Universitat Oberta de Catalunya

uoc.edu

Face Recognition con imágenes de Data Streaming con modelos generados a partir de Transfer Learning y diferentes Funciones de Activación.

Ramírez Subeldia, Elvio
Inteligencia de Negocio y Big Data Analytics
B0.488 - Trabajo Final de Máster (Big Data)

Consultor: Ortiz Santiago, Victor Alejandro
Responsables de la asignatura: Daradoumis Haralabus, Atanasi
Florit Medina, Xavier y Borja Matas, Jaime

25/06/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Face Recognition con imágenes de Data Streaming con modelos generados a partir de Transfer Learning y diferentes Funciones de Activación.*
Nombre del autor:	*Ramírez Subeldia, Elvio*
Nombre del consultor/a:	*Ortiz Santiago, Victor Alejandro*
Nombre del PRA:	*Daradoumis Haralabus, Atanasi* *Florit Medina, Xavier* *Borja Matas, Jaime*
Fecha de entrega (mm/aaaa):	06/2022
Titulación:	*Inteligencia de Negocio y Big Data Analytics*
Área del Trabajo Final:	*B0.488 - Trabajo Final de Máster (Big Data)*
Idioma del trabajo:	*Castellano*
Palabras clave	*Face recognition, Transfer learning, Data streaming, Deep learning.*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Actualmente existen múltiples técnicas para generar y aplicar en las redes neuronales artificiales. Face Recognition es una aplicación de aprendizaje profundo dónde se aplican redes. En el Face Recognition hay que tener en cuenta que se intenta resolver un problema de clasificación. Para resolver estos tipos de problemas se utilizan las Redes Neuronales Convolucionales, presentan una arquitectura de capas, permiten por la forma de convolución una aproximación mayor al objeto que se intenta clasificar. Estas redes poseen diferentes nodos que se encuentran interconectados donde la información circula de un nodo a otro pasando por capas de entrada, capas ocultas y capas de salida. El entrenamiento del modelo puede ser uno de los trabajos más complejos dentro de las tareas en la creación de una red neuronal. Las capas que componen la arquitectura de la red neuronal, además de los valores de peso y sesgo, poseen una Función de Activación, que, dependiendo de cual se use, la información pasara de un nodo a otro de una manera diferente. Este trabajo plantea la aplicación de Transfer Learning para obtener una arquitectura y valores de pesos iniciales pre-entrenados y compara resultado de los modelos para Face Recognition de imágenes en Data Streaming, utilizando las Funciones de Activación ReLu, Swish y Mish. Como método de trabajo se apoya en el enfoque de investigación DSR. Los resultados de las pruebas realizadas no son concluyentes y dan lugar a mejorar los hiperparámetros de la red neuronal en futuras líneas de investigación.

Abstract (in English, 250 words or less):

A challenge when you want to work training neuronal artificial network, is to create a specific architecture and initial values that can start your neural network. This investigation work, using Transfer Learning and different Activation Functions to create a model for Face Recognition with Video Data Streaming, raises a result of comparative between different models. To solve this analysis, three modules are created, the first one dataset creator module, second training module, third Face Recognition module. In dataset creator module we take Transfer Learning from OpenCV. We will use the library for face detection 'Haar Cascades' and capturing images from webcam will be possible to create the dataset. Training module, in this module we used Transfer Learning from VGG16 library, facilitating the creation of the corresponding architecture for a convolutional network. Also, we implemented 'imagenNET' to get the initial values for the network. Adapting each neuronal network for each Activation Function 'ReLu', 'Swish' and 'Mish'. Keras and Tensorflow library will facilitate the use of the different network layers, getting a trained model for Face Recognition with metrics of cost and functioning. Finally, in Face Recognition module, we import the result of the model trained and Face recognition is tested. Analyzed the results of the tests, no conclusive results are obtained on which Activation Function ('ReLu', 'Swish', 'Mish') gives the best result for Face

Recognition, but the model remains for future lines of research where the hyperparameters of the neural network can be tuned.

Índice

Lista de figuras

1. Introducción

1.1 Motivación del proyecto

La elección de este tema deriva de la fase inicial del TFM, donde se investigó para llevar adelante, un sistema de FR de control de acceso a oficinas basado en Deep Learning (DL) y que, los algoritmos para detección de rostros puedan ser mejorados, evitando que arrojen resultados de sesgos raciales en sus predicciones. Con toda la información consultada, es cuando nace el interés por la realización de este trabajo.

El presente proyecto tiene como objetivo principal, la creación de un sistema de Face Recognition (FR), utilizando técnicas Transfer Learning (TL) para imágenes de video en tiempo real, Data Streaming (DS), donde se comparan diferentes Funciones de Activación (FA).

1.2 Contexto y justificación del trabajo

En los últimos años, el DL como herramienta de predicción, ha tomado una relevancia muy importante. Uno de los retos que una persona se puede encontrar al momento de iniciar a trabajar con esta técnica, es acotar la gran variedad de estudios y avances que están disponibles. Hoy en día, las Redes Neuronales Artificiales (RNA) permiten, con un entrenamiento previo, dar respuestas a muchos problemas. Dentro de las RNA, nos encontramos con redes neuronales las cuales, entre otras cosas, son capaces de aprender a reconocer objetos dentro de una imagen y clasificarlos, imitando al córtex visual humano. El avance tecnológico permite entrenar estas RNA a gran velocidad. También, dentro del campo del DL, el conocimiento adquirido para la resolución de un problema puede facilitar la resolución de otro problema, que tenga similares características, para esto existen técnicas de TL.

Así como todo lo relacionado con el DL avanza, también lo hacen sus algoritmos y la forma que resuelven los problemas.
Dentro de las RNAs ,no encontramos con diferentes funciones de activación (FA), las cuales, al igual que en una red neuronal biológica (RNB), van a permitir que la comunicación entre diferentes neuronas se active con diferentes potenciales. Las FA en los últimos años, se ha ido ajustando y aparecieron modelos que prometen mejores resultados en la predicción de diferentes tareas.

En este trabajo se analizan los resultados obtenidos de comparar la creación de un sistema de FR con TL con imágenes de DS a partir de RNA que se activan con diferentes FA.

Para conseguir los objetivos de este proyecto, se plantea la utilización de técnicas de DL para FR y TL.

1.3 Objetivos del trabajo

Los siguientes son objetivos medibles que se esperan conseguir con este trabajo.

- Detectar rostros en imágenes para ser utilizados en entrenamientos de modelos de FR, con una precisión mayor al 90%. (Requerido).
- Creación de dataset de entrenamiento para un modelo para FR con una precisión mayor al 90%. (Requerido).
- Reconocer rostros en imagen de videos en tiempo real con una precisión mayor al 70%. (Requerido).
- Detectar el óptimo resultado de aprendizaje de modelo para FR utilizando TL con diferentes FA, con una precisión mayor al 80%. (Deseado)
- Detectar la FA óptima, con una precisión mayor al 80%. (Deseado)

1.4 Enfoque y método seguido

El presente trabajo se plantea como un proyecto de investigación, donde para comprender y responder a las hipótesis detalladas en el siguiente capitulo, están organizadas una serie de pasos y tareas soportadas y definidas dentro de una metodología. [1] El cumplimiento de esta metodología permite seguir los pasos de investigación para generar conocimiento, tal como lo definido en Design Science Research (DSR), aprendiendo a través de la construcción de artefactos y contribuyendo a la generación de nuevos y verdaderos conocimientos.

Este trabajo se apoya de la metodología para el proceso de investigación según lo definido en DSR, para lo cual, se realizan las siguientes tareas: enunciado, hipótesis, desarrollo, prueba y evaluación, conclusión.

Enunciado: esta etapa proyecta la problemática encontrada, que motivó a la realización de la investigación. Lo podemos ver en apartado 1.2, donde se hace referencia a todo lo observado hasta llegar a este trabajo, y el capítulo 2 que contiene el estado del arte sobre todos elementos que se incluyen en esta investigación.

Hipótesis: se plantea nuestra propuesta de investigación. La propuesta debe ser novedosa o bien una nueva adaptación de algo existente. El fin es plantear una propuesta que surja de la curiosidad o la creatividad sobre un fenómeno a investigar, planteando una hipótesis nula y una alternativa. La H0 o hipótesis nula, es la que se intenta refutar, rechazar o anular. La H1 o hipótesis alternativa representa la conclusión a la que se desea llegar. En este trabajo, en el capítulo 3, se detallan estas hipótesis.

Desarrollo: etapa destinada a desarrollar un diseño tentativo, el cual, va a permitir dar respuesta a la hipótesis planteada. El capítulo 4 es donde se plantea el desarrollo.

Prueba y evaluación: se ejecutan las pruebas sobre el desarrollo planteado, contendrá el análisis de los resultados de las pruebas realizadas. Se deja constancia de las pruebas, de qué manera y en qué condiciones de ejecutaron. Permitirá contrastar o refutar las hipótesis planteadas. En el capítulo 5 es donde se trazan y evalúan las pruebas.

Conclusión: Para finalizar la investigación, es necesario registrar el nivel de satisfacción que se encontró respecto a las iniciales hipótesis planteadas, el cumplimiento de expectativas, la contribución del conocimiento y líneas futuras de investigación. El capítulo 6 es donde se detallan las conclusiones.

1.5 Planificación del trabajo

A continuación, se indican las tareas e hitos necesarios, para el cumplimiento de los objetivos del proyecto.

Definición de las tareas e hitos

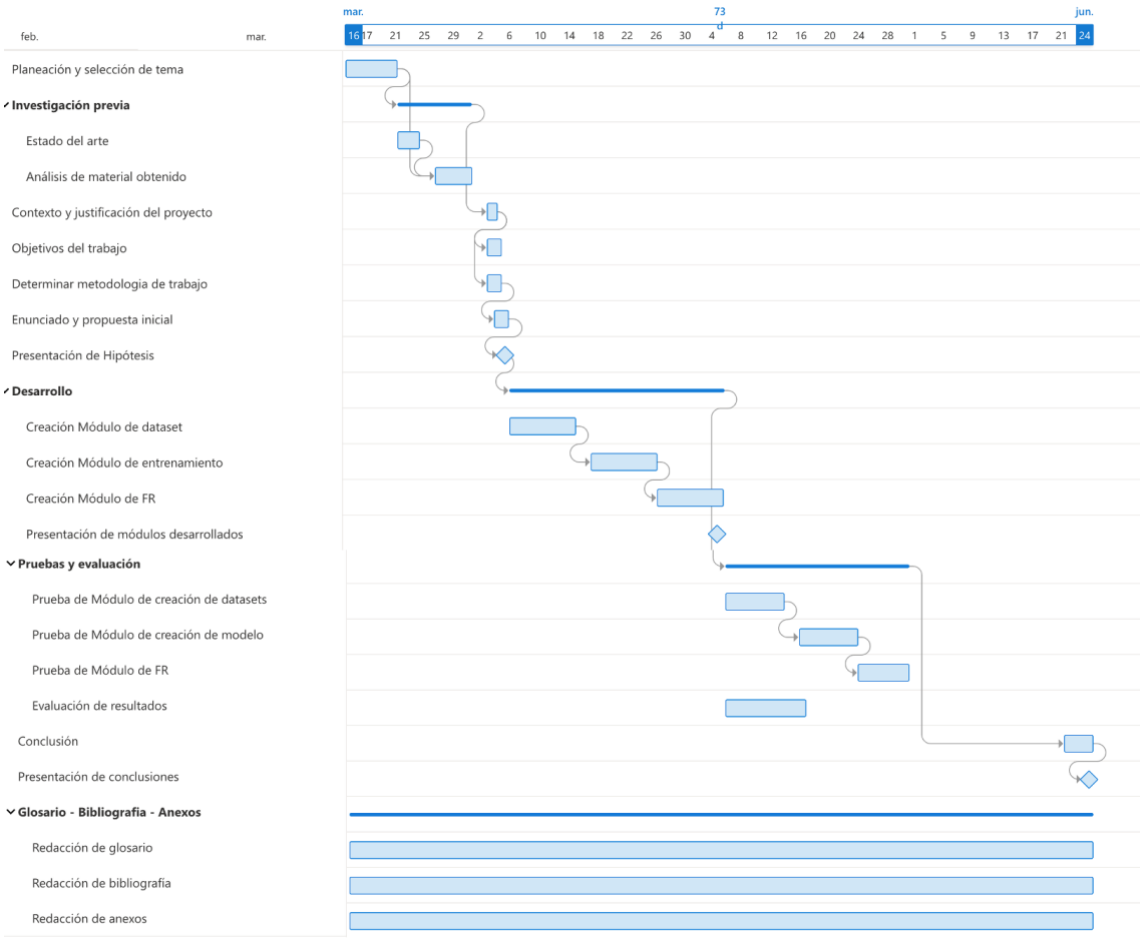

Figura 1: Tareas del proyecto e hitos

1.6 Riesgos

A continuación, se detallan posibles riesgos para llevar adelante este proyecto. Esta lista debe ser actualizada constantemente para encontrar la mitigación que corresponda.

Impacto / Probabilidad

Figura 2: Matriz de probabilidad de impacto

Con la siguiente lista, se pueden obtener los casos más problemáticos combinando el impacto y la probabilidad de que suceda.

1: Riesgo: Posibles problemas para la obtención de imágenes de videos. Imágenes de personas. Permisos para trabajar sobre esas imágenes.
Medida mitigadora: Se utilizarán imágenes propias y/o de personas cercanas que autoricen su uso.
Impacto/ Probabilidad: Alto/ Media

2: Riesgo: Utilización de frameworks y/o herramientas desconocidas
Medida mitigadora: Se deberá dedicar un extra de tiempo al conocimiento de los nuevos frameworks o herramientas con las cuales se decida trabajar.
Impacto/ Probabilidad: Alto/ Media

3: Riesgo: Disponibilidad de conexión a internet y acceso a materiales para la realización del proyecto.
Medida mitigadora: Se deberá estar en un entorno de trabajo compatible con las herramientas mínimas necesarias para llevar adelante este proyecto, en este caso conexión a internet
Impacto/ Probabilidad: Alto / Media

4: Riesgo: Alta dificultad para obtener resultados en tiempo y forma.
Medida mitigadora: Optar por el cambio de investigación. Acotar la investigación.
Impacto/ Probabilidad: Alto / Media

5: Riesgo: Tiempo escaso para completar el proyecto
Medida mitigadora: Se deberá maximizar el tiempo para poder alcanzar tiempo y forma los objetivos planteados.
Impacto/ Probabilidad: Medio / Alta

6: Riesgo: Cambios de alcance
Medida mitigadora: Se deberá realizar una definición inicial clara de todo lo esperado.
Impacto/ Probabilidad: Medio / Alta

7: Riesgo: Lectura de material en otro idioma
Medida mitigadora: Se deberá poder leer en otro idioma, posiblemente inglés, o bien saber trabajar con herramientas de traducción.
Impacto/ Probabilidad: Medio /Media

8: Riesgo: Falta de claridad al momento de diseño de las pruebas.
Medida mitigadora: Se deberá tener una visión clara de lo que se espera en el momento del desarrollo de las pruebas.
Impacto/ Probabilidad: Medio / Media

9: Riesgo: Falta de claridad al momento de la evaluación de resultados.
Medida mitigadora: Se deberá tener una visión clara de lo que se espera con los objetivos de las pruebas.
Impacto/ Probabilidad: Medio / Media

10: Riesgo: No poder tener un ritmo de trabajo adecuado
Medida mitigadora: Se deberá planificar la mayoría de las tareas con tiempo de dedicación para planificarlas.
Impacto/ Probabilidad: Medio / Media

11: Riesgo: Cambio en situación laboral actual
Medida mitigadora: Asegurar la alineación del posible nuevo trabajo con las tareas confirmadas para llevar adelante el proyecto
Impacto/ Probabilidad: Medio / Media

12: Riesgo: Pérdida de documentos
Medida mitigadora: Asegurarse de mantener los documentos salvados en diferentes unidades como disco duro, cloud, correo electrónico.
Impacto/ Probabilidad: Bajo / Media

1.7 Breve sumario de los productos obtenidos

Al finalizar este trabajo, se obtienen tres módulos. Un módulo para la creación de dataset de rostros adaptados para el entrenamiento de modelos de FR. Un segundo módulo para entrenar los modelos de RNA, donde cada modelo se genera a partir de la implementación de técnicas de TL, con una FA que varía de acuerdo con los casos hipotéticos planteados. Y, por último, un tercer módulo de FR, que ejecuta los modelos entrenados, detecta e identifica rostros a partir de la captura de imágenes en DS.

1.8 Breve descripción de los otros capítulos de la memoria

Como se mencionó en el apartado 1.4, este trabajo se apoya en una metodología de investigación, con lo que, la mayoría de los otros capítulos de esta memoria, se utilizan para documentar las diferentes tareas de esta metodología.

Capítulo 2, descripción del estado del arte. Indica los estudios previos, técnicas y herramientas analizadas para trabajar con DL, FR y TL. **Capítulo 3**, planteamiento de la hipótesis nula e hipótesis alternativa. **Capítulo 4**, diseño de los artefactos necesarios para realizar las pruebas, validar o refutar las hipótesis planteadas. Estos diseños permiten, por un lado, crear un dataset de rostros y, por otro lado, entrenar y utilizar un modelo de RNA en un sistema de FR, modelado con TL. El **capítulo 5**, documenta las pruebas realizadas, evalúa y analiza los resultados obtenidos. **Capítulo 6**, indica las conclusiones de la investigación, la satisfacción del trabajo en relación con las hipótesis iniciales, lecciones aprendidas y futuras líneas de investigación. En los **capítulos 7, 8 y 9** se presenta el glosario, bibliografía y anexos.

2 Estado del arte

Revisión de trabajos y análisis de técnicas y herramientas utilizadas para trabajar con DL, FR y TL.

2.1 Trabajos de referencia

Los siguientes trabajos contienen diferentes análisis y comparación de DL, FR, TL y FA.

[2] Describe y repasa conceptos acerca del estado del arte del DL. Los últimos campos de la Inteligencia Artificial (AI) donde el DL está dando soporte, así como el funcionamiento de sus algoritmos. Ventajas de utilización del DL. [3] Describe las diferentes implementaciones de DL, por ejemplo: analizar imágenes de productos en una cadena de producción para clasificarlos de forma automática. Detectar tumores en escáneres cerebrales, entre otros. [4] Describe la utilización de las herramientas VGG utilizadas para TL. [5] Describe la implementación del TL. [6] Comparación entre diferentes funciones de activación en las RNA. Función Relu y Swish. [7] Comparación entre diferentes FA en las RNA. Función Mish.

2.2 Tecnologías utilizadas dentro del DL para FR.

Se analizaron los siguientes modelos computaciones y tecnologías para el trabajo dentro de DL con FR.

Redes neuronales artificiales (RNA): basadas en las redes neuronales biológicas (RNB). Son el eje de la metodología del DL. Versátiles, potentes y escalables y van a permitir afrontar tareas más complejas, que se escapan a la utilización se ML. Actualmente, el aumento de la potencia en los ordenadores, por la ley de Moore [12], mejoras en los algoritmos, o mismo, por la industria de los videos juegos, que persiguieron la creación de tarjetas GPU potentes, así como también, las plataformas en la nube, que permiten que el acceso a estas tecnologías, estén al alcance de todo el mundo.

La principal diferencia entre una red neuronal en DL y un algoritmo de ML es la posibilidad que tiene una RNA de evolucionar, calibrar sus algoritmos e ir mejorando con el tiempo. En este tipo de redes, la o las salidas de una capa alimentan la entrada de la siguiente capa. Cada capa calcula la suma ponderada de su entrada (incluido un peso de sesgo) que, dependiendo de la función de activación utilizada, tendrá un valor diferente, el cual se debe adaptar a la necesidad de la investigación.

Al realizarse el entrenamiento, por medio de procesos de propagación y retro-propagación (BP), se irán ajustando los pesos para adaptarse, buscando la optimización de descenso del gradiente. Emulando el funcionamiento de una RNB.

Decimos que una RNA está completamente conectada, cuando cada nodo de la primera capa está conectado con un nodo de la segunda capa.

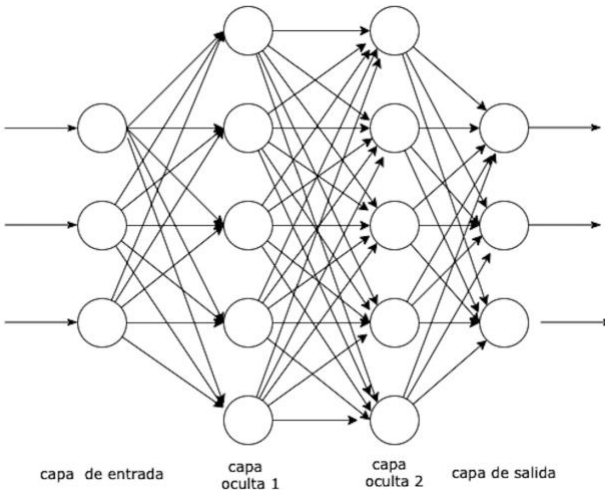

Figura 3: RNA completamente conectada

Deep Learning (DL): Basado en la utilización de RNA, es una metodología utilizada para resolver problemas que no son posible de resolver mediante metodología de machine learning (ML). Donde las neuronas involucradas van a interactuar entre ellas para poder aprender y decidir sin la intervención humana.

Para el caso del FR con imágenes en tiempo real, se utiliza dicha técnica de DL debido a la complejidad y la forma en que se deben buscan los resultados.

Una RNA, al intentar simular una RNB, debe activarse para poder enviar la información al siguiente nodo o capa. Podemos indicar diferentes tipos de entradas / salidas (valores binarios, números).

La siguiente imagen muestra un ejemplo de activación de las neuronas, con valores binarios, dependiendo de lo que se espera (activa / desactivas).

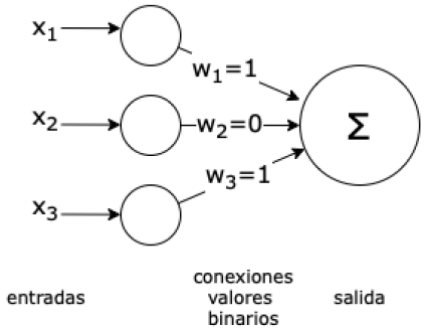

Figura 4: Activación en una RNA con valores binarios

Perceptrón: Es una neurona dentro de una RNA. Este consta de una o más unidades lógicas de umbral (ULU), a diferencia de las RNA que reciben valores binarios, en estas las entradas y salidas son números. Las entradas constituyen la capa entrada y la capa ULU generan resultados, denominada capas de salida.

Estas neuronas pueden incluir, un valor de entrada con sus respectivos pesos, el valor del sesgo, una función de paso y un valor de salida.

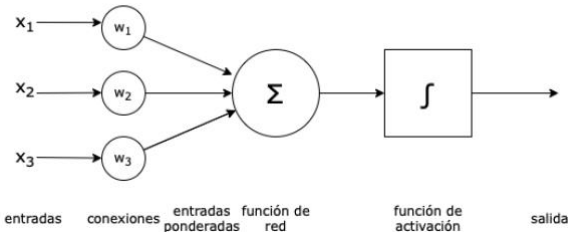

Figura 5: Activación de neuronas con pesos, ponderación y FA

Perceptrón multicapa (PMC): Consta de una capa de entrada, una o varias capas de ULU, llamadas capas ocultas y una última capa ULU llamada capa de salida. Las capas cercanas a la entrada se llaman “capas inferiores” y las próximas a la salida se llaman “capas superiores”. Cada capa, salvo las de salida, incluyen neuronas con valor de sesgo y está completamente conectada a la siguiente capa.

Cuando una RNA está formada por una pila profunda de capas ocultas, decimos que es red neuronal profunda (RNP). Como resultado de un PMC de regresión vamos a recibir un valor resultado de un cálculo. Como resultado de un PMC de clasificación vamos a recibir un porcentaje de clasificación, por ejemplo, el porcentaje en que una imagen representa a una persona. La probabilidad estimada positiva.

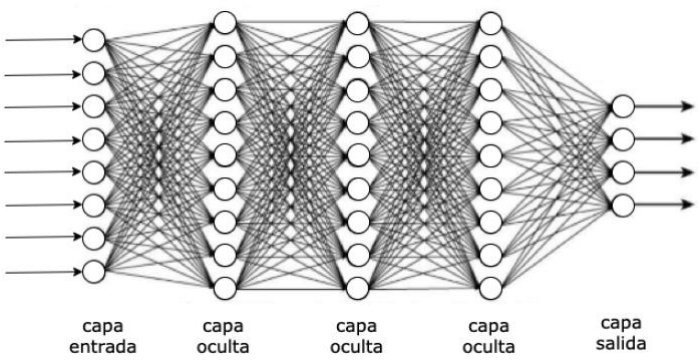

Figura 6: RNA, PMC.

La configuración típica para un PMC de regresión, pueden ser con los siguientes valores de hiperparámetros estimados. Pueden variar según el tipo de tarea a realizar:

Hiperparámetro	Valor típico
# capas ocultas	Entre 1 y 5
# de neuronas por capa oculta	Entre 10 y 100
# de neuronas de salida	1 neurona por dimensión predicción
Tipo de activación	ReLu
Tipo activación de salida	None o ReLu/softplus (positvo) o. Sigmoide / tangente (valor acotado)
Función de perdida	MSE o Huber (outliers)

Tabla 1: configuración típica PMC de regresión.

La configuración típica para un PMC de clasificación, pueden ser con los siguientes valores de hiperparámetros estimados Pueden variar según el tipo de tarea a realizar:

Hiperparámetro	Binaria	Multi-etiqueta Binaria	Multi-etiqueta Clasificación
# capas ocultas	Entre 1 y 5	Entre 1 y 5	Entre 1 y 5
# de neuronas por capa oculta	Entre 10 y 100	Entre 10 y 100	Entre 10 y 100
# de neuronas de salida	1	1 por etiqueta binaria	1 por clase
Tipo activación capa de salida	Sigmoide	Sigmoide	Softmax
Función de perdida	x-entropy	x-entropy	x-entropy

Tabla 2: configuración típica PMC de clasificación.

En este trabajo vamos a trabajar con PMC de clasificación multi-etiqueta. En el cual, se va a utilizar la función de activación Relu, Swish y Mish junto con una FA Softmax para la capa de salida y generar 3 modelos. Donde la función de activación Softmax va a calcular las probabilidades de cada clase objetivo sobre todas las clases posibles, indicando un porcentaje de posibilidad de que un rostro exista entre los valores entrenados.

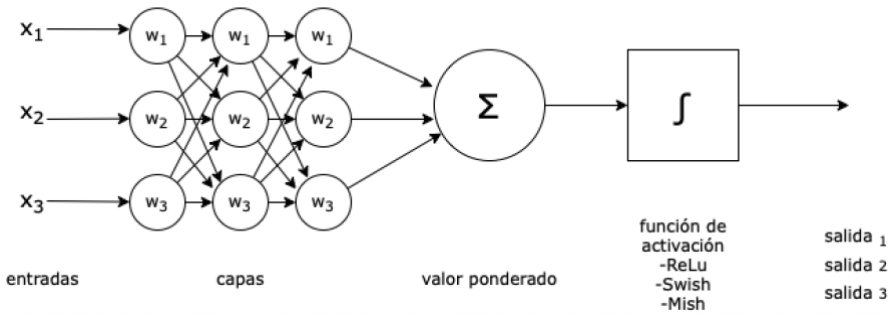

Figura 7: PMC con capas de entrada, capas ocultas, capas de salida, funciones Relu, Swish, Mish.

Retro-propagación (BP): El algoritmo de retro-propagación calcula el gradiente de error de la red respecto a cada parámetro del modelo. Averigua como se debe ajustar cada peso de conexión y cada sesgo para reducir el error. Lo primero que hace es una predicción (paso hacia adelante) y mide el error, luego vuelve por cada capa en el orden inverso para medir la contribución a cada error (paso hacia atrás) y, con esta información, ajusta los pesos de conexión para reducir el error

(paso de descenso del gradiente). El BP permite que la red aprenda la asociación existente entre las capas de entradas y salidas

Red neuronal artificial convolucional (CNN): [8] Indica la eficacia de la utilización de las CNN en FR, por esto se optó por la utilización de una red convolucional (CNN) para resolver este problema de detección y clasificación de rostros. Este tipo de red se corresponden con un tipo de RNA y es una variación de un PMC. Utiliza matrices bidimensionales, efectivas para las tareas de visión artificial, clasificación y segmentación de imágenes, imágenes en tiempo real, entre otras aplicaciones.

Las CNN pueden aprender a reconocer objetos dentro de una imagen, para lo cual necesitan ser entrenadas. Estas redes pueden, dentro de un proceso de aprendizaje de un algoritmo, captar características únicas y generalizarlo. Son redes de clasificación que procesan la información imitando al córtex visual humano, para lo cual utiliza capas convolucionales.

Capas de convoluciones: una de las tareas de las CNN, es la realización de operaciones matemáticas de producto escalar de matrices, para generar una nueva matriz de salida, esta tarea se llama “convulsiones”, donde se define una matriz más pequeña o kernel que ira recorriendo los pixeles de la imagen de entrada y generando una matriz de salida que se pasará a la siguiente capa oculta. Al conjunto de kernel lo llamaremos filtro y a las matrices de salidas la llamaremos “feature mapping” o mapeo de características

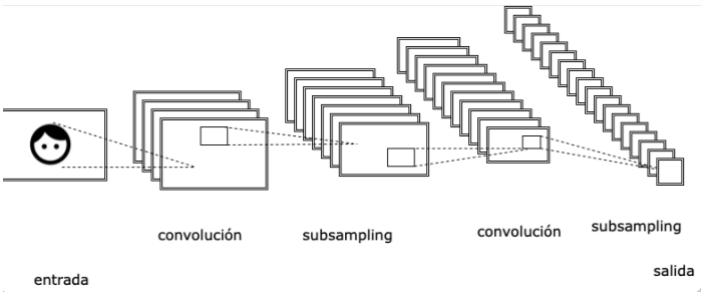

Figura 8: Ejemplo de convolución

Capas pooling: estas capas son necesarias para simplificar o reducir las matrices que se generan en las capas convolucionales. Un ejemplo de pooling puede ser el tipo Max pooling donde por cada filtro se toma el valor más grande en la matriz origen para generar la matriz de salida, haciendo que se reduzca la cantidad de neuronas necesarias y que se requiera menor capacidad de procesamiento.

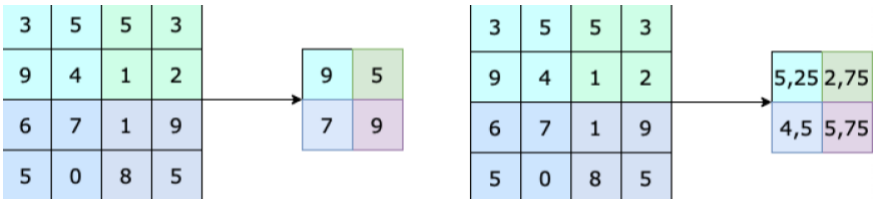

Figura 9: Ejemplo de capa pooling max y averange

Funciones de activación: las funciones de activación van a permitir adaptar los algoritmos para obtener el valor que se necesita, limitando el rango de salida de la neurona. Es una solución para los casos donde el resultado no se puede obtener en una operación lineal.

Al utilizar funciones de activación, cada neurona, enviará el resultado de la función de activación, junto con los valores de la suma ponderada y de sesgo a la siguiente capa.

Las funciones de activación van a dar diferentes formas a los valores de salida.

Una RNA se podrá permitir, con el número indicado de capas, neuronas y funciones de activación, adaptarse mejor a la solución de un problema.

A continuación, se grafican las funciones de activación para RNA analizadas. Las más utilizadas para las CNN como es la FA Relu o las más actuales como son, Mish o Swish.

Figura 10: Tipos de FA para RNA: ReLu, Mish, Swish

2.3 Técnicas para el entrenamiento de RNA

Las siguientes son técnicas utilizadas para el entrenamiento de RNA

Aprendizaje supervisado (AS) y no supervisado (ANS): Dentro del ML y DL la técnica de AS busca reproducir un valor conocido dentro de un conjunto de datos entrenados. En el caso de ANS no se conocer el valor que se está buscando. Este es un trabajo que se corresponde con un AS.

Transferencia de conocimiento (TL): Dentro del ML y DL se intenta aprovechar el conocimiento previo. En las RNA mediante la utilización del TL un modelo pre-entrenado para resolver un problema puede ser reutilizado para una nueva tarea que tenga similares características [14]. Por ejemplo, una CNN que se entrenó para detectar y reconocer un tipo de animal, se puede reutilizar para otro tipo de animales. Esto va a permitir aprovechar modelos que fueron pre-entrenados con gran cantidad de datos, reduciendo en gran medida los tiempos de aprendizajes, sacando provecho del conocimiento previamente adquirido. Consiste en tomar características aprendidas y aplicarlas a un problema similar, principalmente se utiliza en los casos donde la complejidad para entrenar un modelo desde cero es alta.

Las tareas normales para un proceso de TL son:
1- Tomar el modelo con las capas del modelo pre-entrenado.
2- Mantener las capas congeladas para no perder la información.
3- Por encima de las capas congeladas, agregar las nuevas capas necesarias. Las capas nuevas y las existentes aprenderán a generar predicciones sobre un nuevo conjunto de datos.
4- Entrenar el modelo adaptado con el conjunto de datos del problema a solucionar.
5- Predicción con el nuevo modelo.

Otra opción es descongelar las capas o parte de las capas pre-entrenadas, para volver a entrenarlas y conseguir resultados más finos sobre los valores esperados.

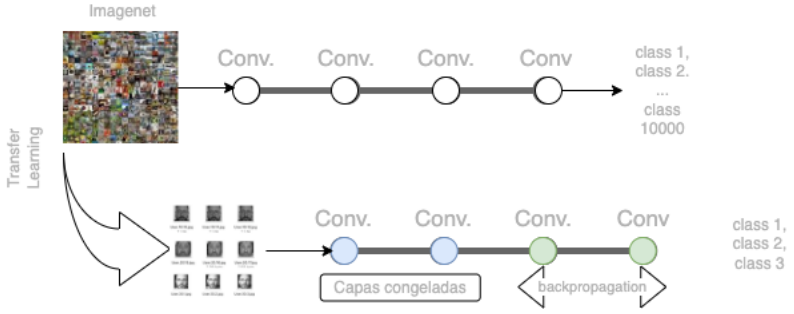

Figura 11: CNN con TL

En este trabajo se va a utilizar el TL para obtener un modelo de CNN que permitan reconocer rostros, junto con la librería de OpenCV con haar-cascades y la CNN VGG con ImageNet. Para el entrenamiento se utilizará Google Colab, el cual va a permitir trabajar con el lenguaje Python para ajustar estas CNN, con diferentes FA y realizar entrenamientos de mayor velocidad, ya que posee la opción de utilizar una máquina virtual con GPU para entrenar nuestro modelo.

Para la Ejecución local del programa Python se utilizará la herramienta de libre distribución, Anaconda.

2.4 Frameworks y herramientas de DL para FR.

A continuación, se detallan los frameworks y herramientas de DL analizados para ser utilizados con FR.

Tensorflow (TF): Framework creado por Google Brain Team. Actualmente es una interfaz de código abierto, que permite la implementación y creación de algoritmos de Machine Learning (ML) y DL. Los algoritmos realizados en TF son válidos para ser utilizados en diferentes dispositivos y sistemas tanto distribuidos a gran escala y/o dispositivos como tarjetas CPU / GPU / TPU y se encuentra disponible para Windows, Linux, MAC OS y plataformas móviles que incluyan Android y IOS.
TF permite entrenamientos e inferencias para modelos de RNA. La última versión estable es la versión 2.0.0 (publicada el 2019-10-01). [23].
En TF los nodos en el grafo representan operaciones matemáticas, mientras que las conexiones o links del grafo representan los conjuntos de datos multidimensionales (tensores).

Keras: Es una librería de código abierto, desarrollada en Python, para DL a alto nivel, que va a permitir crear, entrenar, evaluar y ejecutar todo tipo de RNA. Keras tiene una implementación multi-motor, así como también, una implementación exclusiva de TF para facilitar el trabajo con RNA. Esta librería va a permitir construir CNN y poder entrenar modelos de DL. Actualmente con la biblioteca de TL es posible disponer de Keras, teniendo acceso a capas, funciones objetivas, funciones de activación y/o optimizadores matemáticos. Keras será la opción para este trabajo junto con TF para agilizar los resultados.

Sklearn: biblioteca para aprendizaje automático, de uso libro para Python. Proyecto de Google Sumer of Code. Incluye y permite trabajar con varios algoritmos de clasificación, regresión y análisis de grupos.

OpenCV – cascades classifier [15]: OpenCV es una librería de código abierta para análisis de visión artificial (Visión artificial abierta). Originalmente creada por Intel corporation. Multiplataforma: GNU/Linux, Windows, Mac OS x, Android, IOs. Cuenta con una documentación actualizada para diferentes lenguajes de programación como Python, Java, C++ y Javascript.
Entre sus diferentes usos nos centramos en la opción de cascades classifier. Este clasificador, basado en el método de clasificación propuesto por Paul Viola and Michael Jones [21]. Este algoritmo se puede utilizar para la detección de objetos en tiempo real pero también para la detección de rostros. Se considera que su tasa detección es alta [22] en comparación con otros algoritmos para la detección de rostros.

Visual geometric group (VGG) – ImageNet [16][17]: Creada por el departamento de ciencias la Universidad de Oxford. Permite la convolución de las imágenes ingresadas. Funciona con el ingreso de una imagen RGB del tamaño 224x224, filtro de 3x3 o 1x1 y paso de convolución fijo.
El VGG11 mínimo tiene 8 capas convolucionales y 3 capas completamente conectadas, mientras que, el VGG19 máximo tiene 16 capas convolucionales y

3 capas totalmente conectadas. Además, la red VGG no es seguida por una capa de agrupación detrás de cada capa convolucional, un total de 5 capas de agrupación distribuidas bajo diferentes capas convolucionales.

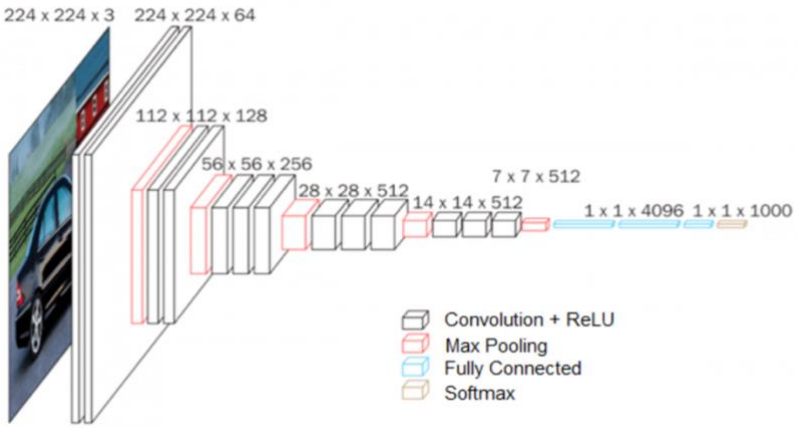

Figura 12: VGG - Diagrama de estructura

Método Secuencial: Todas las capas del modelo se organizan de forma secuencial.

Capas en CNN de VGG:

- Dense: capa de cálculo, conecta una cada neurona de una capa con todas las salidas de la capa anterior.
- Conv2D: Capa que genera las convoluciones entre la capa de entrada y la de salida dentro de un CNN.
- Maxpool2d: Agrupación máxima de señales en 2d, para una entrada que puede tener varios planos de entrada.
- Flatten: aplana la entrada. No afecta el tamaño del lote.

ImageNet: al iniciar la configuración para un CNN se espera que se le indiquen los valores de peso que se envían entre capas. Valores útiles para poder clasificar imágenes. Imaginet, va a brindar los pesos iniciales tomados en el preentrenamiento con miles de imágenes. Estos pesos demandan mucha precisión y tiempo en ser conseguidos. Seguido a esto, se pueden agregar capas que se podrán entrenar con nuevos valores.

Python [18]: lenguaje de programación de alto nivel, permite trabajar con todas las librerías necesarias para la creación de modelos de RNA. Conocido por su simplicidad para su utilización y aprendizaje. Tiene un gran uso para desarrollar programas de ML o DL, entre otro tipo de sistemas.

Google Collab [19]: creado por Google research. Permite desarrollar y ejecutar código Python. Utilizado diferentes técnicas dentro del ML. Para los entrenamientos de modelos de RNA, posee máquinas virtuales que aumentan la velocidad del aprendizaje trabajando de forma de CPU o GPU. Permite compartir proyectos desde la nube.

Anaconda [20]: Entorno para lenguajes de programación Python y R. De uso libre, normalmente, está relacionado a las ciencias de datos, ML, DL. Permite trabajar con grandes volúmenes de datos y logra simplificar el manejo de los despliegues de entornos de trabajo, creación y ejecución de código.

3. Hipótesis

En esta sección describimos las interrogantes planteadas en esta investigación.

En este caso se analiza si en un sistema de FR, modelado a partir de TL, obtiene resultados más satisfactorios al ser entrenado con una FA moderna que al ser entrenado con una FA de las que se utilizan frecuentemente.

3.1 Formulación de hipótesis

La base de esta investigación se centra en comparar diferentes modelos de CNN creados a partir de TL para ser utilizados en FR con imágenes en DS.

¿Es posible obtener un sistema de FR modelado a partir de TL para imágenes en DS que obtengan resultados aceptables de clasificación?, ¿Se podría sustituir el entrenamiento de modelos de CNN y utilizar FA más actuales para obtener resultados óptimos? ¿Supone una ventaja el utilizar las FA Swish o Mish, en los sistemas de FR creados con TL para imágenes con DS en relación con la FA ReLu?

En un sistema de FR modelado por TL, la utilización de una FA Swish o Mish obtiene resultados más satisfactorios que una FA ReLu.

4. Diseño

> En este capítulo se detalla la funcionalidad de los diferentes módulos, su configuración y los resultados esperados.

4.1 Detalle para el diseño de las pruebas

Módulo de creación del dataset

Para este caso, el dataset, se crea con imágenes de rostros de personas. Por cada persona se obtiene un dataset con 100 imágenes, donde pare de las fotos son utilizadas para el entrenamiento del modelo y el restante son utilizado para la validación del modelo.
El dataset guarda imágenes de rostros. Para evitar cualquier tipo de sesgo en relación con el color de la piel de las personas, estas imágenes serán capturadas en una escala de grises. Aprovechando la librería pre-entrenada de OpenCV – haar cascade, se creará el dataset solo capturando los rostros dependiendo de

los landmarks correspondientes. Se almacenan en dos carpetas, “Train” y “Test”, por cada carpeta se crean directorios con los nombres o etiqueta de las personas a las que corresponden cada set de imágenes.

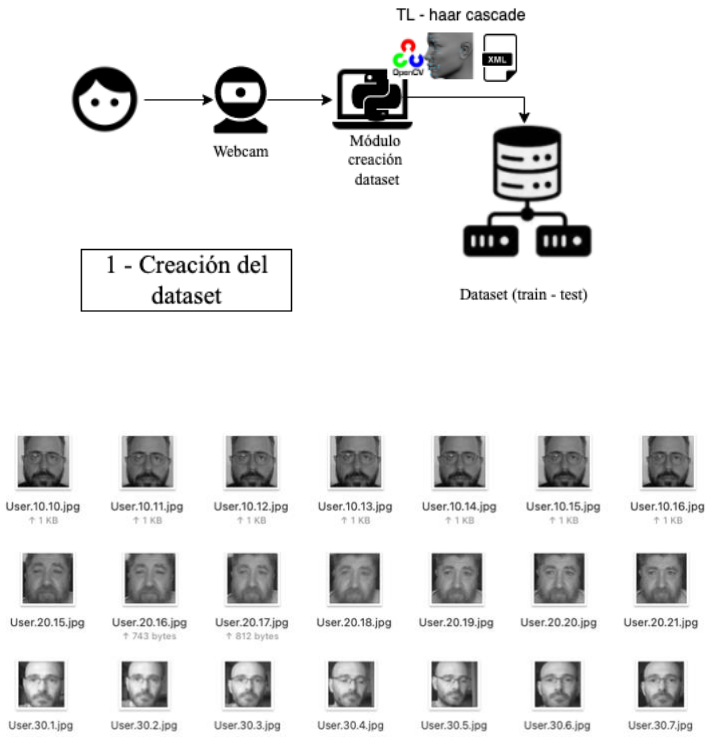

Figura 13: Módulo de creación del dataset

Módulo de entrenamiento del modelo

Obtención de modelo pre-entrenado: Para obtener el modelo pre-entrenado de la CNN, aprovechando la utilización de TL de VGG con “imagenNET” es obtiene los valores de pesos para la inicialización del entrenamiento de la CNN.

Reentrenamiento del modelo con FA (Relu / Swish / Mish) + Softmax: Se toma el modelo pre-entrenado. Por medio de Keras y Tensorflow, Se agregan capas necesarias con la FA: Relu / Swish / Mish, junto la FA Softmax. Se vuelve a generar un modelo por cada una de las FA que se están analizando, un modelo con FA Relu y las capas de la CNN necesarias y una FA Softmax de salida. y que cumplan con el porcentaje de reconocimiento esperado. Se obtienen los resultados de las métricas accuracy, loss y AUC del entrenamiento con cada FA como del porcentaje de la correcta clasificación de las clases de rostros entrenados.

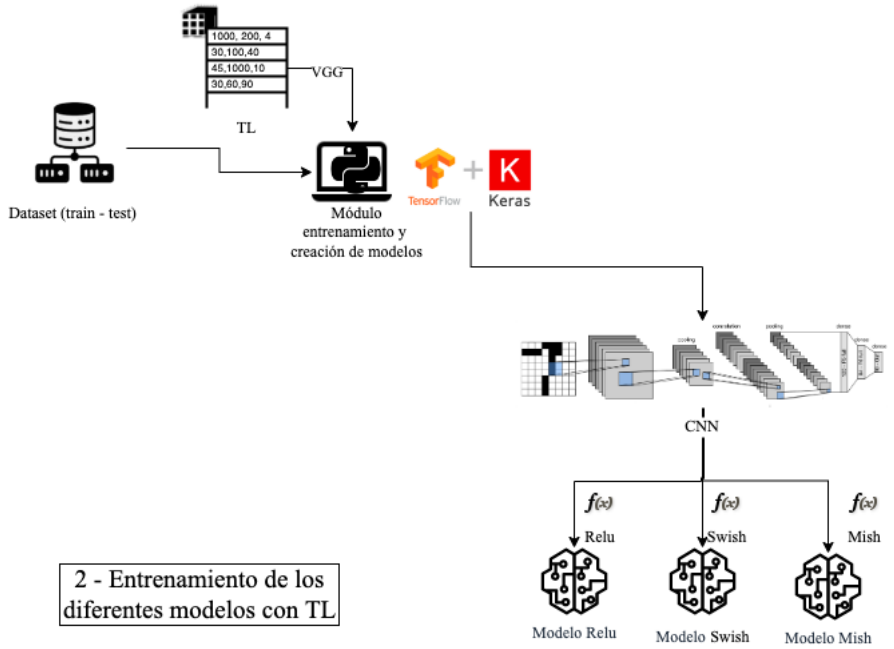

Figura 14: Módulo de entrenamientos del modelo. ReLu, Swish y Mish

Módulo de FR

Obtención de imágenes y reconocimiento facial: Por medio de una cámara web se capturan las imágenes DS. Se realiza el reconocimiento facial con cada uno de los modelos entrenados, el modelo-Relu, el modelo-Swish y el modelo-Mish.

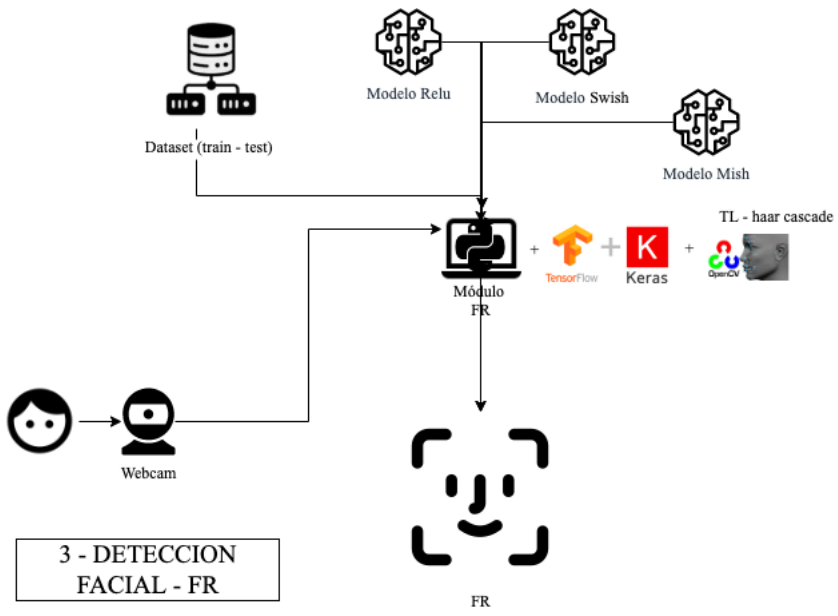

Figura 15: Módulo de detección y FR

5. Pruebas y análisis de resultados

Como parte de las validaciones de las hipótesis planteadas, se detallan las pruebas realizadas, donde se busca comparar el rendimiento de diferentes modelos de CNN, entrenados para ser utilizados en FR, creado a partir de TL, con diferentes FA.

Estas pruebas se componen de los pasos de creación de dataset, entrenamiento y FR. En el apartado de anexos se encuentran indicados los enlaces para acceder al código de los diferentes módulos utilizados para estas pruebas.

5.1 Pruebas

Creación de dataset: ejecutando el módulo de creación de dataset. Se genera inicialmente un dataset de 100 fotos por cada usuario. Cada usuario se genera con el prefijo “User.” y un valor de 10 en 10. Se clasificaron los usuarios: ['User.10', 'User.20', 'User.30']. Se crean dos carpetas, una para almacenar las fotos de entrenamiento y otra las fotos de validación [‘train’,’test’]. Dentro de la carpeta /datasets/train se crean una carpeta por cada usuario, lo mismo dentro de la carpeta /datasets/test. Las imágenes de rostros están adaptadas para ser ingresadas como parte de una CNN y cuya captura está limitada al rostro del usuario por el uso de OpenCV haar-cascade, ‘haarcascade_frontalface_default.xml’, pre-entrenado para detectar los landmarks del rostro.

Entrenamientos del modelo: se entrenan 3 modelos, el primero con FA-Relu, el segundo con FA-Swish, el tercer con FA- Mish. Para cada entrenamiento se toma el conjunto de dataset generado en el módulo de creación de dataset [‘train’,’test’]. Para la creación de la CNN se utiliza la red pre-entrenada convolucional de VGG16, que define las diferentes capas necesarias para un CNN. En la CNN obtenida por TL, se mantienen sin modificar sus capas. Se agregan 2 capas densas, en la prueba 1 y 3 capas densas en el resto de las pruebas, que se configuran con un tipo de FA (‘Relu’, ‘Mish’ o ‘Swish’), que varía dependiendo del entrenamiento. En el primer entrenamiento la FA es ‘ReLu’, en el segundo la FA es ‘Swish y en el tercero la FA es ‘Mish. También se agrega una capa ‘softmax’, que va a permitir obtener un valor de porcentual como resultado de la clasificación. Para cargar el valor inicial de pesos de la CNN se utiliza los valores pre-entrenados de ‘Imagenet’. Como ingreso en CNN pre-entrenada, las imágenes se adaptan a una forma “224,224,3”, esto es, 224x224 pixeles y del tipo RGB. Se definen para el entrenamiento que se compila con los parámetros de loss='categorical_crossentropy', por tratarse de una clasificación de categorías no binaria, algoritmo de optimización 'Adam', utilizado de forma estándar para las clasificaciones, con las métricas ‘accuracy’, para la frecuencia de aciertos en la clasificación, ‘mse’ (media del error cuadrático), promedio de errores al cuadrado y ‘AUC’ (área bajo la curva) medida de precisión del funcionamiento del modelo.

Se ejecutan, por cada entrenamiento entre 10 y 12 epochs, con una cantidad de pasos por epochs y los pasos de validación igual a las clases a clasificar. Una vez entrenado, el modelo se guarda para ser utilizado en el módulo de FR.

Pruebas de FR

Para las pruebas de este módulo, se incluye el modelo entrenado en el módulo en el módulo de entrenamiento dentro del módulo de FR. Este módulo accede a la webcam conectada en el ordenador de ejecución. Utilizando el modelo pre-entrenado de reconocimiento de landmarks de rostros OpenCV, haar-cascade haarcascade_frontalface_default.xml, va a reconocer los rostros que aparecen en el DS. En caso de que los rostros detectados corresponden con los entrenados, hace su reconocimiento indicando el usuario, ['User.10', 'User.20', 'User.30'].

Prueba 1: Se inicia la primera prueba configurando la arquitectura de la CNN con una FA ReLu. Se mantienen la arquitectura de capas de VGG16. Se cargan los pesos iniciales de Imagenet. Con total de 134,272,835 parámetros, de los cuales son los entrenados 134,272,835. Se utiliza un dataset con un total de 210 imágenes para el entrenamiento y 90 para la validación. 3 es el total de clases.

TL. Configuración inicial	Valores
CNN - Inputs Layer	VGG16 – Conv2D [224,224 …14, 14] max_pooling2d
CNN - Outputs Layer	2 Dense, ReLu 1 Softmax, ReLu
FA	Relu
Weights	Imagenet
Total params:	134,272,835
Trainable params:	134,272,835
Non-trainable params	0
datasets	Total 210 imágenes de entrenamiento y 90 imágenes de validación. 70 imagen - 3 clases - train 30 imagen - 3 clases - test
Epochs	10

Tabla 3: Configuración entrenamiento prueba 1.

Prueba 2: Para la siguiente prueba se modifica el dataset aumentando el número de imágenes incluidas en el dataset. Se pasan de 210 a 1497 imágenes train y de 90 a 537 training, en total para las 3 clases. Se mantienen la arquitectura de capas de VGG16. Se agregan al final de la CNN 3 capas densas con FA 'Relu' y una más con FA 'Softmax'. Se cargan los pesos iniciales de Imagenet. Con total de 19,990,743 parámetros, de los cuales son los entrenados 5,276,055, el resto 14,714,688 se mantienen con los valores obtenidos por TL. Se utiliza un

dataset con un total de 1497 imágenes para el entrenamiento y 537 para la validación. 3 es el total de clases.

TL. Configuración inicial	Valores
CNN - Inputs Layer	VGG16 – Conv2D [224,224 …14, 14] max_pooling2d
CNN - Outputs Layer	3 Dense - Relu 1 Softmax - Relu
FA	Relu
Weights	Imagenet
Total params:	19,990,743
Trainable params:	5,276,055
Non-trainable params	14,714,688
datasets	1500 imagen - 3 classes. 600 imagen - 3 classes.
Epochs	12

Tabla 4: Configuración entrenamiento prueba 2 - ReLu.

Prueba 3: se configura una arquitectura CNN TL de VGG16. Se agregan al final de la CNN 3 capas densas con FA 'Swish' y una capa más con FA 'Softmax'. Pesos iniciales de Imagenet. Con un total de 19,990,743 parámetros, de los cuales son los entrenados 5,276,055, el resto 14,714,688 se mantienen con los valores obtenidos por TL. Se utiliza un dataset con un total de 1497 imágenes para el entrenamiento y 537 para la validación. 3 es el total de clases.

TL. Configuración inicial	Valores
CNN - Inputs Layer	VGG16 – Conv2D [224,224 …14, 14] max_pooling2d
CNN - Outputs Layer	3 Dense - Swish 1 Softmax - Swish
FA	Swish
Weights	Imagenet
Total params:	19,990,743
Trainable params:	5,276,055
Non-trainable params	14,714,688
datasets	1500 imagen - 3 clases. 600 imagen - 3 clases.
Epochs	12

Tabla 5: Configuración entrenamiento prueba 3 - Swish.

Prueba 4: se configura una arquitectura CNN TL de VGG16. Se agregan al final de la CNN 3 capas densas con FA 'Mish', una capa más con FA 'Softmax'. Pesos iniciales de Imagenet. Con un total de 19,990,743 parámetros, de los cuales son los entrenados 5,276,055, el resto 14,714,688 se mantienen con los valores

obtenidos por TL. Se utiliza un dataset con un total de 1497 imágenes para el entrenamiento y 537 para la validación. 3 es el total de clases.

TL. Configuración inicial	Valores
CNN - Inputs Layer	VGG16 – Conv2D [224,224 …14, 14] max_pooling2d
CNN - Outputs Layer	3 Dense - Mish 1 Softmax - Mish
FA	Mish
Weights	Imagenet
Total params:	19,990,743
Trainable params:	5,276,055
Non-trainable params	14,714,688
datasets	1500 imagen - 3 clases. 600 imagen - 3 clases.
Epochs	12

Tabla 6: Configuración entrenamiento prueba 3 - Swish.

Prueba 5: repetición de la prueba 1 aumentando el número de epochs.

5.2 Análisis de resultados

Prueba 1:

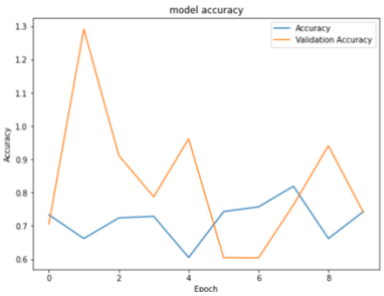

Figura 16: Coste de la ejecución prueba 1

```
Confusion Matrix
[[ 0 30  0]
 [ 0 30  0]
 [ 0 30  0]]
Classification Report
              precision    recall  f1-score   support

     User.10       0.00      0.00      0.00        30
     User.20       0.33      1.00      0.50        30
     User.30       0.00      0.00      0.00        30

    accuracy                           0.33        90
   macro avg       0.11      0.33      0.17        90
weighted avg       0.11      0.33      0.17        90
```

Figura 17 Matriz de confusión / métricas clasificación

Al ejecutar el modelo con esta configuración. El resultado en la matriz de confusión arrojo valores que indican que el modelo inicial, con la primer FA Relu necesitaba ser ajustado. Se decide aumentar y mejorar la cantidad de imágenes en el dataset, y modificar la arquitectura de la CNN.

Prueba 2: ReLu

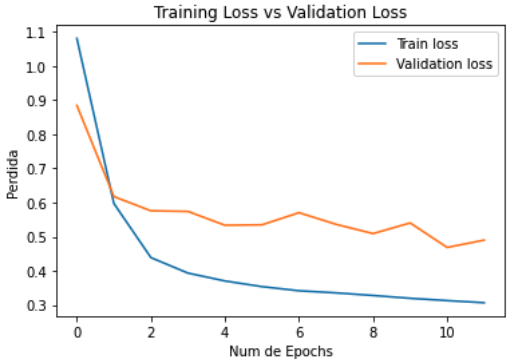

Figura 18

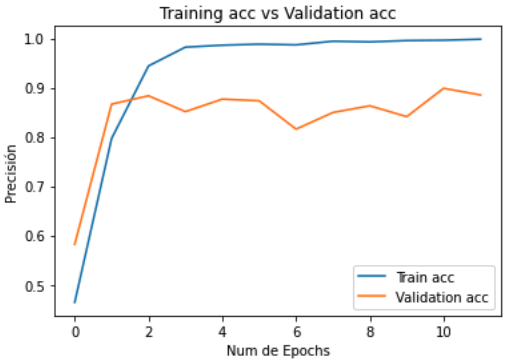

Figura 19

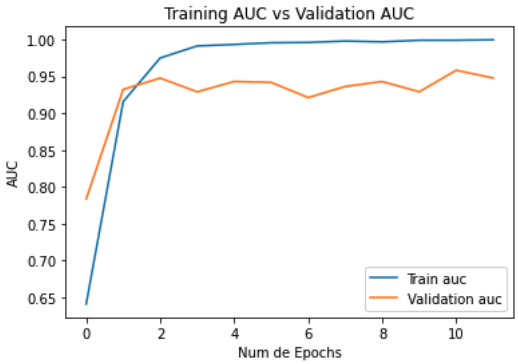

Figura 20

```
Confusion Matrix
[[81 53 61]
 [64 61 74]
 [80 50 67]]
Classification Report
              precision    recall  f1-score   support

     User.10       0.36      0.42      0.39       195
     User.20       0.37      0.31      0.34       199
     User.30       0.33      0.34      0.34       197

    accuracy                           0.35       591
   macro avg       0.35      0.35      0.35       591
weighted avg       0.35      0.35      0.35       591
```

Figura 21

```
6/6 [==============================] - 3s 492ms/step - loss: 0.4899 - acc: 0.8849 - mse: 0.0933 - auc_9: 0.9477
[0.4899207353591919,
 0.8849408030509949,
 0.09326927363872528,
 0.9477096796035767]
```

Figura 22

Figura 18: Prueba 2 – Loss
Figura 19: Prueba 2 – Accurancy
Figura 20: Prueba 2 – AUC
Figura 21: Prueba 2 – Confusion Matrix
Figura 22: Prueba 3 – AUC

En esta prueba observamos que, la perdida en el entrenamiento disminuye progresivamente, llegando al epochs 12 con una perdida alrededor de 0.3. En cuanto a la precisión a medida que avanzan los epochs, se puede ver que va aumento y llegado el epochs 3 la precisión es cercana al 1. Desde el epochs 3 al 12 la precisión se mantiene elevada para finalizar con un valor de 0.9980. En

cuanto al AUC al finalizar el entrenamiento obtiene un valor de 0.9477. Tomando los valores clasificados como verdaderos positivos, se observa que para la primera clase de 195 imágenes, 81 fueron clasificados correctamente, de la segunda clase con un total de 199, 61 fueron clasificadas correctamente y la tercera clase de un total de 197 clases 67 fueron clasificadas como verdaderos positivos.

Prueba 3: Swish

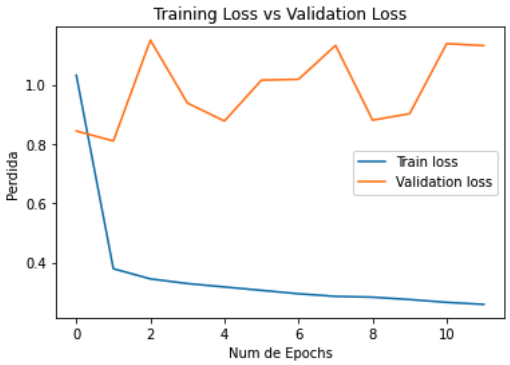

Figura 23

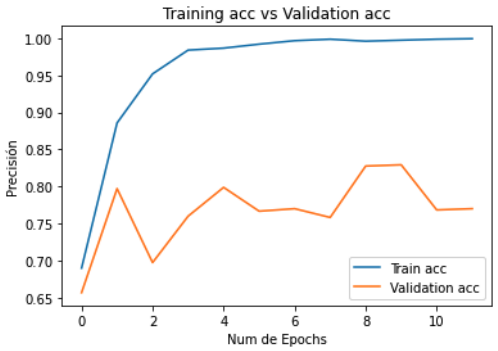

Figura 24

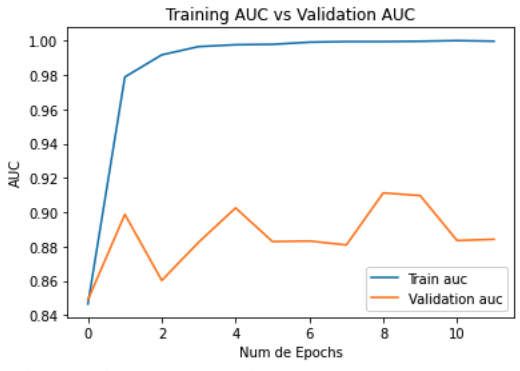

Figura 25

```
Confusion Matrix
[[ 56  30 109]
 [ 68  30 101]
 [ 65  32 100]]
Classification Report
              precision    recall  f1-score   support

     User.10       0.30      0.29      0.29       195
     User.20       0.33      0.15      0.21       199
     User.30       0.32      0.51      0.39       197

    accuracy                           0.31       591
   macro avg       0.31      0.32      0.30       591
weighted avg       0.32      0.31      0.30       591
```

Figura 26

```
6/6 [==============================] - 3s 493ms/step - loss: 1.1318 - acc: 0.7699 - mse: 0.1348 - auc_10: 0.8843
[1.1317634582519531,
 0.769881546497345,
 0.13476160168647766,
 0.88426285982131961]
```

Figura 27

Figura 23: Prueba 3 - Loss
Figura 24: Prueba 3 - Accurancy
Figura 25: Prueba 3 – AUC
Figura 26: Prueba 3 – Confusion Matrix
Figura 27: Prueba 3 – AUC

En esta prueba observamos que llegado al segundo epochs de entrenamiento hay una gran disminución de la perdida, la cual continúa disminuyendo hasta el epochs 12 donde alcanza un valor cercano al 0. En cuanto a la precisión en el entrenamiento, llegado el epochs 4 alcanza un valor cercano al 1 y continua de esa manera hasta el final del entrenamiento donde termina con un valor de 0.993. En cuanto al AUC finalizado el entrenamiento obtiene un valor de 0.8843.
Para la primera clase, con un total de 195 imágenes, considera verdaderas positivas 56. Para la segunda clase de 199 son clasificadas 30 como verdaderas positivas. Para la tercera clase de 197, como verdaderas positivas se clasifican 100.

Prueba 4: Mish

Figura 28

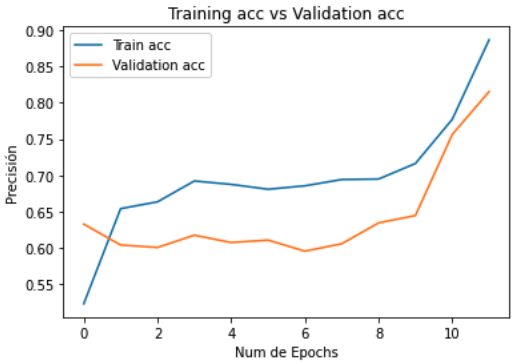

Figura 29

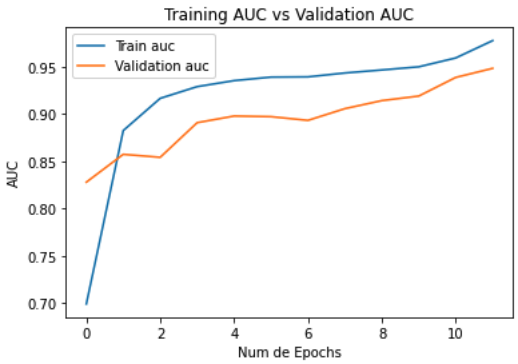

Figura 30

```
Confusion Matrix
[[56 70 69]
 [35 84 80]
 [50 78 69]]
Classification Report
              precision    recall  f1-score   support

    User.10       0.40      0.29      0.33       195
    User.20       0.36      0.42      0.39       199
    User.30       0.32      0.35      0.33       197

    accuracy                          0.35       591
   macro avg       0.36      0.35      0.35       591
weighted avg       0.36      0.35      0.35       591
```

Figura 31

```
6/6 [==============================] - 3s 505ms/step - loss: 0.5724 - acc: 0.8156 - mse: 0.1119 - auc_11: 0.9482
[0.572442352771759, 0.8155668377876282, 0.11185085028409958, 0.948157787322998]
```

Figura 32

Figura 28: Prueba 4 - Loss
Figura 29: Prueba 4 – Accurancy
Figura 30: Prueba 4 – AUC
Figura 31: Prueba 4 – Confusion Matrix
Figura 32: Prueba 4 – AUC

En esta prueba la perdida desde el epochs 1 comienza a disminuir y lo hace hasta el epochs 12 donde termina con una pérdida de 0.4640. En cuanto a la precisión en el entrenamiento, el desde el epochs 2 al 9 se mantiene con un valor estable cercano al 0.7, para finalizar con un valor de 0.08870. El AUC en el entrenamiento genera un valor 0.948. En cuanto a los verdaderos positivos para la primera clase, de 195 se consideran 56, para la segunda clase 84 de 199 y para la tercera clase 69 sobre 197.

Prueba 5:

```
Confusion Matrix
[[67 44 84]
 [65 61 73]
 [57 43 97]]
Classification Report
              precision    recall  f1-score   support

     User.10       0.35      0.34      0.35       195
     User.20       0.41      0.31      0.35       199
     User.30       0.38      0.49      0.43       197

    accuracy                           0.38       591
   macro avg       0.38      0.38      0.38       591
weighted avg       0.38      0.38      0.38       591
```

Figura 33: Prueba 5 – Matriz de confusión.

Tras aumentas el número de epochs, con el fin de mejorar los valores de la clasificación. Estos niveles, para el modelo inicial con FA ReLu, continúan con valores que tiene amplia de perdida.

Prueba 6: Face Recognition

Con los modelos entrenados, estos se cargaron en el módulo de FR. De las pruebas realizadas mediante la captura de imágenes por DS. Se obtuvieron resultados que se puede corresponder con lo esperado luego del análisis de los modelos. El modelo fue capaz de reconocer rostros frontales indicando el nombre de usuario que le correspondía. Por otro lado, al no tener un rostro frontal, considera que no hay rostros en la cámara. También, al momento de identificar al usuario cometió errores identificando al ‘User.30’ cuando en realidad era el ‘User.10’, en ningún momento clasifico al ‘User.10’ como ‘User.20’.

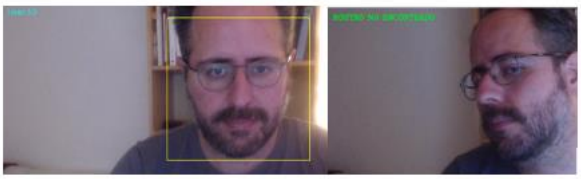

Figura 33: Prueba 6 – Ejemplo prueba FR.

6. Conclusión

Se obtiene como conclusión inicial que, en la actualidad existe la suficiente información y herramientas de DL para dar soporte a múltiples áreas y problemas. Relacionado con la investigación realizada, sobre las hipótesis planteadas, el resultado obtenido no es concluyente, si bien en el entrenamiento de los modelos se aprecia una pequeña ventaja de las CNN en cuanto a los niveles de perdida, precisión y AUC de los modelos creados a partir de TL con FA más actuales como ‘Mish’ en relación con ‘ReLu’, no podemos concluir que con una FA ‘Mish’ o ‘Swish’ obtiene resultados más satisfactorios al ser entrenado con una FA moderna que al ser entrenado con una FA de las que se utilizan frecuentemente. En parte, la no obtención de un resultado concluyente se debe a las faltas de pruebas de alternativas, en el código programado, configuraciones y así también, a las formas que se entrenaron los modelos. Adaptar el dataset utilizado, el código para los modelos requiere de tiempo. Como aprendizaje, agregar que, analizar el tiempo que puede llevar ajustar los hiperparámetros al crear una red neuronal, mejora el resultado y debe ser considerado al momento de la planificación del proyecto.

Como líneas de trabajo a futuro, se considera la adaptación de los modelos planteados en este trabajo. También, la investigación inicial de este TFM, acerca de las mejoras en los algoritmos de DL para FR, detectando rasgos faciales que pueden ser el origen de problemas de sesgos en el momento de la clasificación.

7. Glosario

Face Recognition (FR): Aplicación informática capaz de detectar, rastrear, identificar o verificar un rostro en una imagen o video digital.

Funciones de activación (FA):

Data Streaming (DS): datos que se generan de forma continua.

Design Science Research (DSR)

Deep Learning (DL) [5]: es un área emergente de la investigación del aprendizaje automático y comprende múltiples capas ocultas de redes neuronales artificiales. La metodología de aprendizaje profundo aplica transformaciones no lineales y abstracciones de modelos de alto nivel en grandes bases de datos.

Design science research (DSR): investigación en la ciencia del diseño" o "design science research", es un paradigma de investigación que se centra en el desarrollo y la validación del conocimiento.

One-to-one": técnica de FR donde buscan coincidencias de una foto de un rostro, con otra foto de la misma persona en una base de datos.

One-to-many: técnica de FR que intenta determinar si una foto de una persona coincide con alguna foto dentro de una base de datos.

Requerido: Son objetivos Requeridos ya que su cumplimiento se considera obligatorio para que el proyecto sea considerable exitoso.

Deseables: Son objetivos Deseables cuando su cumplimiento no afecta al éxito del proyecto, pero se entiendo que aporta un valor al cierre general del proyecto.

Escala de Fitzpatrick: La escala de fototipos de Fitzpatrick permite conocer la sensibilidad de la piel frente a la luz ultravioleta.

Redes neuronales artificiales (RNA) [11]: Modelo computacional de nodos interconectados. Neuronas artificiales conectadas entre sí, capaces de aprender sin necesidad de la participación de humanos.

Redes neuronales biológicas (RNB) [11]: Compuesta por un cuerpo celular rodeado de dendritas que contiene un núcleo, un axón y un extremo con botones sinápticos. Las dendritas se comunican con otras neuronas enviando y recibiendo información por medio de impulsos eléctricos o potenciales de acción que viajan a lo largo del axón y generando impulsos químicos que llegan a la siguiente neurona.

GPU: Unidad de procesamiento gráfico o graphics processing unit. Permite la realización de cálculos complejos. Comúnmente utilizadas para la ejecución gráfica y cálculos en videos juegos.

Ley de Moore [12]: Esta ley, indica que el número de transistores incluidos en un chip se duplicara cada 24 meses.

Convolutional neural network o redes neuronales convolucionales (CNN): Tipo de red neuronal artificial (RNA). Posee capas de entradas, ocultas y de salida. Permite técnicas no lineales de detección. Utilizadas para el análisis visual de imágenes.

Retro-propagación o backpropagation (BP): método de propagación utilizado para entrenar una RNA, donde se busca ajustar los valores del gradiente en el entrenamiento.

Unidades lógicas de umbral (ULU) o threshold logic unit (TLU). Valor numérico dentro de las neuronas de una RNA.

Perceptrón multicapa (PMC): RNA compuesta por una capa de entrada, capa o capas ocultas y cada de salida.

Red neuronal profunda (RNP): un PMC que contiene gran cantidad de capas ocultas.

Hiperparámetros: en ML es parámetro cuyo valor se utiliza para controlar el proceso de aprendizaje, así como los valores de otros parámetros se obtienen del entrenamiento.

Red neuronal artificial convolucional (CNN): tipo de RNA, comúnmente utilizada para resolver problemas de clasificación de imágenes.

Aprendizaje supervisado (AS): Se conoce el valor al que se desea obtener en un entrenamiento ya que tienen relación con los valores entrenados.

Aprendizaje no supervisado (ANS): Aprendizaje donde no hay conocimiento previo de lo que se de

Transferencia de conocimiento (TL) [14] o transfer learning: en las RNA es la metodología donde un modelo entrenado previamente, se reutiliza en la resolución de un problema similar.

OpenCV [15]: Open Computer Visión. Librería desarrollada por Intel para la visión artificial.

VGG [16]: es una RNA del tipo convolucional.

Tensorflow (TF) [23]: Biblioteca de código abierto para entrenamiento de RNA creada por Google.

Pytorch (PT) [24]: Biblioteca de código abierto para entrenamiento de RNA creada por Facebook.

Landmark: Puntos de referencias que van a permitir la detección de rostros, ojos, nariz, boca.

8. Bibliografía

[1] Vaishnavi, V., Kuechler, W. y Petter, S. (Eds.) (2004/19). “Design Science Research in Information Systems” 20 de enero de 2004 (creado en 2004 y actualizado hasta 2015 por Vaishnavi, V. y Kuechler, W.); última actualización (por Vaishnavi, V. y Petter, S.), 30 de junio de 2019. URL:
http:// www.desrist.org/design-research-in-information-systems/

[2] QUT eprints - https://eprints.qut.edu.au/127354/ [Fecha de consulta 4 de abril de 2022].

[3] Oreilly - https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ / [fecha de Consulta 30 de Marzo de 2022]

[4] Shermin, T., Murshed, M., Lu, G., & Teng, S. W. (2018). Transfer learning using classification layer features of CNN. *arXiv preprint arXiv:1811.07459*.
https://arxiv.org/pdf/1811.07459.pdf

[5] towardsdatascience - https://towardsdatascience.com/how-to-utilize-transfer-learning-to-classify-images-with-state-of-the-art-deep-learning-models-d8e5d5bb35d4- [Consultado 2 de Junio del 2022]

[6] Buolamwini, Joy; Gebru, T. (2018). Gender Shades: Intersectional Accuracy Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
https://arxiv.org/pdf/1710.05941.pdf

[7] Misra, D. (2019). Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*.
https://arxiv.org/pdf/1908.08681.pdf

[8] Singh, K., Seth, A., Sandhu, H. S., & Samdani, K. (2019, March). A comprehensive review of convolutional neural network based image enhancement techniques. In 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN) (pp. 1-6). IEEE.

[9] Jain, A. K., & Li, S. Z. (2011). Handbook of face recognition (Vol. 1). New York: springer. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.3918&rep=rep1&type=pdf

[10] MIT Technology - https://www.technologyreview.com/2019/12/20/79/ai-face-recognition-racist-us-government-nist-study/ [fecha de Consulta 3 de Abril de 2022]

[11] European Commission - https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:52021PC0206&from=EN [fecha de Consulta 30 de marzo de 2022]

[12] Intel - https://www.intel.es/content/www/es/es/history/museum-gordon-moore-law.html / [fecha de Consulta 30 de Marzo de 2022]

[13] GIMP - https://docs.gimp.org/2.6/es/plug-in-convmatrix.html / [fecha de Consulta 15 de Mayo de 2022]

[14] Sciencedirect - https://www.sciencedirect.com/ [fecha de Consulta 15 de Mayo de 2022]

[15] OpenCv - https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html / [fecha de Consulta 15 de Mayo de 2022]

[16] Neuroarchivo - https://neurohive.io/en/popular-networks/vgg16/ [fecha de Consulta 15 de Mayo de 2022]

[17] Imagenet - https://www.image-net.org [fecha de Consulta 15 de Mayo de 2022]

[18] Van Rossum, G., & Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

[19] Google Colab - https://colab.research.google.com/ [fecha de Consulta 15 de Mayo de 2022]

[20] Anaconda - https://www.anaconda.com/ [fecha de Consulta 15 de Mayo de 2022]

[21] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[22] repositorio.utn.edu.ec - http://repositorio.utn.edu.ec/handle/123456789/7315 [fecha de Consulta 15 de Mayo de 2022]

[23] Tensorflow - https://www.tensorflow.org/ [fecha de Consulta 15 de Mayo de 2022]

[24] Pytorch - https://pytorch.org [fecha de Consulta 15 de Mayo de 2022]

[25] TL y Keras - https://www.tensorflow.org/guide/keras?hl=es-419 [fecha de Consulta 15 de Mayo de 2022]

[26] Keras Api - https://keras.io/api/ - [fecha de Consulta 15 de Mayo de 2022]

[27] Kaggle - https://www.kaggle.com/code/imokuri/mish-activation-function/notebook- [fecha de Consulta 2 de Junio de 2022]

[28] http://www.ijeast.com - https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf - [fecha de Consulta 3 de Junio de 2022]

9. Anexos

Acceso al código del proyecto Python

Módulo de creación del dataset: Módulo de creación del dataset

Módulo de entrenamiento del modelo: Módulo de entrenamiento

Módulo de FR: Módulo FR

Dataset: Dataset

Modelos generados: Modelos_H5

Descomprimir el dataset desde Jupyter

!tar xzvf dataset_TFM.tar.gz

Descomprimir fichero de los modelos desde Jupyter

!tar xzvf modelos_TFM.tar.gz

Instalación en Anaconda AnacondaWEB

Creación de ambientes de trabajo de Anaconda Anaconda Docs

Arquitectura de la CNN

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_10 (Flatten)	(None, 25088)	0
dense_50 (Dense)	(None, 210)	5268690
dense_51 (Dense)	(None, 30)	6330
dense_52 (Dense)	(None, 30)	930
dense_53 (Dense)	(None, 3)	93
dense_54 (Dense)	(None, 3)	12

Total params: 19,990,743
Trainable params: 5,276,055
Non-trainable params: 14,714,688

Figura 34: Anexo – Ejemplo CNN.

Resultados de los entrenamientos

ReLu

En la siguiente figura se muestra los resultadoes de la ejecución del entrenamiento del modelo con una FA Relu y sus diferentes metricas de perdida, precisión, AUC.

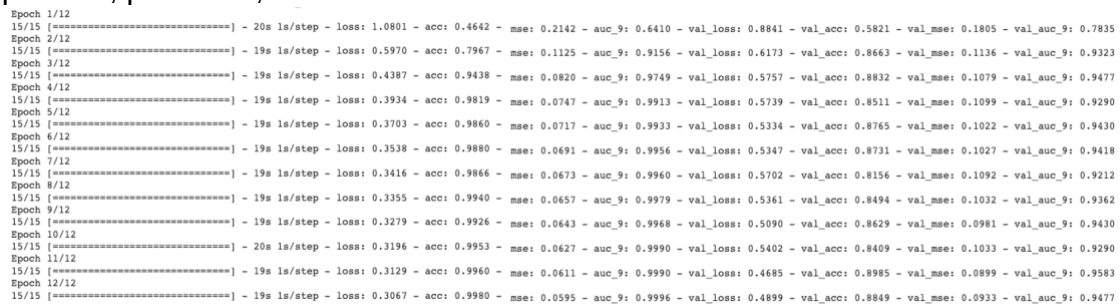

Figura 35: Anexo – Resultado entrenamiento ReLu.

Swish

En la siguiente figura se muestra los resultadoes de la ejecución del entrenamiento del modelo con una FA Swish y sus diferentes métricas de perdida, precisión, AUC.

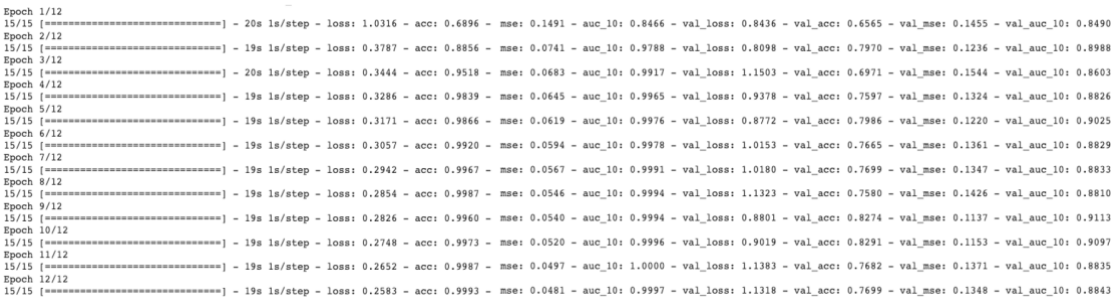

Figura 36: Anexo – Resultado entrenamiento Swish.

Mish

En la siguiente figura se muestra los resultadoes de la ejecución del entrenamiento del modelo con una FA Mish y sus diferentes métricas de perdida, precisión, AUC.

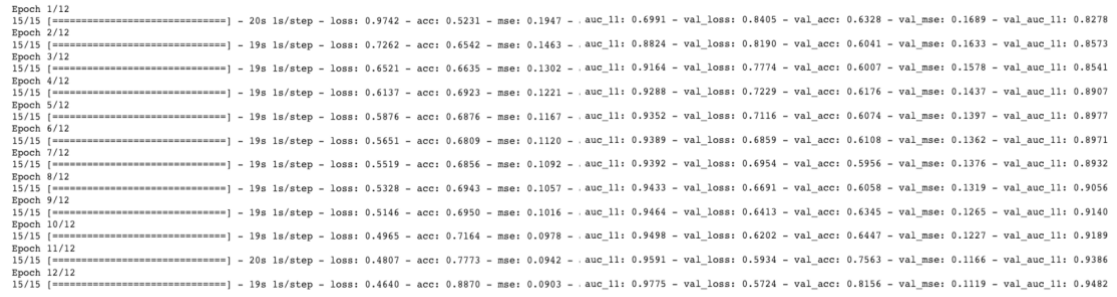

Figura 37: Anexo – Resultado entrenamiento Mish