

UNIVERSIDAD ANDRÉS BELLO
FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL INFORMÁTICA

Proyecto Ciberseguridad: Verificación Facial

Ricardo Sebastián Reyes Palma

Profesor guía: Billy Peralta Márquez

SANTIAGO - CHILE

Marzo - 2022

Índice

1. Resumen **3**

2. Abstract **4**

3. Introducción **5**

4. Marco teórico **6**

4.1. Definiciones Importantes 6

4.1.1. Redes Neuronales Artificiales 6

4.1.2. Redes Neuronales Convolucionales 6

4.1.3. Fine Tuning 7

4.1.4. ArcFace 8

4.2. Redes Neuronales Utilizadas 8

4.2.1. ResNet 8

4.2.2. MobileNet 9

4.2.3. GhostNet 9

4.3. Trabajos Relacionados 10

4.3.1. Identity Verification of Ticket Holders at Large-scale Events Using Face Recognition 10

4.3.2. Hybrid Deep Learning for Face Verification 11

4.3.3. Deep Learning Face Representation by Joint Identification-Verification 12

4.3.4. Unconstrained Face Verification using Deep CNN Features 13

4.3.5. von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification 13

5. Hipótesis **15**

6. Objetivo del trabajo **16**

6.1. Objetivos Específicos 16

7. Metodología **17**

7.1. Preparación del Dataset 17

7.2. Entrenamiento del Sistema 22
7.2.1. Primer Entrenamiento 23
7.2.2. Segundo Entrenamiento 24
7.2.3. Tercero Entrenamiento 25
7.2.4. Cuarto Entrenamiento 27
7.2.5. Entrenamientos Extras 27

8. Resultados **29**
8.1. ResNet34 29
8.2. MobileNet 31
8.3. GhostNet 32
8.4. ResNet34 (Fine-tuning) 34
8.5. Entrenamientos Extra 36
8.5.1. ResNet34 36
8.5.2. MobileNet 37
8.5.3. GhostNet 39

9. Análisis y discusión de los datos **41**
9.1. Análisis de ResNet34 (Fine-tuning) 43

10. Conclusiones **46**

11. Limitaciones, ideas y trabajos futuros **47**
11.1. Limitaciones 47
11.2. Ideas y Trabajos Futuros 47

Bibliografía **49**

1. Resumen

En la actualidad la ciberseguridad se ha convertido en uno de los temas más importantes, debido a la creciente evolución tecnológica y a la utilización de plataformas digitales por parte de bancos, empresas, clínicas y todas aquellas entidades que utilicen información privada y/o sensible de los usuarios.

Es por esto que implementar una red neuronal que logre verificar la identidad de la persona, a través de una imagen de su cédula de identidad junto con una fotografía del usuario y esta pueda ser utilizada en el registro e ingreso a cualquier tipo de plataforma, para evitar fraudes y suplantación de identidad, es algo que se vuelve necesario por estos días.

De la necesidad mencionada anteriormente nace este proyecto de investigación, en donde se utilizó el dataset "FaceID-550", que contiene de más de 11.550 imágenes y cada persona incluida en esta base de datos posee una fotografía de su pasaporte, que simula la calidad y color que posee la imagen de la cédula de identidad chilena, junto con imágenes en distintos ángulos de la misma persona, para realizar el entrenamiento óptimo de algunas redes neuronales seleccionadas y ver cual de todas cuenta con el mejor rendimiento.

2. Abstract

Currently cybersecurity has become one of the most important issues, due to the growing technological evolution and the use of digital platforms by banks, companies, clinics and all those entities that use private and / or sensitive information of the users.

This is why implementing a neural network that manages to verify the identity of the person, through an image of their identity card together with a photograph of the user and this can be used in the registration and entry to any type of platform, to avoiding fraud and identity theft is something that becomes necessary these days.

From the aforementioned need, this research project was born, where the "FaceID-550"dataset was used, which contains more than 11.550 images and each person included in this database has a photograph of their passport, which simulates the quality and color that the image of the Chilean identity card possesses, together with images from different angles of the same person, to carry out the optimal training of some selected neural networks and see which of all has the best performance.

3. Introducción

El registro en aplicaciones o sitios web es una tarea sencilla, ya que sólo requiere información básica, como por ejemplo: correo electrónico, número telefónico, una contraseña, que en muchas plataformas no se le da importancia necesaria al nivel de seguridad que esta posee, entre otras cosas. Todo lo anterior mencionado, es información que se puede alterar o inventar sin mayores complicaciones, una persona puede utilizar el nombre de otra, junto con un correo que no necesariamente sea el personal y una fecha de nacimiento falsa y no tendría ningún problema para registrarse, como ocurre muchas veces con las aplicaciones de transporte, ya sea Uber, Beat, etc. Se han visto muchos delitos en los cuales el conductor utilizaba un perfil falso o la persona utilizaba la cuenta de otra. También ocurre con los pasajeros que para cometer delitos solicitaban un vehículo desde un perfil falso.

Pero ¿Qué pasaría si al momento de registrarte en este tipo de aplicaciones se solicitase la fotografía del carnet de identidad y que para poder iniciar sesión se verifique la identidad de la persona utilizando la cámara del celular? Es por esto por lo que nace la idea de implementar una red neuronal capaz de realizar este tipo de verificación facial "carnet-selfie" y que se pueda utilizar en aplicaciones de este tipo. Para ello se decidió realizar el entrenamiento de varias redes neuronales, las cuales son: ResNet34, MobileNet y GhostNet, utilizando imágenes que tengas las mismas características de la fotografía de la cédula de identidad o del pasaporte, junto con fotografías casuales de la persona. Por lo que se decidió utilizar el dataset llamado "FaceID-550", este contiene imágenes de personas Asiáticas, tanto de su pasaportes o cédula junto con fotografías de la misma persona, lo que lo convierte en una base de datos ideal para el entrenamiento de las redes que se desean entrenar. También cabe mencionar que se utilizará Insightface, que es una biblioteca de código abierto, que cuenta con algoritmos para llevar a cabo el entrenamiento de las redes neuronales.

Es importante destacar que ya existen redes neuronales entrenadas para realizar reconocimiento facial, por lo que con esta investigación se busca obtener resultados mejores en comparación con los que ya existen.

4. Marco teórico

4.1. Definiciones Importantes

4.1.1. Redes Neuronales Artificiales

Es un sistema matemático complejo [1], que simula las neuronas de los seres humanos y al igual que estas, fluye gran cantidad de información. Estas redes neuronales artificiales tienen una gran característica, la cual es que son capaces de aprender a través de la experiencia, es decir, mediante un entrenamiento.

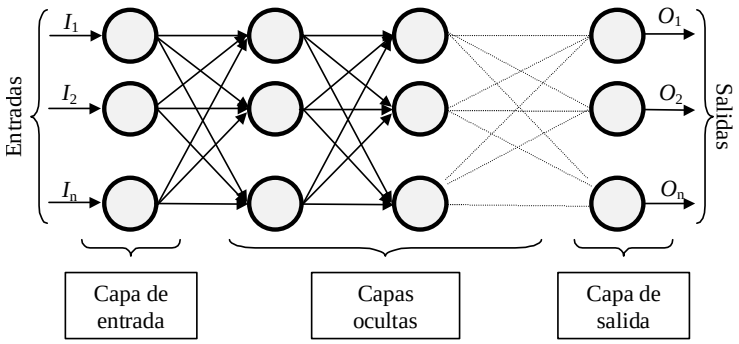


Figura 1: Estructura de una red neuronal artificial [1]

Cada red neuronal cuenta con tres capas, capa de entrada, capas ocultas y por ultimo capa de salida. La capa de entrada es quien recibe la información, que pueden ser fotografías, vídeos o cualquier otro tipo de información. Las capas ocultas no tienen contacto directo con el exterior, se encarga de procesar los datos recibidos por la primera capa. También hay que mencionar que el numero de estas puede ir entre cero y un gran numero de capas. Por ultimo tenemos la capa de salida, al cual entrega la información ya procesada al exterior.

4.1.2. Redes Neuronales Convolucionales

Es un tipo de red multicapa, que posee diferentes capas de *pooling* alternadas y convolucionales y en su parte final cuenta con una serie de capas "full-connected". Este tipo

de redes, actualmente son las más usadas en problemáticas de visión computacional, ya que es capaz de extraer todo el contenido de una imagen o vídeo y posteriormente clasificarla, detectando objetos o personas en ella. Cada uno de los parámetros de los filtros de este tipo de red puede ser modificados y optimizados para obtener mínimos errores en la clasificación final [2].

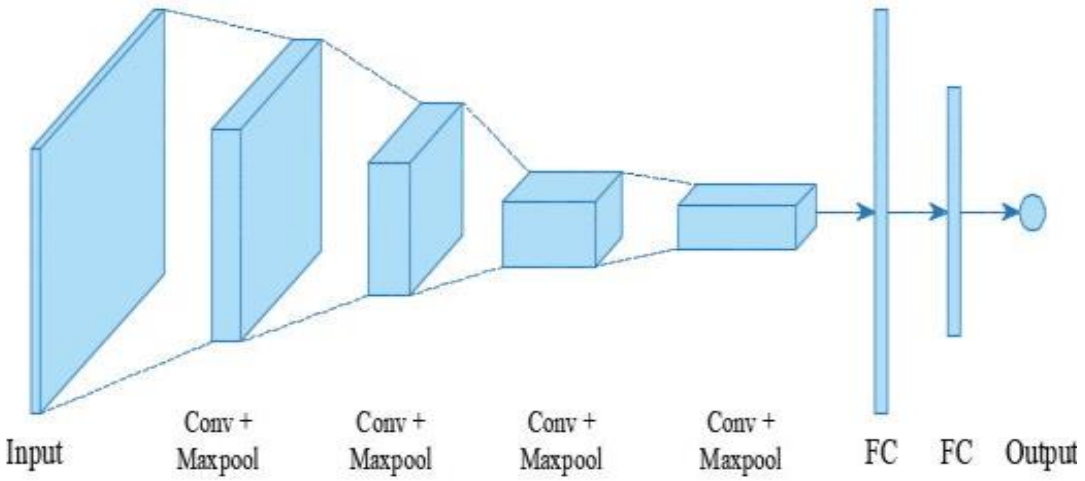


Figura 2: Arquitectura básica de una red neuronal convolucional [3]

Las redes utilizadas en este proyecto de investigación son un tipo de red neuronal convolucional y serán detalladas más adelante.

4.1.3. Fine Tuning

Esta técnica se utiliza para ahorrar recursos y tiempo en el entrenamiento de una red neuronal. Si se tiene acceso a una red pre entrenada que sea capaz de resolver algo similar a lo que se está buscando se puede utilizar como punto de partida, de esta forma se puede obtener mejores resultados en un menor tiempo.

4.1.4. ArcFace

Es una función de perdida utilizada para reconocimiento facial [4]. Esta utiliza la función arco-coseno para hacer el calculo del ángulo entre la característica actual de la imagen y el peso actual de la red neuronal, además de utilizar una baja complejidad computacional.

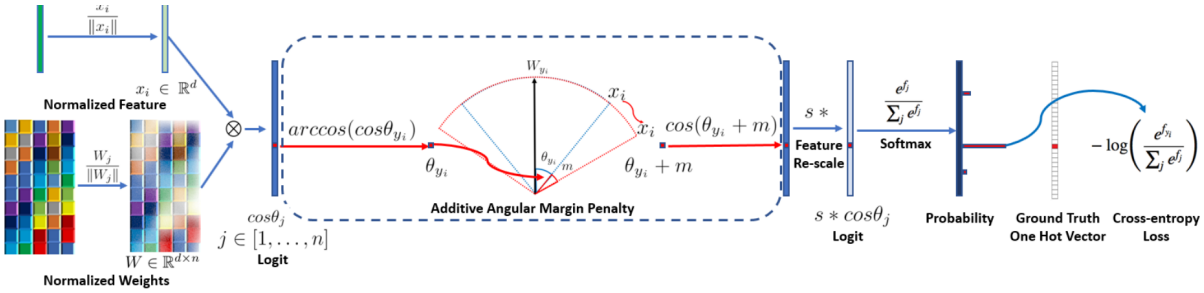


Figura 3: Arquitectura de ArcFace Loss [4]

4.2. Redes Neuronales Utilizadas

4.2.1. ResNet

También conocida como Residual Network [5]. Esta red neuronal, mediante las "Skip connections" para saltar capas convolucionales en cada bloque básico de la red, permite entregar caminos alternativos para los datos originales y derivados, lo que hace que a la hora de realizar entrenamientos, este sea más rápido y efectivo, aliviando uno de los problemas de las redes neuronales profundas, el cual es que en muchas ocasiones los gradientes desaparecen al momento de realizar entrenamientos.

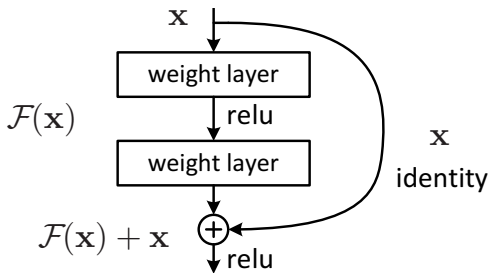


Figura 4: Bloque de aprendizaje de ResNet [6]

4.2.2. MobileNet

Esta es un tipo de red neuronal convolucional [7] utilizado mayormente en aplicaciones de visión móvil. Una de las características de esta red es que utiliza convoluciones separables en profundidad, esta se compone de dos capas: convolución en profundidad y convolución en punto. La primera la utiliza para realizar un solo filtro para los canales de entrada de información, esto normalmente no es así, ya que se aplican filtros en todos los canales. La segunda capa se combinan todas las salidas de la primera.

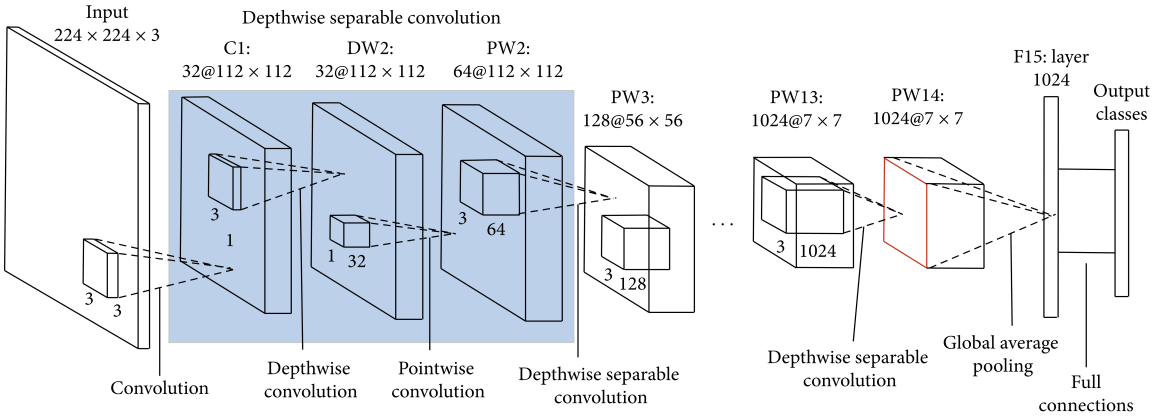
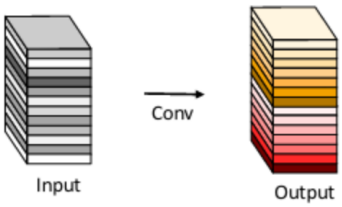


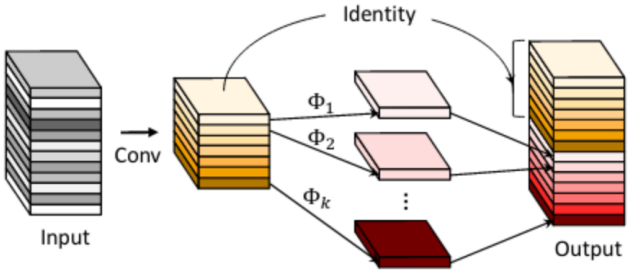
Figura 5: Arquitectura de MobileNet [8]

4.2.3. GhostNet

Esta red [9] cuenta con un modulo que tiene el nombre de "Ghost". Dentro de este modulo ocurren dos tipos de convoluciones, la convolución primaria y la convolución fantasma. La convolución primaria involucra convoluciones normales pero su cantidad se controlará de manera estricta, las salidas que entregue este serán concatenadas en la convolución fantasma.



(a) The convolutional layer.



(b) The Ghost module.

Figura 6: Descripción del modulo de GhosNet [10]

4.3. Trabajos Relacionados

4.3.1. Identity Verification of Ticket Holders at Large-scale Events Using Face Recognition

En este articulo [11] se expone un sistema para verificar la identidad de personas que poseen entradas a eventos de gran escala a traves de reconocimiento facial, este es llamado *Ticket-ID*. Este sistema se ha implementado con la necesidad de frenar la reventa ilegal de estas entradas, debido a que en Japón, donde se llevó a cabo este desarrollo, utilizan rigurosos para verificar a las personas en las entradas de los eventos, para comprobar que sea la persona, la compradora original de la entrada.

El entrenamiento de este sistema se efectuó gracias a la base de datos de la pagina vendedora de las entradas, ya que al momento de registrarse en la plataforma, el usuario debía subir una foto de perfil. Este proceso debía ser previamente aceptado por el usuario, así de esta forma se evitaban problemas legales. Ahora bien, gracias a *Ticket-ID* han logrado verificar la identidad de más de 50.324 asistentes a un gran concierto, en donde la precisión promedio fue del 90 %, sumado a lo anterior, también lograron reducir el tiempo en los controles de verificación que antes, labor que realizaban anteriormente personas.

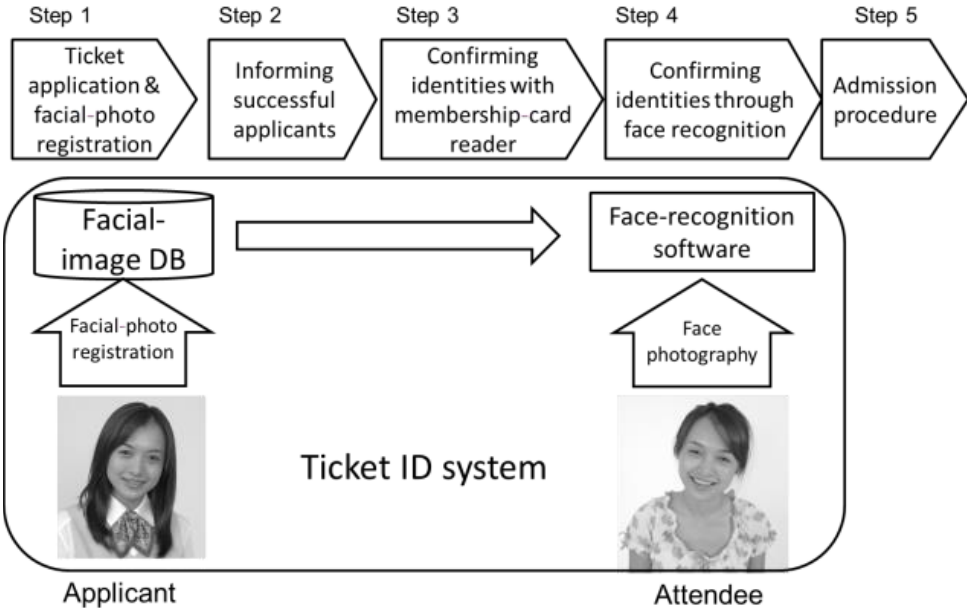


Figura 7: Proceso de sistem *Ticket-ID* [11]

4.3.2. Hybrid Deep Learning for Face Verification

Se propone en este articulo [12] una red neuronal convolucional híbrida llamada ConvNet- Restricted Boltzmann Machine (RBM), el cual este ultimo es un modelo que se utiliza para verificación facial en condiciones salvajes. Esta red aprende directamente y extrae al mismo tiempo información visual de pares de rostros, mediante píxeles sin procesar y los compara con los filtros aprendidos con anterioridad. Cuenta con múltiples capas ConvNet para lograr solidez y caracterizar similitudes faciales las fotografías.
Esta red neuronal convolucional fue entrenada con el dataset conocido como *LFW*. Utilizando el 80 % de la base de datos para entrenar las ConvNet profundas y en 20 % restante se utilizó para capacitar la capa superior de la red, que corresponde a la capa RBM. Se generaron tanto pares positivos como negativos, un total de 40.000 para realizar el entrenamiento de las redes y 8.000 para entrenar la capa superior. En entre proyecto también se entrenó utilizando un nuevo dataset creado por ellos, el cual llamaron "CelebFaces". Esta red cuenta con un total de 87.628 imagenes de personas famosas que no están dentro de la base de datos LFW. Los resultados obtenidos luego de realizar el entrenamiento fue

de un 91.75 % de *accuracy* con los pares de la primera base de datos usada y un 92.52 % con CelebFaces.

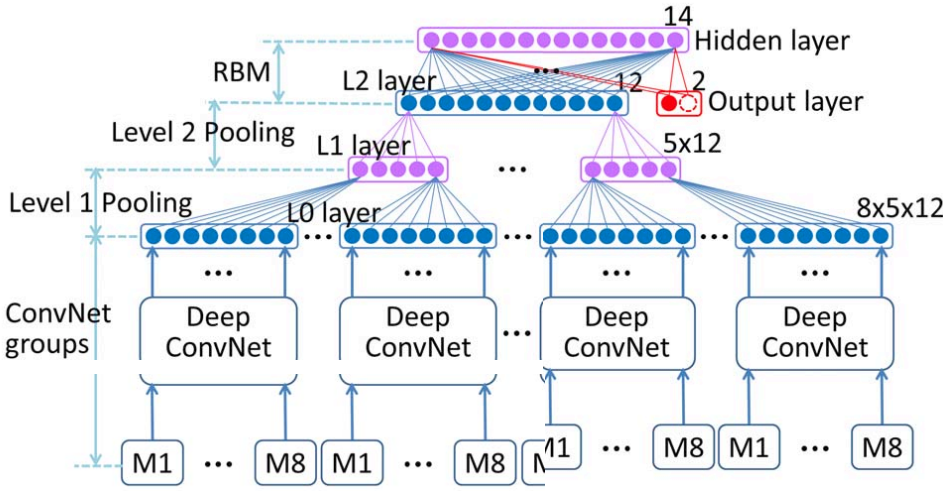


Figura 8: Arquitectura de modelo híbrido ConvNet-RBM [12]

4.3.3. Deep Learning Face Representation by Joint Identification-Verification

En este articulo [13] se implementó el modelo *DeepID2* junto con la red neuronal ConvNet, donde este se aprovecha extrayendo las características aprendidas de DeepID2 de toda la región del rostro. Los autores afirman que LFW, la base de datos utilizada en el articulo mencionado anteriormente, es un conjunto de pruebas solido pero no es adecuado para realizar un entrenamiento, ya que la mayoría de personas presentes en esta base de datos, solo contiene una imagen de rostro, con lo cual recurrieron a una base de datos llamada "CelebFaces+", que contiene 202.599 imágenes de caras de 10.177 famosos y ninguno de ellos es parte del dataset LFW.

Para llevar a cabo el entrenamiento se hizo una división de la base de datos, en donde 8192 identidades escogidas aleatoriamente sería destinado para entrenar el modelo DeepId2 y los 1985 restantes, llamado CelebFaces+B, se utilizaron como un set de validación para poder decidir la cantidad de épocas para el entrenamiento y la tasa de aprendizaje. Posterior a esto CelebFaces+B se separa en otro conjunto de entrenamiento con 1485 identidades y los 500 restantes para la validación. Finalmente se utiliza una ConvNet

profunda para extraer todas las características aprendidas por DeepID2 y se construye el sistema.

Finalmente se realizó un testeo final, utilizando LFW, logrando un *accuracy* de 99.15 %.

4.3.4. Unconstrained Face Verification using Deep CNN Features

El autor propone [14] un algoritmo de verificación facial basado en características de redes convolucionales profundas (DCNN). Implementa esta red utilizando *coffe* [15] y se entrena con la base datos CASIA-WebFace. Esta base de datos cuenta con 494.414 imágenes de 10.575 personas diferentes.

Se estableció un tamaño de bacht de 128 para la fase de entrenamiento, el decaimiento de peso de todas las capas de definió en 0, la tasa de aprendizaje se estableció inicialmente en 1e-2 y se reducia a la mitad cada 100.000 iteraciones completadas, llegando a completar un total de 1.000.000 de iteraciones.

Luego de concluir con el entrenamiento, en un total de nueve días completos, se puso a prueba la red neuronal realizando evaluaciones con las bases de datos LFW e IARPA Janus Benchmark A (IJB-A), obteniendo un *accuracy* de 97.45 % y 97.7 % respectivamente.

4.3.5. von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification

En este articulo [16] los autores proponen un modelo de mezcla llamado *von Mises-Fisher (vMF)* como base para realizar un entrenamiento profundo y efectivo. Para realizarlo, utilizaron la base de datos MS-Celeb-1M, que contiene un total de 4.61 millones de imágenes de 61.240 personas diferentes.

Utilizaron el 95 % del dataset para llevar a cabo el entrenamiento y el 5 % restante

para el monitoreo y evaluación de la perdida y el *accuracy*. Para el comienzo se definió una tasa de aprendizaje de un 0.1 para las dos primeras épocas, luego fue disminuyendo en un factor de 10 en cada iteración. Finalizaron el entrenamiento al completarse cinco épocas. También realizaron un aumento de datos durante el entrenamiento, volteando horizontalmente las fotografías.

Luego de finalizar el proceso de entrenamiento, pusieron a prueba el modelo ejecutando una evaluacion del rendimiento con cuatro base de datos distintas: LFW, YouTube faces, CACD e IJB-A. Con la primera database mencionada, se obtuvo un *accuracy* de 99.63 %, con la segunda fue de un 96.46 %., el desempeño con CACD fue de un 99.20 % y por ultimo, con IJB-A el *accuracy* final fue de 89.7 %.

5. Hipótesis

Al concluir con los entrenamientos de todas las redes neuronales, ResNet34 será capaz de realizar una verificación facial sobre el 90 % de *accuracy*.

6. Objetivo del trabajo

El objetivo de este proyecto es implementar una red neuronal para que sea capaz de verificar la identidad de una persona con un *accuracy* igual o superior a un 90 %.

6.1. Objetivos Específicos

- Estudio de documentos relacionados y búsqueda de Dataset ideal.
- Realizar un preprocesamiento en los datos para adaptar las imágenes incluidas en el dataset.
- Entrenar las diferentes redes neuronales con el dataset seleccionado.
- Validar los resultados y compararlos mediante métricas de clasificación.

7. Metodología

7.1. Preparación del Dataset

Para poder realizar un correcto entrenamiento de la red neuronal se deben generar tres archivos: un archivo .rec, .idx y por ultimo uno .bin. Este ultimo se consigue generando pares de imágenes, cada uno de estos debe incluir obligatoriamente una fotografía de pasaporte, ya que como se ha mencionado anteriormente, la finalidad de este proyecto es que el sistema logre predecir si es o no la misma persona, analizando una imagen de la cédula de identidad, junto con una selfie del usuario. Es por esto que se debe llevar a cabo un preprocesamiento en la base de datos, adaptando la cantidad de píxeles de las fotografías y el tamaño centrando la imagen solamente en la cara de la persona. Para hacer esto se utilizó la librería del lenguaje de programación Python *detect face*, que permitió que las imágenes quedaran de la siguiente manera:

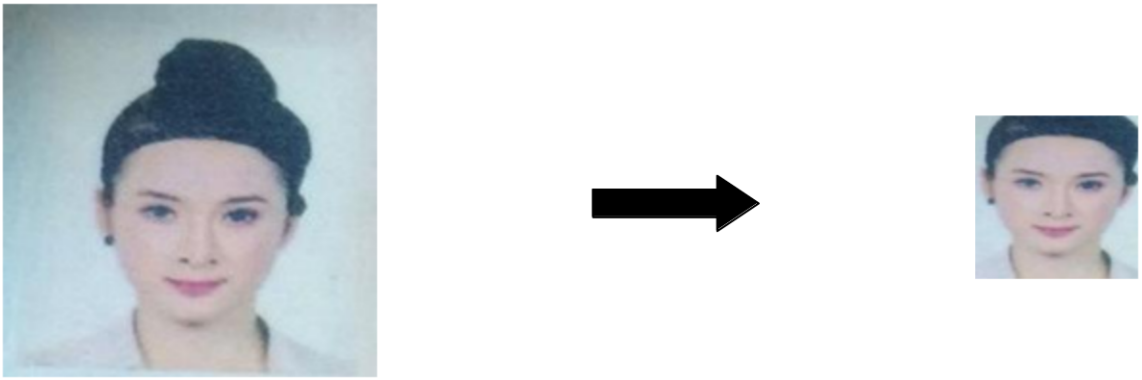


Figura 9: El antes y después de una de las imágenes incluidas en la base de datos "FaceID-550."

La imagen de la izquierda es como originalmente se encuentra en el dataset y en la derecha se puede observar como queda luego de efectuar el preprocesamiento. Una fotografía que originalmente era de 256 píxeles x 256 píxeles quedó de 112 píxeles x 112 píxeles. Cabe destacar que la extensión de las fotografías es indiferente, puede ser tanto .jpg como .png, lo que si es importante y también se debió cambiar, son los nombres de las carpetas, debido a que cada una de ellas contaba con dos o más guiones bajos y lo que necesita el algoritmo, que se realizó para poder modificar el nombre de todas las imágenes,

es que las carpetas solo tengan un guión bajo.

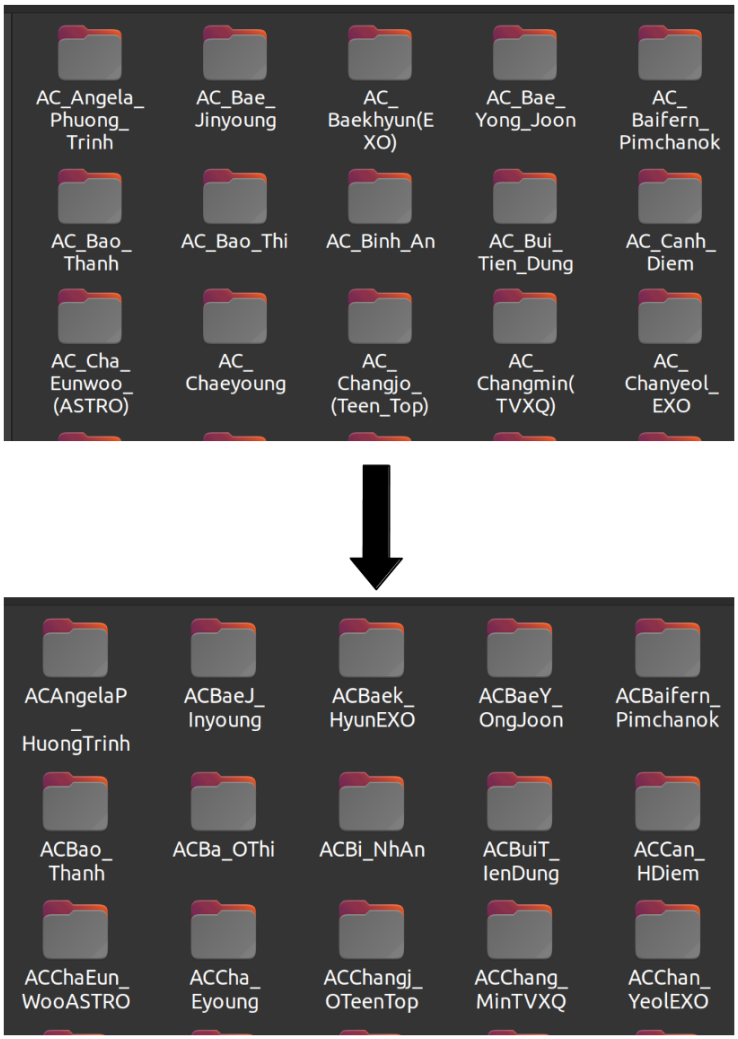


Figura 10: Cambio de nombre de una pequeña parte de las carpetas incluidas en el dataset

En la parte de arriba de la figura se puede observar como vienen las carpetas por defecto y en la parte de abajo como quedan luego del cambio realizado, en donde solo se dejó un guión bajo. Posterior a esto, se debe continuar con el cambio de las imágenes, ya que estas deben seguir un formato especifico, en donde la fotografía de pasaporte debe contener el número 0 al final de su nombre y el resto puede finalizar con una numeración aleatoria que parta del uno hacía adelante, separados por un guión bajo.



Figura 11: Selección de una de las personas incluidas en el dataset, donde se puede observar el antes y el después, tanto de la fotografía como del nombre.

Para lograr aquello, se creó un algoritmo en Python en donde, como primera etapa, se ordenase alfabéticamente cada una de las carpetas e imágenes, de esta manera, siempre la fotografía de pasaporte quedará como primera imagen, ya que cuenta con la terminación "0ID". Luego de esto, se recorren todas las carpetas y en cada una de ellas, se hace el reemplazo mencionado anteriormente.

```
import os, pathlib, random


for path, subdirs, files in os.walk(root):
    i = 1
    e = 0
    for name in sorted(files):
```

```
        extension = name.split(".")[-1].lower()
        if extension != "jpg":
            continue
        stri = str(random.randint(1,50))
        if i == 1:
             os.rename(os.path.join(path, name),
             os.path.join(path, os.path.basename(path)
             + "_" + str(e) + '.' + extension))
        else
             os.rename(os.path.join(path, name),
             os.path.join(path, os.path.basename(path)
             + "_" + stri + '.' + extension))
        i = i + 1
```

Los cambios de nombre en las imágenes, tuvieron como finalidad dar el formato requerido por el algoritmo generador de pares que se creó, gracias a lo realizado, era sencillo crear una condición en la que siempre, como primera imagen del par, escogiera una terminada en 0 y como segunda, cualquiera que tuviera una terminación distinta a la primera. Pero antes de comenzar con la generación de pares, se debe hacer la división del dataset modificado, en lo que será entrenamiento y testeo. El primero debe contar con la mayor parte de información, por lo que para este proyecto se dejó en un 70 % del dataset, luego el 30 % restante para el último mencionado.

Luego de realizar esta repartición, hay que continuar creando los archivos .idx y .rec, que fueron mencionados con anterioridad. Para ello primero hay que generar un archivo .lst con la carpeta de entrenamiento, este archivo contiene el listado completo con las ubicaciones de todas las imágenes incluidas en esta carpeta. Una vez generado, se utilizó un algoritmo *open source* que importa y utiliza varias librerías, como por ejemplo: mxnet, cv2, traceback, entre otras y que permite generar tanto el archivo .idx como el .rec.

Posteriormente se debe continuar con la generación de pares utilizando el algoritmo creado, generando un archivo .txt que contiene todos los pares posibles y que será necesario para la creación del archivo .bin, para esto también se utilizó un código *open source*. Cada par creado se realizó de manera aleatoria, generando un total de 25454 pares, los cuales mezclaba a cada una de las personas incluidas en el dataset, así como también parejas de fotografías de la misma persona, para que el testing, luego de realizar el entrenamiento, fuese robusto.

```
import mxnet as mx
from mxnet import ndarray as nd
import argparse, pickle, sys, os
sys.path.append(os.path.join(os.path.dirname(__file__), '..', 'eval'))
import src.lfw as lfw

parser = argparse.ArgumentParser(description='Package LFW images')

parser.add_argument('--data-dir', default='', help='')
parser.add_argument('--image-size', type=str, default='112,112')
parser.add_argument('--output', default='', help='path to save.')
args = parser.parse_args()
lfw_dir = args.data_dir
image_size = [int(x) for x in args.image_size.split(',')]
lfw_pairs = lfw.read_pairs(os.path.join(lfw_dir, 'pairs.txt'))
print(lfw_dir)
lfw_paths, issame_list = lfw.get_paths(lfw_dir, lfw_pairs)
lfw_bins = []

i = 0
for path in lfw_paths:
  with open(path, 'rb') as fin:
```

```
    _bin = fin.read()
    lfw_bins.append(_bin)
    i+=1
    if i%1000==0:
      print('loading lfw', i)

with open(args.output, 'wb') as f:
  pickle.dump((lfw_bins, issame_list), f,
  protocol=pickle.HIGHEST_PROTOCOL)
```

Este algoritmo originalmente fue utilizado para realizar el .bin del dataset LFW, pero que para este proyecto fue adaptado para generarlo con FaceID-550. Al finalizar y teniendo los archivos es posible comenzar con el entrenamiento de la red neuronal.

7.2. Entrenamiento del Sistema

Para realizar los entrenamientos se utilizó una biblioteca *open source* de Python, llamada "InsightFace" y que cuenta con algoritmos de detección facial, verificación facial, etcétera y que está optimizado para realizar entrenamientos, ideal para lo que busca este proyecto. Por lo que para cada uno de estos, se desarrolló un código, importando cada una de las librerías que serían necesarias para llevar a cabo de manera eficiente el proceso.

```
1 from tensorflow import keras
2 import losses, train, models
3 import tensorflow_addons as tfa
```

Figura 12: Librerías importadas y necesarias para la ejecución del entrenamiento de la red neuronal.

Otro punto importante a mencionar es que cada uno de los entrenamientos se llevaron a cabo en un Computador de escritorio con una tarjeta de vídeo GTX 1660 SUPER (6GB de memoria VRAM), 16 GB de RAM y un procesador Ryzen 5600X. Si bien no son las

mejores especificaciones para realizar entrenamientos con una gran cantidad de datos, no hubo mayores complicaciones a la hora de efectuar los experimentos, estos pudieron ser completados sin complicaciones. Cada uno de ellos duró aproximadamente entre cinco y seis días, debido a que se estableció un total de cuarenta y un "epoch", también conocidas como épocas o iteraciones, que son cada uno de los ciclos de corrección para reducir las perdidas en el entrenamiento de la red.

7.2.1. Primer Entrenamiento

Este entrenamiento se realizó con la red neuronal ResNet34. En los primeros intentos, surgieron varios inconvenientes, ya que el proceso terminaba inesperadamente al ejecutar el código, lo que imposibilitaba la realización de los experimentos, por lo que se comenzó a ajustar los diferentes parámetros que el algoritmo contenía, realizando diversas pruebas, hasta que se llegó al elemento que hacía que el equipo no lograra realizar el entrenamiento y este era el "batch". Este se había definido con un valor de 256, lo que para un computador con las características mencionadas anteriormente, era demasiado, por lo que se comenzó a disminuir esa cantidad poco a poco, hasta que finalmente se consiguió llegar a una cifra en la que el entrenamiento se podía llevar a cabo sin ningún problema, eso si, el equipo no se podía utilizar para nada más, debido a que a la mínima apertura de otro programa terminaba el proceso de entrenamiento instantáneamente. También se debe mencionar que este experimento no solo se realizó con cuarenta épocas, si no que se utilizó además, una época más para realizar una optimización de la red neuronal y lograr obtener mejores resultados en el entrenamiento.

```
from tensorflow import keras
import losses, train, models
import tensorflow_addons as tfa

data_path = 'Resnet34_folder'
eval_paths = ['FaceID.bin']
```

```
basic_model = models.buildin_models("r34",
              dropout=0.4, emb_shape=512, output_layer='E',
              bn_momentum=0.9, bn_epsilon=1e-4, scale=True,
              use_bias=False, use_se=True)
basic_model = models.replace_add_with_stochastic_depth(basic_model)
basic_model = models.add_l2_regularizer_2_model(basic_model,
              weight_decay=5e-4, apply_to_batch_normal=False)
basic_model = models.replace_ReLU_with_PReLU(basic_model,
              target_activation='swish')

tt = train.Train(data_path, eval_paths=eval_paths,
                 save_path='Resnet34_V1.h5',
                 basic_model=basic_model, lr_base=0.1, lr_decay=0.5,
                 lr_decay_steps=16, lr_min=1e-5,batch_size=128,
                 random_status=0, eval_freq=2000, output_weight_decay=1)

optimizer = keras.optimizers.SGD(learning_rate=0.1, momentum=0.9)
sch = [
    {"loss": losses.ArcfaceLoss(scale=16), "epoch": 1, "optimizer":
    optimizer},
    {"loss": losses.ArcfaceLoss(scale=64), "epoch": 40},
]
tt.train(sch, 0)
```

7.2.2. Segundo Entrenamiento

Para el segundo entrenamiento que se realizó, se utilizó la red neuronal *MobileNet* y al igual que en el entrenamiento anterior, la cantidad de épocas que se estipuló fueron cuarenta, más una extra para la optimización. EL algoritmo utilizado fue el siguiente:

```
from tensorflow import keras
import losses, train, models
import tensorflow_addons as tfa

data_path = 'FaceID_folder'
eval_paths = ['FaceID.bin']

basic_model = models.buildin_models("mobilenet",
                dropout=0, emb_shape=256, output_layer='GDC')
tt = train.Train(data_path, save_path='mobilenetv2.h5',
        eval_paths=eval_paths,
     basic_model=basic_model, lr_base=0.001,
        batch_size=128, random_status=0)
optimizer = tfa.optimizers.AdamW(learning_rate=0.001,
                weight_decay=5e-5)
sch = [
    {"loss": keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
            "epoch": 1, "optimizer": optimizer},
    {"loss": losses.ArcfaceLoss(scale=64), "epoch": 40,
            "bottleneckOnly": True},
]
tt.train(sch, 0)
```

7.2.3. Tercero Entrenamiento

La ultima red escogida para realizar el entrenamiento fue la conocida como: *GhostNet*. Para esta red neuronal se utilizó un algoritmo parecido a los anteriores, pero adaptándolo para esta.

```
from tensorflow import keras
import losses, train, models
import tensorflow_addons as tfa

data_path = 'FaceID_folder'
eval_paths = ['FaceID.bin']

basic_model = models.buildin_models("ghostnet", dropout=0,
                emb_shape=512, output_layer='GDC',
                bn_momentum=0.9, bn_epsilon=1e-5)
basic_model = models.add_l2_regularizer_2_model(basic_model,
                weight_decay=5e-4, apply_to_batch_normal=False)
basic_model = models.replace_ReLU_with_PReLU(basic_model)

tt = train.Train(data_path, eval_paths=eval_paths,
        save_path='Ghostnetv2.h5', basic_model=basic_model,
        model=None, lr_base=0.1, lr_decay=0.5, lr_decay_steps=16,
        lr_min=1e-5, batch_size=128, random_status=0, eval_freq=2000,
        output_weight_decay=1)

optimizer = keras.optimizers.SGD(learning_rate=0.1, momentum=0.9)
sch = [
    {"loss": losses.ArcfaceLoss(scale=32), "epoch": 1,
        "optimizer": optimizer},
    {"loss": losses.ArcfaceLoss(scale=64), "epoch": 40},
]
tt.train(sch, 0)
```

7.2.4. Cuarto Entrenamiento

En este ultimo entrenamiento se volvió a utilizar ResNet34, pero esta vez se utilizó la red con *Fine-tuning*, esto quiere decir que estaba previamente entrenada y que podría significar una mejoría notoria en los resultados obtenidos previamente. El algoritmo utilizado, un tanto diferente a los utilizados anteriormente, es el siguiente:

```
from tensorflow import keras
import losses, train, models
import tensorflow_addons as tfa

data_path = 'FaceID_folder'
eval_paths = ['FaceID.bin']

tt = train.Train(data_path, save_path='ArcFace_V3.h5',
                 eval_paths=eval_paths, basic_model=None,
                 model='arcface_weights.h5', batch_size=128,
                 random_status=0, lr_base=0.1, lr_decay=0.5,
                 lr_decay_steps=16, lr_min=1e-5)
optimizer = tfa.optimizers.SGDW(learning_rate=0.1,
                                momentum=0.9, weight_decay=5e-5)
sch = [
  {"loss": losses.ArcfaceLoss(scale=16), "epoch": 1, "optimizer":
  optimizer},
  {"loss": losses.ArcfaceLoss(scale=64), "epoch": 40},
]
tt.train(sch, 0)
```

7.2.5. Entrenamientos Extras

Luego de concluir con el entrenamiento de todas las redes neuronales y ver los resultados en las que no se utilizó *Fine-tuning*, se decidió volver a realizar un entrenamiento a

las tres primeras, esta vez utilizando veinte iteraciones adicionales para observar la evolución del aprendizaje de estas, de esta forma se podría comprobar si era posible igualar o superar los resultados obtenidos por ResNet con Fine-tuning.

8. Resultados

Antes de comenzar a realizar los entrenamientos de las redes neuronales, se agregó una función en uno de los algoritmos de entrenamiento y evaluación, la cual entregaría por pantalla parámetros que son fundamentales a la hora de hacer un análisis del rendimiento, tales como: *precision*, *recall*, *specificity*, desviación estándar del *accuracy*, así como también la curva ROC y la curva PR para cada una de las redes utilizadas.

8.1. ResNet34

Como resultado de este primer entrenamiento, no se consiguió uno de los objetivos de este proyecto de investigación, el cual era obtener un *accuracy* sobre el 90 %. También hay que mencionar que de todos los entrenamientos realizados, esta fue la que más demoraba por cada iteración, tanto de entrenamiento, como de evaluación y validación, aproximadamente una hora por cada una, por lo que tomó más tiempo en comparación con el resto.

Métricas	**Resultados**
Specificity	0.49
Precision	0.50
Recall	0.45
Accuracy	**0.54**
STD Accuracy	0.081

Tabla 1: Resumen de resultados obtenidos del entrenamiento de la red neuronal ResNet34.

Ahora bien, a continuación se puede observar tanto la curva ROC como la curva RP (Precision/Recall) de ResNet34.

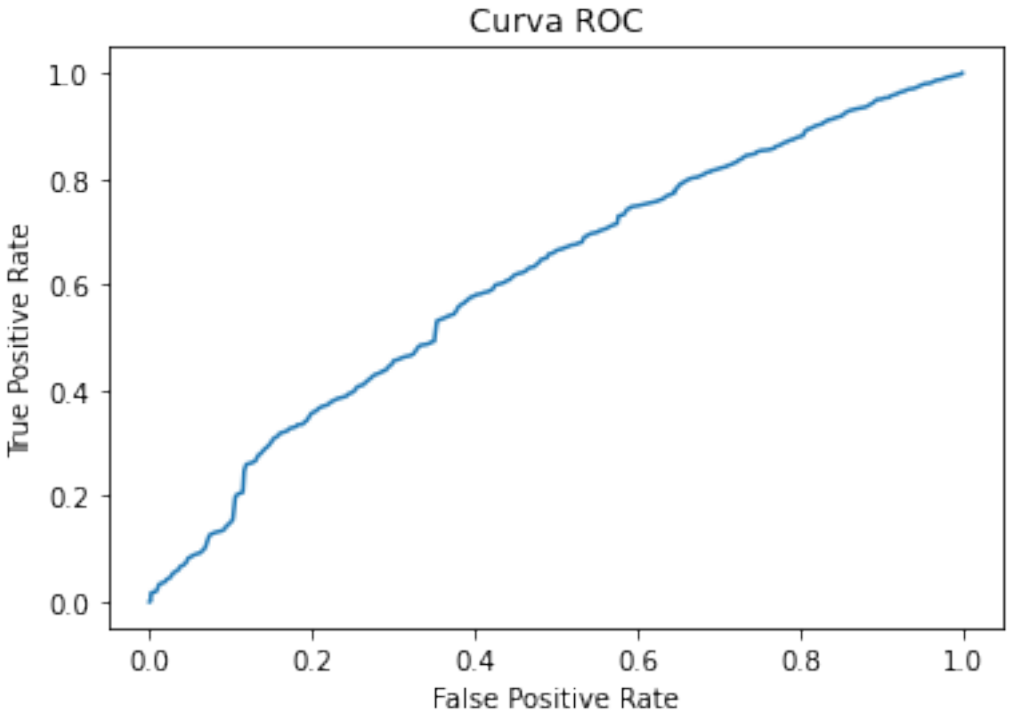


Figura 13: Curva ROC de la red neuronal ResNet34

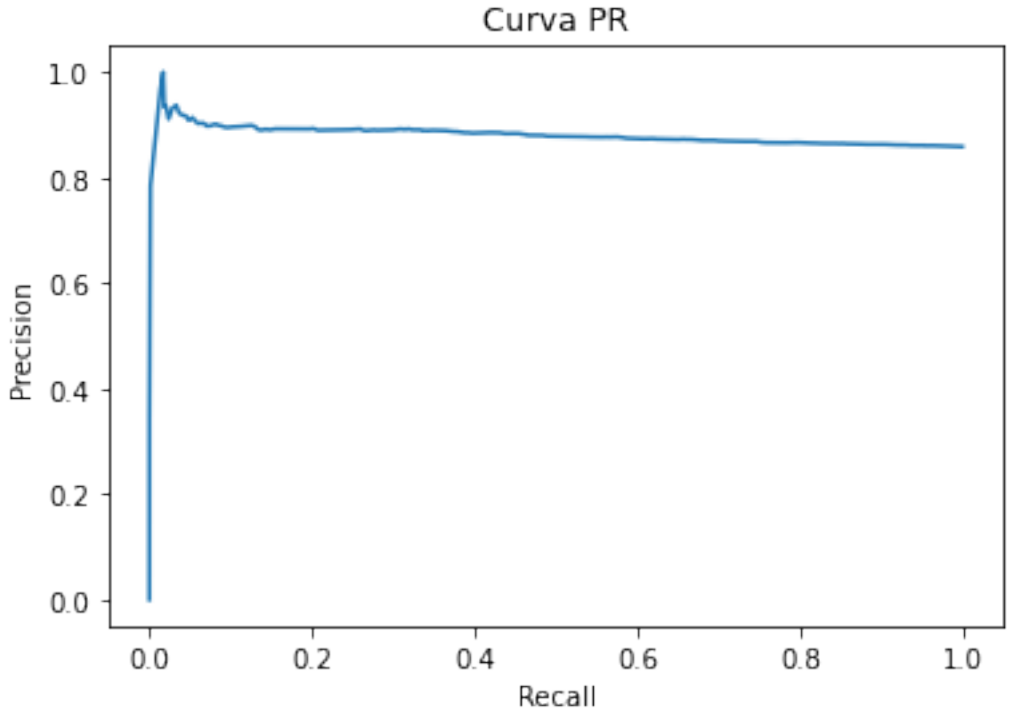


Figura 14: Curva PR de la red neuronal ResNet34

8.2. MobileNet

Al igual que con la red anterior, con tan solo cuarenta y un épocas, no fue capaz de alcanzar el resultado esperado, pero en comparación con ResNet34, esta fue la red neuronal que más rápido completaba cada iteración, cada una terminaba en aproximadamente diez minutos, por lo que fue la red que más rápido completo el ciclo de entrenamiento completo. Los métricas obtenidas de esta red fueron las siguientes:

Métricas	**Resultados**
Specificity	0.63
Precision	0.52
Recall	0.71
Accuracy	**0.68**
STD Accuracy	0.019

Tabla 2: Resumen de resultados obtenidos del entrenamiento de la red neuronal Mobilenet.

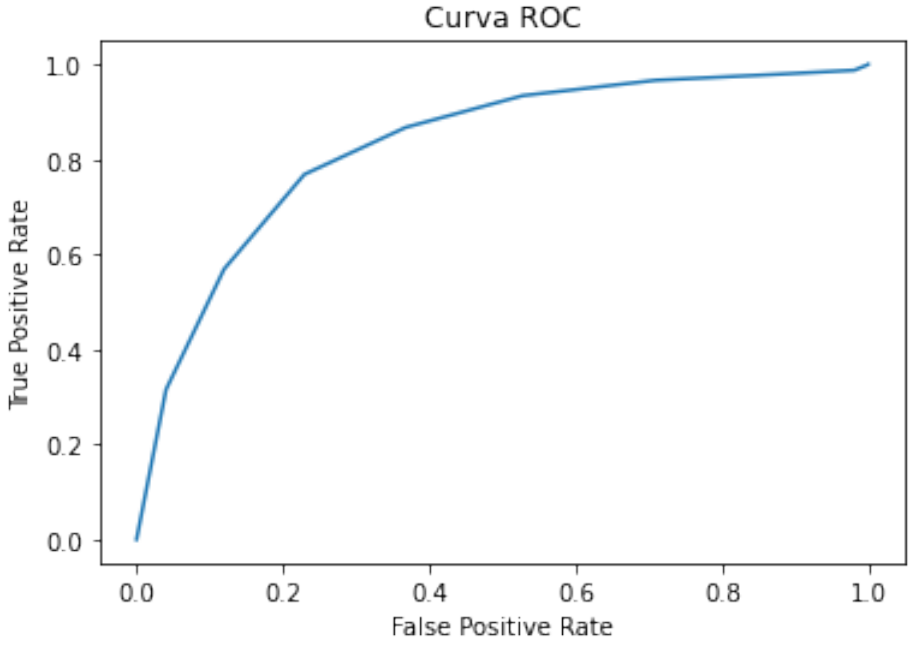


Figura 15: Curva ROC de la red neuronal Mobilenet

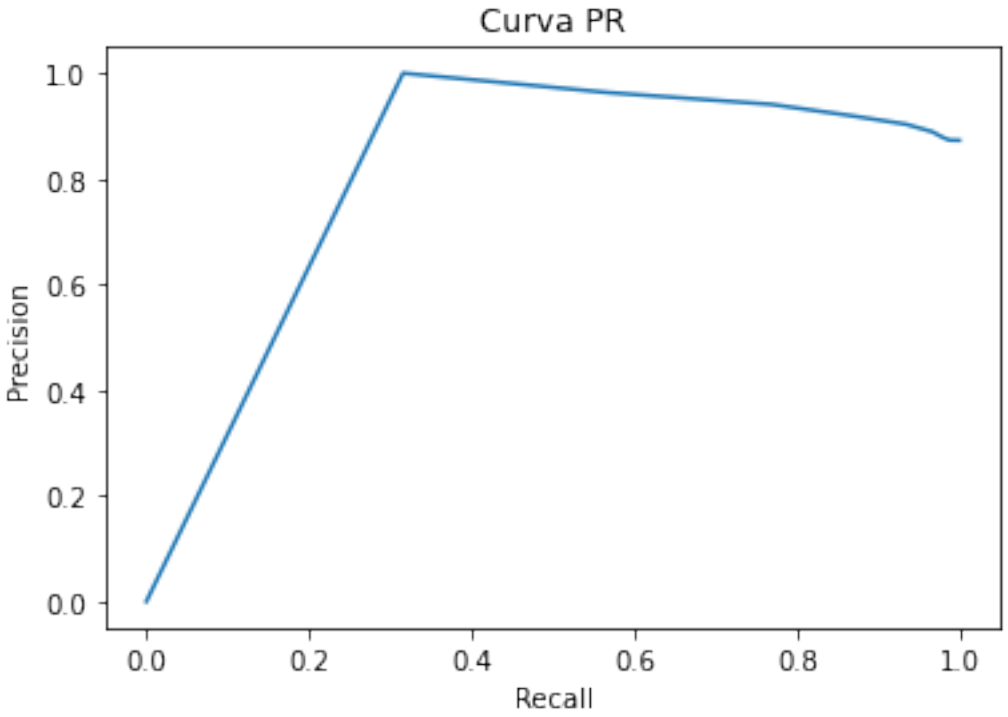


Figura 16: Curva PR de la red neuronal Mobilenet

8.3. GhostNet

De las tres primeras redes, esta fue la que tuvo el mejor resultado de *accuracy*, pero como sus predecesoras, tampoco logró los resultados esperados en un comienzo y solamente en este parámetro fue superior a la anterior. Al igual que Mobilenet, fue una de las más rápidas en completar iteraciones, con un tiempo aproximado de entre doce y quince minutos por cada una de las épocas. Los resultados que se obtuvieron son los siguientes:

Métricas	**Resultados**
Specificity	0.57
Precision	0.52
Recall	0.74
Accuracy	**0.72**
STD Accuracy	0.055

Tabla 3: Resumen de resultados obtenidos del entrenamiento de la red neuronal Ghostnet.

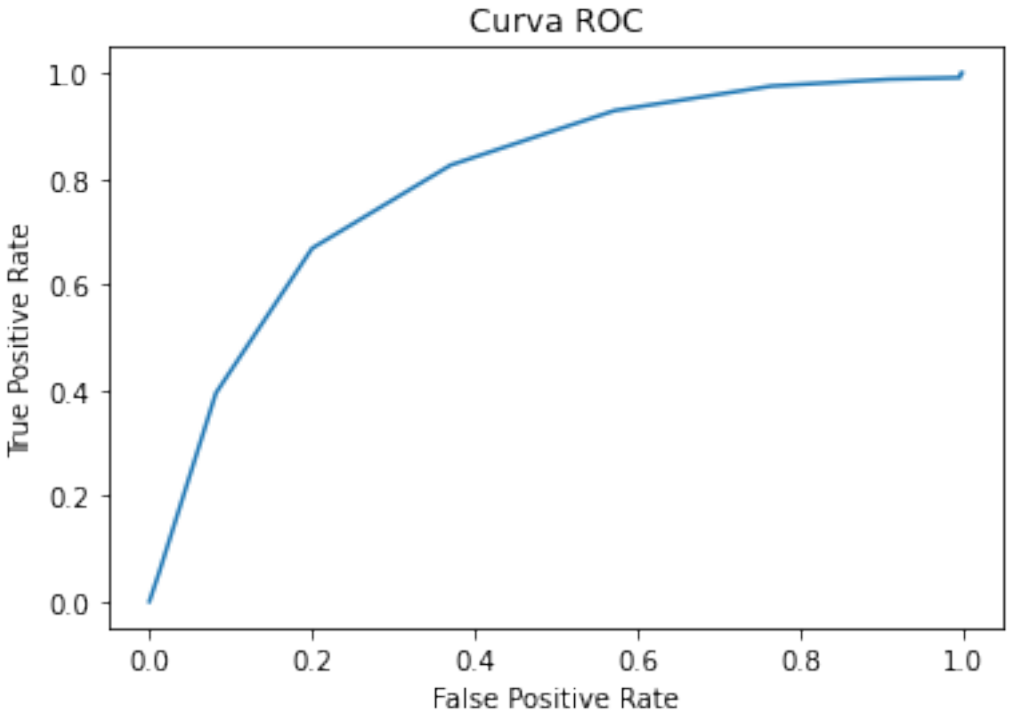


Figura 17: Curva ROC de la red neuronal Ghostnet

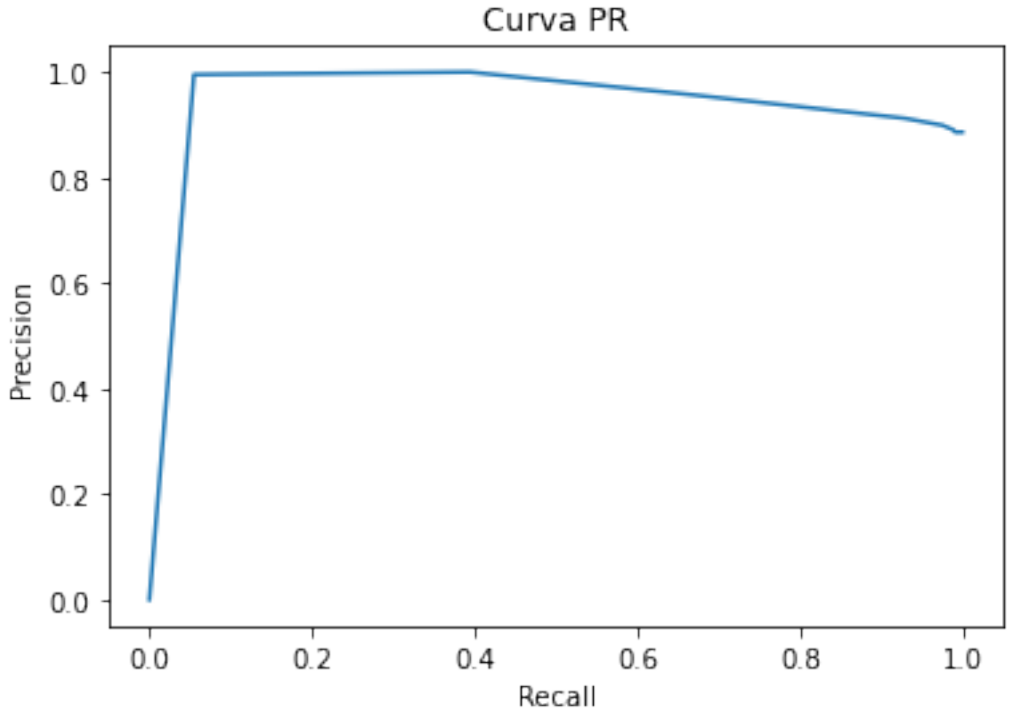


Figura 18: Curva PR de la red neuronal Ghostnet

8.4. ResNet34 (Fine-tuning)

Este ultimo entrenamiento fue en el que se obtuvo mejor resultado, en tan solo cuarenta y un épocas, no sólo logró alcanzar un *accuracy* del 90 %, si no que lo superó, llegando a un 99 %. Esto gracias a que se utilizó ResNet34 con pesos pre entrenados, lo que facilitó el entrenamiento de la red neuronal con la base de datos FaceID-550. El único inconveniente de la red neuronal que es que al igual que su versión sin finetunig, cada iteración demoró entre cuarenta y cinco minutos hasta una hora. Los resultados para cada una de las métricas son:

Métricas	**Resultados**
Specificity	0.90
Precision	0.88
Recall	0.90
Accuracy	**0.9987**
STD Accuracy	0.003

Tabla 4: Resumen de resultados obtenidos del entrenamiento de la red neuronal ResNet34 con finetuning.

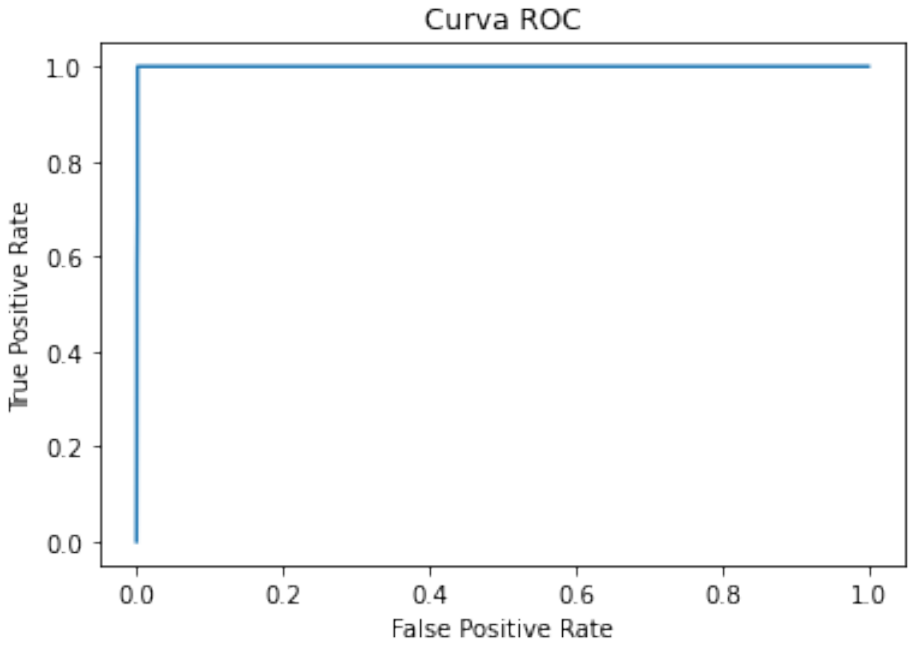


Figura 19: Curva ROC de la red neuronal ResNet34

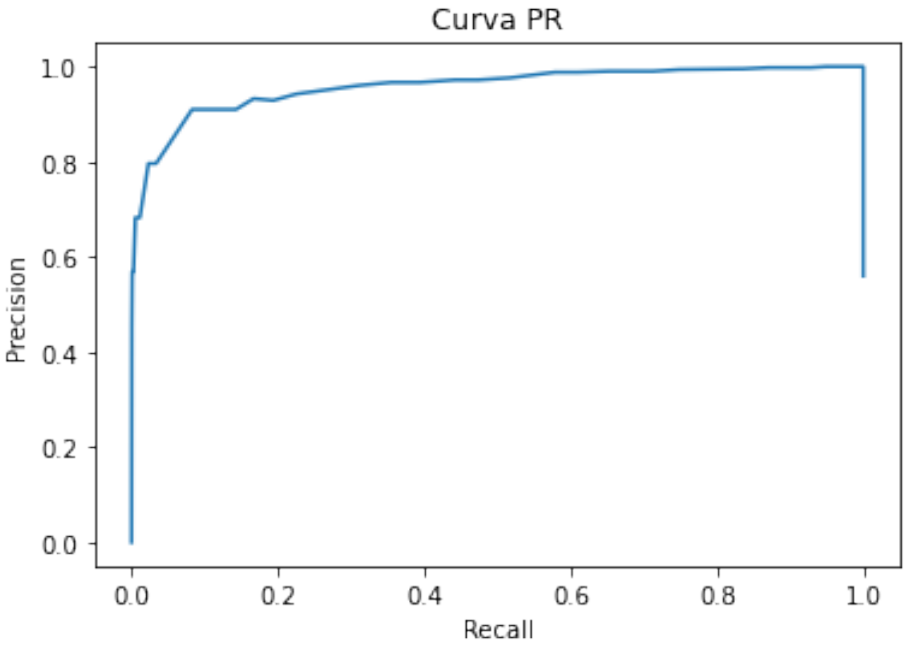


Figura 20: Curva PR de la red neuronal ResNet34

8.5. Entrenamientos Extra

Se puede apreciar una leve mejora en cada uno de los parámetros luego de las veinte iteraciones extra, pero aún así esta no logró superar el rendimiento de ResNet34 con Fine-Tuning. Esto se repite también con las otras dos redes neuronales, las cuales tienen un pequeño aumento en el su rendimiento.

8.5.1. ResNet34

Métricas	**Resultados**
Specificity	0.50
Precision	0.50
Recall	0.68
Accuracy	**0.59**
STD Accuracy	0.083

Tabla 5: Resumen de resultados obtenidos del segundo entrenamiento de la red neuronal ResNet34.

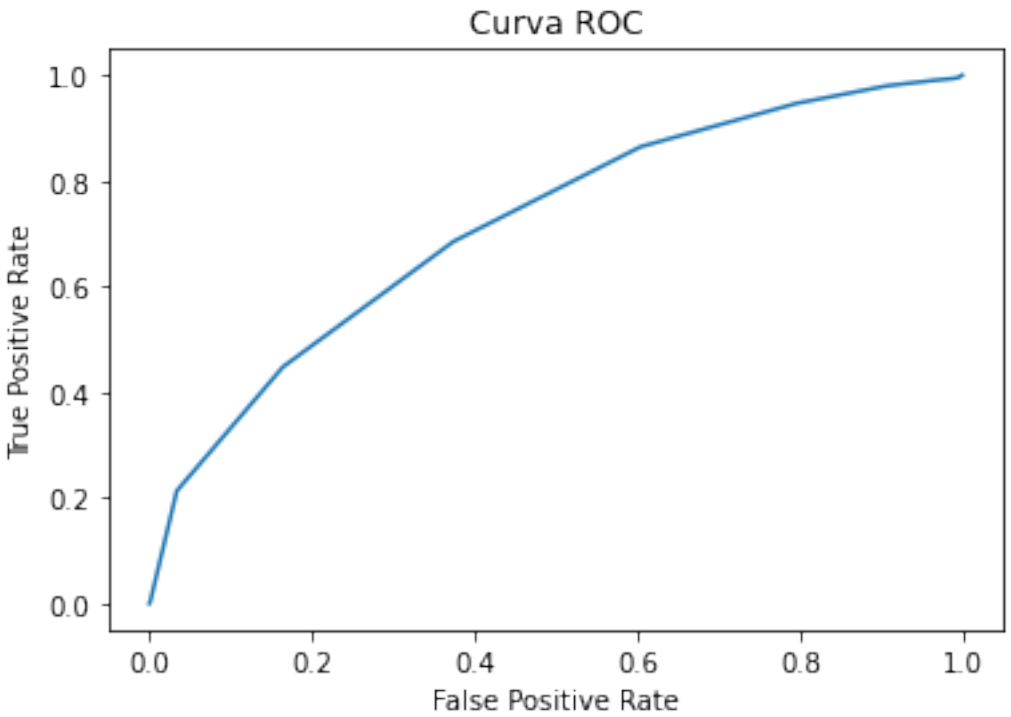


Figura 21: Curva ROC de la red neuronal ResNet34

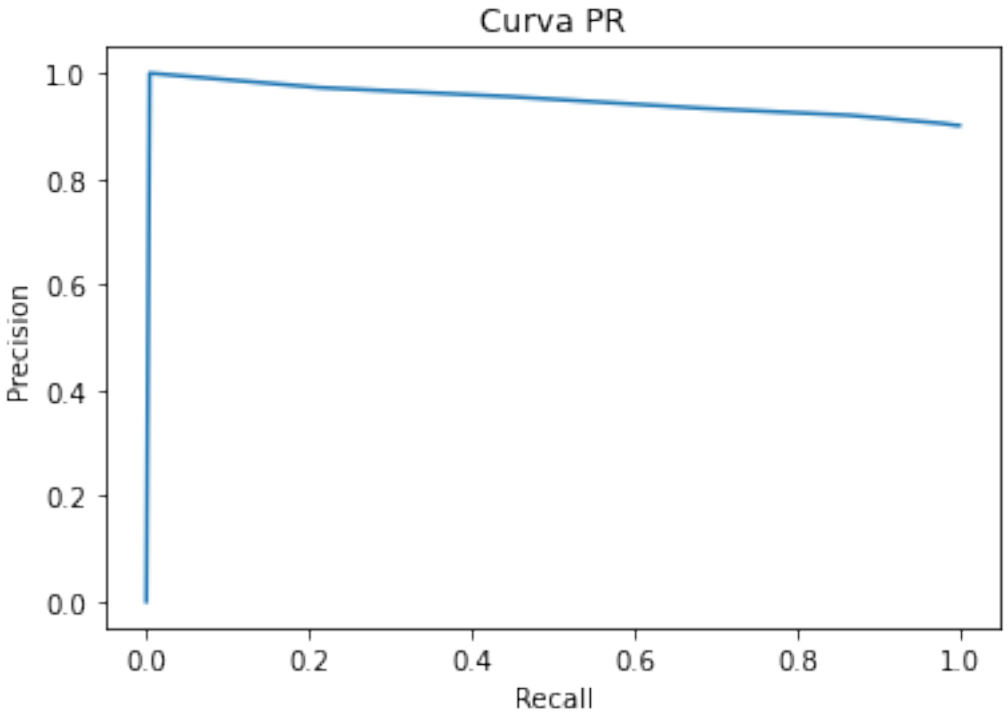


Figura 22: Curva PR de la red neuronal ResNet34

8.5.2. MobileNet

Métricas	**Resultados**
Specificity	0.64
Precision	0.52
Recall	0.72
Accuracy	**0.70**
STD Accuracy	0.037

Tabla 6: Resumen de resultados obtenidos del segundo entrenamiento de la red neuronal MobileNet.

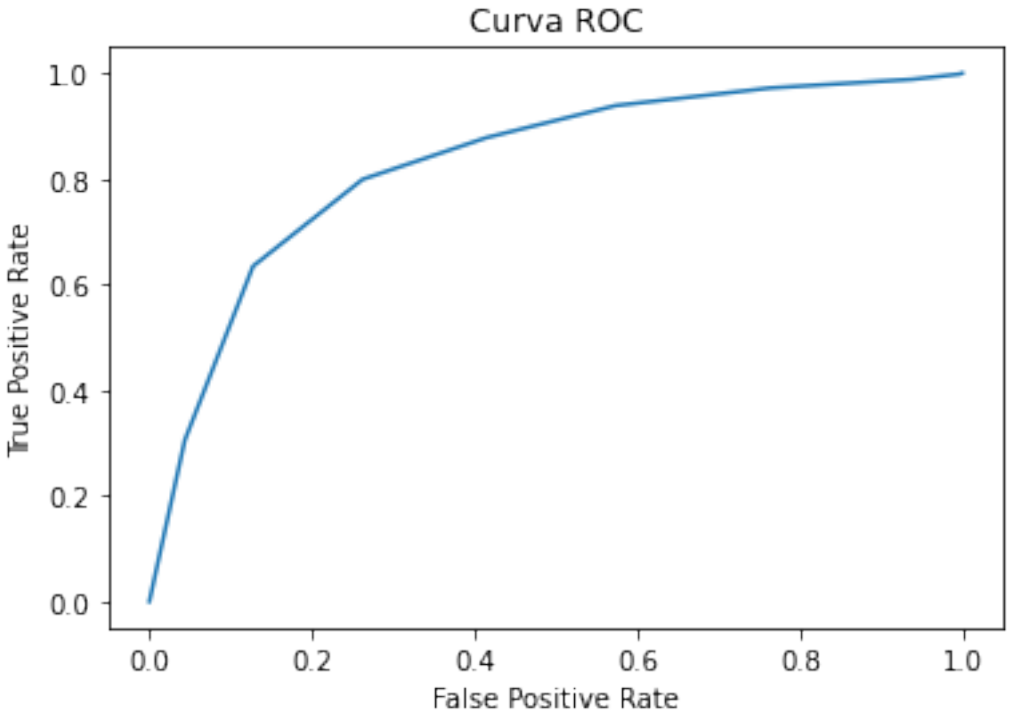


Figura 23: Curva ROC de la red neuronal MobileNet luego de su segundo entrenamiento

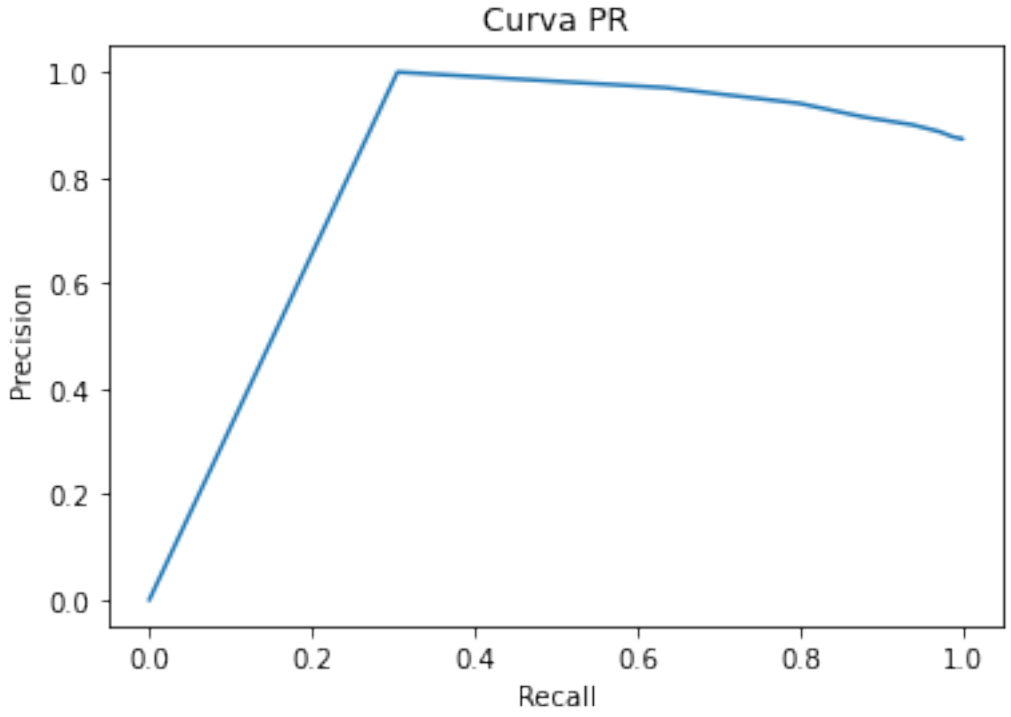


Figura 24: Curva PR de la red neuronal MobileNet luego de su segundo entrenamiento

8.5.3. GhostNet

Métricas	Resultados
Specificity	0.62
Precision	0.53
Recall	0.77
Accuracy	**0.76**
STD Accuracy	0.033

Tabla 7: Resumen de resultados obtenidos del segundo entrenamiento de la red neuronal GhostNet.

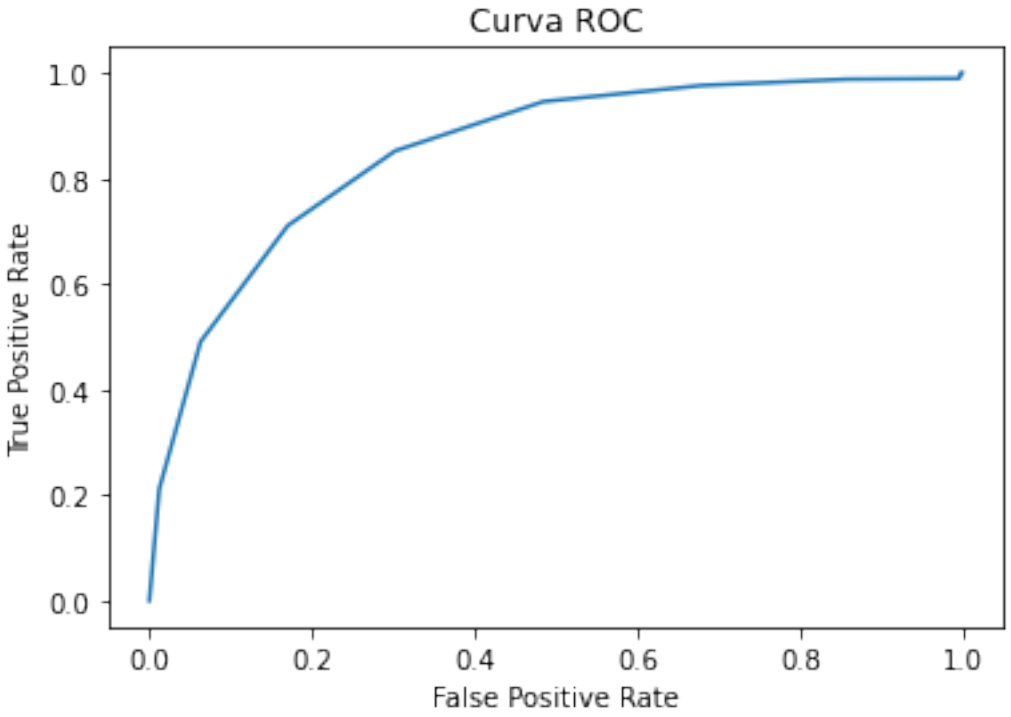


Figura 25: Curva ROC de la red neuronal GhostNet luego de su segundo entrenamiento

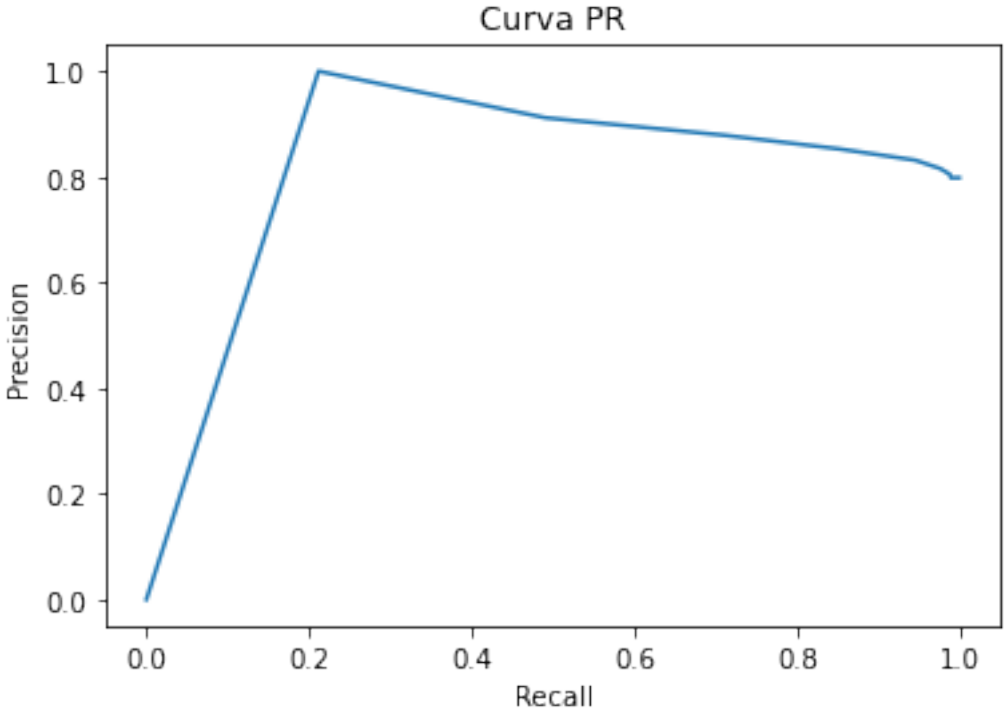


Figura 26: Curva PR de la red neuronal GhosNet luego de su segundo entrenamiento

9. Análisis y discusión de los datos

Luego de haber concluido los entrenamientos de cada una de las redes neuronales, incluido el entrenamiento extra que se le realizo a cada una de aquellas en las que no se utilizó Fine-tuning, se pudo apreciar a simple vista que la red neuronal ResNet34 con fine-tuning era superior en todos los sentidos a las demás, como se aprecia en el siguiente cuadro y los siguientes gráficos:

	ResNet34	**Mobilenet**	**Ghostnet**	**ResNet34(Finetuning)**
Specificity	0.50	0.64	0.62	0.90
Precision	0.50	0.52	0.53	0.88
Recall	0.68	0.72	0.77	0.90
Accuracy	0.59	0.70	0.76	0.9987
STD Accuracy	0.083	0.037	0.033	0.003

Tabla 8: Resumen de las métricas obtenidas del entrenamiento de todas las redes neuronales.

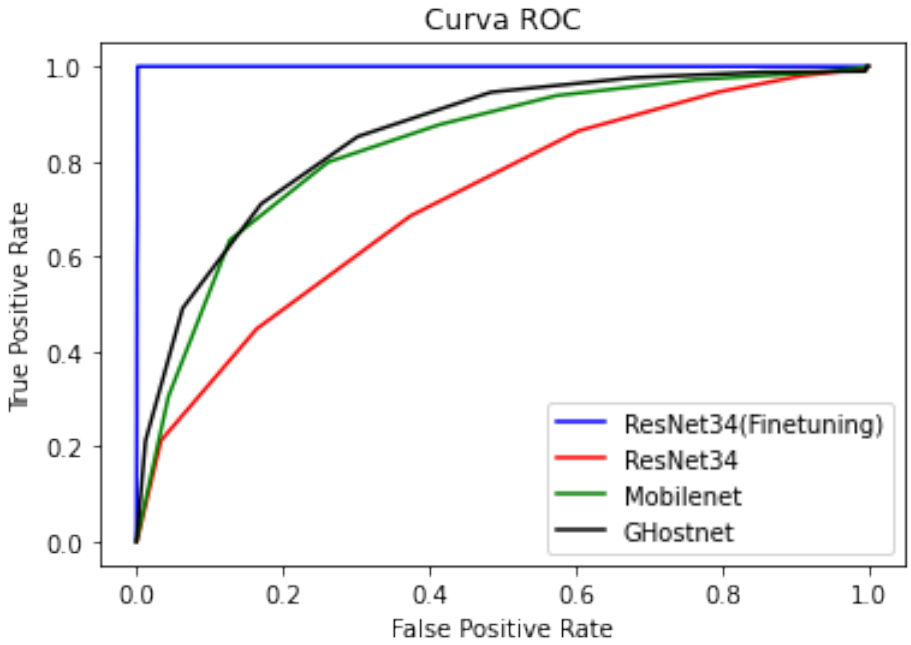


Figura 27: Curva ROC de todas las redes utilizadas

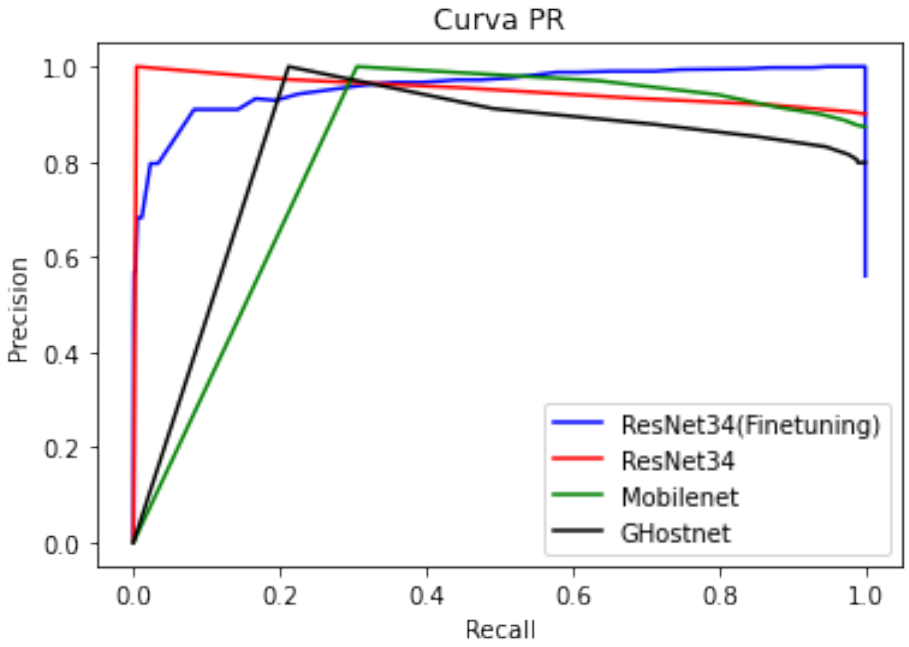


Figura 28: Curva PR de todas las redes utilizadas

ResNet34 con pesos pre entrenados es superior en todos los parámetros, esto demuestra la importancia de utilizar este tipo de redes para ahorrar recursos computacionales y de tiempo, ya que realmente no requirió de las cuarenta y un épocas para alcanzar esas métricas superiores al objetivo, sino que aproximadamente en la iteración veinticinco ya había comenzado a obtener resultados superiores a 90 %.

Esto no quiere decir que las demás redes sean malas o que no sirvan para el tipo de verificación facial "carnet-selfie", si no que se necesita aún más tiempo para lograr resultados mejores. Las tres redes neuronales aún seguían aprendiendo al final de las veinte iteraciones extra que se agregaron al entrenamiento, por lo que si la cantidad de épocas fuese aún mayor, estas posiblemente obtendrían mejores resultados, comparables a los que obtuvo la red con Fine-tuning.

9.1. Análisis de ResNet34 (Fine-tuning)

Luego de obtener los resultados esperados con esta red, se escogió aleatoriamente catorce pares utilizados para realizar el test y poder observar en cuales de estos la red neuronal no predecía correctamente.

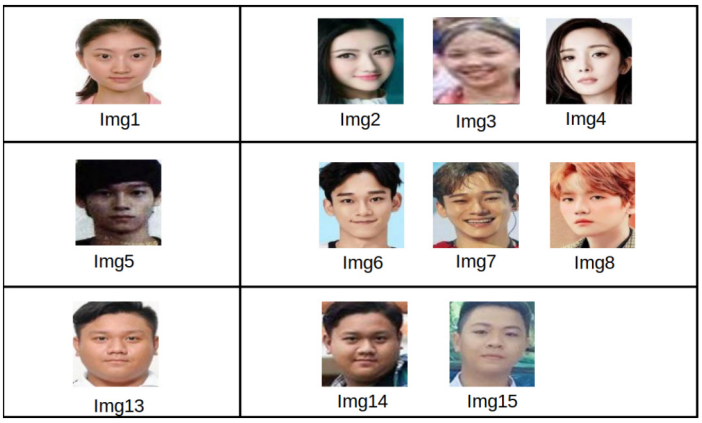


Figura 29: Primer grupo de pares del dataset FaceID-550

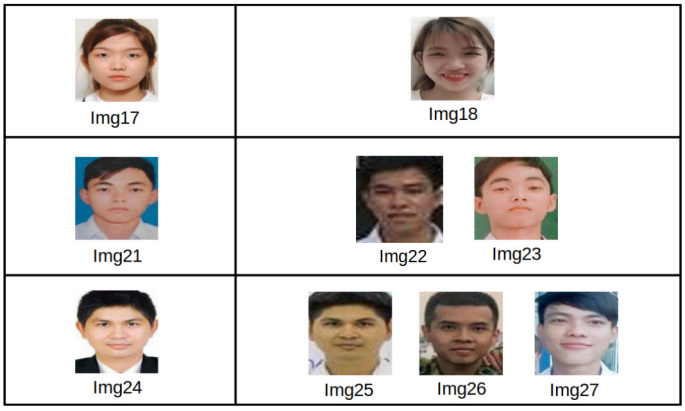


Figura 30: Segundo grupo de pares del dataset FaceID-550.

En el lado izquierdo de las figuras 29 y 30 tenemos las imágenes ID del dataset, las que simulan una fotografía de carnet y/o de pasaporte y junto a ellas, en la parte derecha, las que fueron utilizadas para completar los pares y realizar la predicción. Los resultados obtenidos fueron los siguientes:

```
Test con ResNet34

Debería ser TRUE (img1-img2) {'Verificación de identidad': True}
Debería ser FALSE (img1-img3) {'Verificación de identidad': True}
Debería ser FALSE (img1-img4) {'Verificación de identidad': False}
Debería ser TRUE (img5-img6) {'Verificación de identidad': True}
Debería ser TRUE (img5-img7) {'Verificación de identidad': True}
Debería ser FALSE (img5-img8) {'Verificación de identidad': False}
Debería ser TRUE (img13-img14) {'Verificación de identidad': True}
Debería ser FALSE (img13-img15) {'Verificación de identidad': False}
Debería ser TRUE (img17-img18) {'Verificación de identidad': True}
Debería ser FALSE (img21-img22) {'Verificación de identidad': True}
Debería ser TRUE (img21-img23) {'Verificación de identidad': True}
Debería ser TRUE (img24-img25) {'Verificación de identidad': True}
Debería ser FALSE (img24-img26) {'Verificación de identidad': True}
Debería ser FALSE (img24-img27) {'Verificación de identidad': False}
```

Figura 31: Resultados obtenidos del test realizado a los pares.

En la parte izquierda de la figura 31 se muestran los resultados que debería arrojar la predicción, junto con el par en cuestión y en la parte derecha lo que predice la red neuronal. De todos los pares, en sólo tres hay errores. Como se puede observar, solamente en tres pares hay errores de predicción, en los pares: “img1-img3”, “img21-img6” e “img24-img26”.



Figura 32: Pares en los que la predicción de la red fue fallida.

Lo que ResNet34 debería haber arrojado en estos tres pares era un “False”, ya que no son la misma persona, pero en todos se repitió la misma tónica y es que la red neuronal consideraba que si lo eran. Como se puede observar en la figura 32, tanto la luz, como la calidad de las imágenes que están posicionadas en la parte derecha de los pares no es considerablemente óptima, lo que pudo influir a la hora de realizar la predicción.

A día de hoy, los teléfonos móviles poseen cámaras cada vez mejores, por lo que este tipo de casos no se vería reflejado en caso de que se quisiera implementar esta red en un sistema de verificación facial para una aplicación Android o IOS.

Ahora bien, también hay que destacar algunos casos en los que si acertó la red neuronal en situaciones en las que la "selfie" cambiaba en comparación con la fotografía de carnet.



Figura 33: Pares en los que la predicción de la red acertó.

Como se puede observar en la figura 33, en el primer par, concretamente en la selfie de la persona, esta se encuentra en un ángulo distinto, además de que está con maquillaje, comparada con la imagen de carnet y aún así ResNet34 logró predecir exitosamente. Mismo caso ocurre en segundo par, en donde la persona o más bien, la iluminación, hace que se vea su cabello de un color distinto al original, además de que está sonriendo y a pesar de ello, se logró obtener el resultado correcto en la predicción.

10. Conclusiones

El objetivo de este proyecto era implementar una red neuronal que sea capaz de verificar la identidad de una persona con un *accuracy* igual o superior a un 90 %, por lo cual se recurrió a diferentes algoritmos para adaptar la base de datos escogida, que en este caso fue FaceID-550. Este dataset contenía una gran cantidad de imágenes de personas, de origen Asiático, cada una contaba con una fotografía de pasaporte o de cédula de identidad y otras en distintos ángulos.

Para poder encontrar la mejor red se realizaron entrenamientos a todas las redes neuronales, estableciendo las mismas condiciones para cada una de ellas. No obstante, luego de observar los resultados y dar con una red neuronal capaz de superar el objetivo planteado, llegando a un *accuracy* de 99 %, se decidió agregar veinte iteraciones extras de entrenamiento a las redes neuronales que no contaban con pesos pre entrenados para observar el crecimiento de su rendimiento para cada una. A pesar de obtener mejoras en cada una de ellas, no fue demasiado grande el crecimiento, la red que mejor crecimiento en su *accuracy* fue GhostNet, con un 5 % aproximadamente, por lo que se decidió concluir con los entrenamientos, ya que los recursos computacionales y de tiempo, eran superiores a los esperados.

Por ultimo, la hipótesis planteada se puede dar por confirmada, debido a que, como se mencionó anteriormente, la red neuronal ResNet34 no solo logró igualar lo planteado, si no que lo superó llegando a obtener un rendimiento cercano al 100 %. También es importante destacar que las demás redes utilizadas en esta investigación, en las cuales no se aplicó la técnica de Fine-tuning, son capaces de seguir aumentando su *accuracy*, por lo que esto da paso a seguir trabajando con estas redes, ya que si se establecen muchas más épocas de entrenamiento, posiblemente estas obtengan resultados iguales o superiores a los esperados.

11. Limitaciones, ideas y trabajos futuros

11.1. Limitaciones

Una de las grandes limitaciones de este proyecto de investigación, fue el poder entrenar cómodamente las redes sin Fine-tuning. Si bien se logró con el objetivo de entrenar una red neuronal con resultados superiores al 90 %, el resto de redes no lo consiguió, aún cuando al finalizar los entrenamientos se realizaron otras veinte iteraciones extras para su entrenamiento. Debido a que cada entrenamiento consumía prácticamente todos los recursos del equipo, por lo que no se podía utilizar para nada más, lo que limitaba el acceso a clases online o para reuniones durante el semestre.

11.2. Ideas y Trabajos Futuros

Una de las grandes ideas que surgió finalizando este proyecto es la de implementar no solo dos fotografías para realizar la verificación, si no que añadir una más, a la cédula de identidad y a la selfie de frente de la persona, se le sumaría una foto de perfil, de esta manera se haría mucho más robusto el sistema y dificultaría aún más la suplantación de identidad. Para este caso se logró encontrar una base de dato que contaba con fotos tanto de frente como de perfil y se le realizó sin problemas el pre procesamiento de las fotos, así como se hizo con FaceID-550, el problema fue que esta base de datos no contaba con fotos similares a la cédula de identidad. Se pensó en escoger fotos de frente de cada una de las personas y reducirles la calidad, así como añadirle un efecto blanco y negro para simular de manera correcta la imagen del carné, pero al ser una cantidad muy grande de imágenes y al poco tiempo que se contaba, no se logró llevar a cabo el entrenamiento de la red.

Otra idea que también se consideró, también a pocos meses de finalizar el proyecto, fue la de enfocar la atención del entrenamiento en el fondo de las fotos del usuario, no en las caras, así se podría corroborar que la persona que está intentando utilizar el sistema se encuentra en tiempo real y no esta utilizando fotografías falsificas o sacadas de las redes

sociales y/o internet.

Por ultimo, otra idea que se consideró en un punto del proyecto, fue la de hacer que el sistema tuviera reconocimiento de movimiento, gracias a esto se podría evitar hacer un registro utilizando tres fotografías y solamente utilizar la imagen del carnet de identidad junto con este "sensor", así se podría evitar la utilización de fotografías impresas para suplantar identidad.

Bibliografía

[1] Damián Jorge Matich. Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, 41:12–16, 2001.

[2] Manlio Massiris, Claudio Delrieux, and Juan Álvaro Fernández Muñoz. Detección de equipos de protección personal mediante red neuronal convolucional yolo. In *XXXIX Jornadas de Automática*, pages 1022–1029. Área de Ingeniería de Sistemas y Automática, Universidad de Extremadura, 2018.

[3] Shadman Sakib, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. An overview of convolutional neural network: its architecture and applications. 2019.

[4] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.

[5] Noor M Al-Moosawi M Al, Raidah Salim Khudeyer, et al. Resnet-34/dr: A residual convolutional neural network for the diagnosis of diabetic retinopathy. *Informatica*, 45(7), 2021.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Debjyoti Sinha and Mohamed El-Sharkawy. Thin mobilenet: An enhanced mobilenet architecture. In *2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON)*, pages 0280–0285. IEEE, 2019.

[8] Wei Wang, Yutao Li, Ting Zou, Xin Wang, Jieyu You, and Yanhong Luo. A novel image classification approach via dense-mobilenet models. *Mobile Information Systems*, 2020, 2020.

[9] Bangtong Huang, Hongquan Zhang, Zihong Chen, Lingling Li, and Lihua Shi. Research on efficient deep learning algorithm based on shuffleghost in the field of virtual reality. *Wireless Communications and Mobile Computing*, 2021, 2021.

[10] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020.

[11] Akitoshi Okumura, Takamichi Hoshino, Susumu Handa, Yugo Nishiyama, and Masahiro Tabuchi. Identity verification of ticket holders at large-scale events using face recognition. *Journal of Information Processing*, 25:448–458, 2017.

[12] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Hybrid deep learning for face verification. In *Proceedings of the IEEE international conference on computer vision*, pages 1489–1496, 2013.

[13] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. *Advances in neural information processing systems*, 27, 2014.

[14] Jun-Cheng Chen, Vishal M Patel, and Rama Chellappa. Unconstrained face verification using deep cnn features. In *2016 IEEE winter conference on applications of computer vision (WACV)*, pages 1–9. IEEE, 2016.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[16] Md Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, Liming Chen, et al. von mises-fisher mixture model-based deep learning: Application to face verification. *arXiv preprint arXiv:1706.04264*, 2017.