640x640: una familia de detectores con una precisión y eficiencia significativamente mejores en un amplio espectro de limitaciones de recursos. Los detectores EfficientDet son detectores de un solo disparo. Los pesos de la red de clase y caja se comparten en todos los niveles de funciones.

2.2.4. Lenguaje de Programación Python

"Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas." (Rossum, 2016)

Como se dice Python es un lenguaje muy poderoso, que, si bien es fácil de aprender y de usar, es un verdadero lenguaje de programación, en el que puedes realizar operaciones muy complejas y este te dará soporte, además de que para cosas matemáticas es el lenguaje indicado. En este caso, para la ejecución de modelos fue el lenguaje ideal, ya que posee un par de módulos o paquetes como son:

NumPy: Es una librería de Python con muchas funciones de algebra, algebra lineal, cálculo matricial y transformadas de Fourier.

TensorFlow: Librería de código abierto especializada en el aprendizaje automático, permite a los sistemas entrenar y construir redes neuronales para detectar patrones

Capítulo 3
Procedimiento

El CIMAT es un centro de investigación enfocado en las matemáticas, ciencias de la computación y física en Guanajuato, Gto., donde se llevan a cabo diversos proyectos en áreas de Matemáticas, Estadística y Computación, además de ofrecer posgrados en estas mismas áreas. En esta institución se llevó a cabo el proyecto ya mencionado, por parte del equipo de investigación del Dr. Rogelio Hasimoto Beltrán del área de Ciencias de la Computación en colaboración con el consorcio de Inteligencia Artificial (IA). Dicho proyecto contaba con un desarrollo avanzado a lo largo de aproximadamente 3 años, pero estaba aún en etapa de desarrollo, y contaba con una aplicación móvil funcional, así como el inició del desarrollo de una página web de administración para los concesionarios del proyecto.

Se trata de una investigación aplicada, la cual, tuvo varias etapas tales como: identificación del problema seguido de la obtención del conjunto de datos o imágenes a través de una investigación de campo con la captura de videos en el transporte público de la ciudad de Guanajuato. Posteriormente, se plantearon 4 experimentos diferentes con modelos que podrían resolver nuestra problemática, los cuales fueron entrenados por una red neuronal, para finalizar con un análisis de los resultados obtenidos en cada experimento. Una vez que todas estas etapas fueron realizadas se obtuvieron resultados que discutiremos posteriormente, en la *Imagen 17* tenemos la representación en orden cronológico de las distintas etapas realizadas.

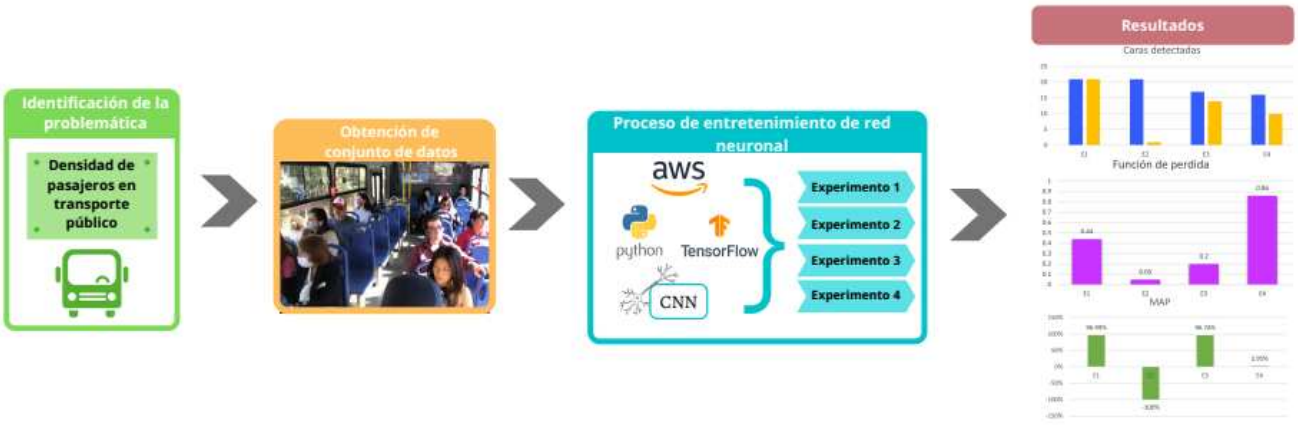


Imagen 17 Representación del proceso de resolución del problema

3.1. Integración e introducción al Aprendizaje Máquina

La integración al proyecto fue mediante una serie de cursos que echaron a andar las pautas para poder resolver el objetivo principal, estos pusieron en contexto las herramientas que se usarían en la resolución de la problemática una de estas fue Python, el lenguaje de programación que se usó para desarrollar todo el código. Claramente al hablar de la problemática se habla del conteo de las personas mediante procesamiento de imágenes, las cuales podemos entender matemáticamente como una representación en forma de matriz donde tenemos una escala donde 0 representa en la matriz un pixel con poca saturación (en escala de grises es negro) y el 255 gran saturación (en escala de grises es blanco) y cada elemento dentro de ella representa un pixel, esto procesado mediante Aprendizaje Maquina (Machine Learning) por medio de un modelo matemático que se ejecutará en Python. En otras palabras, haremos uso de los recursos que nos ofrece Python en redes neuronales, para poder procesar imágenes tomadas de un previo muestreo del transporte público que nos sean útiles para identificar a las personas y tener un conteo de ellas.

3.1.1. Aprendizaje profundo

Para tener el conocimiento pertinente sobre las redes neuronales, se inició un curso sobre ellas. Su principal objetivo era introducirnos a las redes neuronales, teniendo énfasis en su composición y su comportamiento. Se comenzó con la básico, conociendo lo que son las neuronas, su funcionalidad y estructura, para ir avanzando poco a poco. Todo ello encaminado a la problemática, para ello se optó por usar redes de neuronales convolucionales (CNN), debido que al procesar una imagen era la mejor manera de hacerlo, por la cantidad de datos que contiene y además de la cantidad de filtros que usa automáticamente, en compañía de esto se seleccionó la función de activación conocida como sigmoide, durante ese periodo de tiempo se hicieron experimentos previos de ensayo en las propias maquinas, como una introducción y adaptación a lo que sería la practica oficial.

Otras de las herramientas se usaron fueron los modelos pre-entrenados EfficientDet y Faster R-CNN para imágenes de una resolución de 640x640 pixeles, que nos ayudaron en el entrenamiento de la red neuronal en conjunto con Python.

3.2. Preparación de materiales y entorno de desarrollo

Durante esta fase se comenzaron las pautas de cómo se trataría de resolver la problemática, se pusieron en marcha los conocimientos obtenidos y se realizaron los experimentos mediante el uso del procesamiento de imágenes digitales y el uso de las redes neuronales, con lo cual se pudo tener las pruebas y bases para una solución a la problemática.

3.2.1. Investigación de campo: Recolección de imágenes y video en el transporte público

Como se menciona el principal objetivo del proyecto es poder tener un estimado de la densidad de las personas que están a bordo del transporte público. Siendo como primer paso, una investigación de campo a bordo del transporte público, la cual consiste en monitorear todo el recorrido de una ruta, tomando videos en ángulos

específicos que permitieran grabar a las personas tanto dentro como en la entrada del autobús. En la *Imagen 18* e *Imagen 19* tenemos un ejemplo de la colocación de las cámaras en el transporte público que hicieron la grabación de donde se obtuvieron las imágenes.

Imagen 18. Ejemplo de cámaras en el transporte público (Entrada)

Imagen 19. Ejemplo de cámaras en el transporte público (Atrás)

El objetivo de estas imágenes durante el recorrido es tener material para comenzar a realizar experimentos de implementación y pruebas, además de que sean de utilidad para la creación del conjunto de datos. Esto se puede lograr tras descomponer los videos en una serie de imágenes, que a su vez serán procesadas mediante procesamiento de imágenes tomando como referencia un fondo (camión vacío) de varios puntos, y este a su vez sometiendo los datos obtenidos a una red neuronal programada en Python con ayuda de TensorFlow.

3.2.2. Obtención de las imágenes y creación de los conjuntos de datos

De manera general conocemos el concepto de un conjunto de datos, pero al momento de la programación no es tan simple como un conjunto de imágenes, si no que estas contienen marcadores que permiten identificar lo que queremos reconocer o clasificar, estos marcadores son contenidos en un script (código de programación) de XML el cual mediante coordenadas de la imagen crea el área que contiene el objeto que se desea reconocer, y tambien los etiqueta de manera que se pueda reconocer que del pixel “a” al “b” del eje x, y, z es el área de cierto objeto con la etiqueta “Cara”. Para los conjuntos de datos se tomaron varias imágenes de distintas procedencias, esto con el fin de tener una variación, uno de los lugares donde se recopilo información fue la página de Kaggle.com, la cual provee de diferentes conjuntos de datos. En este caso, usamos uno que contuviera distintas cantidades de personas por imagen y lo interesante es que contenía no solo la característica de “Cara” si no que a su vez estaban clasificados con “mascarilla” y “sin mascarilla”, a grandes rasgos esto nos servirá dado al contexto actual de la vida diaria, es decir, ya que ahora por medidas de salubridad el uso de cubrebocas o mascarilla es de uso obligatorio, al hacer uso del transporte público las personas la usaran para subir, lo cual dificultara a al modelo identificar las “caras” asi que para poder cubrir tambien esta nueva característica a clasificar, se optó por usar conjunto de datos con gente usándola. Otro conjunto de datos se obtuvo desde una cuenta de GitHub que sin más que agregar solo etiqueta como “Cara”. Y por último el resultado de la investigación de campo, mediante la herramienta FFmpeg, que es

una herramienta de procesamiento de imágenes, se tomó el video y se descompuso en diferentes imágenes, obteniendo de un video de aproximadamente 5 minutos alrededor de 500 imágenes de las cuales solo se usaron en promedio de 250-300 y en algunos casos solo se usaron 50 de ellas, esto para variar en las experimentaciones. Pero como se comentó anteriormente un conjunto de datos no solo es el conjunto de imágenes como tal si no que se debe etiquetar y crear el documento XML correspondiente a cada imagen, entonces para adaptar correctamente a este nuevo conjunto de datos se usó una herramienta en línea llamada RoboFlow (roboflow.com), que permite crear tu propio conjunto de datos, etiquetarlo y aplicarle todas aquellas transformaciones que se requieran, y al final te genera el XML de cada imagen.

Para que se tenga una mejor idea de las características de los nuevos conjuntos de datos, se crearon dos derivados de los tres anteriores. Uno de ellos en al menos un 85% son imágenes de los encontrados en internet, y el otro 15% son imágenes sacadas de la unidad donde se grabó, mientras que el otro es lo contrario, esto porque se debe tener las dos perspectivas donde el modelo aprende de imágenes que serán más a pegadas a las que analizará y a la vez adaptarse a todo tipo de complexiones y ambientes.

Ahora que tenemos nuestros conjuntos de datos se inició la experimentación creando 4 grupos con diferentes las dos variaciones en el conjunto de datos y en la herramienta de Redes Neuronales.

3.2.3. Preparación del material y entorno de experimentación

Anteriormente se ha mencionado todo el proceso de las Redes Neuronales, asi como que los procesos de programación y matemático son de mucho costo computacional, por lo mismo un equipo convencional no sería de mucha ayuda, ni siquiera siendo un equipo de gama alta, el poder computacional que se necesita debe ser un poco más robusto si se quiere obtener resultados que sean confiables. Para ello se rentaron los servicios de Amazon AWS, tanto de almacenamiento como

de máquinas virtuales, dado que la memoria y procesador que se necesitan para hacer un entrenamiento y validación de una Red Neuronal son muy altas.

Las especificaciones de la máquina virtual son las siguientes:

- Seleccionar la opción de EC2 (Elastic Compute Cloud)
- La AMI (Amazon Machine Image): ami-03911bc455239df8d: Deep Learning Base AMI (Ubuntu 18.04) Version 32.0.
- Para todo el hardware (memoria y procesador): g4dn.xlarge

Los requerimientos establecidos son para una maquina con sistema operativo Linux, de tal forma que se puedan hacer unas 8 mil iteraciones sin que la memoria se llene o haya problemas de procesamiento. Para esto tambien se debe crear una red de conexión, con ello nos podemos conectar desde cualquier terminal a la máquina desde SSH con una IP privada asignada especialmente a la máquina.

Con el equipo listo y verificado que funcione correctamente, se preparó el ambiente y los recursos necesarios para poder hacer la experimentación. Como la base de nuestra experimentación es el lenguaje de Python se debe verificar que esté instalado y que la versión sea la correcta, puesto que lo siguiente es preparar la librería de TensorFlow es importante que tengamos el intérprete de disponible. Para hacer la instalación de la API de detección de objetos TensorFlow 2, se obtuvieron los comandos y pasos a seguir del repositorio en GitHub donde está al público por medio del siguiente enlace:

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2.md

Mediante el enlace anterior tenemos acceso a todos los recursos disponibles que nos ofrece de instalación, los comandos básicos son los siguientes:

- git clone https://github.com/tensorflow/models.git
- cd models/research
- protoc object_detection/protos/*.proto --python_out=.
- cp object_detection/packages/tf2/setup.py .

- /usr/bin/python3 -m pip install .

Estos comandos ayudan en la instalación del módulo con el que trabajaremos, aquel que nos ayudara a crear, entrenar y poner a prueba el modelo que usaremos para el reconocimiento de rostros llamado TensorFlow 2, es importante que se instale correctamente y en las carpetas indicadas, de lo contrario puede crear problemas.

Hasta este punto aún no hemos puesto nuestros conjunto de datos en la máquina, y dado que dependiendo del equipo de cómputo y el Sistema Operativo o la capacidad de almacenamiento es recomendable hacer la "cubeta" o contenedor de almacenamiento desde la nube donde tengamos fácil acceso y un respaldo adecuado, tanto la máquina virtual como el servicio de almacenamiento (EC2) fueron rentados de Amazon, por su escalabilidad y seguridad además de que hablando de computación es el mejor prestador de servicios y de mejores costos, en el cual debemos tener carpetas ordenadas de manera que sean identificables. En la *imagen 20* se muestra un ejemplo de cómo se puede organizar el material. Dentro de la carpeta Data tendremos la carpeta Train que lleva consigo el archivo que contiene todas las imágenes compresas con su respectivo XML y el TXT donde vienen las etiquetas o "características" que se identificaran, igualmente de la carpeta Test, lleva consigo aquellas imágenes que son para verificar o, mejor dicho, evaluar si el modelo aprendió correctamente. La ultima carpeta de Data es el modelo pre-entrenado en detección de objetos, aunque esto se explicará de manera más profunda a continuación por ahora podemos decir que es un punto de partida para poder iniciar a entrenar nuestro propio modelo, trayendo consigo los "puntos de guardado" (check-point), siendo estos, aquellos puntos de guardado de los pesos y errores que ya "aprendió", esto de cierta manera agilizara a nuestras iteraciones al momento de entrenar.

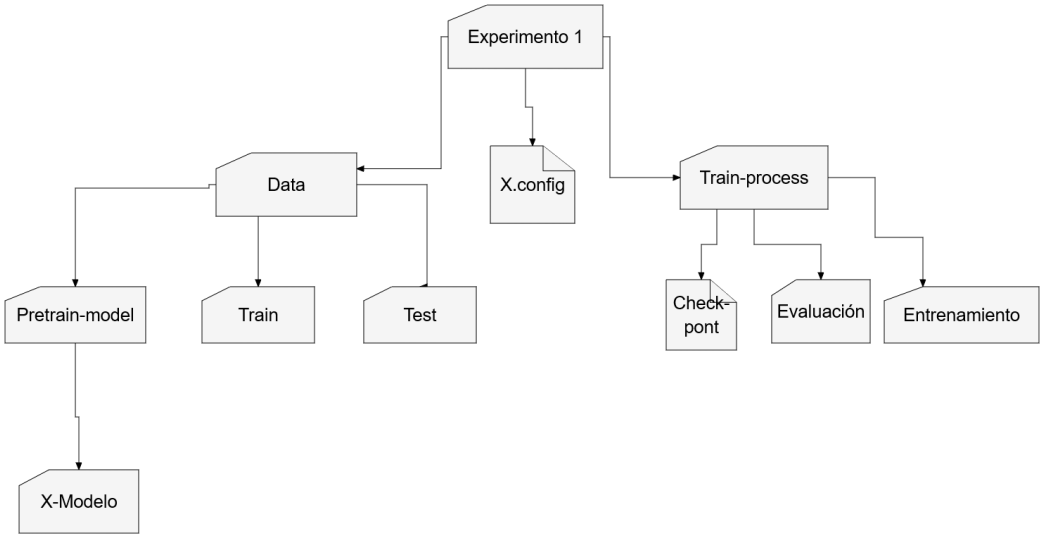


Imagen 20 Ejemplo de distribución de carpetas

Por otra parte, tenemos el archivo .config, el cual es indispensable configurar antes de subirlo o de hacer experimentos ya que puede afectar tanto a los resultados como en el espacio de almacenamiento, la primera precaución es cambiar todos las variables de entorno por las que usaremos, al pedir el conjunto de datos de evaluación se da la respectiva variable de entorno, es importante mencionar que el inicio de la variable no es el de la maquina propia sino que es en base a la raíz de la máquina virtual, ya que sobre ella se trabajó. Otro aspecto importante fue asegurar que la resolución de la imagen sea la correcta tanto del modelo y como de nuestro conjunto de datos, en este caso es de 640x640 pixeles, el número de pasos o iteraciones que se estableció es de 8,000 esto para no saturar la memoria, pero asegurar buenos resultados. Un punto importante y es el tipo de punto de guardado a usar, será el de “detección”.

Por último, la carpeta de Train-process es donde se guardó: el modelo entrenado, su evaluación y los diferentes puntos de guardado generados durante el proceso. Esta última carpeta al estar vacía no se sube a la “cubeta” cuando se migra asi que se creó desde la máquina virtual una vez que estuvo toda la estructura, bien otra alternativa era crear un archivo de texto dentro.

Cuando se generó una estructura similar a la anterior, esta se migro a la instancia de almacenamiento del servicio de Amazon, y más tarde se copió dentro de la máquina virtual. Esto fue la preparación del material que se necesitó previo a la experimentación, fue necesario revisar cada cosa antes de llevarla a cabo, ya que lo primordial era obtener buenos resultados y usar el mínimo de recursos en el menor tiempo posible, por lo que un imprevisto o error puede ser costoso. Durante el experimento de prueba, se generaron varios contratiempos provocados por error al colocar las variables de entorno, o la configuración del archivo del modelo, estos contratiempos fueron de ayuda durante las 4 experimentaciones oficiales, aprovechando los recursos al máximo.

3.3. Entrenar redes neuronales enfocadas a la solución del problema

La experimentación consto de 4 partes en las que se cambiaban algunas variables como el modelo de detección de objetos o el conjunto de datos, esto con el fin de identificar cual genera mejores resultados y en la cantidad de tiempo usado. Una cosa importante a mencionar es que los comandos usados durante el experimento por cuestiones de confidencialidad no serán mencionados o mostrados, pero el proceso que se llevó a cabo si y se mostraran algunos de los resultados obtenidos.

Cada vez que se fue generando un nuevo experimento estos fueron siendo almacenados en la cubeta separadas dentro del mismo dominio de almacenamiento con el fin de tener todo reunido. Por último, se usó una conexión mediante SSH (Secure Shell), por la seguridad de las conexiones y sobre todo por la facilidad de acceso remoto, aunque esta puede terminar matando los procesos que estén corriendo en la terminal donde accedamos a la máquina virtual. Para evitar eso es esencial el uso de sesiones de terminal ya que en caso de que nuestra conexión sea detenida u ocurra cualquier incidente que afecte el funcionamiento, esta seguirá funcionando, la razón es que el proceso de aprendizaje puede llevar algunas horas y puede afectar a los resultados, además de que el uso del recurso al reanudar puede elevar los costos.

3.3.1. Experimento 1 con el modelo Faster R-CNN ResNet50 V1 640x640

En el primer experimento se usó el modelo pre-entrenado Faster R-CNN de imágenes de 640x640 y con el conjunto de datos donde el 85% de las imágenes fueron las tomadas de los videos del transporte público local y el otro 15% de los conjuntos de datos obtenidos de Internet, el conjunto de imágenes de evaluación fue de las de la investigación. Una vez que se tuvo todo el material en orden y dentro de la máquina virtual accedemos a la carpeta models y research que anteriormente se instaló junto al TensorFlow, ahí configuramos la variable de entorno de acceso al archivo de configuración del modelo, una vez hecho se ejecuta el comando de ejecución del código de la detección de objetos que se encuentra en la carpeta donde accedimos anteriormente. El proceso de aprendizaje puede llegar a tardar horas, en este caso tardo aproximadamente 1 hora y 20 minutos, para visualizar lo que se estaba ejecutando se recurrió al buscador y justo ahí se colocó la IP que nos asignaron a la máquina virtual y el puerto que asignamos si creamos una red para ella, esto claramente se hizo desde otra terminal de sesión. Como se muestra en la *imagen 21* tenemos varias gráficas, las cuales representan las funciones de perdida, la gráfica que más peso tiene para analizar los resultados es la primera y es quien nos ira mostrando si el modelo va aprendiendo correctamente o no, además de que nos indica el número de iteraciones realizadas, para llevar un mejor control.

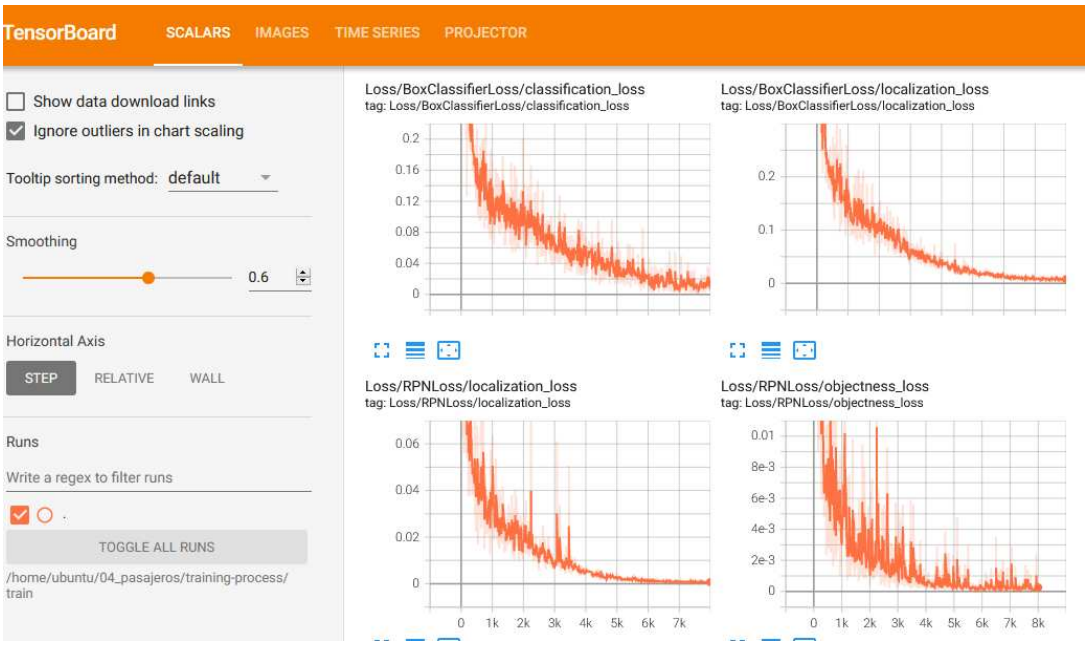


Imagen 21 Pantalla de supervisión de entrenamiento

Cuando el proceso de entrenamiento terminó se realizó el de evaluación del modelo, para ello se creó una nueva terminal de sesión, y la anterior de entrenamiento fue terminada, para esta última fase se sitúa en la misma carpeta que para el entrenamiento pero ahora se crea una nueva variable de entorno, aparte de los dos anteriores, este apunta a la carpeta de Training-process que es donde están guardados los puntos de guardado, estos contienen la información del último estado de aprendizaje con el cual se evalúa si el modelo está funcionando correctamente. Al crear las variables de entorno se usa un comando similar al de entrenamiento solo que esta vez usa tres variables. El proceso de evaluación tambien puede demorar, pero en este caso solo tardó aproximadamente 30 min, tambien se puede visualizar el proceso de este mediante una pantalla similar a la *imagen 21,* solo que esta vez contiene imágenes y no gráficas.

Cuando se terminó el proceso la parte importante y que no se debe olvidar, fue migrar o sincronizar todo lo que se obtuvo en el servicio de almacenamiento, ya que el modelo y la evidencia puede servir en futuros experimentos.

3.3.2. Experimento 2 con el modelo Faster R-CNN ResNet50 V1 640x640

El segundo experimento con este modelo constó de un nuevo conjunto de datos este al contrario del anterior cuenta con un 85% de imágenes del conjunto de datos encontrados en internet, mientras que el resto fueron imágenes de las que, recolectadas en la investigación de campo, es un tanto redundante explicar todo el proceso nuevamente, por lo que se puede decir que el proceso de realización fue exactamente el mismo. En cuanto a los tiempos necesitados fueron de aproximadamente 1 hora y media en aprendizaje y 30 minutos en evaluación.

Los resultados claramente variaron dada la diferencia de material con el que se le "enseño" al modelo. Es importante recordar la sincronización de las carpetas con la "cubeta", en una carpeta separada de la del experimento anterior.

3.3.3. Experimento 3 con el modelo EfficientDet D1 640x640

Para el experimento numero 3 cambiaron algunas cosas, el conjunto de datos fue el que tiene mayoría de imágenes de la investigación que se realizó mientras que el resto eran de las encontradas en Internet, para el modelo de pre-entrenamiento cambio al EfficientDet, la razón del cambio el modelo, es porque se debía tener una mayor cobertura en las posibilidades de tener éxito, por lo que se eligieron dos de los modelos más usados y efectivos en la detección de objetos. Para la configuración del archivo del modelo, sigue totalmente igual, si bien la estructura cambia ligeramente, son identificables las variables a cambiar.

El proceso de realización fue similar a los experimentos anteriores, en cuanto a tiempos este por ser un método más robusto y pesado tardo aproximadamente 1 hora y 40 minutos en el aprendizaje y 45 min en la evaluación y los resultados fueron totalmente diferentes al experimento 1 con quien comparte características.

Para este experimento se sincronizó en una carpeta diferente, de manera que se pudiera identificar de los demás experimentos.

3.3.4. Experimento 4 con el modelo EfficientDet D1 640x640

Este fue el último experimento realizado, el cual conto con el conjunto de datos donde la mayoría de las imágenes son de las encontradas en Internet y el resto de las que se obtuvieron de la investigación, al igual que en los demás nada más fue diferente, solo que durante esta experimentación ocurrieron percances de los que no se estaba previsto y la máquina virtual fallo a punto de terminar el aprendizaje, debido a los costos de los recursos se optó por dejarlo en la iteración que concluyo, que fueron cerca de las 6 mil, además de que las gráficas podrían darnos proyecciones de cómo pudo haber termino y los resultados no fueron totalmente alterados. Por lo que este experimento fue valido, en cuanto a sus tiempos supero la hora y 45 min hasta el momento de quiebre y la evaluación tardo alrededor de 35 min. De igual manera se sincronizo con su respectiva carpeta para guardar el contenido.

Estos fueron los 4 experimentos que se realizaron durante esta etapa, cada uno de los 4 trataba de cubrir la mínima diferencia que existiera entre ellos como punto de inflexión para tomar una decisión sobre cuál es la mejor herramienta en costos y tiempo que nos permitan resolver la problemática. Cabe mencionar que los experimentos y los conjuntos de datos elegidos solo fueron la prueba, ya que para obtener resultados 100% o más cerca posible de confiables, se necesitan más iteraciones e imágenes para entrenar el modelo, para cubrir otras posibles situaciones y lugares. Pero eso se dejará para el momento de la implementación.

Capítulo 4

Evaluación

4.1 Resultados de Experimentación

En cuanto a los cuatro experimentos, se obtuvieron buenos resultados en al menos tres, el experimento donde no se obtuvieron resultados satisfactorios es un indicador de que pudo haber un error que altero alguno de los resultados o que simplemente los resultados tuvieron poca cantidad de iteraciones.

4.1.1. Experimento 1 con el modelo Faster R-CNN ResNet50 V1 640x640

Este experimento es del cual se obtuvieron los mejores resultados, ya que obtuvo una tasa de éxito muy alta y poco margen de error. Como se observa en la *Imagen 22* nos muestra la función de perdida graficada, donde el eje x es la cantidad de iteraciones y el eje y el valor de pérdida o costo (Loss).

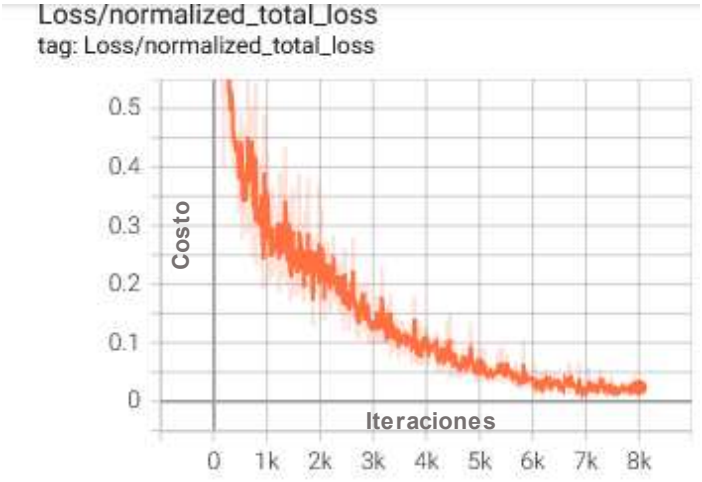


Imagen 22 Gráfica del Experimento 1

Entre menor es la perdida podemos deducir que mejores son los resultados de aprendizaje, y aquí obtenemos que en la iteración 8 mil tenemos una pérdida de

aproximadamente 0.025 un numero realmente cerca del 0.

Mientras que en la evaluación obtenemos una imagen donde hace la detección de los objetos, en este caso las caras que reconoce. En la *imagen 23* se muestran dos imágenes, donde *b)* es la evaluación previamente etiquetada y *a)* es el resultado de la evaluación del modelo, en la que podemos concluir que el modelo fue capaz de reconocer al 100% a las personas que iban a bordo y las clasifico de manera correcta.

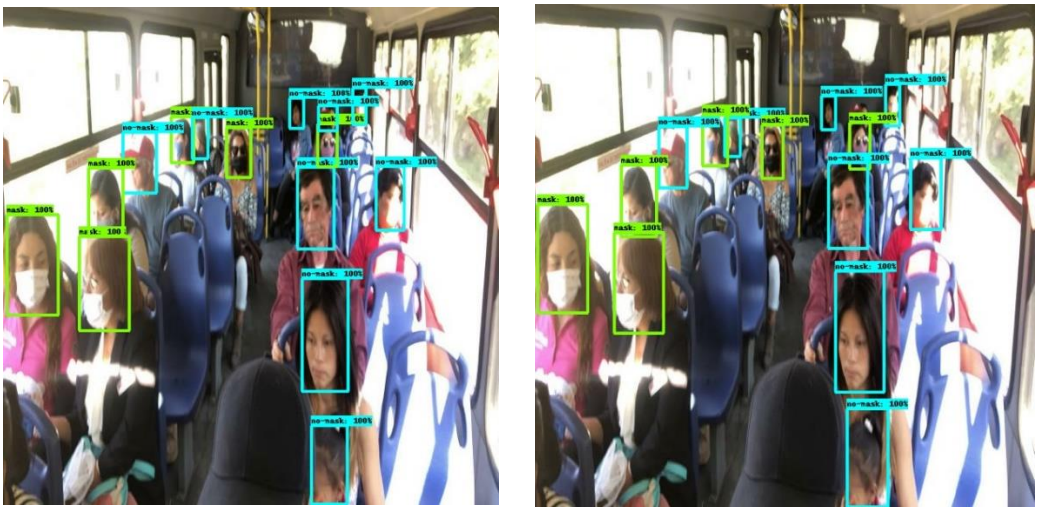


Imagen 23 Resultados de detección de rostros del Experimento 1. a) y b)

Y por si fuera poco identifica a una persona que no estaba etiquetada anteriormente, por lo que podemos concluir que es capaz de identificar rostros relativamente pequeños.

4.1.2. Experimento 2 con el modelo Faster R-CNN ResNet50 V1 640x640

A pesar de usar el mismo modelo que el Experimento 1 el que tuvo menor tasa de éxito fue este. En la *Imagen 24* tenemos la gráfica de la función de perdida, y aun que la escala es ligeramente diferente a la del Experimento 1, podemos notar que nuestra pendiente no va disminuyendo si no que tiene un ruido muy notable y es inestable, por lo que se puede deducir que el modelo no aprendió correctamente, al terminar las 8 mil iteraciones tenemos un costo de aproximadamente 0.1.

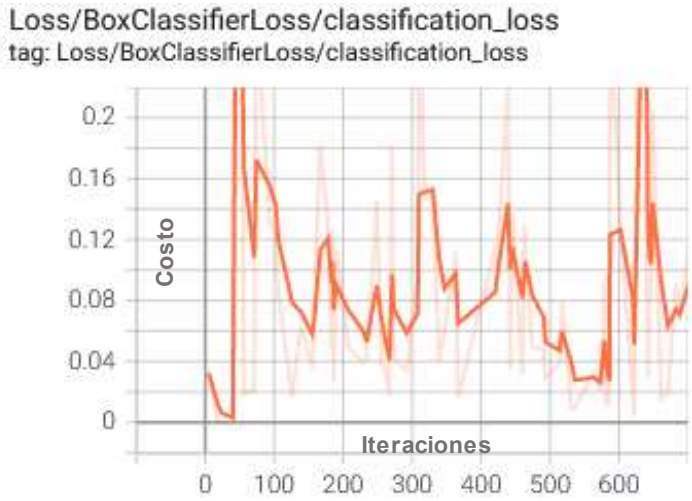


Imagen 24 Gráfica del Experimento 2

La imagen de comparación en la que se muestra en la *Imagen 25* a simple vista podemos notar que la detección ha fallado, sin embargo, no podemos concluir que fue meramente que el modelo estaba mal o que este método no funciono, ya que, al contrario, al hacer uno de los siguientes experimentos se llegó a la conclusión de que el archivo de etiquetado estaba incorrecto, al menos un 5% del conjunto de datos, lo que pudo ser la causante en que este experimento fallara. Dado a la cantidad de recursos que se deben usar para recrear el experimento y el poco tiempo que se tenía fue imposible hacerlo nuevamente, ya que recordemos que se estaban rentando máquinas virtuales y su costo se puede elevar dependiendo del tiempo de uso y recursos.

Imagen 25 Resultados de detección de rostros Experimento 2

4.1.3. Experimento 3 con el modelo EfficientDet D1 640x640

Otro experimento del que se obtuvieron buenos resultados, además de que aquí se emplea el uso de un nuevo modelo, más robusto que el anterior. En la *Imagen 26* se observa la gráfica de la función de pérdida o costo, a diferencia de las demás esta se detuvo en cierto número de iteración de forma manual una vez que se observó la falta de mejoría, ya que a pesar de ir en descenso en cierto punto su comportamiento es muy estable y puede llegar a convertirse en una recta o disminuir muy lentamente, consumiendo más recursos y espació que podríamos no tener. Al contrario de las demás tenemos 7 mil iteraciones y el valor que obtuvo al final fue de 1.5. Por lo que podemos decir que va aprendiendo de manera lenta pero no necesariamente puede ser lo que necesitemos por los costos y recursos utilizados.

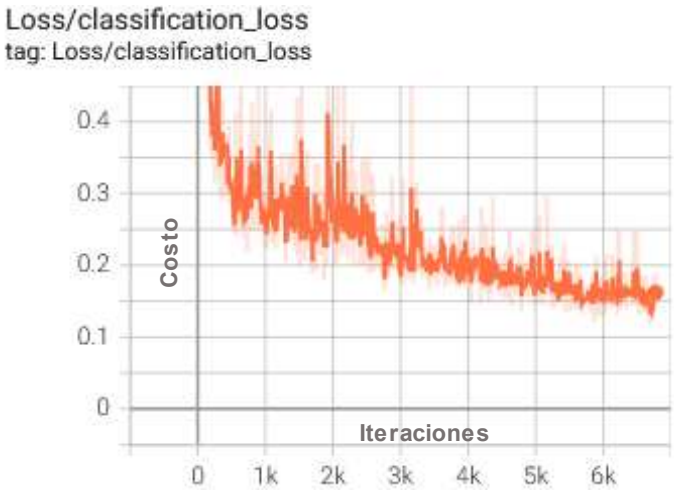


Imagen 26 Gráfica del Experimento 3

Los resultados de la *Imagen 27* tienen dos puntos importantes que resaltar: 1. el señor de lentes no es identificado por el modelo mientras que 2. la niña que esta abajo a la derecha es identificada, por ello podemos deducir que el modelo está aprendiendo correctamente ya que logro identificar objetos que no estaban previstos y que a parecer la falta de aprendizaje hizo que no tomara a la persona que no marco, en un contexto en el que tenemos los recursos suficientes y podríamos dejar iterar al menos 10 mil veces probablemente esta opción sería una de las más viables.

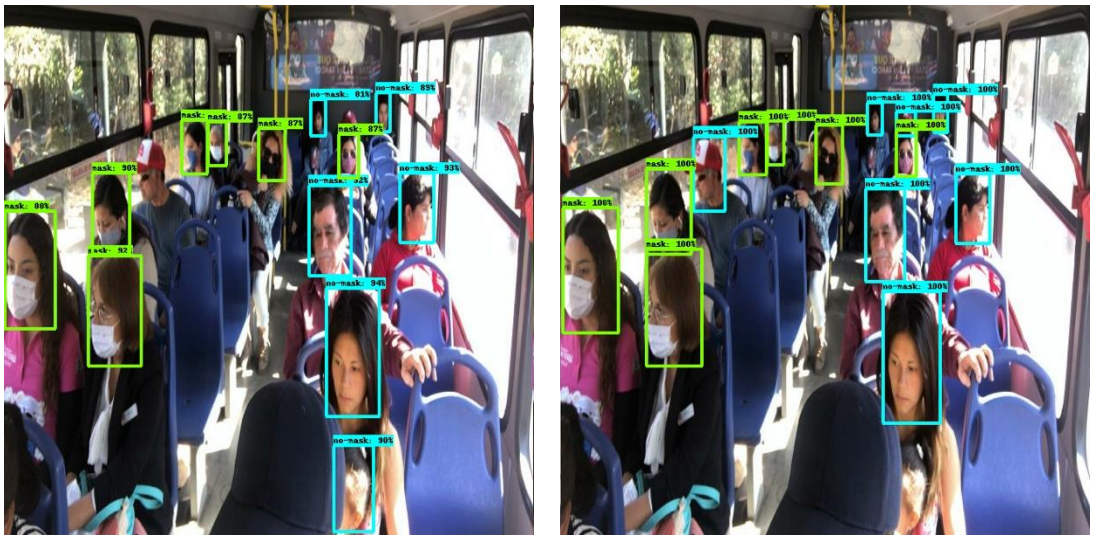


Imagen 27 Resultados de detección de rostros Experimento 3. a) y b)

4.1.4. Experimento 4 con el modelo EfficientDet D1 640x640

El ultimo experimento consto de tan solo 6 mil iteraciones por una interrupción no predicha, por lo que fue imposible reanudarla, sin embargo, los resultados que se obtuvieron fueron buenos, y con la función de costo se puede tener una predicción cercana al comportamiento que podía adoptar. En la *Imagen 28* podemos ver la gráfica y notar que hay demasiado ruido en algunas iteraciones, esto puede ser porque al ser un modelo más robusto y pesado el cálculo es aún más exhaustivo, por lo que se podrían ver este tipo de alteraciones repentinas, si bien la gráfica tiene un buen descenso, tiene demasiadas alteraciones lo que dificulta tomar un valor exacto o aproximado de su función de costo, siendo el ultimo valor registrado un 0.18, la pendiente de esta curva en comparación con el experimento anterior casi no tiene cambios y da la sensación que no podrá tomar valores más bajos, por lo que este experimento fue el tercero en obtener resultados favorables.

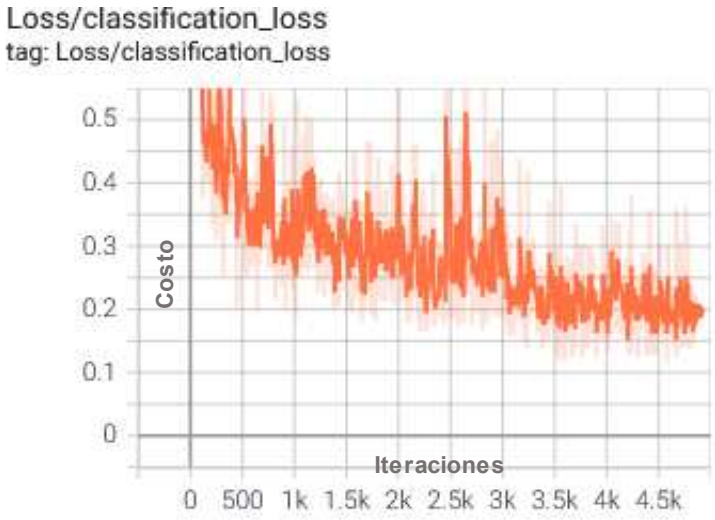


Imagen 28 Gráfica del Experimento 4

En la *Imagen 29* se muestra el resultado, y pese a que el número de iteraciones estaba cerca se puede notar como a las personas del fondo no las detecta, este fenómeno se pueden deber a varios factores y uno de ellos es el conjunto de datos usado, se identificó un patrón donde las imágenes que se apegaban más a la realidad son los experimentos que más éxito tienen, mientras que este que tenía un conjunto de datos con imágenes de una base de datos oficial en internet tiene menos tasa de éxito, esto tambien se puede deber al etiquetado. La razón del fallo más probable es sencillamente que la cantidad de iteraciones no fue suficiente, por otra parte, la cantidad de tiempo que se uso fue similar a la de los demás experimentos.

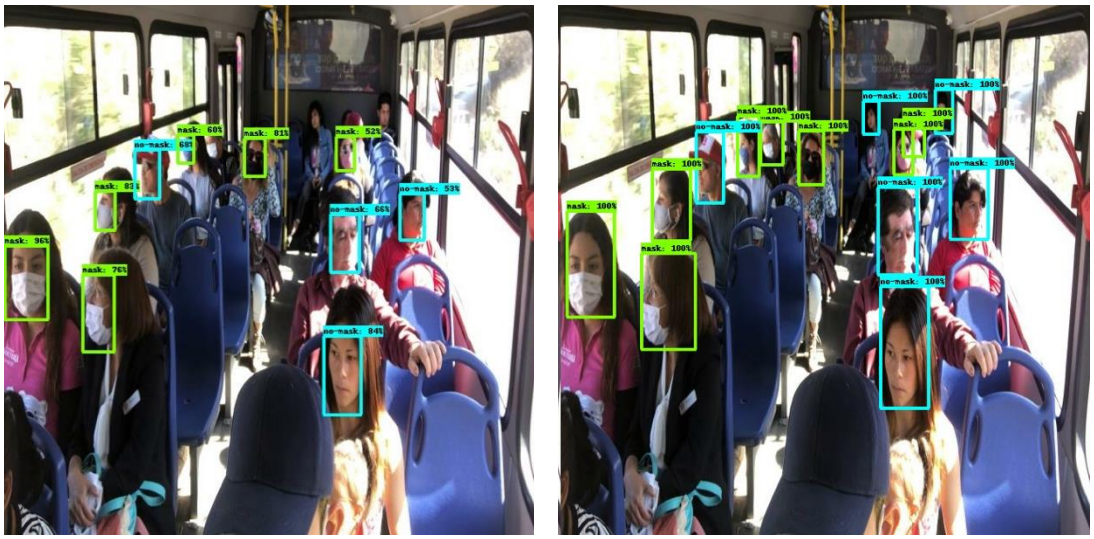


Imagen 29 Resultado de detección de rostros Experimento 4. a) y b)

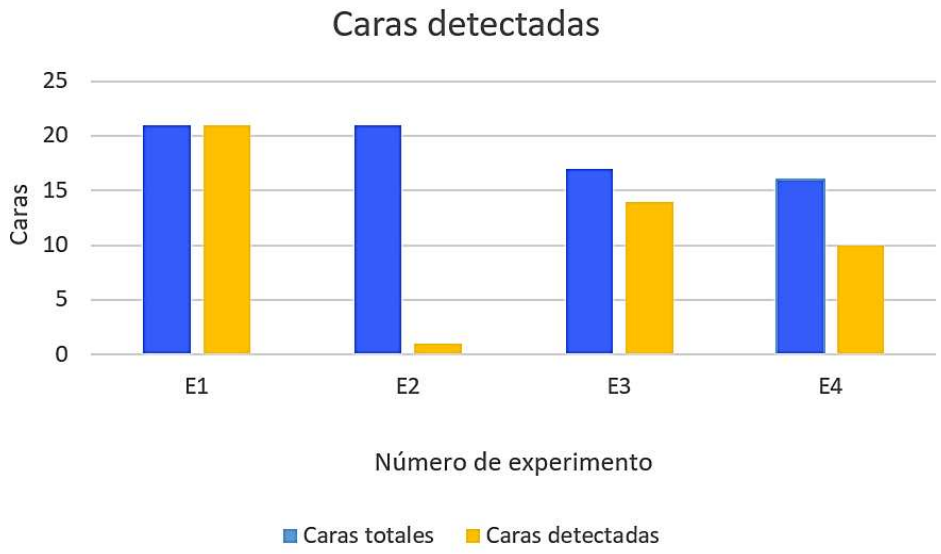
4.5 Interpretación de resultados

Es importante tener el panorama completo de los cuatros experimentos que se realizaron con el fin de poder comparar algunos puntos que puedan ser de real importancia al elegir el modelo y el tipo de conjunto de datos que se debe crear para tener un modelo capaz de detectar al 100% o lo más cercano, a cualquier rostro humano en cualquier situación. En la *Tabla 1* tenemos la descripción de los resultados de detección, siendo el experimento 1 con Faster como herramienta la que más caras pudo detectar en la mayoría de los casos, mientras que el mismo experimento, pero con diferente conjunto de datos obtuvo el peor resultado de detección.

	Modelo	Total de caras	Total de caras detectadas	Acierto
Experimento 1	Faster R-CNN ResNet50 V1 640x640	21	21	100%
Experimento 2		21	1	5%
Experimento 3	EfficientDet D1 640x640	17	14	81.25%
Experimento 4		16	10	62.5%

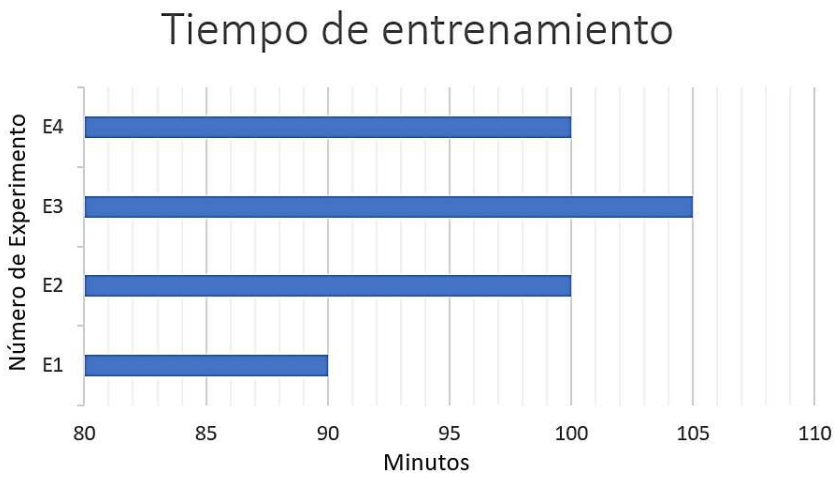
Tabla 1 Tabla comparativa entre los resultados de los experimentos

En la *Gráfica 1* tenemos un resumen, donde se ve claramente como el experimento 1 obtuvo el mayor porcentaje de acierto seguido del experimento 3, que contaba con otra herramienta, pero el mismo conjunto de datos.

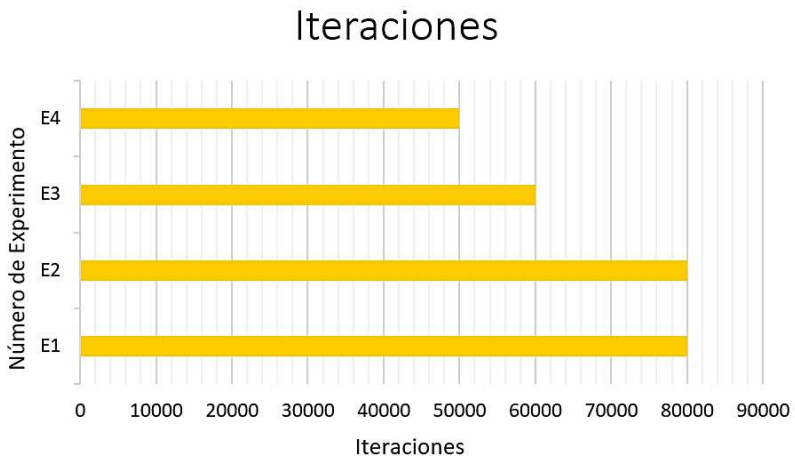


Gráfica 1 Porcentaje de acierto de detección

Otras de las características importantes a avaluar es el tiempo de entrenamiento, ya que puede ahorrar recursos y valga la redundancia tiempo, podemos comparar en la Gráfica 2 los tiempos de entrenamiento que tardo cada una, claro esto está estrechamente relacionado a la Grafica 3 que representa el número de iteraciones, ya que ambas mantienen una correlación, siendo esto el experimento 1 fue quien más iteraciones hizo y menor tiempo de entrenamiento necesito, mientas que el experimento 3, pese a que hizo menos iteraciones necesito un tiempo mayor, consumiendo más recursos y con resultados buenos, pero no los mejores.



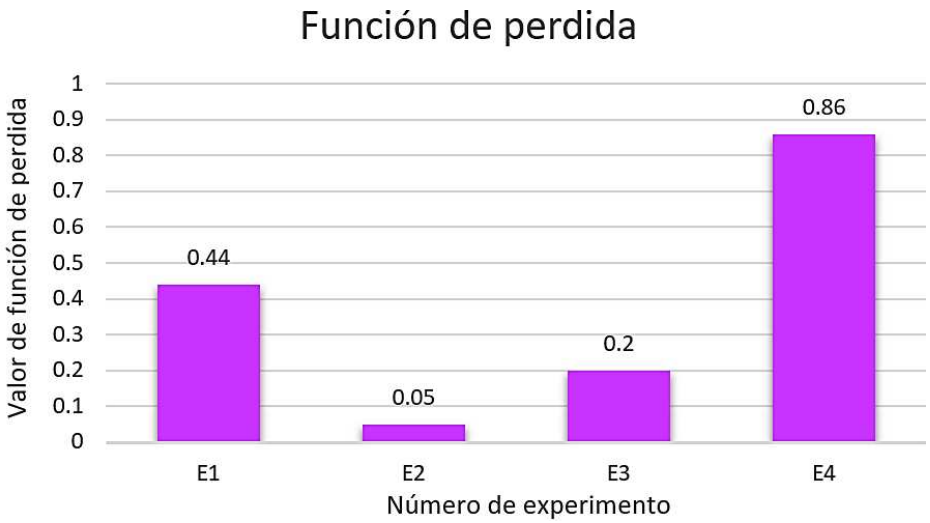
Gráfica 2 Representación del tiempo de entrenamiento en minutos



Gráfica 3 Representación de la cantidad de iteraciones por experimento

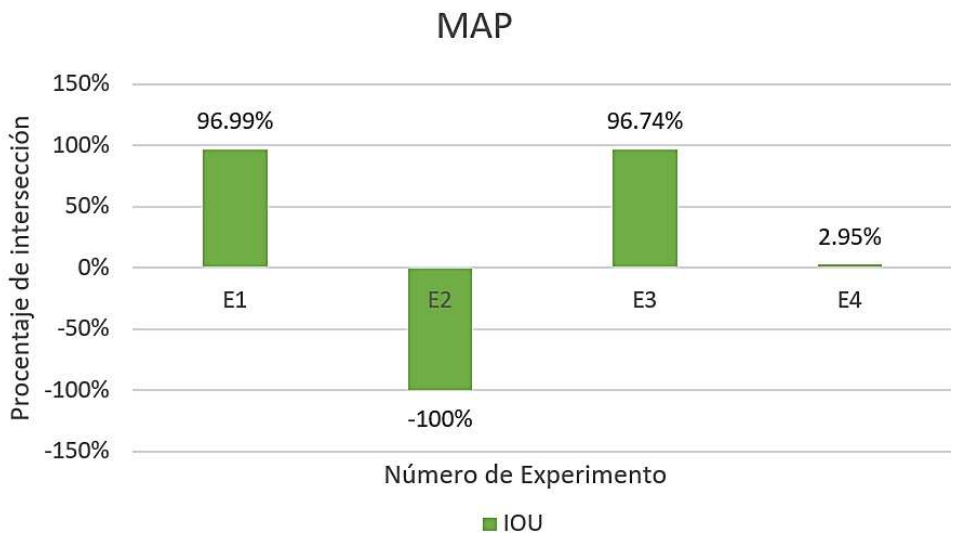
En la Gráfica 4 tenemos la representación de los valores obtenidos de la función de pérdida o costo de cada experimento y como previamente se ha mencionado, mientras menor es la función de perdida es mejor el aprendizaje del modelo, en la gráfica se puede saber que el experimento 1 es quien obtuvo el menor valor, por lo que se puede deducir que es quien mejor entreno el modelo y quien obtendrá mejores resultados, mientras que el experimento 4 tuvo el peor, una de las razones por lo que los últimos dos experimentos tuvieron datos tales como mayor tiempo de

entrenamiento pero menor número de iteraciones y mayor valor de perdida puede ser a que es una herramienta que se conoce por ser exhaustiva, es decir lleva todo el procesamiento a fondo por lo que tiende a ser más tardada y consume más recursos. Otro dato a tener a consideración puede ser que si es demasiado pequeño puede significar que se sobre entreno o que no obtuvo resultados adecuados.



Gráfica 4 Representación del valor de la función de perdida por experimento

Otro de los datos importantes que se consideraron es el MAP (Mean Average Precision) que más que nada nos dice que tanta es su precisión en porcentaje, es decir que tanto el etiquetado o caja de las imágenes de prueba del modelo coincide con las que designamos previamente etiquetadas para la prueba, o llamado IOU (Intersectión Over Union), que es que tanto en porcentaje se intersectan ambas cajas, en la *Grafica 5* tenemos los datos de los 4 experimentos donde podemos comprobar que el experimento 1 obtuvo un mejor resultado y casi a la par el experimento 3, siendo el experimento 2 quien obtuvo un porcentaje negativo, lo que significa que no hubo aprendizajes o aprendió mal en otras palabras



Gráfica 5 Gráfica del MAP (Mean Average Precision)

Con el contexto de las gráficas podemos deducir que quien mejores resultados obtuvo fue el experimento 1 de quien se aprovecharan las características para poder hacer en un futuro experimentos con mejores recursos y que obtengan mejores resultados.

4.6 Aplicación e impacto

Al término de la experimentación y análisis se decidió usar el ambiente del primer experimento, tomando más imágenes de diferentes rutas de transporte y de diferentes regiones. Por lo que una vez implementado el modelo entrenado de manera más robusta, este contara con una implementación en la aplicación que se está desarrollando para el proyecto, que mostrara la cantidad de pasajeros por unidad que van a bordo, lo cual será un impacto importante en el uso de la herramienta. La aplicación de este modelo es de gran ventaja ya que actualmente y debido a la pandemia por COVID-19, la cantidad de personas a bordo del transporte público se encuentra restringida y es importante tomar las medidas de prevención de contagio como un hábito, por lo que el saber la cantidad de personas a bordo de un autobús puede ser de ayuda en este ámbito, además los usuarios

podrán saber si el autobús cuanta con lugares disponibles o no. Por otra parte, del lado de los administradores esto les permite llevar un conteo de la cantidad de personas que prestan servicio a “x” hora en “x” ruta, además de tener un estimado de las ganancias.

Si bien el impacto que tendrá en la sociedad es positivo, se debe de tener consciencia que será algo nuevo a lo que comúnmente no estamos acostumbrados pero por ende se esperan resultados favorables en la aceptación por parte de los usuarios ya que será una herramienta de uso diario que tiene consigo bastantes beneficios.

Capítulo 5

Conclusión y recomendaciones

5.1 Conclusión

Todo el proceso de integración al proyecto y durante las etapas de investigación fueron fundamentales para poder entender el funcionamiento e importancia que tenía el poder obtener buenos resultados. La mayoría de la población hace uso del transporte público siendo este en cierta medida fundamental para la vida diaria, algunos dependen de eso para poder llegar a sus lugares de trabajo o centros de estudio, por lo que de manera muy sutil es de importancia para cada uno de ellos, que la aplicación es tan importante, el poder brindar una herramienta que sea de ayuda y que tenga un impacto positivo para cientos de mexicanos que diario salen de sus casas por razones diferentes, pero que podrían hacer uso de aplicación como herramienta de ayuda, es el principal motivo por lo que el éxito de la implementación es importante.

Para ello el conocimiento básico de Aprendizaje máquina y Redes Neuronales fue de mucha importancia, en el tiempo contemporáneo estas herramientas tienen aplicaciones muy prácticas que pueden ayudar a la sociedad de innumerables maneras, por lo que su empleo en la detección de personas en el transporte público es un paso importante. Si bien son temas complejos y complicados de entender de buenas a primeras, es algo que vale la pena aprender, ya que su dominio o uso puede abrir muchas puertas y generar nuevas ideas de aplicación.

Como se ha dado a entender, aun no se implementó en la aplicación móvil, dado a que para que se pueda tener éxito rotundo se debe tener una mejor implementación, desarrollo, recursos y tiempo para que la red neuronal de nuestro modelo sea capaz de identificar cualquier rostro.

Hay muchas alternativas que se pudieron haber tomado en el proceso de experimentación, pero se basó en los buenos resultados y manera de implementación que ofrecía TensorFlow que fue elegido, y contemplando los resultados se puede asumir que fue correcto tomarlo, ya que realmente no necesitas

ser un experto programador en Python o conocedor de la Inteligencia Artificial para comprender su funcionamiento, bastaba con entender los conceptos básicos, por lo que fue la herramienta perfecta, para que en futuras investigaciones o bien las personas que leen esto, puedan interesarse en intentar hacer implementaciones con esta herramienta, sin necesidad de ser expertos. Se tuvo pensado hasta cierto momento de la finalización de los experimentos usar una herramienta alternativa para tener al menos más variables y comparaciones, pero dado a que el tiempo no lo permitió y que TensorFlow era realmente es una librería que cualquiera puede manejar se optó por seguir con esta misma.

Para ir directo al punto principal de la conclusión, se obtuvieron resultados satisfactorios, tanto que los resultados fueron inesperadamente positivos, inicialmente se tenía la idea que, con las iteraciones elegidas dada a la limitación de los recursos, se podía carecer de resultados que nos permitirán tener material para tomar decisiones, pero este no fue el caso. Los resultados arrojados fueron mejores que los esperados, tiendo como consecuencia que se logró detectar rostros que no se tenían contemplados, siendo estos perfiles y rostros semi completos, se puede llegar a la conclusión de que esto se debió al uso de cubre bocas o que la diversidad de conjunto de datos y de rostros que aprendió le permitió reconocer que ciertos patrones o áreas eran rostros o parte de ellos, además de que en todos los experimentos exitosos sin errores de índole humana, fueron identificados correctamente si la persona usaba el cubrebocas, reconocía rostros aun en alteraciones de luz y sombra, más los ya nombrados que nos tomaron por sorpresa, por lo que se puede decir que fueron un éxito y se puede adelantar a hacer la predicción que ante una maquina lo suficientemente equipada y potente, con más de mil imágenes de conjunto de datos y un aproximado de 20 mil iteraciones, siempre y cuando no dejemos que aprenda excesivamente el modelo, podemos obtener los mejores resultados: que sea capaz de reconocer perfiles, personas de espalda, menores, o partes superiores de la cabeza, como comúnmente se tiene en un transporte público, esto ayudaría que esos márgenes de error que podría tener

de que alguna persona no se contemplada sean eliminados y tener un conteo al menos en un 90%-99% efectivo de la densidad de las personas que van a bordo.

Como persona común, el uso de esta app es un paso adelantado a mejorar la calidad de vida cotidiana de las personas, administración de tiempo y sobre todo para las personas que son administrativos de tener un mejor manejo de sus ingresos y de evitar robos. Como investigadora y desarrolladora es un gran paso tecnológico, el implementar cosas que ayudan a la vida cotidiana con herramientas que siempre han estado ahí y que tienen una utilidad enorme, siendo que además sería la primera aplicación de ser especie que resuelva una problemática de la cotidianidad que se crea, sería un tema relevante y que se espera inspire a miles de crear y de adentrarse a la inteligencia artificial con el fin de mejorar y apoyar al desarrollo de la sociedad.

5.2 Investigación a futuro

Para futuras investigaciones se espera entrenar un modelo con un conjunto de datos que contenga más de mil imágenes, que sean en su mayoría situaciones reales del transporte público y lo restante imágenes genéricas sobre rostros con y sin cubrebocas. Esto entrenado en un equipo que tenga el poder de procesamiento y almacenamiento de un aproximados de más de 25 mil iteraciones. Con este modelo evaluado correctamente se implementará una nueva sección en la app, que en base a videos de las posibles cámaras que estén instaladas en el trasporte público, se puedan extraer sus imágenes y ser procesadas por la red neuronal, y mediante un programa hacer el conteo de los rostros de las personas que identifico, mostrándose asi en el teléfono de manera individual por unidad.

Bibliografía

Flórez López, R., & Férnandez Férnandez, J. (2008). *Las redes neuronales artificiales.* España: Gesbiblo.

Lastra, M. S. (01 de junio de 2017). *Direccion General de Comunicacion Social*. Obtenido de DGCS: https://www.dgcs.unam.mx/boletin/bdboletin/2017_384.html

Nielsen, M. (2019). *Neural Networks and Deep Learning.*

Rosebrock, A. (2014). *Practical Python and OpenCV.* pyimagesearch.

Rossum, G. V. (2016). *El tutorial de Python.*

Watt, J., Borhani, R., & Katsaggelos, A. K. (2020). *Machine Leraning Refined.* Cambridge University Press.

Woods, R., & Gonzalez, R. (2002). *Digital Image Processing.* New Jersey: Prentice Hall.

Xiaoyue , J., Abdenour , H., Yanwei , P., & Granger, E. (2019). *Deep Learning in Object Detection and Recognition.* Singapure: Springer.