Practical approach of application the Deep Learning Toolbox by Matlab in the facial recognition of students

Nestor Asbel Cayllahua Aquino, Ing[1], Juan Carlos Suárez Macedo, Ing[2], Pedro Huamaní-Navarrete, Dr[3],
[1,2,3]Ricardo Palma University, Peru, ncayllahua102@gmail.com, juancarlos.suarezmacedo@gmail.com,
phuamani@urp.edu.pe

Abstract–The purpose of this article was to provide a practical approach to the application of the Toolbox Deep Learning, from the Matlab software, for facial recognition of a group of students from the Mechatronics Engineering Program, at the Ricardo Palma University, Lima-Peru. To do this, the methodology used consisted of defining the architecture, configuration and training of a deep learning convolutional neural network, to extract the relevant data of facial features in the photographs taken of the group of students. The sample used was 426 photographs corresponding to 14 students; Likewise, different tests were carried out to select the most appropriate number of layers, the highest percentage of precision and the shortest training time. And, the best result corresponded to the use of two convolution layers, of 16 and 32 filters, respectively, with a precision percentage of 94.00% in the facial recognition of the group of students.

Keywords: ***facial recognition, Deep Learning toolbox, convolutional neural network, convolution, precision percentage.***

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2020.1.1.603
ISBN: 978-958-52071-4-1 ISSN: 2414-6390

Enfoque práctico de la aplicación del Toolbox Deep Learning del Matlab en el reconocimiento facial de estudiantes

Nestor Asbel Cayllahua Aquino, Ing[1], Juan Carlos Suárez Macedo, Ing[2], Pedro Huamaní-Navarrete, Dr[3],
[1,2,3]Ricardo Palma University, Peru, ncayllahua102@gmail.com, juancarlos.suarezmacedo@gmail.com, phuamani@urp.edu.pe

***Abstract*– The purpose of this article was to provide a practical approach to the application of the Toolbox Deep Learning, from the Matlab software, for facial recognition of a group of students from the Mechatronics Engineering Career, at the Ricardo Palma University, Lima-Peru. For this, the methodology used consisted of defining the architecture, configuration and training of a convolutional neural network of deep learning, to extract the relevant data of facial features in the photographs taken of the group of students. The sample used were 426 photographs corresponding to 14 students; Likewise, different tests were carried out to select the most appropriate number of layers, the highest percentage of precision and the shortest training time. And, the best result corresponded to the use of two convolution layers, of 16 and 32 filters, respectively, with a precision percentage of 94.00% in the facial recognition of the group of students.**

Keywords—Facial recognition, Deep Learning Toolbox, Convolutional Neural Network, Percentage of precision, Matlab.

I. INTRODUCCIÓN

Con el desarrollo computacional y los algoritmos de inteligencia artificial, el reconocimiento de rostros se ha convertido en uno de los temas más interesantes e importantes en la literatura, en lo que concierne a visión por computadora. Por tal razón, en [1], se muestra el análisis de algunas técnicas de reconocimiento facial y la línea de tiempo en cuanto a diferentes métodos para manejar problemas de reconocimiento facial; asimismo, se hace una comparación entre varias técnicas de procesamiento de imágenes tales como Eigen Faces, Hidden Markov Model (HMM), algoritmos basados en geometría, y algoritmos Template Matching. De esta manera, en [1], se afirma que todas esas técnicas mejoran la calidad, eliminan el ruido, son de naturaleza versátil, y conservan la precisión original de los datos de la imagen.

Por otro lado, con el surgimiento de la inteligencia artificial, son diversas las topologías de redes neuronales que son posibles de utilizar para el reconocimiento facial. Sin embargo, es necesario contar con un número suficiente de rostros para lograr el entrenamiento de la red neuronal. Es así que, en [2], se concluye que para realizar el sistema de reconocimiento facial se debe contar con una base de datos, el número de usuarios, y de preferencia estos deben tener aproximadamente la misma edad. Por tal razón, en este artículo propuesto, se está considerando un grupo de estudiantes del mismo ciclo de estudios, entre hombres y mujeres.

Asimismo, según [3], el reconocimiento facial es una tecnología actualmente en desarrollo con múltiples aplicaciones en la vida real; por lo cual, en dicho trabajo se consideró como objetivo desarrollar un sistema de reconocimiento facial para la empresa basada en inteligencia artificial, GoldenSpear LLC. Es así que, dicho sistema fue desarrollado utilizando redes neuronales convolucionales con 8 capas para extraer datos relevantes, y permitir comparar los rasgos faciales de forma eficiente. De igual manera, dicho sistema fue entrenado para reconocer a un conjunto de personas y aprender en línea sin intervención humana, integrando a las nuevas personas que procesa y mejorando sus predicciones sobre las que ya posee.

Por otro lado, en [4], se alcanza una precisión del 97.35% en el Data Set de Labeled Faces in the Wild (LFW), al utilizar una red neuronal profunda con más de 120 millones de parámetros utilizando varias capas conectadas localmente, pero sin compartir los pesos en lugar de las capas convolucionales convencionales.

De la misma forma, en [5] se señala que el entrenamiento de una Red Neuronal Convolucional (CNN) puede ser problemática en términos de tiempo, recursos y disposición de la base de datos. En un entrenamiento completo, los parámetros de red (los pesos de las diferentes capas) se inicializan aleatoriamente y se aprenden utilizando el algoritmo de descenso de gradiente. Entonces, si el número de parámetros es grande, se necesita una gran cantidad de imágenes para aprender estos parámetros. Por otro lado, un entrenamiento completo puede no ser factible en aplicaciones donde el conjunto de entrenamiento sea pequeño. Esto es cierto especialmente para una CNN muy profunda, que debería ser entrenada con millones de imágenes para obtener muy buenos resultados. Por lo tanto, un conjunto de datos de tamaño limitado puede limitar fuertemente el rendimiento de la red.

Otro punto importante tal como se ha hallado a través de las pruebas realizadas en [6], el desbalanceo en la base de datos de entrenamiento afecta mucho a los resultados obtenidos en el entrenamiento de las redes neuronales convolucionales profundas. Pues, el simple balanceo copiando aleatoriamente las imágenes y equiparando el número de imágenes entre las diferentes clases de entrenamiento mejora los resultados, y si, además, no se hace una simple copia, sino que se realizan transformaciones aleatorias de traslación, rotación y zoom, se mejoran todavía más tales resultados.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2020.1.1.603
ISBN: 978-958-52071-4-1 ISSN: 2414-6390

Y, como también, en [7], se detalla que el estado del arte del reconocimiento facial ha avanzado significativamente con la aparición del aprendizaje profundo; pues, las redes neuronales de aprendizaje profundo han logrado recientemente un gran éxito en el reconocimiento general de objetos debido a su excelente capacidad de estudio. Esto motiva a investigar su efectividad en el reconocimiento facial. Por tal razón, en este artículo se propusieron dos arquitecturas de redes neuronales profundas, las cuales fueron denominadas DeepID3, para el reconocimiento facial. Estas dos arquitecturas se reconstruyen a partir de las capas de convolución e inicio propuestas en la red VGG y Google Net para adecuarlos al reconocimiento facial. De esta manera, las dos arquitecturas propuestas alcanzan un 99.53% de precisión de verificación de rostro LFW y 96.0% de precisión de identificación de rostro de LFW rango 1, respectivamente.

Por lo tanto, debido a que, en los últimos años muchos problemas de clasificación en inteligencia artificial han impulsado su rendimiento mediante el uso de técnicas de aprendizaje profundo, en particular la CNN, el contenido de este artículo se centra en dar un enfoque práctico de la aplicación del Toolbox Deep Learning del Matlab, para el reconocimiento facial de un grupo de estudiantes de la Carrera de Ingeniería Mecatrónica, de la Universidad Ricardo Palma, Lima – Perú, utilizando para ello la teoría del aprendizaje profundo y no de las técnicas de procesamiento digital de imágenes, porque estas últimas necesitan una etapa de pre procesamiento como es la atenuación del ruido y la separación del fondo de la imagen de interés. De esta manera, se inicia con la sección Introducción, seguida de la sección Marco Teórico que contiene los fundamentos de reconocimiento facial, Red Neuronal Convolucional, el Aprendizaje Profundo y el Deep Learning Toolbox del Matlab. Posteriormente, se continúa con la tercera sección titulada como Definición y Entrenamiento de la Red Neuronal Convolucional. Luego, se presenta la cuarta sección denominada Pruebas y Resultados, y se finaliza con las Discusiones, Conclusiones y Agradecimientos.

II. Marco teórico

En esta sección, se hace una breve descripción de los principales conceptos utilizados en este artículo.

A. Fundamentos de reconocimiento facial

Según [8], el primer sistema de reconocimiento facial fue desarrollado en 1973 y desde entonces el estudio de esta parte de la ciencia ha avanzado mucho. Sin embargo, el reconocimiento facial automático aún enfrenta muchos desafíos. Es así que, una de las clasificaciones para los sistemas de reconocimiento facial toma en cuenta la cooperación del usuario. Pues, se tienen escenarios de usuarios cooperativos cuando el usuario está dispuesto a cooperar presentando su rostro de manera adecuada. Este tipo de sistema se encuentra, por ejemplo, en aplicaciones como el control de acceso físico. En los escenarios de usuarios no cooperativos, típico en aplicaciones de vigilancia, el usuario no es consciente de estar siendo identificado. De esta manera, un sistema de reconocimiento facial consta generalmente de cuatro módulos: 1) Detección facial (diferencia lo que es rostro de lo que no lo es como es el caso del fondo de la imagen). En esta fase o módulo se localizan los puntos de referencia del rostro como pueden ser los ojos, la nariz, la boca y el contorno facial. 2) Normalización. Se normaliza o registra el rostro geométrica y fotométricamente. 3). Extracción de las características del rostro. Esta operación se realiza sobre el rostro normalizado, a fin de extraer información relevante que se empleará para buscar coincidencias. 4). Búsqueda de coincidencias. Las características del rostro se comparan con otros rostros registrados en la base de datos.

B. Red Neuronal Convolucional

Según [9] las redes neuronales convolucionales, también denominadas ConvNets, son herramientas ampliamente utilizadas para el aprendizaje profundo. Específicamente son adecuadas para utilizar las imágenes como entradas, aunque también se utilizan para otras aplicaciones como texto, señales y respuestas continuas. Asimismo, según [10], las CNN están inspiradas en la estructura biológica de la corteza visual, que contiene arreglos de células simples y complejas. Además, como es de conocimiento, en una red neuronal con neuronas totalmente interconectadas, el número de parámetros (pesos) puede incrementar rápidamente a medida que aumenta el tamaño de la entrada; sin embargo, una CNN reduce el número de parámetros con el número reducido de conexiones, pesos compartidos, y submuestreo; de esta manera, en forma general, una CNN está compuesta de múltiples capas, como convolucionales, Max-Pooling o Average-Pooling, y Fully-Connected [9]. A continuación, la figura 1 muestra una representación general de una CNN, donde se combinan diferentes tipos de capas para aprender automáticamente las características de las imágenes de entrada. Entre esas capas de neuronas se tiene la de convolución, la de sub-muestreo, y la totalmente conectada.

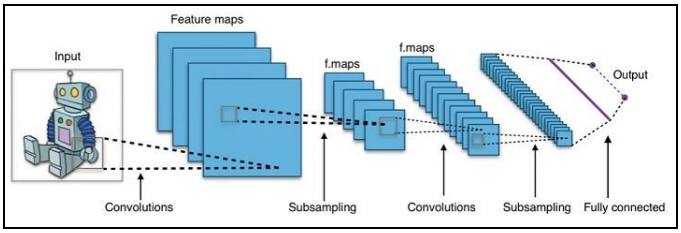

Fig. 1. Representación general de una Red Neuronal Convolucional [11].

C. Deep Learning o Aprendizaje Profundo

Según [12], Deep Learning o Aprendizaje Profundo es una técnica del Machine Learning que utiliza una red neuronal profunda. Asimismo, la red neuronal profunda es una red neuronal multi-capa que está compuesta por dos o más capas ocultas. En general, se acostumbra a utilizar decenas e inclusive cientos de capas, quienes aprenden automáticamente

conforme se realiza el entrenamiento con el conjunto de datos de entrada.

D. *Deep Learning Toolbox*

Según [12], Deep Learning Toolbox™ (anteriormente denominado Neural Network Toolbox™) proporciona un marco de referencia para diseñar e implementar redes neuronales profundas con algoritmos, modelos entrenados y aplicaciones. Por lo cual, se puede utilizar las redes neuronales convolucionales y las redes de memoria a largo plazo (LSTM), para realizar la clasificación y la regresión en imágenes, series temporales y datos de texto. Además, se puede construir arquitecturas de redes avanzadas, como redes de confrontación generativa (GANs) y redes siamesas utilizando bucles de entrenamiento personalizados, pesos compartidos y diferenciación automática. Las aplicaciones y los gráficos ayudan a visualizar activaciones, editar y analizar arquitecturas de redes, y monitorear el progreso del entrenamiento.

III. Definición y Entrenamiento de la Red Neuronal Convolucional

En esta sección se señalan cinco etapas referentes a la definición de la arquitectura de la red neuronal convolucional, las características consideradas para cada una de sus capas de neuronas, el proceso de entrenamiento, y la visualización gráfica del resultado. Asimismo, se incluye el proceso de captura de las imágenes de rostros de los estudiantes. A continuación, la figura 2 representa al diagrama de bloques general de este artículo.

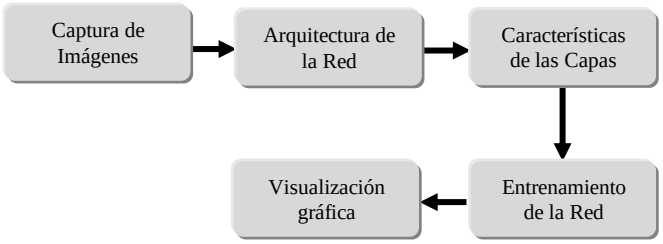

Fig. 2. Diagrama de bloques general.

A. *Primera etapa: captura de imágenes*

Para esta etapa se empleó el objeto "webcam", cuyo fin fue utilizar los dispositivos que capturan imágenes y se encuentran conectados a la PC. Luego, una vez realizada la captura, se creó un repositorio de imágenes, y se procedió a ordenarlas en carpetas dentro de un Data Store con apoyo del Matlab. Posteriormente, se agregaron las respectivas etiquetas a cada imagen para facilitar el análisis de parte de la red neuronal convolucional. A continuación, en la figura 3, se muestra un ejemplo del código de programación empleado para el caso de cuatro estudiantes.

B. *Segunda etapa: arquitectura de la red*

Para definir la arquitectura de la red neuronal convolucional, se realizaron cinco pasos los cuales se presentan a continuación.

```
categories = {'Juan','Luz','Julia','Patrick'};
rootFile = 'rostros';

imds = imageDatastore(fullfile(rootFile,categories),...
    'LabelSource','foldernames');
```

Fig. 3. Declaración del Data Store en Matlab.

- Primer Paso: se modificó el tamaño del conjunto de las imágenes del Data Store, a una resolución de 128x128x3 pixeles utilizando la función "AugmentedImageDatastore" del Toolbox Deep Learning del Matlab.
- Segundo Paso: se definió el tamaño de los "inputs" de la red neuronal convolucional a 128x128x3; luego, se crearon dos capas convolucionales con 16 y 32 filtros de tamaño 4x4; posteriormente, se definieron los parámetros ReLu, MaxPool, drop, fc1, sm1 y class, utilizados en el arreglo "Layers"; y, finalmente, se implementaron las convoluciones y parámetros anteriormente definidos.
- Tercer Paso: se definieron varias opciones de configuración de la red neuronal. Entre ellas tenemos: 'sgdm' (Descenso de gradiente estocástico con impulso), 'InitialRateSchedule' (Rango inicial de la tasa de aprendizaje, 0.0001), 'MaxEpochs' (Cantidad máxima de épocas, 20), 'Mini BatchSize' (Cantidad mínima de lotes observados por cada iteración), 'Plot' (generación gráfica del resultado del aprendizaje de la red neuronal).
- Cuarto Paso: se realizó el entrenamiento a través de la función "trainNetwork" del Add Ons del Toolbox Deep Learning del Matlab, y se visualizaron los resultados del proceso de aprendizaje.
- Quinto Paso: se realizó la prueba al seleccionar las imágenes utilizadas, y posteriormente se validaron las respuestas por medio de cantidades de porcentajes.

C. *Tercera etapa: características de las capas*

En esta etapa se muestran las principales características que tienen las capas de la red neuronal convolucional entrenada.

- Capa de Entrada. Permite definir el tamaño de entrada del grupo de imágenes que fueron parte del entrenamiento de la red neuronal convolucional. Asimismo, para el ingreso de tal información se tuvo en cuenta el tamaño correspondiente al alto, ancho y cantidad de canales de color. Además, en esta capa se realizó una normalización con la finalidad de transformar los valores de los pixeles a un rango comprendido de [0, 1].
- Capa de Convolución. Consiste en una operación lineal que tiene como característica principal la extracción de información de los datos, tal como se realiza en una operación de filtrado espacial [13], [14]. Para ello se utilizaron filtros o kernel que generaron nuevas imágenes llamadas "feature maps", las cuales contienen características importantes de cada imagen evaluada. Ver la figura 4. Cabe señalar que, si la capa de convolución tiene como cantidad "k" filtros, la misma

cantidad “k” se tendrá como feature maps (filtro que mapea la imagen para encontrar las características resaltantes que aprende la red neuronal).

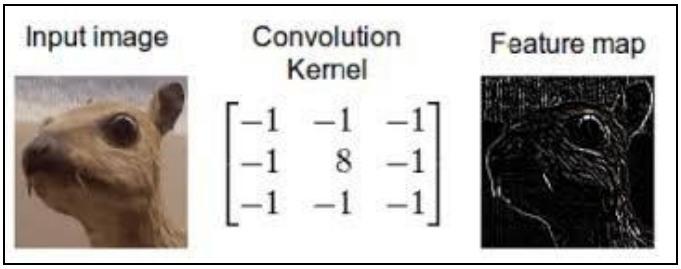

Fig. 4. Ejemplo de aplicación de un filtro o kernel para generar una imagen Feature Map [15].

- Capa ReLu. Consiste en una función de activación que permite a la red CNN, mantener los valores positivos de los resultados obtenidos de la capa precedente. Asimismo, esta función de activación permitió detectar un rango de patrones para predecir con mejor precisión las imágenes que la red CNN iba a entrenar.
- Capa Pooling. Corresponde al muestreo de la información ingresada, al reducir el número de parámetros que ingresan en las siguientes capas. Cabe mencionar que las principales características obtenidas en las capas precedentes, que detectaron los filtros, se mantuvieron.
- Capa Dropout. Toma acción durante la etapa de entrenamiento, estableciendo aleatoriamente los elementos de entrada en cero a partir de la máscara en (1),

$$\text{rand}(\text{size}(X)) < \text{Probabilit y}. \quad (1)$$

Donde X es la capa de entrada, “size” la indicación del tamaño de la capa, y “rand” la indicación de aleatorio. Asimismo, “Probability” señala un factor de probabilidad establecido. Luego, se escalan los elementos restantes en 1/(1-Probability). Esta operación cambia efectivamente la arquitectura de red subyacente entre iteraciones, y ayuda a evitar que la red se sobreajuste.

- Capa Fully Connected. Aquella donde sus neuronas se conectan con las neuronas de capas anteriores. En esta capa se combinan las características que se aprendieron en capas precedentes, particularmente, permiten la clasificación de las imágenes.
- Capas de Salida: Softmax y Classification. La capa Softmax es otra función de activación que generalmente se utiliza a la salida de la red neuronal, y consiste en asignar una probabilidad a cada clase ingresada y que se va a clasificar. La suma de estas probabilidades no debe ser mayor a 1. Por otro lado, a la salida de la red neuronal, se tiene la clasificación de cada uno de los datos que se ingresaron para ser evaluados por la CNN.

D. Cuarta etapa: entrenamiento de la red.

El uso de la librería Deep Learning en Matlab dio la posibilidad y facilidad para el desarrollo de este artículo. Pues, las herramientas de aprendizaje profundo y convolución resultaron adecuados para el reconocimiento facial de estudiantes de la Facultad de Ingeniería de la URP. A continuación, se describen los pasos ejecutados para esta etapa.

- Primero: se definió la ruta donde se encuentran las carpetas que contienen las fotografías de los estudiantes. Adicionalmente, se utilizó una rutina de programación para capturar fotografías de los estudiantes nuevos; luego de ello, se unió al grupo de imágenes (Data Set) en una variable de tipo objeto. Posterior a la agrupación, se ingresó una variable que representó al tamaño de la imagen de entrada para el entrenamiento. Y como también, se utilizó la función “splitEachLevel” del Matlab sobre el Data Set, con el fin de dividirla. Una parte para el entrenamiento y otra para la validación. Seguidamente, se muestra el código de programación empleado.

```
inputSize = [128 128 3];
[imgTrain,imgTest] = splitEachLabel(imds,0.8, 'randomized');
augimgTrain = augmentedImageDatastore(inputSize,imgTrain);
```

- Segundo: se ingresaron las capas de la red neuronal utilizando el comando “imageInputLayer” del Toolbox Deep Learning del Matlab. Luego, se determinó el tamaño de la imagen a entrenar y su respectivo número de canales. A continuación, se muestra el código utilizado.

```
inp = imageInputLayer([128 128 3]);
```

Inmediatamente, se procedió con la declaración de la capa de convolución la cual extrae las principales características. A continuación, se muestra el código utilizado.

```
Conv1= convolution2dLayer(num_categories, …
32, 'Stride', 2, 'Padding', 2, 'BiasLearnRateFactor', 1);
```

Donde, la variable “num_categories” representa la cantidad de clases que la red CNN clasifica. El número de filtros está dado por el valor 32, el parámetro “stride” representa el paso con el cual el filtro se desplaza vertical y horizontalmente sobre la imagen para aprender sus características; luego, el parámetro “padding” aplica a los bordes de la imagen de entrada un vector de relleno de valor 0, y “BiasLearnRateFactor=1” es el factor con el cual el sesgo o bias se multiplica por el ratio global del aprendizaje.

Posterior a la convolución, se ingresaron las capas de función de activación de tipo no lineal y denominada ReLu. Básicamente lo que realiza esta capa es convertir aquellos valores de entrada menores a cero en “0”. A continuación, se presenta el comando utilizado.

```
Relu1 = reluLayer();
```

A continuación, se ingresó la capa Pooling para reducir la información obtenida de las capas anteriores, pero sin perder las características más resaltantes de la imagen inicial entrenada. Para este artículo, se optó por utilizar Max Pooling, que retorna el mayor valor durante el proceso de recorrido del filtro sobre el mapa de características o feature map. El tamaño del filtro pooling usado fue 2x2, con un desplazamiento

"stride" igual a 2. A continuación, se presenta el código utilizado.

```
maxPool = maxPooling2dLayer(2,'Stride',2);
```

Seguidamente, se ingresó la capa Dropout para "eliminar" aleatoriamente los elementos de entrada. Esto permitió mejorar el sobre ajuste que la red neuronal estaba experimentando. El valor elegido fue 0.4 que representa el 40% de elementos retenidos a la salida. A continuación, se presenta el código utilizado.

```
drop = dropoutLayer(0.4);
```

Consecutivamente, se ingresó la capa Fully Connected para conectar las neuronas de las capas previas. De esta manera, se combinan las características principales aprendidas durante el entrenamiento para la identificación de patrones. El tamaño de salida de esta capa debe ser del mismo número de clases de la data de entrenamiento. En este artículo el número fue dado por la variable "num_categories". A continuación, se presenta el código utilizado.

```
fc= fullyConnectedLayer(num_categories, …
'BiasLearnRateFactor',2);
```

Finalmente, para la clasificación, se utilizó la capa Softmax que hace uso de otra función de activación para normalizar la salida de la capa Fully Connected. Y, a su vez, tales salidas, permitieron a la capa Classification seleccionar la clase a la cual la imagen entrenada pertenece. A continuación, se presenta el código utilizado.

```
sm = softmaxLayer();
class = classificationLayer();
```

El resumen de la declaración de las capas, anteriormente mencionadas, se muestra en la figura 5 como una sola variable denominada "layers".

```
layers = [
    imageInputLayer([128 128 3],"Name","imageinput")
    convolution2dLayer([4 4],16,"Name","conv_1","Padding", ...
    [2 2 2 2],"Stride",[2 2])
    reluLayer("Name","relu_1")
    maxPooling2dLayer([2 2],"Name","maxpool_1","Stride",[2 2])
    convolution2dLayer([4 4],32,"Name","conv_2","Stride",[2 2])
    reluLayer("Name","relu_2")
    maxPooling2dLayer([2 2],"Name","maxpool_2","Stride",[2 2])
    dropoutLayer(0.4,"Name","dropout")
    fullyConnectedLayer(5,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
```

Fig. 5. Capas ingresadas para la etapa de entrenamiento.

- Tercero: se procedió a establecer las opciones del entrenamiento. Para ello, el Matlab cuenta con la función "trainingOptions" para especificar los valores con los cuales la red neuronal es entrenada. En ella, el término Descenso de Gradiente Estocástico (SGD) se utiliza de forma iterativa para ajustar aleatoriamente los parámetros obtenidos del entrenamiento, hasta tener un mínimo error. El valor de la tasa de aprendizaje (initial learn rate) utilizada fue de 0.0001; pues, si la tasa de aprendizaje es demasiado baja, el entrenamiento lleva mucho tiempo. Y, si la tasa de aprendizaje es demasiado alta, se podría alcanzar un resultado sub-óptimo. A continuación, se presenta el código utilizado.

```
trainingOptions('sgdm','InitialLearnRate', 0.0001, ...
```

Adicionalmente, se utilizó la característica L2 Regularization para ayudar a mejorar el "overfitting" (sobre entrenamiento de la red) causado por el nuevo conjunto de datos ingresados durante el entrenamiento. Entonces, con el valor de 0.003 como tasa de regularización, se adecuó mejor la obtención de los resultados para el entrenamiento, ya que la pérdida de prueba fue más baja. Cabe precisar también que, si la precisión del entrenamiento no supera la precisión de prueba, significa que el modelo está aprendiendo detalles y ruidos de los datos de entrenamiento. A continuación, se presenta el código utilizado.

```
'L2Regularization',0.003, ...
```

Por otro lado, el parámetro "maxepoch" fue igual a 20, y correspondió a la cantidad de pasadas que dio el algoritmo de entrenamiento sobre el conjunto completo de imágenes entrenadas. Y, el tamaño que tuvo el "miniBatch" utilizado en la red neuronal fue igual a 64. Tener presente que el MiniBatch, es un grupo más reducido de imágenes que es usado para evaluar la pérdida del gradiente SGD lo cual permite actualizar los pesos de cada neurona. A continuación, se presenta el código utilizado.

```
'MaxEpochs',20,'MiniBatchSize',64, ...
```

Luego, el parámetro "shuffle" otorga a la red una mezcla aleatoria de la data entrenada cada vez que ejecuta una pasada completa; el valor recomendado para ello es "every-epoch". Sin embargo, para este artículo no fue necesario utilizarlo, porque inicialmente se combinaron las imágenes. A continuación, se presenta el código utilizado.

```
'Shuffle','every-epoch', ...
```

Asimismo, durante el entrenamiento se mide la precisión y la pérdida de los datos entrenados (variable augimgTrain); por ello, el comando "validationdata" permitió medir ese proceso cada "x" iteraciones; es decir cada vez que se realizaba la iteración completa. A continuación, se presenta el código utilizado.

```
'ValidationFrequency',1, ...
```

En último lugar, el parámetro "plots" muestra el progreso del entrenamiento en una gráfica que cuenta con información de la pérdida de error, y la precisión con que la red va entrenando. También facilita en la línea de comando la misma información de manera textual haciendo uso del comando "verbose" igual a "true". A continuación, se presenta el código utilizado.

```
'Plots','training-progress','Verbose', true);
```

- Cuarto: se procedió con la ejecución del entrenamiento de la red neuronal. El algoritmo utilizado para el entrenamiento

fue dado por la herramienta Add Ons de Deep Learning [16]; a continuación, se presenta el código utilizado.

```
[net,info] = trainNetwork(augimgTrain,layers,opts);
```

Y, tal como se observa, cada parámetro tiene un significado. En primer lugar, "augimgTrain" contiene el Data Store pre procesado. En segundo lugar, "layers" indica las capas que tendrá la red neuronal, y en tercer lugar "opts" contiene todas las opciones para el desarrollo de la red neuronal. A continuación, la figura 6, muestra la agrupación de las capas Convolución, Pooling y ReLu en un rectángulo, mientras que la figura 7 muestra la continuación de la figura anterior, pero agrupando las capas Dropout (flatten), Fully Connected, Softmax y Classification, en otro rectángulo.

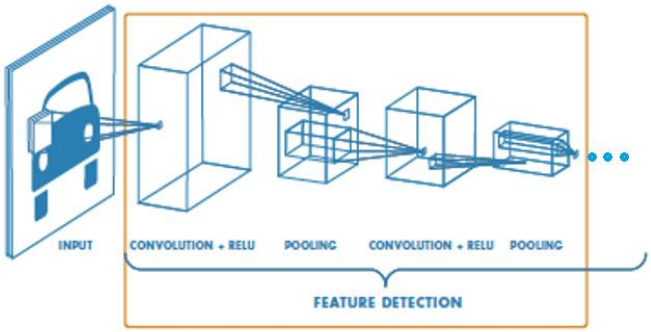

Fig. 6. Capas intermedias de la CNN [17].

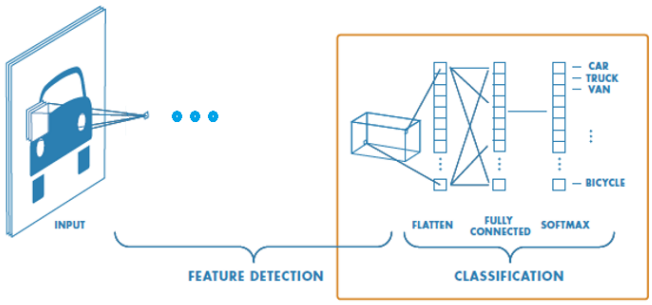

Fig. 7. Capas finales de la CNN [17].

E. Quinta etapa: visualización gráfica de resultados.

Una vez realizada la etapa de entrenamiento, fue necesario conocer el resultado de las iteraciones. De esta manera, tal como se observa en la figura 8, la red neuronal manifestó un total de 20 épocas, un tiempo de entrenamiento de aproximadamente 6 minutos y una precisión de entrenamiento del 100%.

```
Training on single GPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:28 |       8.24% |      12.6488 |          0.0010 |
|      20 |          20 |       00:05:42 |     100.00% |  -0.0000e+00 |    8.0000e-06  |
|=========================================================================================|
```

Fig. 8. Resultado del proceso de entrenamiento.

Asimismo, es importante validar las imágenes de prueba con la finalidad de corroborar la precisión de entrenamiento de la red neuronal. Por ejemplo, el resultado del análisis de una imagen representada por el número 11 identifica correctamente a la persona, y por lo tanto le asigna la precisión del 100%. De esta manera, a partir del Toolbox Deep Learning fue posible invocar el proceso de entrenamiento de forma gráfica y evolutiva, a través de 2 representaciones: Exactitud y Pérdida. Ver la figura 9.

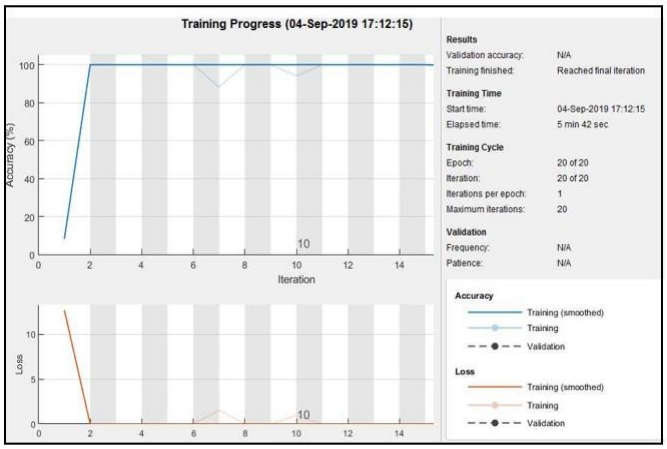

Fig. 9. Representación gráfica del comportamiento durante el entrenamiento de la Red Neuronal.

A continuación, en la Tabla I, se presenta el listado de participantes más el número de imágenes capturadas que fueron utilizadas en las etapas de entrenamiento y validación.

TABLA 1
LISTADO DE ESTUDIANTES CON EL NÚMERO DE FOTOGRAFÍAS CAPTURADAS PARA LA ETAPA DE ENTRENAMIENTO Y VALIDACIÓN.

	ESTUDIANTES	GRUPO DE IMÁGENES	
		ENTRENAMIENTO	VALIDACIÓN
1	Andres	29	5
2	Carlos	30	6
3	Cristian	36	7
4	Enrique	30	6
5	Guillermo	40	8
6	Gustavo	39	7
7	Juan	19	3
8	Julia	32	6
9	Luis	35	7
10	Luz	29	5
11	Marco	28	5
12	María	26	5
13	Miguel	27	5
14	Patrick	26	5

TOTAL	**426**	**80**

IV. Pruebas y Resultados

Las pruebas fueron realizadas con un universo de 14 estudiantes matriculados en una asignatura que hace uso del Laboratorio de Control de la URP. Para ello, se prepararon carpetas diferentes con correspondencia a cada uno de los estudiantes. Luego, se separaron a través del comando "splitEachLabel" del Matlab, donde el 80% de las imágenes fueron seleccionadas para el entrenamiento y el 20% para la validación de la red neuronal entrenada. Esto quiere decir que el 20%, no entraron al entrenamiento y fueron utilizadas de manera aleatoria para validar la red neuronal, tal como se observó en la tabla 1.

Por lo tanto, para evaluar los resultados obtenidos, se tuvo en cuenta tres criterios de evaluación. El primero fue el porcentaje de precisión (PR%) que se muestra en (2), y mide la exactitud del método empleado dividiendo todos los casos clasificados correctamente (Casos_ClasCorr), entre el número total de casos evaluados (Casos_Ev). De esta manera, es posible conocer el porcentaje de acierto.

$$PR\% = \left(\frac{Casos\_ClasCorr}{Casos\_Ev} \right) * 100 \quad (2)$$

El segundo criterio fue el porcentaje de falsos positivos (FP%), el cual se define como el conjunto de imágenes clasificadas de estudiantes que no correspondían. Esto es mostrado en (3) y se obtiene del cociente del número de casos clasificados erróneamente, entre los casos clasificados como correctos por parte de la red CNN.

$$FP\% = \left(\frac{Casos\_ClasErr}{Casos\_Clas\_Corr\_CNN} \right) * 100 \quad (3)$$

Y, el tercer criterio fue el porcentaje de falsos negativos (FN%), el cual se define como el conjunto de imágenes que por el contrario sí correspondían a los estudiantes, pero fueron clasificados por otros. Esto es mostrado en (4) y se obtiene del cociente del número de casos clasificados como erróneos por parte de la red CNN, entre los casos clasificados correctamente.

$$FN\% = \left(\frac{Casos\_Clas\_Err\_CNN}{Casos\_ClasCorr} \right) * 100 \quad (4)$$

Asimismo, se optó por realizar varias pruebas de entrenamiento con diferentes características para la red neuronal. En la tabla 2 se muestra cuatro de ellas. Además, una de las pruebas, utilizó una capa Dropout que permitió mejorar el overfitting.

Luego, al utilizar los tres criterios señalados en las ecuaciones 2, 3 y 4, se obtuvieron los resultados visualizados en la tabla 3. Asimismo, en la figura 10, se muestran las imágenes que no fueron utilizadas en el entrenamiento, y por lo tanto se emplearon para la etapa de la validación de la red neuronal.

Tabla 2
Principales características para cada una de las pruebas realizadas.

	CARACTERÍSTICAS	NÚMERO DE PRUEBAS			
		1ra	2da	3ra	4ta
1	Número de Capas	2	3	2	2
2	Tamaño de Filtros	16, 32	16, 32, 64	16, 32	16, 32
3	Tasa de Aprendizaje	0.001	0.001	0.001	0.0001
4	Número de Épocas	10	20	15	10
5	Porcentaje de Precisión	62%	38%	45%	94%

Tabla 3
Resultados de las pruebas con cada uno de los criterios.

NÚMERO DE PRUEBAS	Precisión (PR%)	Falsos Positivos (FP%)	Falsos Negativos (FN%)
1	62	87	13
2	38	74	26
3	45	65	35
4	94	0	0

Fig. 10. Grupo de imágenes no entrenadas.

V. Discusiones

Para una mejor efectividad en el proceso de reconocimiento, se recomienda contar con una mayor cantidad de fotografías de estudiantes en diferentes situaciones. Esto permitirá que la red neuronal identifique con mayor precisión los patrones que forman las siluetas y características principales. Asimismo, para un caso posterior, se recomienda utilizar otras funciones de activación, incrementar las capas de convolución por encima de cinco, y cambiar el tamaño de los filtros. Y, por el contrario, se recomienda probar el algoritmo en un hardware computacional acondicionado con tarjeta de

video Nvidia para aprovechar su compatibilidad de procesamiento con el software Matlab, y de esa manera ahorrar el tiempo de entrenamiento.

VI. Conclusiones

Para realizar las etapas de entrenamiento y aplicación de la red neuronal convolucional de aprendizaje profundo, se empleó el software Matlab y su Toolbox Deep Learning. Asimismo, se realizaron dos procedimientos para la captura de imágenes de los rostros de los estudiantes. El primero a partir de una galería de fotografías previamente almacenadas en un Data Set, y el segundo a partir de una captura en tiempo real utilizando una WebCam. Asimismo, se validó en tiempo real el funcionamiento de la red neuronal de aprendizaje profundo, con la finalidad de lograr un reconocimiento facial de un grupo de estudiantes. No obstante, el uso de una computadora personal no fue el más adecuado, porque no se aprovechó todo el potencial que el Matlab puede entregar cuando se utiliza hardware computacional con procesador gráfico.

Por otro lado, el Data Set formado por los rostros de los estudiantes de Ingeniería Mecatrónica, fue utilizado para el entrenamiento y la validación de la red neuronal convolucional utilizada en este artículo. Sin embargo, se debe precisar que, al no contar con fotografías con fondos de diferentes lugares o contextos, la red neuronal únicamente aprendió lo que "se le mostraba", por ello que en ocasiones durante la prueba se generaron los falsos positivos. Por lo tanto, se concluye que las redes neuronales de aprendizaje profundo necesitan de grandes volúmenes de información para un correcto entrenamiento de las imágenes.

Agradecimientos

Al grupo de estudiantes del curso Taller de Ingeniería Mecatrónica Básica, en el semestre 2019-2, de la Carrera de Ingeniería Mecatrónica de la Universidad Ricardo Palma, quienes brindaron las facilidades para la realización de las diferentes capturas de las imágenes de sus rostros. Asimismo, a la profesora de dicho curso la Dra. Margarita Murillo Manrique, por cedernos el permiso de ingresar a su clase para llevar a cabo la tarea de captura de fotografías.

Referencias

[1] M. Selvapriya and J. KomalaLakshmi, "Face Recognition Using Image Processing Techniques: A Survey," *International Journal of Engineering and Computer Science*, vol. 3-12, pp.9704-9711, june 2014.

[2] María Osnaya B., "Estudio y análisis del sistema de reconocimiento facial aplicando redes neuronales y el software Matlab", Tesis de Pregrado, Instituto Politécnico Nacional. M, México D.F., 2016.

[3] Xavier Serra, "Face recognition using Deep Learning", Thesis of Master, Universidad Politécnica de Cataluña, Barcelona, 2017.

[4] Taigman, Yaniv et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". *In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '14*. Washington, DC, USA: IEEE Computer Society, pp. 1701–1708.

[5] Vilardi, A. L. Very Deep Convolutional Neural Networks, Tesis de Posgrado, Escuela Técnica de Ingeniería de Telecomunicaciones, Universidad Politécnica de Cataluña, Barcelona, 2016.

[6] O. Picazo, Redes neuronales convolucionales profundas para el reconocimiento de emociones en imágenes, Tesis de Posgrado, Escuela Técnica Superior de Ingenieros Informáticos, Madrid, 2018.

[7] S. Yi, "DeepID3: Face Recognition with Very Deep Neural Networks". *In Hong Kong: The Chinese University of Hong Kong*. 2019.

[8] G. García. (2018, december 16). Naps Tecnología y Educación. Fundamentos del reconocimiento facial [Online]. Available: https://naps.com.mx/blog/fundamentos-del-reconocimiento-facial/

[9] M. Hudson, M. Hagan and H. Demuth. *Neural Network Toolbox™ User's Guide*. Natick: The MathWorks Inc., 2018.

[10] Hubel, H. D. and Wiesel, T. N. "Receptive Fields of Single neurones in the Cat's Striate Cortex." *Journal of Physiology*. Vol 148, pp. 574-591, 1959.

[11] D. Nelson. (2019, december 28). What are Convolutional Neural Networks? [Online]. Available: https://www.unite.ai/what-are-convolutional-neural-networks/

[12] Mathworks. (2020). Deep Learning Toolbox. Create, analyze, and train deep learning networks [Online]. Available: https://www.mathworks.com/products/deep-learning.html

[13] O. Marques e H. Vieira. *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999.

[14] E. Alegre, G. Pajares y A. de la Escalera. *Conceptos y Métodos en Visión por Computador*. España: Grupo de Visión del Comité Español de Automática, 2016.

[15] Tim Dettmers. (2015, march 26). Understanding Convolution in Deep Learning. [Online]. Available: https://timdettmers.com/2015/03/26/convolution-deep-learning/

[16] Adaptive Vision. Deep Learning Add-on [Online]. Available: https://www.adaptive-vision.com/en/software/deep-learning/

[17] S. Saha. (2018, december 15). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).