UNIVERSIDAD RICARDO PALMA
FACULTAD DE INGENIERÍA
PROGRAMA DE TITULACIÓN POR TESIS
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

REDES NEURONALES DE APRENDIZAJE PROFUNDO PARA EL RECONOCIMIENTO FACIAL Y CONTROL DE ACCESO DE ESTUDIANTES A UN LABORATORIO

TESIS
PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:

Bach. CAYLLAHUA AQUINO, NESTOR ASBEL
Bach. SUÁREZ MACEDO, JUAN CARLOS

ASESOR: Dr. Ing. HUAMANÍ NAVARRETE, PEDRO

LIMA - PERÚ
2019

DEDICATORIA

Dedico esta investigación a mis padres, a mi hermano y a mis abuelos, a mis compañeros de aula con quienes compartí muchos momentos durante la etapa universitaria y finalmente a todas aquellas personas que me empujaron a culminar y cerrar esta etapa de mi vida, con esta investigación.

Juan Carlos Suárez

Dedico esta tesis a mi madre por su amor, sacrificio y fe que siempre ha tenido en mí y que me ha permitido llegar hasta aquí. También a todas aquellas personas que han hecho posible cumplir con este anhelo. Esto simboliza una muestra de mi gratitud y el derecho a un merecido reconocimiento.

Nestor Cayllahua

RECONOCIMIENTO

Nuestro aprecio y gratitud al Dr. Pedro Huamaní por brindarnos sus conocimientos, su paciencia y el tiempo que dedicó a orientarnos en el proceso de investigación de la presente tesis.

Nestor C.
Juan Carlos S.

ÍNDICE GENERAL

RESUMEN……ix
ABSTRACT……..x
INTRODUCCION……1
CAPÍTULO I – Planteamiento del Problema 3
1.1 Formulación y Delimitación del Problema 3
1.1.1 Problema General 3
1.1.2 Problemas Específicos 3
1.2 Importancia y Justificación del Estudio 3
1.3 Limitaciones del Estudio 4
1.4 Objetivos 4
1.4.1 Objetivo General 4
1.4.2 Objetivos Específicos 4
1.5 Estado del Arte 5
Instituto Politécnico Nacional Zacatenco 5
Universidad de Cataluña 5
Universidad de Sevilla 6
Universidad Politécnica de Madrid 6
CAPÍTULO II - Marco Teórico 7
2.1 Marco Histórico 7
2.2 Fundamentos Básicos de Redes Neuronales 7
2.2.1 Tipos de Redes Neuronales 8
2.2.2 Deep Learning o Aprendizaje Profundo 9
2.2.3 Ajuste excesivo y ajuste insuficiente 9
2.2.4 Regularización para Aprendizaje Profundo 11
2.2.5 Parámetro de Regularización L2 13
2.3 Fundamentos Básicos de Reconocimiento Facial 13
2.3.1 Operadores de Puntos 14
2.3.2 Filtrado Lineal 15
2.4 Cómo aprende una Red Neuronal Profunda Convolucional 16
CAPÍTULO III – RED NEURONAL PARA EL RECONOCIMIENTO FACIAL 19
3.1 Matlab y toolbox Deep Learning 19
3.1.1 Matlab 19
3.1.2 Toolbox Deep Learning 20
3.2 Procedimiento para la Captura de Imágenes 20
3.2.1 A través de una galería de imágenes ya disponible 20
3.2.2 A través de un dispositivo que capture imágenes 21
3.3 Procedimiento para el entrenamiento de la red neuronal 22
3.3.1 Paso 1 Re-muestreo de Imágenes 22
3.3.2 Paso 2 Capas 22

3.3.3 Paso 3 Opciones 23
3.3.4 Paso 4 Entrenamiento 23
3.3.5 Paso 5 Prueba 23
3.4 Características de las Capas de la Red Neuronal 23
3.4.1 Capa de ingreso de imagen 24
3.4.2 Capa de convolución 24
3.4.3 Capa ReLU (función de activación) 24
3.4.4 Capa de pooling 25
3.4.5 Capa de dropout 26
3.4.6 Capa fully connected 26
3.4.7 Capa de salidas: softmax y clasificación 26
3.5 Etapas de entrenamiento 26
3.5.1 Definición de data 27
3.5.2 Ingreso de capas de la red neuronal 27
3.5.3 Algoritmo para especificar las opciones de entrenamiento 30
3.5.4 Ejecución para entrenar la red neuronal 33
3.6 Visualización gráfica de salida 34
CAPÍTULO IV – PRUEBAS Y RESULTADOS 38
4.1 Criterios de Evaluación 38
4.2 Resultados Experimentales 39
4.2.1 Prueba 1: 40
4.2.2 Prueba 2: 41
4.2.3 Prueba 3: 42
4.2.4 Prueba 4: 43
CONCLUSIONES 46
RECOMENDACIONES 47
REFERENCIAS BIBLIOGRÁFICAS 48
ANEXOS 50

ÍNDICE DE FIGURAS

Figura 1. Modelo Simplificado de una Neurona Biológica 8
Figura 2. Modelo de una Neurona Artificial 8
Figura 3. Tipos de redes Neuronales 9
Figura 4. Ajuste excesivo y ajuste insuficiente 11
Figura 5. Reconocimiento de rostros de baja resolución de personas conocidas 14
Figura 6. Filtrado de vecindad (convolución) 16
Figura 7. Red Neuronal Convolucional 17
Figura 8. Diagrama de una red de aprendizaje profundo para reconocimiento 18
Figura 9. Extracción de características durante el proceso de aprendizaje 18
Figura 10. Diagrama de bloques general de la tesis 19
Figura 11. Declaración del datastore en Matlab 21
Figura 12. Carpetas utilizadas para formar el Data store 21
Figura 13. Algoritmo para captura de imágenes con dispositivo 22
Figura 14. Capa de convolución y feature maps, antes de ingresar a la capa ReLu 24
Figura 15. Representación gráfica de la función de activación ReLu 25
Figura 16. Filtro o kernel 25
Figura 17. Capas ingresadas para el entrenamiento 29
Figura 18. Diagrama de bloques de la arquitectura de la red neuronal 30
Figura 19. Capas intermedias 34
Figura 20. Capas finales 35
Figura 21. Resultado de las iteraciones durante el entrenamiento 35
Figura 22. Visualización del resultado de la imagen 11 evaluada 36
Figura 23. Gráfico del comportamiento durante el entrenamiento de la red neuronal 36
Figura 24. Variables creadas al procesar el algoritmo de la red neuronal 37
Figura 25. Comportamiento de la prueba 1 40
Figura 26. Comportamiento de la prueba 2 41
Figura 27. Comportamiento de la prueba 3 42
Figura 28. Comportamiento de la prueba 3 43
Figura 29. Grupo de imágenes no entrenadas 43
Figura 30. Representación del resultado de la evaluación 44

ÍNDICE DE TABLAS

Tabla 1. Cantidad de Imágenes tomadas y validadas por alumno 37
Tabla 2*:* Opciones utilizadas para las pruebas. 45
Tabla 3. Resultados de las pruebas en diferentes escenarios. 45
Tabla 4*:* Resultados obtenidos al usar la versión elegida de red neuronal. 45

ÍNDICE DE ECUACIONES

Ecuación 1. Representación matemática de la convolución 16
Ecuación 2. Expresión matemática SGD 31
Ecuación 3. Fórmula precisión 38
Ecuación 4. Fórmula falsos positivos 38
Ecuación 5. Fórmula falsos negativos 39

RESUMEN

Las redes neuronales convolucionales (CNN siglas en inglés) de aprendizaje profundo han demostrado en los últimos años una gran capacidad para resolver problemas de clasificación de imágenes, gracias al aprendizaje que realizan con millones de datos que se pueden encontrar en diferentes motores de almacenamiento en el internet. Asimismo, el reconocimiento facial representa un importante campo de aplicación desde hace algunos años y actualmente, por su diversificación en aplicaciones que van desde la mejora de la seguridad, programas digitales de telefonía celular y aplicaciones industriales. Por lo cual, este trabajo tuvo como objetivo la implementación de una red neuronal convolucional de aprendizaje profundo, para el reconocimiento facial y el control de acceso de estudiantes de la carrera de Ingeniería Mecatrónica; para esto, la metodología empleada consistió en el entrenamiento de la red neuronal, con el propósito de extraer los datos relevantes de los rasgos faciales en las fotografías tomadas al grupo de estudiantes, y cuando se probó con un nuevo grupo de fotos de las mismas personas, se consiguió el reconocimiento de las personas. La muestra utilizada fue de 426 fotografías correspondiente a 14 alumnos que utilizaron el Laboratorio de Control en el semestre 2019-I; igualmente, el software empleado para el entrenamiento de la red fue el MATLAB y su Toolbox Deep Learning. Así como también, se realizaron diferentes pruebas para la selección de la red, iniciando con una capa de convolución, luego dos, y finalmente tres capas, las cuales otorgaron los siguientes porcentajes de precisión 15.63%, 94.00% y 67.13% , respectivamente. De esta manera, se optó por elegir la red neuronal con dos capas de convolución, de 16 y 32 filtros, para realizar el reconocimiento facial.

Palabras claves: Aprendizaje profundo, reconocimiento facial, Matlab, toolbox Deep Learning, convolución, capas.

ABSTRACT

The convolutional neural networks (CNN) of deep learning have demonstrated in recent years a great capacity to solve image classification problems, thanks to the learning they do with millions of data that can be found in different storage engines in the Internet. Likewise, facial recognition represents an important field of application for past years and nowadays, for its diversification in applications ranging from improved security, digital cell phone programs and industrial applications. Therefore, this work aimed at an implementation of a convolutional neural network of deep learning, for facial recognition and access control of students of the Mechatronics Engineering; For this, the methodology used consisted of the training of a neural network, with the purpose of extracting the relevant data of the facial features in the photographs taken to a group of students, and when it was tested with a new group of photos of the same people , the recognition of people was achieved. The sample used was 426 photographs corresponding to 14 students who used the Control Laboratory in the 2019-I semester; plus, the software used for network training was the MATLAB with its Deep Learning Toolbox. In addition, different tests were performed for the selection of the network, starting with one convolution layer, then two, and finally three layers, which gave the following accuracy percentages 15.63%, 94.00% and 67.13%, respectively. In this way, we chose to select the neural network with two layers of convolution, 16 and 32 filters, to perform facial recognition.

Keywords: Deep learning, facial recognition, Matlab, Deep Learning toolbox, convolution, layers.

INTRODUCCIÓN

En los últimos años, varios problemas de clasificación en inteligencia artificial han impulsado su rendimiento mediante el uso de técnicas de aprendizaje profundo, en particular las redes neuronales convolucionales (CNN).
El tema de este proyecto se centra en el reconocimiento de rostros y explora el estado de arte del aprendizaje profundo.

La presente investigación se desarrolló para aplicar una Red Neuronal de Aprendizaje Profundo para el reconocimiento facial y el control de acceso de estudiantes al Laboratorio de Control de la Universidad Ricardo Palma, para ello se construyó un algoritmo a través del uso de la librería Deep Learning que se encuentra en el programa de computación MATLAB.

La investigación presenta cuatro capítulos que a continuación explicamos de manera breve:
En el capítulo I se expondrá el problema que encontramos para el control de ingreso al laboratorio de control de la universidad Ricardo Palma; así también, detallaremos los problemas específicos para la elaboración de esta investigación, explicaremos los objetivos que responderán la problemática planteada y finalmente la justificación y estado del arte que usaremos como apoyo al estudio desarrollado.

En el capítulo II se detallará el marco teórico, el cual explicará los fundamentos básicos de redes neuronales y los fundamentos básicos del reconocimiento facial, y finalmente se terminará el contenido de este capítulo con información de cómo el sistema o red neuronal aprende gracias a la técnica de aprendizaje profundo.

En el capítulo III comenzaremos explicando el software MATLAB y su librería Deep Learning, continuaremos explicando el procedimiento utilizado en esta investigación para la captura de imágenes, detallaremos los pasos que seguimos para armar lo que será la arquitectura de nuestra red neuronal de aprendizaje profundo, explicaremos luego de ello las características que tienen cada una de las capas que tendrá la red

neuronal y finalmente se mostrarán las funciones del algoritmo desarrollado para el aprendizaje y la visualización del entrenamiento realizado por la red neuronal.

En el capítulo IV se mostrarán las simulaciones, pruebas y ajuste de parámetros para seleccionar la red neuronal de aprendizaje profundo, que facilite el entrenamiento y testeo para el reconocimiento facial y de control de acceso de los estudiantes del laboratorio.

Para finalizar esta investigación, expondremos las conclusiones y recomendaciones sobre la red neuronal de aprendizaje profundo utilizada, listaremos las referencias bibliográficas que apoyarán el desarrollo de esta investigación y adicionaremos información de apoyo y el algoritmo desarrollado.

CAPÍTULO I – Planteamiento del Problema

Para el acceso al laboratorio de control de la universidad Ricardo Palma se cuenta con un sistema de validación de acceso simple y de registro manual. Esto produciría que personas ajenas al laboratorio de control de la universidad ingresen sin autorización, ocasionando daños y/o pérdida de equipos de alto valor para las carreras de ingeniería Electrónica e ingeniería Mecatrónica.

1.1 Formulación y Delimitación del Problema

1.1.1 Problema General

De lo expuesto anteriormente la siguiente pregunta de investigación es planteada:
¿Cómo aplicar una red neuronal de aprendizaje profundo para el reconocimiento facial y el control de acceso de estudiantes al Laboratorio de Control de la Universidad Ricardo Palma?

1.1.2 Problemas Específicos

a) ¿Es posible entrenar y utilizar una red neuronal convolucional de aprendizaje profundo para el reconocimiento facial de los estudiantes que acceden al Laboratorio de Control de la Universidad Ricardo Palma?
b) ¿Cómo validar en tiempo real el funcionamiento de la red neuronal para el reconocimiento facial de los estudiantes y permitir el control de acceso al Laboratorio de Control?
c) ¿Es posible establecer una base de datos conformada por los rostros de estudiantes de la carrera de Ingeniería Mecatrónica que hacen uso del Laboratorio de Control, para utilizarla en el entrenamiento y validación de la red neuronal convolucional?

1.2 Importancia y Justificación del Estudio

En referencia al proyecto de tesis, se tiene como finalidad aplicar la técnica de redes neuronales de aprendizaje profundo para el reconocimiento facial y el control de acceso de los estudiantes al Laboratorio de Control de la Universidad Ricardo Palma.

También se tiene por finalidad, el establecimiento de manera digital del acceso para la optimización de los recursos administrativos mediante el reconocimiento de rostros de personas.

1.3 Limitaciones del Estudio

En la presente investigación se cuenta con las siguientes limitaciones:

- Contar con un hardware computacional con tarjeta de video Nvidia para la aceleracion del entrenamiento de la red neuronal.
- De acuerdo con investigaciones previas, el uso de una gran cantidad de imágenes ayuda a mejorar la predicción durante el entrenamiento de la red neuronal
- Las fotografias tomadas fueron 30 en promedio para cada uno de los estudiantes, y delimitadas unicamente a un solo ambiente con un aforo total de 15 alumnos. Se considera una limitante dado que todas las imágenes corresponden a un mismo contexto, reduciendo el aprendizaje de la red neuronal.

1.4 Objetivos

1.4.1 Objetivo General

Implementar una red neuronal de aprendizaje profundo para el reconocimiento facial y el control de acceso de estudiantes al Laboratorio de Control de la Universidad Ricardo Palma.

1.4.2 Objetivos Específicos

a) Entrenar y utilizar una red neuronal convolucional de aprendizaje profundo, a través del toolbox Deep Learning del software Matlab, para el reconocimiento facial de los estudiantes que acceden al Laboratorio de Control de la Universidad Ricardo Palma.
b) Desarrollar un algoritmo que permita utlizar un dispositivo de captura de imágenes, para interactuar en tiempo real con una red neuronal entrenada para el reconocimiento facial de los estudiantes y permitir el control de acceso al Laboratorio de Control.
c) Establecer una base de datos conformada por los rostros de estudiantes matriculados en dos asignaturas de la carrera de Ingeniería Mecatrónica que hacen uso del Laboratorio de Control, para utilizarla en el entrenamiento y validación de la red neuronal convolucional.

1.5 Estado del Arte

Instituto Politécnico Nacional Zacatenco

Osnaya M. (2016) En su tesis concluye que: "Para realizar el sistema de reconocimiento facial; se debe tener una base de datos, el número de usuarios, y al tiempo que se requiere para realizar la identificación". (P.93)

Universidad de Cataluña

Serra X. (2017) En su tesis manifiesta que: "La parte de reconocimiento facial se ha probado utilizando dos conjuntos de datos de diferentes tamaños, y obteniendo resultados estables en torno al 90% de precisión, alcanzando un máximo del 95%". (P.65)

Vilardi A. (2016) En su tesis para graduarse de Master en Ingeniería Electrónica, en la descripción de optimización de la red señala que: La capacitación de una CNN muy profunda 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. Si el número de parámetros es grande, se necesita una gran cantidad de imágenes para aprender estos parámetros. 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. Cuando el conjunto de datos no es tan grande, usar un modelo pre-entrenado y luego ajustarlo, puede ser la mejor opción para obtener mejores resultados con un pequeño conjunto de datos. Esto significa construir una nueva red neuronal, tomando los parámetros aprendidos pre-entrenados previamente como inicialización, e inicializar aleatoriamente los parámetros del clasificador, para clasificar la cantidad correcta de etiquetas. También es posible inicializar solo unas pocas capas de red neuronal con los parámetros pre-

entrenados, y usar una inicialización aleatoria para algunos de las últimas capas. (P.28)

Universidad de Sevilla

Jimenez I. (2018) En su tesis explica que: Para realizar experimentos con la red neuronal y evaluar sus resultados, se va a recurrir a la técnica denominada validación cruzada o cross-validation. Esta se trata de un método estadístico dedicado a evaluar y comparar algoritmos y consiste en tomar el conjunto de datos disponible y dividirlo en dos partes, que serán denominadas datos de entrenamiento y datos de validación o de prueba. Como sus propios nombres indican, el primer subconjunto será utilizado para llevar a cabo el entrenamiento de la red neuronal, y el segundo, generalmente de menor tamaño (aproximadamente un 20% del conjunto total de datos) se destinará a realizar pruebas o análisis para comprobar los resultados del entrenamiento. (P.29)

Universidad Politécnica de Madrid

Picazo O. (2018) En su tesis concluye que: Como se ha visto a través de las pruebas realizadas, y siempre teniendo en cuenta que se ha realizado sobre el conjunto de datos de bajo estudio, el desbalanceo en la base de datos de entrenamiento afecta bastante a los resultados obtenidos en redes neuronales convolucionales profundas. El simple balanceo copiando aleatoriamente las imágenes y equiparando el número de imágenes entre las diferentes clases de entrenamiento mejora bastante 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 los resultados. (P.43)

CAPÍTULO II - Marco Teórico

2.1 Marco Histórico

López R. (2019) Explica que el Deep Learning o aprendizaje profundo es un subcampo dentro del Machine Learning, el cuál utiliza distintas estructuras de redes neuronales para lograr el aprendizaje de sucesivas capas de representaciones cada vez más significativas de los datos. El aprendizaje profundo o Deep Learning hace referencia a la cantidad de capas de representaciones que se utilizan en el modelo; en general se suelen utilizar decenas o incluso cientos de capas de representación, las cuales aprenden automáticamente a medida que el modelo es entrenado con los datos. (P.14)

Yi S. (2019) En una investigación, detalla que el estado del arte del reconocimiento facial ha avanzado significativamente con la aparición del aprendizaje profundo. Las redes neuronales de aprendizaje profundo lograron recientemente un gran éxito en el reconocimiento general de objetos debido a su excelente capacidad de estudio. Esto nos motiva a investigar su efectividad en el reconocimiento facial. Este artículo propone dos arquitecturas de redes neuronales muy profundas, 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. Las señales de supervisión de la identificación de la cara conjunta se agregan a las capas de extracción de características intermedias y finales durante el entrenamiento. Un conjunto de las dos arquitecturas propuestas alcanza 99.53% de precisión de verificación de cara LFW y 96.0% de precisión de identificación de cara de LFW rango 1, respectivamente. (P.1)

2.2 Fundamentos Básicos de Redes Neuronales

Haykin S. (1999) Señala que las Redes Neuronales Artificiales (RNA) surgen como una necesidad de contar con máquinas capaces de aprender y recordar, tal como la inteligencia humana. Dicho de otra manera, imitar el

funcionamiento del cerebro humano, ya que este posee un procesamiento altamente paralelo y con capacidad de resolver problemas complejos.

Existen muchas definiciones acerca de las RNA. Por ejemplo, algoritmos computacionales inspirados en la naturaleza: Neuronas Biológicas. O como también, sistemas altamente paralelos no lineales y densamente interconectados y basados en procesadores simples: modelo simplificado de la neurona biológica. (P.23)

A continuación, en las figuras 1 y 2 se muestran los modelos de una neurona biológica y una neurona artificial.

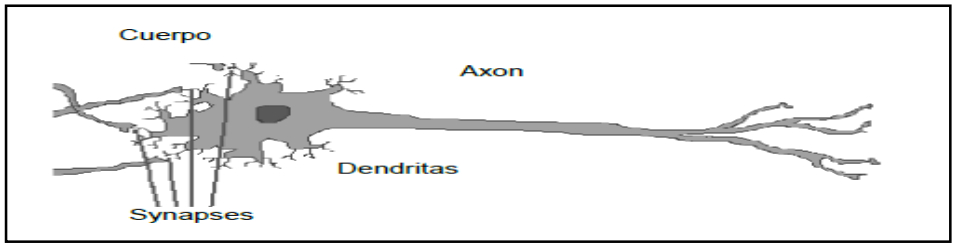


Figura 1. **Modelo Simplificado de una Neurona Biológica**

Fuente: (Huamaní, 2008)

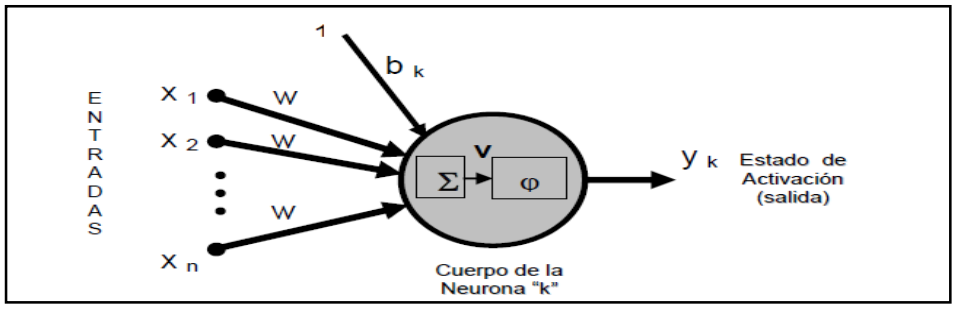


Figura 2. **Modelo de una Neurona Artificial**

Fuente: (Huamaní, 2008)

2.2.1 Tipos de Redes Neuronales

Kim P. (2017) Explica que "se han desarrollado muchos tipos de técnicas de aprendizaje automático para resolver problemas en varios campos. Estas técnicas de aprendizaje automático pueden ser clasificadas en tres tipos según el método de entrenamiento" (ver Figura 3). (P.12)

- Aprendizaje supervisado
- Aprendizaje No supervisado

- Aprendizaje reforzado

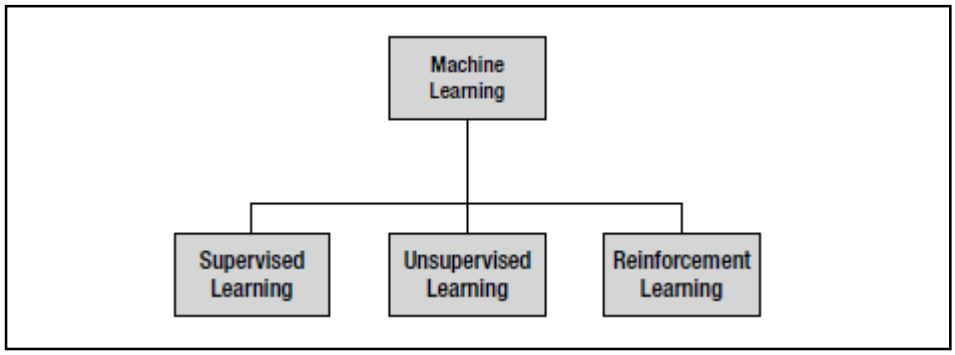


Figura 3. **Tipos de redes Neuronales**

Fuente: (Kim, 2017)

Para el presente trabajo se va a utilizar la técnica de aprendizaje supervisado, dentro del cual se encuentra la técnica de aprendizaje profundo (Deep Learning).

2.2.2 Deep Learning o Aprendizaje Profundo

Kim P. (2017) Define que Deep Learning "es una técnica de Aprendizaje Profundo que emplea una red neuronal profunda. Como saben, la red neuronal profunda es una red neuronal multicapa que contiene dos o más capas ocultas. Aunque esto puede ser decepcionantemente, esta es la verdadera esencia del Deep Learning". (P.103)

2.2.3 Ajuste excesivo y ajuste insuficiente

I. Goodfellow, Y. Bengio, A. Courville (2016) Explican que el desafío central en el aprendizaje automático es que nuestro algoritmo debe funcionar bien en entradas nuevas y nunca antes vistas, no solo en aquellas en las que se entrenó nuestro modelo. La capacidad de desempeñarse bien en entradas no observadas anteriormente se denomina "generalización".

Usualmente, cuando se entrena un modelo de aprendizaje automático, tenemos acceso a un conjunto de entrenamiento, se calcula alguna medida de error en dicho conjunto, llamado "error de entrenamiento"; y se reduce este error de entrenamiento. Hasta ahora, lo que hemos descrito es simplemente un problema de optimización. Lo que separa el aprendizaje automático de la optimización es que queremos que el error de generalización, o también llamado "error de prueba", también sea bajo. El

error de generalización se define como el valor esperado del error en una nueva entrada. Aquí, la expectativa se toma a través de diferentes entradas posibles, extraídas de la distribución de entradas que esperamos que el sistema encuentre en la práctica.

Por lo general, estimamos el error de generalización de un modelo de aprendizaje automático midiendo su rendimiento en un conjunto de ejemplos de prueba que se recopilaron por separado del conjunto de capacitación.

Una conexión inmediata que podemos observar entre el error de entrenamiento y el error de prueba es que el error de entrenamiento esperado de un modelo seleccionado al azar es igual al error de prueba esperado de ese modelo. Supongamos que tenemos una distribución de probabilidad p (x, y) y tomamos muestras de ella repetidamente para generar el conjunto de entrenamiento y el conjunto de pruebas. Para algún valor fijo w, el error esperado del conjunto de entrenamiento es exactamente el mismo que el error esperado del conjunto de prueba, porque ambas expectativas se forman usando el mismo proceso de muestreo de conjunto de datos. La única diferencia entre las dos condiciones es el nombre que asignamos al conjunto de datos que muestreamos.

Por supuesto, cuando usamos un algoritmo de aprendizaje automático, no ajustamos los parámetros con anticipación, sino tomamos muestras de ambos conjuntos de datos. Muestreamos el conjunto de entrenamiento, luego lo usamos para elegir los parámetros para reducir el error del conjunto de entrenamiento, después tomamos muestras del conjunto de prueba. Bajo este proceso, el error de prueba esperado es mayor o igual al valor esperado del error de entrenamiento. Los factores que determinan qué tan bien funcionará un algoritmo de aprendizaje automático son su capacidad para:

- Hacer el error de entrenamiento pequeño
- Hacer que la brecha entre el entrenamiento y el error de prueba sea pequeña

Estos dos factores corresponden a los dos desafíos centrales en el aprendizaje automático: el ajuste excesivo y el ajuste insuficiente. El ajuste insuficiente ocurre cuando el modelo no puede obtener un valor de error suficientemente bajo en el conjunto de entrenamiento. El sobreajuste ocurre cuando la brecha entre el error de entrenamiento y el error de prueba es demasiado grande. (ver figura 4)

Podemos controlar si un modelo es más probable que se sobreajuste o subadapte alterando su capacidad. Informalmente, la capacidad de un modelo es su capacidad para adaptarse a una amplia variedad de funciones. Los modelos con baja capacidad pueden tener dificultades para adaptarse al conjunto de entrenamiento. Los modelos con alta capacidad pueden sobreajustarse memorizando las propiedades del conjunto de entrenamiento que no les sirven bien en el conjunto de prueba. (P.108-110)

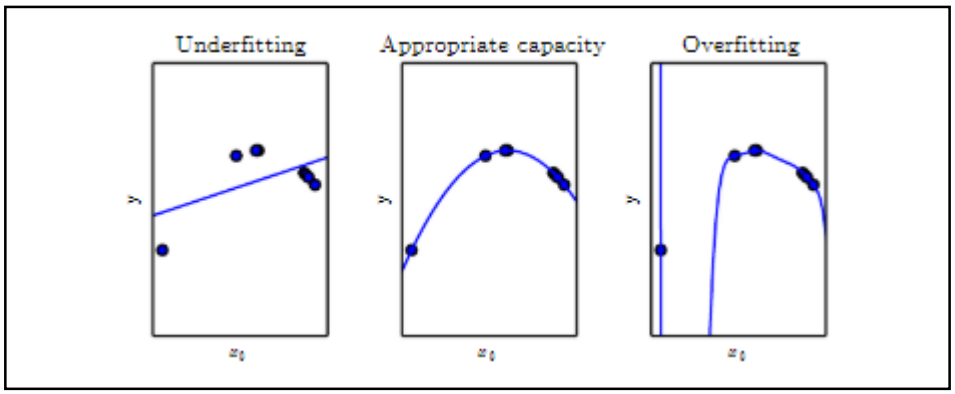


Figura 4. **Ajuste excesivo y ajuste insuficiente**

Fuente: (Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016)

2.2.4 Regularización para Aprendizaje Profundo

I. Goodfellow, Y. Bengio, A. Courville (2016) Indican que un problema central en el aprendizaje automático es cómo hacer un algoritmo que funcione bien no solo en los datos de entrenamiento, sino también en las nuevas entradas. Muchas estrategias utilizadas en el aprendizaje automático están explícitamente diseñadas para reducir el error de prueba, posiblemente a expensas de un mayor error de entrenamiento. Estas estrategias se conocen colectivamente como regularización. Una gran cantidad de formas de

regularización están disponibles para el profesional de aprendizaje profundo.

Definimos la regularización como "cualquier modificación que hagamos al algoritmo de aprendizaje que tiene la intención de reducir su error de generalización pero no su error de entrenamiento". Existen muchas estrategias de regularización. Algunos ponen restricciones adicionales en un modelo de aprendizaje automático, como agregar restricciones en los valores de los parámetros. Algunos agregan términos adicionales en la función objetivo que pueden considerarse como correspondientes a una restricción suave en los valores de los parámetros. Si se elige con cuidado, estas restricciones y penalizaciones adicionales pueden mejorar el rendimiento en el conjunto de pruebas. A veces, estas restricciones y sanciones están diseñadas para codificar tipos específicos de conocimiento previo. Otras veces, estas restricciones y penalizaciones están diseñadas para expresar una preferencia genérica por una clase de modelo más simple para promover la generalización. A veces, las sanciones y restricciones son necesarias para determinar un problema determinado. Otras formas de regularización conocidos como métodos de ensamblaje, combinan múltiples hipótesis que explican los datos de entrenamiento.

En el contexto del aprendizaje profundo, la mayoría de las estrategias de regularización se basan en estimadores de regularización. La regularización de un estimador funciona mediante el intercambio de aumento del sesgo o bias por una varianza reducida. Un regularizador efectivo es aquel que hace un comercio rentable, reduciendo la varianza significativamente sin aumentar demasiado el sesgo.

Controlar la complejidad del modelo no es una simple cuestión de encontrar el modelo del tamaño correcto, con el número correcto de parámetros. En cambio, podríamos encontrar (y de hecho en escenarios prácticos de aprendizaje profundo, casi siempre encontramos) que el mejor modelo de adaptación (en el sentido de minimizar el error de generalización) es un modelo grande que se ha regularizado adecuadamente. (P.116-117)

2.2.5 Parámetro de Regularización L2

La penalización del parámetro L2, comúnmente conocida como pérdida de peso, es una estrategia de regularización que lleva los pesos más cerca del origen al agregar un término de regularización $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ a la función objetivo. En otras comunidades académicas, la regularización L2 también se conoce como regresión del cartucho o regularización Tikhonov.

2.3 Fundamentos Básicos de Reconocimiento Facial

Szeliski R. (2010) Indica que algunos de los enfoques más tempranos para el reconocimiento facial incluyeron la ubicación de características distintivas de la imagen, como los ojos, la nariz y la boca, y la medición de las distancias entre estas ubicaciones (Fischler y Elschlager 1973; Kanade 1977; Yuille 1991). Enfoques más recientes se basan en la comparación de imágenes de nivel de gris proyectadas en subespacios de dimensiones inferiores denominados interfaces propias (eigenfaces) y en el modelado en conjunto de las variaciones de forma y apariencia (a la vez que se descuentan las variaciones de postura) utilizando modelos de apariencia activa.

Se pueden encontrar descripciones adicionales de técnicas de reconocimiento facial en una serie de encuestas y libros sobre este tema (Chellappa, Wilson y Sirohey 1995; Zhao, Chellappa, Phillips et al. 2003; Li y Jain 2005), así como en el sitio Web de Reconocimiento Facial http://www.face-rec.org/.

La encuesta sobre el reconocimiento facial de los humanos por Sinha, Balas, Ostrovsky et al. (2006) también vale la pena leerlo; incluye una serie de resultados sorprendentes, como la capacidad de los humanos para reconocer imágenes de baja resolución de caras conocidas (Figura 5) y la importancia de las cejas en reconocimiento. (P.668-670)

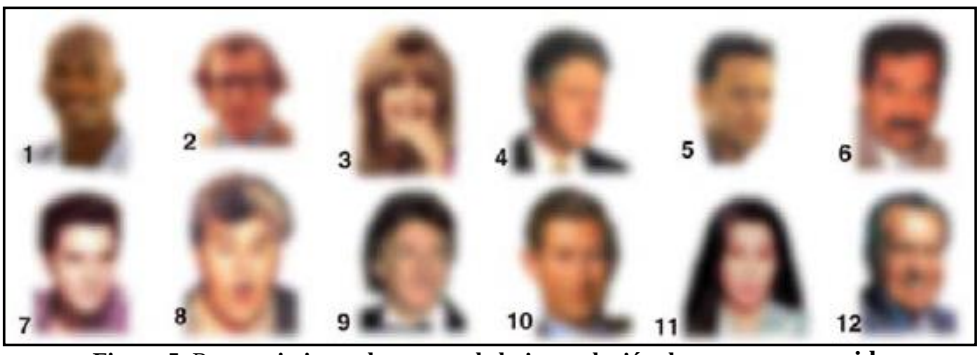


Figura 5. R**econocimiento de rostros de baja resolución de personas conocidas**

Fuente: Sinha, Balas, Ostrovsky et al. (2006)

S. Li, A. Jain (2005) Explican que se espera que un sistema de reconocimiento de rostros identifique rostros presentes en imágenes y videos automáticamente. Puede funcionar en uno o ambos modos: (1) verificación facial (o autenticación), e (2) identificación facial (o reconocimiento). La verificación facial implica una comparación uno a uno que revisa una imagen de cara de consulta con una imagen de cara de plantilla cuya identidad está siendo llamada. La identificación facial implica comparaciones de uno a muchos que verifican una imagen de rostro de consulta con todas las imágenes de la plantilla en la base de datos para determinar la identidad de la cara de consulta. Otro escenario de reconocimiento facial implica una validación de la lista de observación, donde una cara de consulta se compara con una lista de sospechosos (coincidencias de uno a pocos). (P.1)

2.3.1 Operadores de Puntos

Szeliski R. (2010) Señala que los tipos más simples de transformaciones de procesamiento de imágenes son los operadores de puntos, donde para cada salida el valor de píxel depende solo del valor de píxel de entrada correspondiente (y, alguna información o parámetros recopilados globalmente). Ejemplos de tales operadores incluyen brillo y ajustes de contraste, así como corrección de color y transformaciones. En la literatura de procesamiento de imágenes, tales operaciones también se conocen como procesos puntuales.

Si bien los controles de brillo y ganancia descritos anteriormente pueden mejorar la apariencia de una imagen, ¿cómo podemos determinar

automáticamente sus mejores valores? Un enfoque podría ser para mirar los valores de píxel más oscuros y brillantes en una imagen y asignarlos a negro puro y blanco puro. Otro enfoque podría ser encontrar el valor promedio en la imagen, empujarlo hacia el gris medio, y expandir el rango para que llene más de cerca los valores visualizables.

¿Cómo podemos visualizar el conjunto de valores de luminosidad en una imagen para probar algunos de estas heurísticas? La respuesta es trazar el histograma de los canales de color individuales y valores de luminancia. A partir de esta distribución, podemos calcular estadísticas relevantes como los valores de intensidad mínima, máxima y media.

¿No sería mejor si pudiéramos aclarar simultáneamente algunos valores oscuros y oscurecer algunos valores claros, sin dejar de utilizar toda la extensión de los valores disponibles? ¿Puedes pensar en un mapeo que pueda hacer esto?

Una respuesta popular a esta pregunta es realizar la ecualización del histograma. (P.101-106)

2.3.2 Filtrado Lineal

Szeliski R. (2010) Indica que la ecualización de histograma adaptativo local es un ejemplo de un operador de vecindario u operador local, que utiliza una colección de valores de píxel en la vecindad de un píxel dado para determinar su valor de salida final (Figura 6). Además de realizar el ajuste de tono local, los operadores de vecindario se pueden usar para filtrar imágenes para agregar desenfoque suave, agudizar detalles, acentuar bordes o eliminar ruido.

El tipo más común de operador de vecindario es un filtro lineal, en el cual el valor del píxel de salida se determina como una suma ponderada de los valores de píxel de entrada.

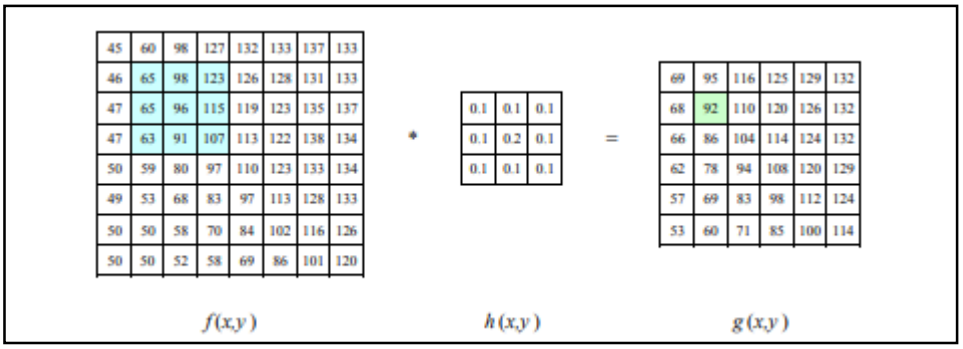


Figura 6. **Filtrado de vecindad (convolución)**

Fuente: Li y Stan (2005)

En la figura 6, la imagen de la izquierda está convolucionada con el filtro en el medio para producir la imagen de la derecha. Los píxeles azules claros indican la vecindad de origen para el píxel verde claro de destino. (P.107)

2.4 Cómo aprende una Red Neuronal Profunda Convolucional

Según Hilera J. (1994), "una red neuronal se puede definir como un modelo matemático compuesto por un gran número de elementos de procesado organizados en niveles, estando basada en el funcionamiento de las neuronas biológicas. Se trata de una tecnología computacional emergente de la que existen una gran diversidad de tipos y aplicaciones". (P.9)

Así también, según I. Goodfellow, Y. Bengio, A. Courville (2016) la convolución, "(…) es una operación sobre dos funciones de un argumento de valor real."

Además, según I. Goodfellow, Y. Bengio, A. Courville (2016), "La operación de convolución es típicamente denotado con un asterisco (1): Seguidamente se muestra la representación matemática de la operación de convolución:"

$$s(t) = (x * w)(t). \quad \ldots\ldots \quad (1)$$

Ecuación 1. Representación matemática de la convolución

Fuente: (Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016)

"En la terminología de red convolucional, el primer argumento (en este ejemplo, la función x) de la convolución a menudo se conoce como la entrada, y el segundo argumento (en este ejemplo, la función w) como el

núcleo. La salida a veces se conoce como el mapa de características.” Esto puede ser observado en la siguiente figura 7:

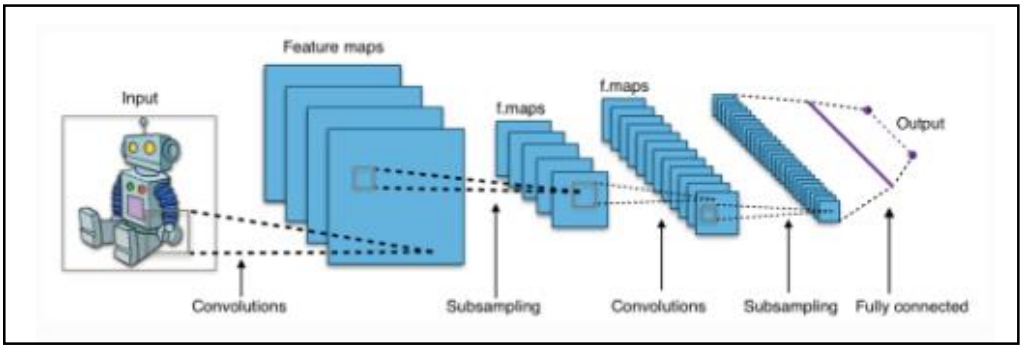


Figura 7. **Red Neuronal Convolucional**

Fuente: Ian Goodfellow, Yoshua Bengio, Aaron Courville

Además I. Goodfellow, Y. Bengio, A. Courville (2016), señala que “la entrada suele ser una matriz de datos multidimensional y el núcleo suele ser una matriz de parámetros multidimensional que se adaptan mediante el algoritmo de aprendizaje”. (P327-329)

Mathworks (2018) Dice que teniendo un conjunto de imágenes donde cada imagen contiene una de las cuatro categorías diferentes de objetos, y queremos que la red de aprendizaje profundo reconozca automáticamente qué objeto es cada imagen, se rotula las imágenes para tener datos de entrenamiento para la red (Figura 8).

Usando estos datos de entrenamiento, la red puede comenzar a entender las características específicas del objeto y asociarlas con la categoría correspondiente. (Figura 9)

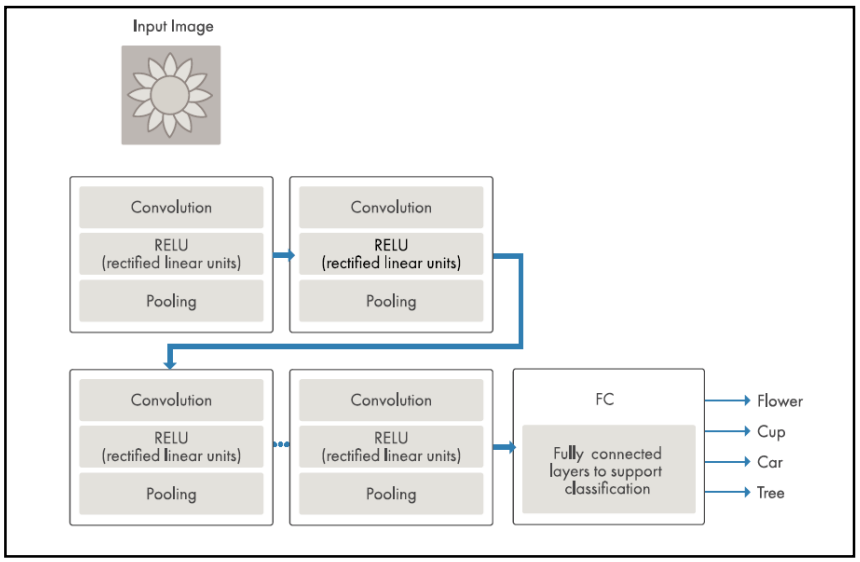


Figura 8. **Diagrama de una red de aprendizaje profundo para reconocimiento**

Fuente: (Mathworks, 2018)

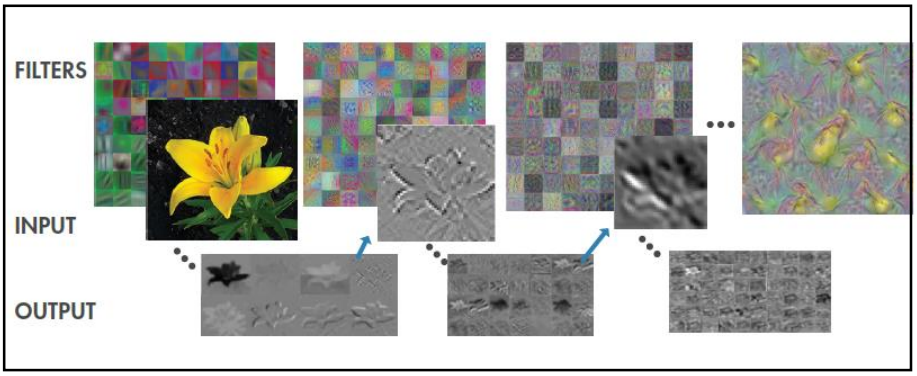


Figura 9. **Extracción de características durante el proceso de aprendizaje**

Fuente: (Mathworks, 2018)

Cada capa en la red toma datos de la anterior capa, lo transforma y lo pasa. La red aumenta el complejidad y detalle de lo que está aprendiendo de capa a capa.

Tenga en cuenta que la red aprende directamente de los datos: no tenemos influencia sobre qué características se están aprendiendo. (P.6)

CAPÍTULO III – RED NEURONAL PARA EL RECONOCIMIENTO FACIAL

El capítulo siguiente muestra una breve descripción del software MATLAB y su librería Deep Learning, se continúa con la explicación del procedimiento utilizado para la captura de imágenes, detallaremos también los pasos que seguimos para armar la arquitectura de la red neuronal de aprendizaje profundo, explicaremos las características que tienen cada una de las capas y finalmente mostraremos las funciones para el algoritmo desarrollado. (ver Figura 10)

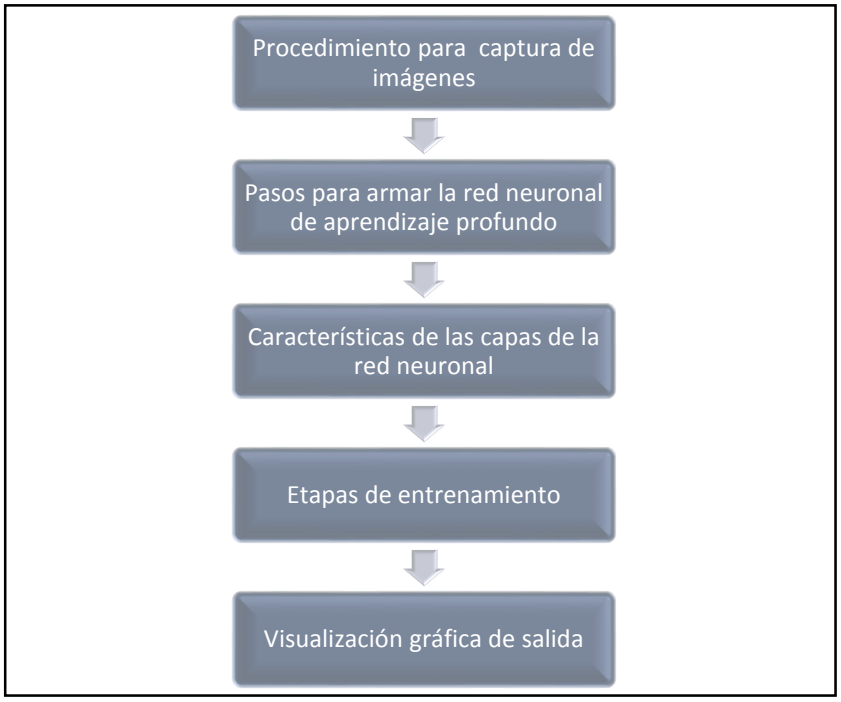


Figura 10. Diagrama de bloques general de la tesis

Fuente: Elaboración propia

3.1 Matlab y toolbox Deep Learning

3.1.1 Matlab

Mathworks (2019) Millones de ingenieros y científicos en todo el planeta utilizan MATLAB® para analizar y diseñar los sistemas y productos que transforman nuestro mundo. El lenguaje de MATLAB, basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Las gráficas integradas facilitan la visualización de los

datos y la obtención de información a partir de ellos. El entorno de escritorio invita a experimentar, explorar y descubrir. Todas estas herramientas y funciones de MATLAB están probadas rigurosamente y diseñadas para trabajar juntas.

MATLAB ayuda a llevar sus ideas más allá del escritorio. Puede ejecutar análisis en conjuntos de datos de mayor tamaño y expandirse a clústeres y nubes. El código de MATLAB se puede integrar con otros lenguajes, lo que permite implementar algoritmos y aplicaciones en sistemas web, empresariales o de producción. (P.1)

3.1.2 Toolbox Deep Learning

M. Hudson, M. Hagan, H. Demuth (2018) Explican que el Neural Network Toolbox de MATLAB proporciona comandos simples para crear e interconectar las capas de una red neuronal profunda. Ejemplos y redes pre-entrenadas facilitan el uso de MATLAB para el aprendizaje profundo, incluso sin conocimientos avanzados de algoritmos de visión por computador o redes neuronales. (P.2-2)

3.2 Procedimiento para la Captura de Imágenes

Este procedimiento se realizó de dos formas:

- A través de una galería de imágenes ya disponible.
- A través de un dispositivo que capture imágenes.

3.2.1 A través de una galería de imágenes ya disponible

Al tener un repositorio de imágenes almacenadas en una base de datos, dispositivos de almacenamiento u otros, se pudo extraer estas imágenes y ordenarlas en carpetas dentro de un Data Store mediante Matlab para su posterior uso en las redes neuronales. La Figura 11 contiene el código de Matlab que permitió crear un Data Store

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

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

Figura 11. **Declaración del datastore en Matlab**

Fuente: (MATLAB, 2019)

El data store extraído de una carpeta llamada “rostros” como se muestra en la Figura 12, contiene dos carpetas, una de ellas fue la carpeta “rostros” que contiene todas las carpetas de la parte derecha de la Figura 12.

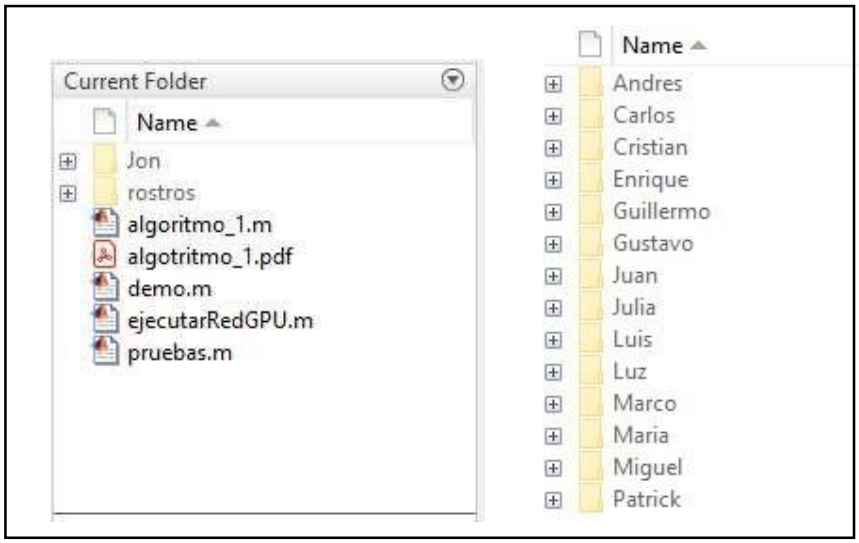


Figura 12. **Carpetas utilizadas para formar el Data store**

Fuente: (MATLAB, 2019)

En el proceso de creación del Data Store se le agregó las etiquetas a cada imagen para que así la red neuronal las utilice en su análisis.

3.2.2 A través de un dispositivo que capture imágenes

Para este proceso se hizo uso del objeto “webcam” proporcionado por uno de los Add Ons de Matlab el cual te permite utilizar los dispositivos que capturan imágenes que estén conectados al ordenador.

Se ingresó un código de Matlab para el inicio de la captura de imágenes el cual se muestra en la siguiente imagen (Figura 13):

```
cam=webcam;

for i = 1:30
    img=snapshot(cam);
    imwrite( img , strcat('pic', num2str(i)  ,'.jpg'));
    pause(0.2)
end
```

Figura 13. **Algoritmo para captura de imágenes con dispositivo**

Fuente: (MATLAB, 2019)

La función del código de la Figura 13 utilizó el Add Ons Webcam con el objetivo de habilitar el dispositivo que captura imágenes; luego, a través de un bucle, tomó 30 fotos con un intervalo de 0.2 segundos de pausa y luego los guardó con el nombre de "pic" - número de foto tomada- ".jpg".

3.3 Procedimiento para el entrenamiento de la red neuronal

La descripción de la arquitectura se realizó por medio de pasos o fases, los cuales se mostrarán a continuación:

3.3.1 Paso 1 Re-muestreo de Imágenes

- Se definió las categorías y la carpeta de origen en donde se extraerán las imágenes.
- Se creó el Data Store agregando las etiquetas del análisis.
- Se modificó el tamaño de las imágenes a analizar de acuerdo con la funcion augmentedImageDatastore a un tamaño de 128 x 128 pixeles.

3.3.2 Paso 2 Capas

- Se definió el tamaño de los inputs de la red, es decir 128 x 128 x 3 canales.
- Se creó dos capas convoluciones para la red, con 16 y 32 filtros de tamaño 4 x 4.
- Se definió los parámetros Relu, MaxPool, drop, fc1, sm1 y class que se usarán en el arreglo "Layers".

- Se implementó las convoluciones y parámetros definidos anteriormente en el arreglo “Layers”.

3.3.3 Paso 3 Opciones

Se definió las siguientes opciones:

- ‘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’: Se encarga de generar una gráfica sobre el aprendizaje de la red neuronal.

3.3.4 Paso 4 Entrenamiento

- Se usó de la función “trainNetwork” del paquete de Add Ons de Deep Learning Matlab la cual entrenó nuestra red según los Layers y Opts ingresados.
- Se visualizó el resultado del proceso de aprendizaje.

3.3.5 Paso 5 Prueba

- Se seleccionó la o las imágenes que se ingresaban para las pruebas, en el caso que sean varias se hizo uso del Data Store.
- Se validó las respuestas por medio de porcentajes.

3.4 Características de las Capas de la Red Neuronal

En esta sección mostramos las principales características que tienen las capas de la red neuronal entrenada. También indicamos el proceso que cada etapa ejecuta durante el tiempo que la red es entrenada. En los siguientes párrafos explicaremos en detalle lo señalado.

3.4.1 Capa de ingreso de imagen

Esta capa nos permite definir el tamaño de entrada del grupo de imágenes que van a ser parte del entrenamiento de la red neuronal convolucional, en diferentes clases a evaluarse. Es preciso señalar que se debe ingresar un tamaño correspondiente al alto, ancho y cantidad de canales de color. En caso se tengan imágenes en blanco y negro el número de canales es 1, si son imágenes en color la cantidad de canales es 3.

Esta capa realiza una normalización a la información ingresada, que consiste en transformar los datos para que se encuentren en el rango de [0, 1].

3.4.2 Capa de convolución

La capa convolucional es una operación lineal que tiene como característica principal la extracción de información de la data entrenada, para ello hace uso de filtros o kernel que generan nuevas imágenes llamadas "feature maps", las cuales contienen características importantes de cada imagen evaluada. (Figura 14)

Cabe indicar 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).

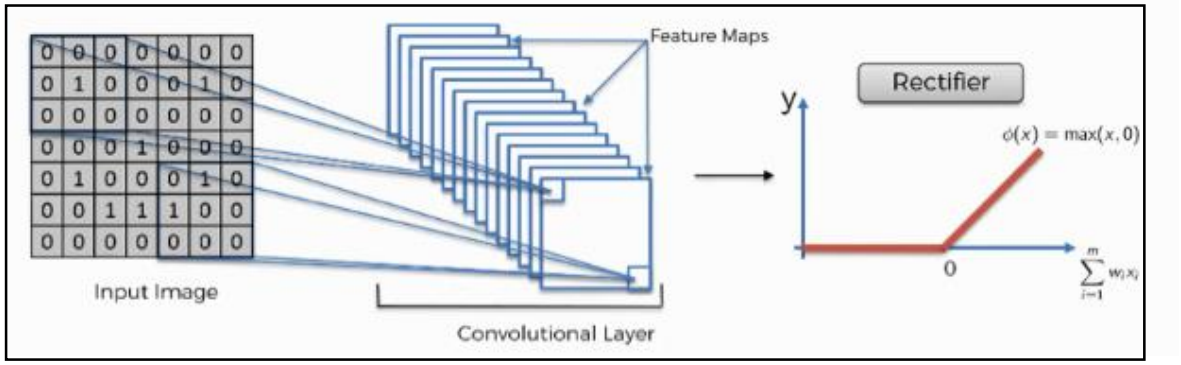


Figura 14. Capa de convolución y feature maps, antes de ingresar a la capa ReLu

Fuente: (Team, 2018)

3.4.3 Capa ReLU (función de activación)

Esta capa consiste básicamente en una función (Figura 15) que permite a la red mantener los valores positivos de los resultados obtenidos de la capa precedente. Esta función de activación nos permite detectar un rango de patrones para predecir con mejor precisión las imágenes que la red neuronal va a entrenar.

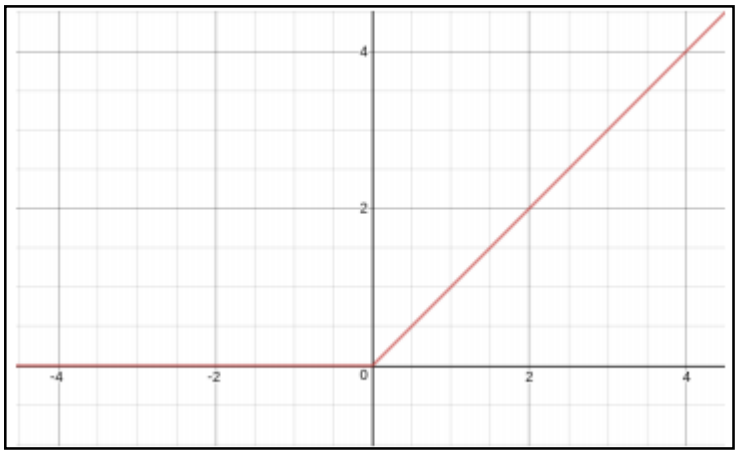


Figura 15. Representación gráfica de la función de activación ReLu

Fuente: (Agarap, 2019)

3.4.4 Capa de pooling

En esta capa se puede notar que el muestreo de la información ingresada va a reducir el número de parámetros que van a ingresar después en las siguientes capas. Cabe mencionar que las principales características obtenidas en las capas precedentes que detectaron los filtros se mantienen. Gráficamente se puede apreciar el proceso en la Figura 16:

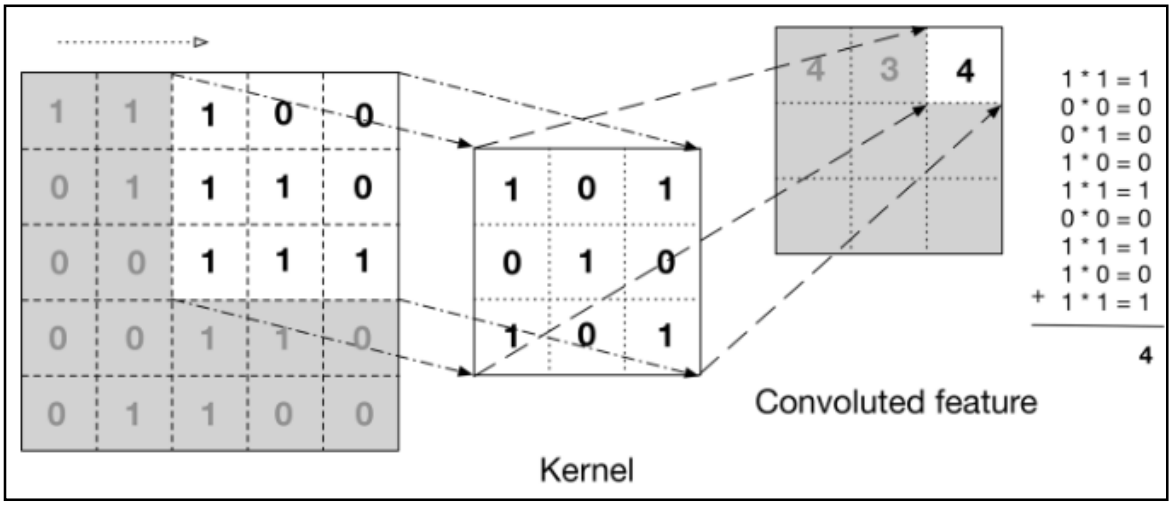


Figura 16. Filtro o kernel

Fuente: (Adam Gibson, Josh Patterson, 2017)

3.4.5 Capa de dropout

En el momento del entrenamiento, la capa establece aleatoriamente los elementos de entrada en cero dados por la máscara *rand(size(X))<Probability*, donde X es la capa entrada y luego escala 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.

3.4.6 Capa fully connected

Las capas de convolución, Relu y dropout tienen a continuación una capa donde las neuronas se conectan con las neuronas de capas anteriores. También en esta capa se combinan las características que se aprendieron en capas precedentes. Para la red neuronal que hemos evaluado, esta capa permite la posterior clasificación de las imágenes.

3.4.7 Capa de salidas: softmax y clasificación

Mediante la capa "classification" a la salida de la red neuronal, se puede obtener la clasificación de cada uno de los datos que se ingresaron para ser evaluados en la red.

Esta capa debe seguir a la capa softmax que es otra función de activación que generalmente se utiliza a la salida de la red neuronal, esta capa asigna una probabilidad a cada clase ingresada como data y que se va a clasificar. La suma de estas probabilidades no debe ser mayor a 1.

3.5 Etapas de entrenamiento

Aplicamos la técnica de aprendizaje profundo (Deep Learning) y convolución de una red neuronal para comenzar a escribir las funciones y variables del algoritmo.

El uso de la librería Deep Learning en Matlab abre la posibilidad y facilidad para la creación del algoritmo que esta tesis presenta.

Cabe recalcar que las herramientas de aprendizaje profundo y convolución son adecuados para el reconocimiento de imágenes, que en esta tesis presentamos para el reconocimiento facial de estudiantes del laboratorio de control.

3.5.1 Definición de data

Primero definimos la ruta donde se encuentran las carpetas que contienen las fotografías de los alumnos, además ingresamos un algoritmo (ver anexo) para tomar fotos en caso existiese un alumno que no haya sido parte del grupo de fotos que inicialmente se preparó; luego de ello juntamos dicho grupo de imágenes (data set) en una variable de tipo Objeto. A continuación se muestra lo señalado:

```
rootFile = 'rostros';
categories = {'Juana','Luis','Pedro','nestor'};
imds = imageDatastore(fullfile(rootFile,categories),'LabelSource','foldernames');
```

Luego de la agrupación, ingresamos una variable que representa el tamaño con el cual deseamos que la red neuronal vaya a entrenar. También separamos con una función (splitEachLevel) la data set, una parte para el entrenamiento de la red y el otro grupo para la validación o testeo del reconocimiento facial.

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

3.5.2 Ingreso de capas de la red neuronal

Para la red neuronal convolucional Matlab permite el uso de librerías que facilitan la construcción de la arquitectura de la red a entrenarse.
Inicialmente colocamos la capa de ingreso a la red, haciendo uso de la variable que contiene el tamaño con la cual las imágenes serán entrenadas y luego probadas, en este caso 128 x 128 pixeles. Debido al uso de imágenes a colores, tener en cuenta que el número de canales es 3.

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

Luego de la capa mencionada, procedemos a declarar la capa de convolución. En esta capa, la red extrae las principales características, las cuales se convierten en “nuevos” mapas de características. Como se describió en el apartado 3.4, ello se produce por el

trabajo que realizan los filtros que recorren las imágenes que se entrenan para el reconocimiento.

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

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

A continuación de la convolución, se recomienda ingresar capas de función de activación. Para la red que entrenamos, dicha función es de tipo no lineal llamada ReLu. Básicamente lo que realiza esta capa es convertir aquellos valores de entrada menores a cero en “0”.

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

La siguiente capa en nuestra red neuronal se llama Pooling. Se declaró esta capa para que ejecutar una reducción de la información obtenida de las capas anteriores pero sin perder las características más resaltantes de la imagen inicial entrenada.
La opción que optamos a declarar es 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 2 x 2, cuyo desplazamiento “stride” fue de 2.

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

Ingresamos una capa de dropout para “eliminar” aleatoriamente los elementos de entrada lo que nos permitió mejorar el sobre ajuste que nuestra red estaba experimentando. El valor elegido fue 0.4 que representa el 40% de elementos retenidos a la salida.

drop = dropoutLayer(0.4);

Luego de las capas mencionadas se recomienda la declaración de la capa "fully connected". El proceso llevado por esta capa es la de conectar las neuronas de las capas previas, combinando las características principales aprendidas durante el entrenamiento para la identificación de patrones. Tener presente que el tamaño de salida de esta capa debe ser el mismo número de clases de la data de entrenamiento, para nuestro caso ese número es dado por la variable "num_categories".

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

Para las capas de salida para la clasificación, se recomienda el uso de "Softmax y Classification". La capa de softmax utiliza otra función de activación para normalizar la salida de la capa Fully Connected. Los valores otorgados por la capa Softmax permitirán a la capa de Classification seleccionar a que clase la imagen entrenada pertenece.

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

Finalmente podemos resumir la declaración de las capas mencionadas en una única variable llamada "layers" (Figura 17 y Figura 18 como representación en bloques):

```
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")];
```

Figura 17. **Capas ingresadas para el entrenamiento**

Fuente: Elaboración propia

3.5.3 Algoritmo para especificar las opciones de entrenamiento

Matlab nos permite usar la opción “trainingOptions” para especificar los valores con los cuales nuestra red neuronal es entrenada para el reconocimiento de imágenes.

En primer término utilizamos el Descenso de Gradiente Estocástico (SGD), el cual es usado de forma iterativa para ajustar aleatoriamente los parámetros obtenidos del entrenamiento, hasta tener un mínimo error.

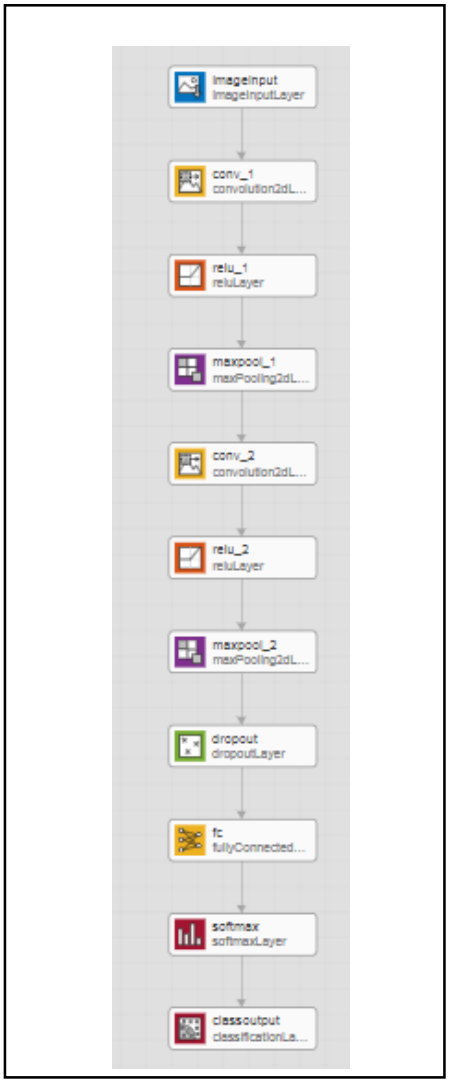


Figura 18. **Diagrama de bloques de la arquitectura de la red neuronal**

Fuente: Elaboración propia

Developers, Google (2019) Explica que en el descenso de gradientes, un lote es la cantidad total de ejemplos que usas para calcular la gradiente en una sola iteración. Hasta ahora, hemos supuesto que el lote era el conjunto de datos completo. Al trabajar a la escala de Google, los conjuntos de datos suelen tener miles de millones o incluso cientos de miles de millones de ejemplos.

Además, los conjuntos de datos de Google con frecuencia contienen inmensas cantidades de atributos. En consecuencia, un lote puede ser enorme. Un lote muy grande puede causar que incluso una sola iteración tome un tiempo muy prolongado para calcularse.

Es probable que un conjunto de datos grande con ejemplos muestreados al azar contenga datos redundantes. De hecho, la redundancia se vuelve más probable a medida que aumenta el tamaño del lote. Un poco de redundancia puede ser útil para atenuar las gradientes inconsistentes, pero los lotes enormes tienden a no tener un valor mucho más predictivo que los lotes grandes.

¿Cómo sería si pudiéramos obtener la gradiente correcta en promedio con mucho menos cómputo? Al elegir ejemplos al azar de nuestro conjunto de datos, podríamos estimar (si bien de manera inconsistente) un promedio grande de otro mucho más pequeño. El descenso de gradiente estocástico (SGD) lleva esta idea al extremo: usa un solo ejemplo (un tamaño del lote de 1) por iteración. Cuando se dan demasiadas iteraciones, el SGD funciona, pero es muy inconsistente. El término "estocástico" indica que el ejemplo único que compone cada lote se elige al azar. (P.14)

A continuación el comando usado SGD con una tasa de aprendizaje (initial learn rate) de valor 0.0001 fue el elegido. Si la tasa de aprendizaje es demasiado baja, el entrenamiento lleva mucho tiempo. Si la tasa de aprendizaje es demasiado alta, la capacitación podría alcanzar un resultado sub-óptimo o divergir.

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

Matemáticamente se puede expresar el descenso del gradiente estocástico de la siguiente manera (ecuación 2), donde “alfa” representa la tasa de aprendizaje:

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

Ecuación 2. Expresión matemática SGD

Fuente: (Stanford University, 2019)

También los siguientes parámetros para el entrenamiento son ingresados:

'L2Regularization',0.003, ...

Esta función L2 Regulatization ayuda a mejorar el "overfitting" (sobre entrenamiento de la red) causado por nueva data ingresada durante el entrenamiento de la red neuronal. El valor de 0.003 viene a ser la tasa de regularización que mejor se adecuó para la obtención de los mejores 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 nuestro modelo está aprendiendo detalles y ruidos de los datos de entrenamiento.

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

El parámetro "maxepoch" igual a 20, fue la cantidad de pasadas que dio el algoritmo de entrenamiento sobre todo el conjunto de imágenes a entrenarse, así también el tamaño que tuvo el "miniBatch" que se utilizó en nuestra red fue 64. Tener presente que el MiniBatch, es un grupo más reducido de imágenes que es usado para evaluar la pérdida del gradiente SGD y así actualizar los pesos de cada neurona.

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

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". Para la red que hemos entrenado no fue necesario este parámetro debido que inicialmente se mezclaron las imágenes.

'ValidationData',augimgTrain, ...
'ValidationFrequency',1, ...

Durante el entrenamiento, se mide la precisión y la pérdida de los datos a entrenarse (variable augimgTrain); el comando "validationdata" permitió medir ese proceso cada "x" iteraciones; en nuestro caso cada vez que se realizaba la iteración completa.

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

El parámetro “plots” muestra el progreso del entrenamiento en una gráfica la cual 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”.

3.5.4 Ejecución para entrenar la red neuronal

El algoritmo utilizado para el entrenamiento es dado por el paquete de Add Ons de Deep Learning en Matlab el cual se muestra a continuación:

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

En el código anterior se puede observar que hay 3 parámetros los cuales tienen los siguientes significados:

- augimgTrain: Esta variable contiene el Data Store pre procesado a 128 x 128 x 3 de las imágenes destinadas al entrenamiento de la red neuronal.
- layers: Indica las capas que tendrá la red neuronal.
- opts: Contiene todas las opciones para el desarrollo de la red neuronal.

Para entender como trabajó la red neuronal de aprendizaje profundo convolucional nos apoyaremos de la breve descripción que nos brinda Mathworks en su guía de usuario disponible en su página web.

Mathworks (2018) Explica que una red neuronal convolucional (CNN o ConvNet) es uno de los algoritmos más populares para el aprendizaje profundo con imágenes y video.

Al igual que otras redes neuronales, una CNN se compone de una capa de entrada, una capa de salida y muchas capas ocultas en el medio.

Digamos que tenemos un conjunto de imágenes donde cada imagen contiene una de las cuatro categorías diferentes de objetos, y queremos que la red de aprendizaje profundo reconozca automáticamente qué objeto está

cada imagen. Rotulamos las imágenes para tener datos de entrenamiento para la red.

Usando estos datos de entrenamiento, la red puede comenzar a entender las características específicas del objeto y asociarlas con la categoría correspondiente.

Cada capa en la red toma datos de la anterior capa, lo transforma y lo pasa. La red aumenta el complejidad y detalle de lo que está aprendiendo de capa a capa. (P.7)

Las capas en el medio (Figura 19) en Matlab estarán agrupadas en las capas convolución, pooling y Relu, las cuales fueron descritas en la sección 3.4 del presente estudio y que se encierran con el recuadro naranja en la figura a continuación.

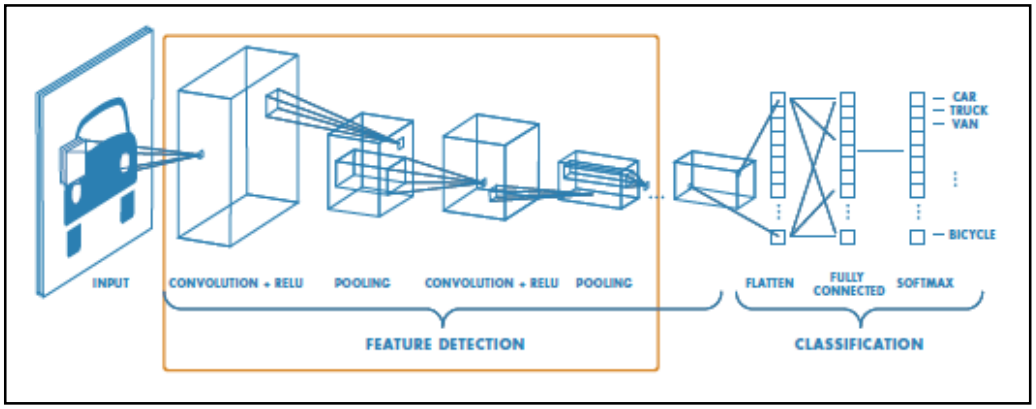


Figura 19. **Capas intermedias**

Fuente: (Mathworks, 2018)

En Matlab las capas a la salida estarán agrupadas en las capas dropout (flatten), fully connected, softmax y classification, las cuales fueron descritas en la sección 3.4 del presente estudio y que se encierran con el recuadro naranja en la Figura 20.

3.6 Visualización gráfica de salida

En la Figura 21 se describe el entrenamiento de la red neuronal para 20 épocas, un tiempo de entrenamiento de aproximadamente 6 minutos y con una precisión de entrenamiento del 100%

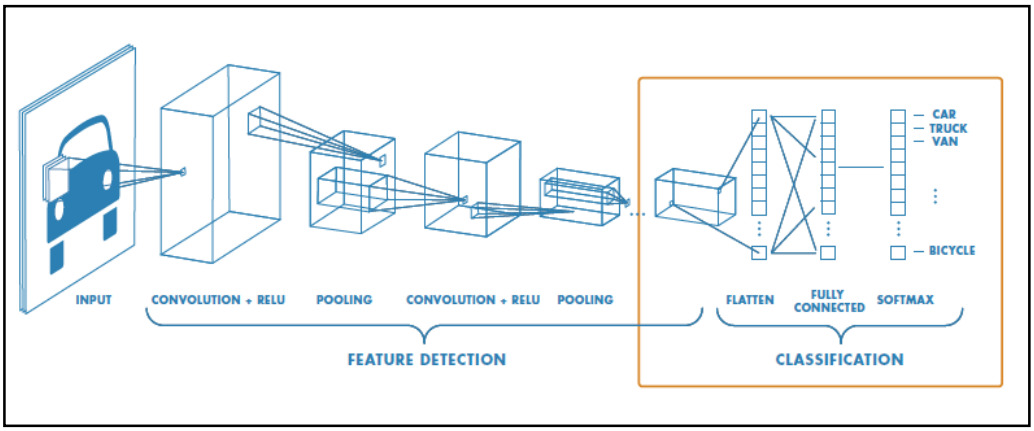


Figura 20. **Capas finales**

Fuente: (Mathworks, 2018)

Proceso de ejecución:

```
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 |
|=========================================================================================|
```

Figura 21. **Resultado de las iteraciones durante el entrenamiento**

Fuente: Elaboración propia

Luego del entrenamiento resulta importante validar las imágenes de testeo para corroborar la precisión de entrenamiento de la red neuronal. Para ello en el código mostramos todas las imágenes de testeo asociadas a un número a ser seleccionado como se muestra a continuación.

Petición de la imagen para testear:

```
Ingrese el número de imagen que desea probar: 11
```

Resultado del análisis de la imagen número 11 identifica correctamente a la persona testeada y le da la precisión calculada en el código del 100% como se muestra en la Figura 22:

Resultado del análisis de la imagen número 11:

Figura 22. **Visualización del resultado de la imagen 11 evaluada**

Fuente: Elaboración propia

Mediante la toolbox de matlab para Deep Learning podemos invocar el Proceso de entrenamiento de forma gráfica y evolutiva expresado en 2 gráficos de Exactitud y Pérdida. (ver Figura 23)

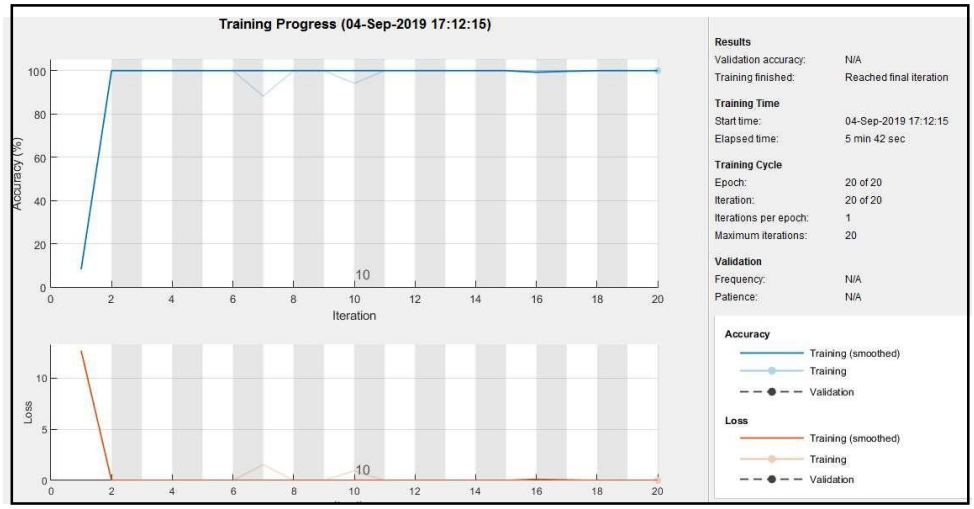


Figura 23. **Gráfico del comportamiento durante el entrenamiento de la red neuronal**

Fuente: Elaboración propia

Durante la ejecución del código en el Matlab se muestra una serie de variables (Figura 24) que se crean durante todo el proceso, entre las que tenemos, por ejemplo: cam que hace referencia a la cámara fotográfica, img la imagen cargada, probs matriz de probabilidades para reconocimiento facial, entre otras que han sido descritas en el capítulo 3.

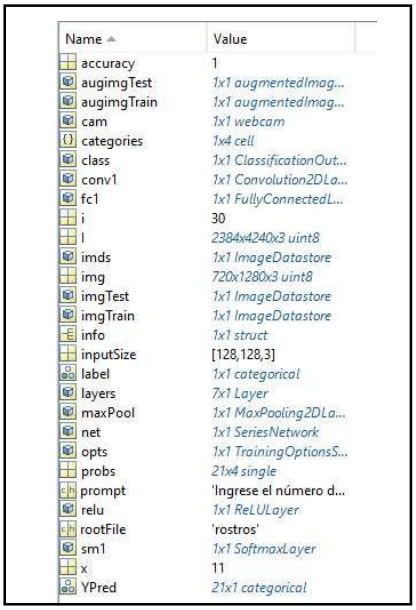


Figura 24. Variables creadas al procesar el algoritmo de la red neuronal

Fuente: Elaboración propia

A continuación, se agrega un cuadro con la cantidad de imágenes utilizadas por alumno para el entrenamiento y prueba. (Tabla 1)

Alumnos	Imágenes para el entrenamiento	imágenes para la validación
Andres	29	5
Carlos	30	6
Cristian	36	7
Enrique	30	6
Guillermo	40	8
Gustavo	39	7
Juan	19	3
julia	32	6
Luis	35	7
Luz	29	5
Marco	28	5
Maria	26	5
Miguel	27	5
Patrick	26	5

Tabla 1. Cantidad de Imágenes tomadas y validadas por alumno

Fuente: Elaboración propia

CAPÍTULO IV – PRUEBAS Y RESULTADOS

Las pruebas que se realizaron con una muestra de la fuente de datos de un universo de 14 alumnos pertenecientes al laboratorio de Control. Dentro de la carpeta que contiene los datos, las imágenes se dividen según el nombre de carpeta que corresponde a cada uno de los alumnos.

La carpeta de nombre "rostros", corresponde a 14 alumnos de la fuente principal donde se tuvo un total de 426 imágenes finales usadas para esta investigación.

A nivel lógico separamos a través de una sentencia del Matlab llamada "splitEachLabel" el 80% de las imágenes 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 (tabla 1).

4.1 Criterios de Evaluación

Para evaluar los resultados se han tenido en cuenta tres criterios de evaluación. El primero ha sido llamado "Precisión". Mide la exactitud del método dividiendo todos los casos clasificados correctamente entre el número total de casos evaluados. Así podemos conocer el tanto por ciento de acierto del método. La ecuación utilizada es la siguiente:

$$\text{Precisión} = \frac{Casos\ Clasificados\ Correctamente}{Casos\ Evaluados} * 100$$

Ecuación 3. Fórmula precisión

En el segundo criterio obtenemos el porcentaje de falsos positivos, "FP", hemos definido los falsos positivos como el conjunto de imágenes clasificadas de alumnos que no correspondían, utilizando la ecuación que se expone a continuación:

$$\text{FP} = \frac{Casos\ Clasificados\ Erroneamente}{Casos\ Clasificados\ como\ Correcto\ en\ el\ Sistema} * 100$$

Ecuación 4. Fórmula falsos positivos

El tercer Criterio lo definimos como falsos negativos es el conjunto de imágenes que por el contrario sí correspondían a los alumnos, pero fueron clasificados por otros. Estos últimos son los que más deben preocuparnos, pues dar un resultado erróneo en este sentido podría suponer el acceso a alumnos que no les corresponde estar en el laboratorio, consiguiendo lo contrario a lo que buscamos. La ecuación usada para calcularlo fue:

$$\text{FN} = \frac{Casos\ Clasificados\ Erroneamente\ en\ el\ Sistema}{Casos\ Clasificados\ Correctamente} * 100$$

Ecuación 5. Fórmula falsos negativos

4.2 Resultados Experimentales

En este apartado exponemos las opciones de entrenamiento utilizadas para realizar cada prueba, también los resultados obtenidos y algunas gráficas del proceso de entrenamiento de la red.

Los resultados obtenidos para la primera versión son los que aparecen en la tabla 2. Para entrenar la red se han usado el 80% de las imágenes, y se han utilizado las siguientes opciones de entrenamiento (mayor información en el punto 3.5.3 de la tesis):

```
opts = trainingOptions('sgdm', ... % uso de descenso de gradiente estocástico
    'InitialLearnRate', 0.0001, ...  % tasa de aprendizaje en 0.0001
    'ValidationData',augimgTrain, ... % mide precisión y pérdida del data set
    'ValidationFrequency',1, ... % medición en cada iteración
    'MaxEpochs', 10, ... % número de épocas
    'MiniBatchSize', 64, ... % tamaño de las imágenes para evaluar el gradiente
    'L2Regularization', 0.003, ... % tasa de regularización
    'Plot', 'training-progress','Verbose', true);  % gráfica de progreso
```

4.2.1 Prueba 1:

Inicialmente se realizó una prueba haciendo uso de dos capas de convolución con 16 y 32 filtros respectivamente, con una tasa de aprendizaje de 0.001, las épocas para las iteraciones fueron 10. El resultado obtenido con estos parámetros se observa en la igura 25, llegando a un entrenamiento con un resultado de precisión de 62% (ver tabla 2).

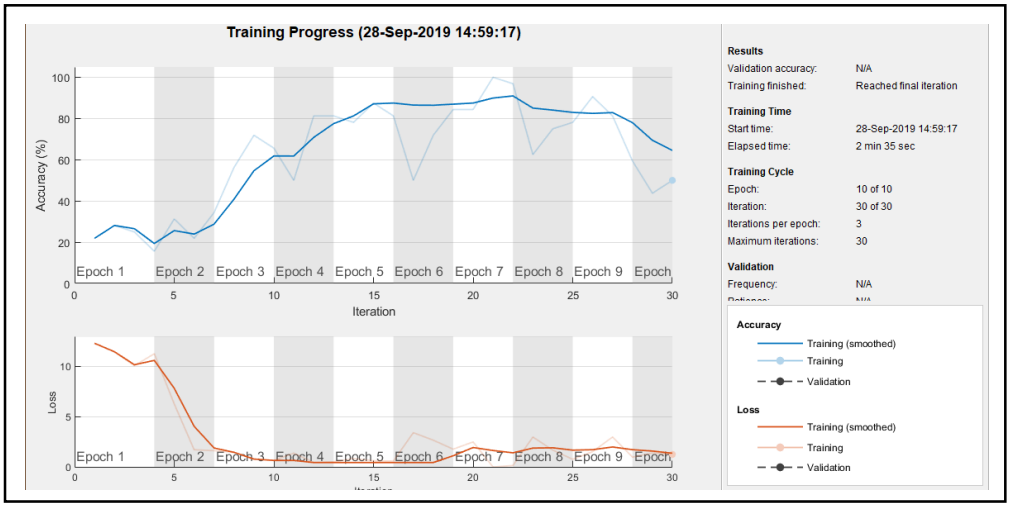


Figura 25. Comportamiento de la prueba 1

Fuente: Elaboración propia

Observaciones:

- La red neuronal convolucional no identifica las fotos tomadas por la webcam (30 fotos para el aprendizaje).
- Al realizar la prueba de la red con otra persona sí reconoce.

Posibles causas:

- Calidad de fotos de la webcam: Al revisar el repositorio de las 30 fotos tomadas por la webcam se puede observar que algunas salen borrosas debido al movimiento que realiza el sujeto para tener variedad de fotos.
- Cantidad de fotos: Al analizar a detalle los pixeles se puede necesitar más de 30 fotos para aprender correctamente.

Soluciones:

- Mostrar en pantalla la foto tomada y dar un tiempo al sujeto de prueba para que cambie de pose al tomar las fotos.
- Probar con 30 y 45 fotos en el entrenamiento.

- Modificar las capas para que analice más a detalle.

Cambios:

- Se reemplazó la fórmula de porcentaje de precisión "accuracy" por otra recomendada.

4.2.2 Prueba 2:

Se realizó una prueba haciendo uso de tres capas de convolución con 16, 32 y 64 filtros respectivamente, con una tasa de aprendizaje de 0.001, las épocas para las iteraciones fueron 20. El resultado obtenido con estos parámetros se observa en la figura 26, llegando a un entrenamiento con un resultado de precisión de 38%. (Ver tabla 2).

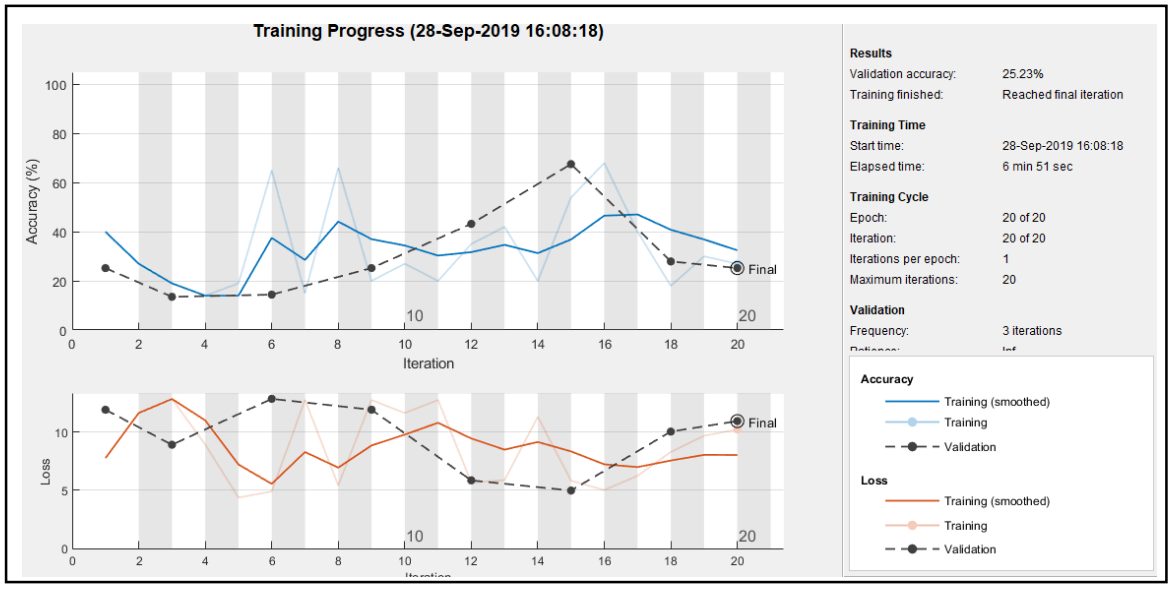


Figura 26. Comportamiento de la prueba 2

Fuente: Elaboración propia

Observaciones:

- La red tuvo un aprendizaje muy bajo según la gráfica de aprendizaje y resultados.
- Al evaluar en otros sujetos el resultado es erróneo:

Soluciones:

- Se eliminó la capa 3 y se volvió a probar la red con la fórmula de acertamiento.

4.2.3 Prueba 3:

Se realizó una prueba haciendo uso de dos capas de convolución con 16 filtros, 32 respectivamente, con una tasa de aprendizaje de 0.001, las épocas para las iteraciones fueron 15. El resultado obtenido con estos parámetros se observa en la figura 27, llegando a un entrenamiento con un resultado de precisión de 45%. (Ver tabla 2)

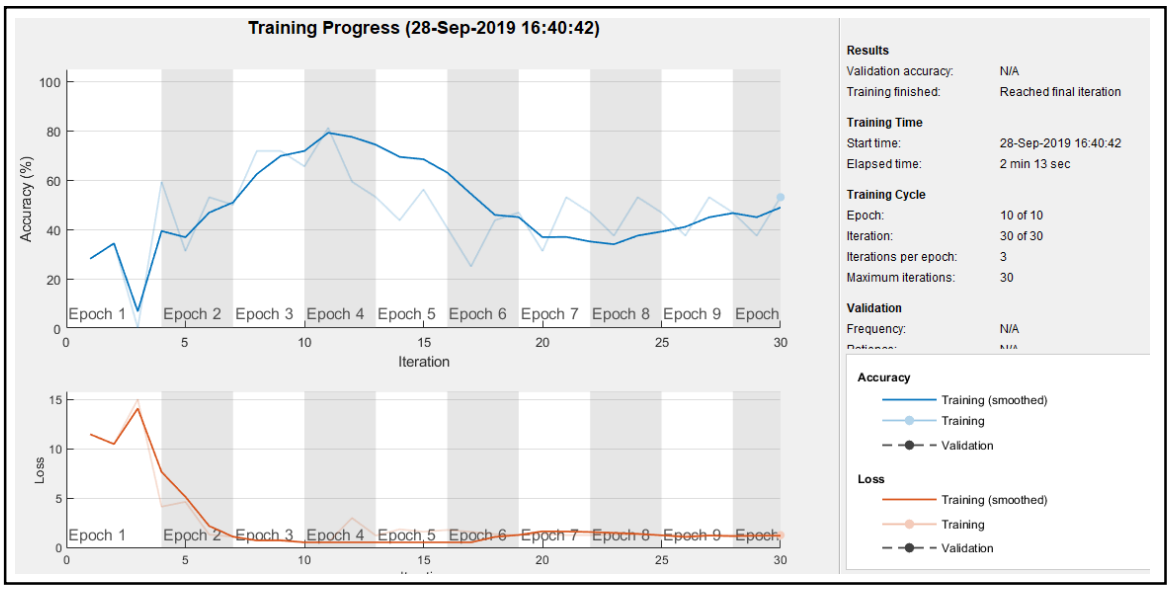


Figura 27. Comportamiento de la prueba 3

Fuente: Elaboración propia

Observaciones:

- La gráfica de aprendizaje esta por debajo del promedio sin embargo el resultado con la webcam es exitoso.
- Al probar con las imágenes predeterminadas de las categorías (categories = {'Andres','Luis','Juan','Maria'}) surge un error constante en donde determina que todos son Luis con un error del 20.5%:

Soluciones:

- Se modificó las opciones de la red, implementando las de la PRUEBA 2.
- Se revisó el cálculo del porcentaje de precisión, para validar la eficacia de la prueba 3.

4.2.4 Prueba 4:

Finalmente, en esta prueba utilizamos dos capas de convolución con 16 y 32 filtros respectivamente, con una tasa de aprendizaje de 0.0001, las épocas para las iteraciones fueron 10, se agregó una capa dropout que permitió mejorar el "overfitting". El resultado obtenido con estos parámetros se observa en la figura 28, teniendo un resultado de precisión del 94% para el entrenamiento de la red neuronal.

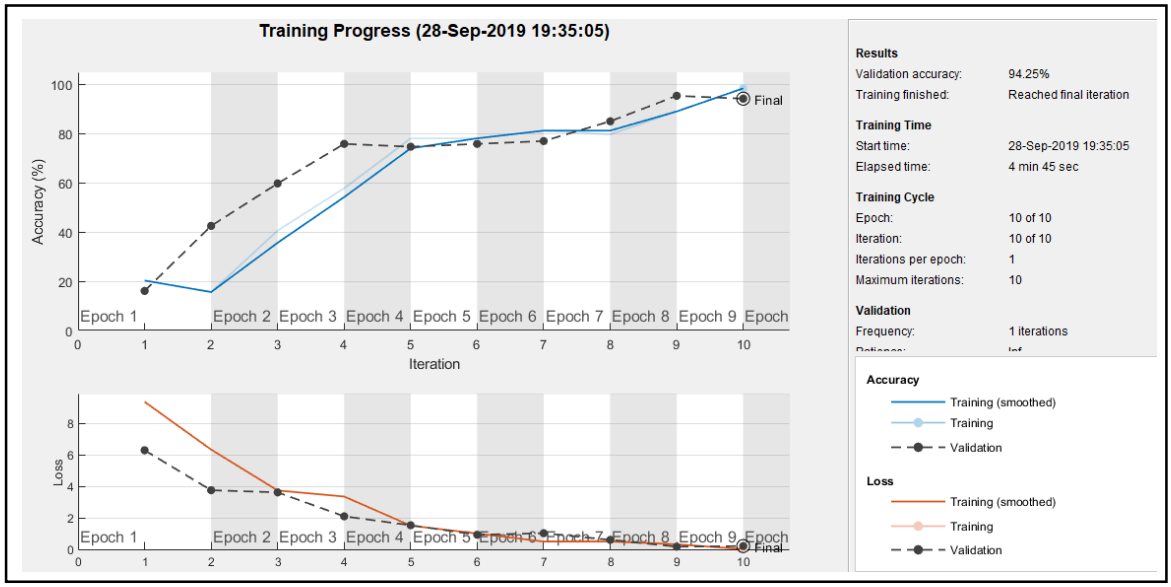


Figura 28. Comportamiento de la prueba 3

Fuente: Elaboración propia

Aquí se muestran todas las imágenes (figura 29) que no han sido entrenadas y que se usaron para la validación, asociándolos con un número.

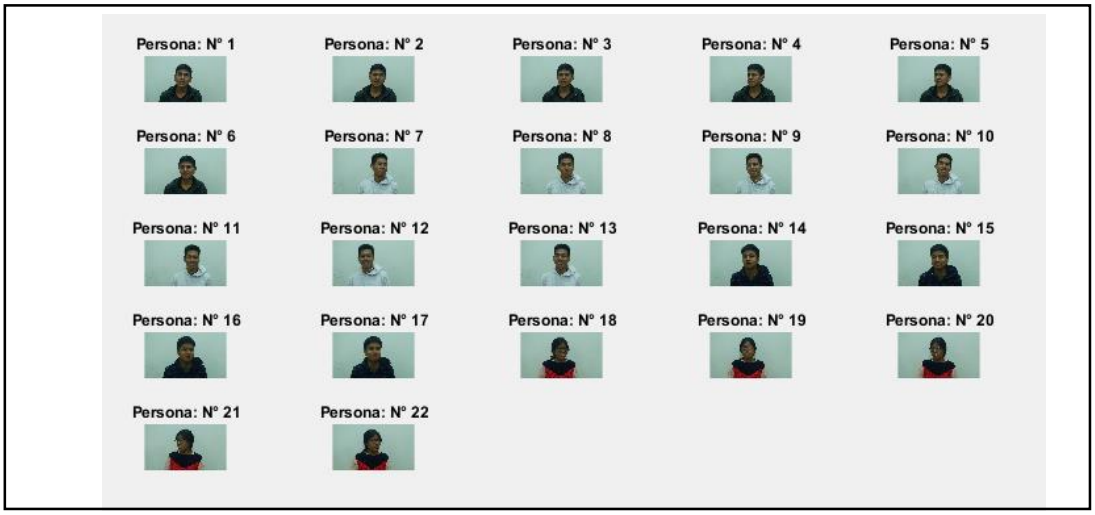


Figura 29. Grupo de imágenes no entrenadas

Fuente: Elaboración propia

Cálculo del porcentaje de predicción

Para determinar el porcentaje de predicción correcto usamos los porcentajes que devuelve la matriz Probs, para ello utilizamos la función MAX tal como se describe en el siguiente código de Matlab.

El "x" escogido es 7, número que representa una de las fotos no entrenadas.

```
% Ingreso de imagen a probar
prompt = 'Ingrese el número de imagen que desea probar: ';
x = input(prompt);
x = double(x);

while( x > size(imgTest.Files) & x <= 0)
    x = input(prompt);
    x = double(x);
end

subplot(1,1,1)
I = readimage(imgTest,x);
imshow(I)
label = YPred(x);
clearvars max
[maximo2,pos2] = max(probs(x,:));

% ASIGNACION DE COLOR AL TITULO DE LA IMAGEN SEGUN EL ACIERTO
if  maximo2 >= 0.999
    colorText = 'g';
    title(string(label) + ", " + num2str((100*maximo2),3) + "%", 'Color', colorText);
else
    colorText = 'r';
    title("IMAGEN NO RECONOCIDA - " + string(label) + ", " + num2str((100*maximo2),3) + "%", 'Color', colorText);
end
```

Por lo tanto, el resultado se muestra con un mensaje de color VERDE, con el nombre de "Luis" y un porcentaje de acierto de 100% tal como muestra la figura 30.

Figura 30. Representación del resultado de la evaluación

Fuente: Elaboración propia

En la presente tabla se resume las opciones utilizadas y los resultados obtenidos en cada una de las pruebas realizadas.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4
Capas	2	3	2	2
Filtros	16, 32	16,32,64	16, 32	16, 32
Tasa de Aprendizaje	0.001	0.001	0.001	0.0001
Épocas	10	20	15	10
Precisión	62%	38%	45%	94%

Tabla 2*:* Opciones utilizadas para las pruebas.

Fuente: Elaboración propia

De la misma manera, analizando los resultados a partir de las imágenes de testeo podemos observar que para la prueba 4 tenemos un nivel de precisión es del 94%; con cero casos de falso positivos y cero de falsos negativos, como se detalla a continuación:

	Precisión (%)	FP (%)	FN (%)
Prueba 1	62	87	13
Prueba 2	38	74	26
Prueba 3	45	65	35
Prueba 4	94	0	0

Tabla 3. Resultados de las pruebas en diferentes escenarios.

Fuente: Elaboración propia

Por otro lado en el siguiente cuadro mostramos todas las iteraciones y pruebas que realizamos para ajustar nuestra red neuronal y encontrar el código más óptimo.

	Opcion 1	Opcion 2	Opcion 3	Opcion 4	Opcion 5	Opcion 6
Capas	1	1	1	2	2	3
Filtros	16	16	16	16, 32	16, 32	16,32,64
InitialLearnRate	0.1	0.001	0.0001	0.001	0.0001	0.01
Epocas	10	10	10	10	20	20
Presicion	15.63%	43.75%	99.99%	99.99%	94.00%	67.13%

Tabla 4*:* Resultados obtenidos al usar la versión elegida de red neuronal.

Fuente: Elaboración propia

Finalmente escogimos la opción 5 con un porcentaje de precisión del 94% y con cero Falsos positivos y negativos.

CONCLUSIONES

1) En el presente estudio se utilizó el Software Matlab y su Toolbox Deep Learning, tal como se explicó en la sección 3.3 y 3.5, para realizar el entrenamiento y aplicación de la red neuronal convolucional de aprendizaje profundo, en el reconocimiento facial de los estudiantes que accedían al Laboratorio de Control de la Universidad Ricardo Palma. También precisamos que realizamos dos procedimientos para la captura de imágenes, uno de ellos fue contar con una galería de fotográficas tomadas a los alumnos de la carrera de Ingeniería Mecatrónica y el segundo procedimiento consistió en la captura de fotografías en tiempo real haciendo uso de un dispositivo de captura.

2) Se validó en tiempo real el funcionamiento de la red neuronal de aprendizaje profundo para el reconocimiento facial de los estudiantes del Laboratorio de Control haciendo uso de un dispositivo de captura de imágenes a través de un algoritmo que permitió integrar dicho dispositivo, la red entrenada y la imagen capturada. Tener presente que el uso de un hardware computacional de hogar no fue el más adecuado, puesto que no se aprovechó todo el potencial que el Matlab puede entregar cuando se utiliza hardware computacional con procesador gráfico, el cual mejora considerablemente el tiempo de entrenamiento.

3) Fue posible establecer una base de datos formada por los rostros de los estudiantes de Ingeniería Mecatrónica que hicieron uso del Laboratorio de Control, la cual fue usada para el entrenamiento y validación de la red neuronal convolucional del presente estudio. Se debe precisar que al no contar con fotografías con fondo de diferentes lugares o contextos, la red neuronal únicamente aprendió lo que “se le mostraba”, por ello que en ocasiones durante el testeo de nueva información se generaron los falsos positivos. Podemos concluir que las redes neuronales de aprendizaje profundo necesitan de grandes volúmenes de información para el correcto entrenamiento de las imágenes.

RECOMENDACIONES

1) Se recomienda utilizar esta solución de reconocimiento facial para la seguridad al acceso a los laboratorios y áreas restringidas de la universidad Ricardo Palma.

2) Es importante contar con una mayor cantidad de fotografías de las personas en diferentes situaciones y lugares, de tal manera que la red neuronal pueda identificar con mayor precisión los patrones que forman las siluetas y características principales de las personas de la fotografías que vaya a entrenar.

3) Para un futuro estudio de la investigación presentada, se sugiere recopilar mayor cantidad de imágenes, probar con otras funciones de activación, incrementar las capas de convolución a mayor de cinco, probar el algoritmo en un hardware computacional que este acondicionado con tarjeta de video Nvidia para aprovechar su compatibilidad de procesamiento con el MATLAB, y de esa manera ahorrar tiempo durante el entrenamiento; finalmente cambiar el tamaño del filtro e imágenes de ingreso.

REFERENCIAS BIBLIOGRÁFICAS

5 Trabajos citados

Adam Gibson, Josh Patterson. (2017). *Deep Learning.* Estados Unidos: O'Reilly.

Agarap, A. (2019). *Arxiv.* Recuperado el 2019, de https://arxiv.org/pdf/1803.08375.pdf

Developers, Google. (2019). *Google*. Recuperado el 2019, de https://developers.google.com/machine-learning/crash-course/reducing-loss/stochastic-gradient-descent?hl=es-419

Haykin, S. (1999). *Neural Networks. A Comprehensive Foundation.* Canada: Prentice Hall.

Huamaní, P. (2008). *Simulación de redes neuronales con Matlab.* Lima: Universidad Ricardo Palma.

Ian Goodfellow, Yoshua Bengio, Aaron Courville. (2016). *Deep Learning (Adaptive Computation and Machine Learning Series).* Estados Unidos: MIT Press. Obtenido de https://www.deeplearningbook.org/contents/convnets.html

Kim, P. (2017). *Matlab Deep Learning.* Estados Unidos: Apress.

López, R. (2019). *Iaarbook.github*. Obtenido de https://iaarbook.github.io/deeplearning/

Mark Hudson, Martin Hagan, Howard Demuth. (2018). *Neural Network Toolbox User's Guide.* Estados Unidos: Mathworks.

Mathworks. (2018). *Mathworks.* Obtenido de https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/80879v00_Deep_Learning_ebook.pdf

Mathworks. (2019). *Mathworks*. Obtenido de https://la.mathworks.com/

MATLAB. (2019). *Producto MATLAB.* Obtenido de https://la.mathworks.com/products/matlab.html

Osnaya, M. (2016). *Estudio y análisis del sistema de reconocimiento facial aplicando redes neuronales y el software Matlab.* México D.F.: (Tesis de pregrado). Instituto Politécnico Nacional.

Picazo, O. (2018). *Redes neuronales convolucionales profundas para el reconocimiento de emociones en imágenes.* Madrid: (Tesis de postgrado). Escuela Técnica Superior de Ingenieros Informáticos.

Serra, X. (2017). *Face recognition using Deep.* Barcelona: (Tesis de postgrado). Universidad Politécnica de Cataluña.

Silva, I. J. (2018). *Reconocimiento Facial basado en redes neuronales convolucionales.* Sevilla: (Tesis de pregrado). Universidad de Sevilla.

Stan Li, Anil Jain. (2005). *Handbook of Face Recognition.* Estados Unidos: Springer.

Stanford University. (2019). *Deep Learning Stanford.* Obtenido de http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/

Szeliski, R. (2010). *SZELISKI.* Estados Unidos: Microsoft Research. Obtenido de http://szeliski.org/Book/

Team, S. D. (2018). *Super Datascience*. Obtenido de https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1b-relu-layer/

Vilardi, A. L. (2016). *VERY DEEP CONVOLUTIONAL NEURAL NETWORKS.* Escuela Técnica de Ingeniería de Telecomunicaciones. Barcelona: (Tesis de postgrado). Universidad Politécnica de Cataluña.

Yi, S. (2019). *DeepID3: Face Recognition with Very Deep Neural Networks.* Hong Kong : The Chinese University of Hong Kong .

ANEXOS

Algoritmo de la Red Neuronal para Reconocimiento Facial y Control de Acceso de Estudiantes a un Laboratorio de la Universidad Ricardo Palma:

```
% ---------------------------------------------------------


categories = {'Juana','Luis','Pedro','nestor','jc'};
rootFile = 'rostros';

% CONSULTA PARA TOMAR FOTOS Y AGREGARLAS AL DATA STORE
question_1 = '¿Desea tomar fotos desde la webcam y agregarlas al
aprendizaje? (s/n) ';
answer_1 = input(question_1,'s');

if strcmp(answer_1, 's')

    % CAPTURA DE IMAGENES POR WEBCAM
    cam=webcam;
    nombre = 'Ingrese su nombre: ';
    picture_name = input(nombre, 's');
    [status, msg, msgID] = mkdir (strcat(rootFile, '/', picture_name));

    if strcmp(msg, 'Directory already exists.')
        answer_2 = input('El directorio ya existe, ¿desea sobrescribirlo?
(s/n) ', 's');
        if strcmp(answer_2,'s')

            % ELIMINACION DE DIRECTORIO ANTERIOR
            [status, message, messageid] = rmdir(strcat(rootFile, '/',
picture_name), 's');
            % CREACION DEL NUEVO DIRECTORIO
            mkdir (strcat(rootFile, '/', picture_name));

            % TOMA DE FOTOS
            disp('SE TOMARAN LAS FOTOS EN: ')
            for i=3:-1:1
                pause(0.5)
                disp(['     ', num2str(i), '...'])
            end
            for i = 1:30
                img=snapshot(cam);
                imwrite( img , strcat(rootFile, '/', picture_name,
'/img', num2str(i)  ,'.jpg'));
                pause(0.2)
            end
            disp('FOTOS TOMADAS CON EXITO.')

            % ACTUALIZA LA LISTA DE CATEGORIAS
            categories = [categories picture_name];
        else
            disp('SE UTILIZARA EL DIRECTORIO EXISTENTE.')

            % ACTUALIZA LA LISTA DE CATEGORIAS
            categories = [categories picture_name];
```

```
        end
    else

        % TOMA DE FOTOS
        disp('SE TOMARAN LAS FOTOS EN: ')
        for i=3:-1:1
            pause(0.5)
            disp(['     ', num2str(i), '...'])
        end
        for i = 1:30
            img=snapshot(cam);
            imwrite( img , strcat(rootFile, '/', picture_name, '/img',
num2str(i)  ,'.jpg'));
            pause(0.2)
        end
        disp('FOTOS TOMADAS CON EXITO.')

        % ACTUALIZA LA LISTA DE CATEGORIAS
        categories = [categories picture_name];
    end
    clearvars cam
    % -------------------------------------
else
    disp('SE USARAN LAS CARPETAS SIGUIENTES: ')
    cant_var = size(categories);
    for i = 1:cant_var(2)
        disp(categories(i))
    end
    clearvars cant_var
end
clearvars question_1 answer_1
% ------------------------------------------------------------

% CREACION DE DATA STORE
[m, num_categories] = size(categories);
imds = imageDatastore(fullfile(rootFile,categories),...
    'LabelSource','foldernames');

inputSize = [128 128 3];

 [imgTrain,imgTest] = splitEachLabel(imds,0.8,'randomized');
augimgTrain = augmentedImageDatastore(inputSize,imgTrain);
%-----------------------

 % INGRESO DE CAPAS DE LA RED NEURONAL
 conv1 =
convolution2dLayer(4,16,'Stride',2,'Padding',2,'BiasLearnRateFactor',1);
 relu = reluLayer();
 maxPool = maxPooling2dLayer(2,'Stride',2);
 conv2 = convolution2dLayer(4,32,'Stride',2);
 relu2 = reluLayer();
 maxPool2 = maxPooling2dLayer(2,'Stride',2);
 %conv3 = convolution2dLayer(4,64,'Stride',2);
 %relu3 = reluLayer();
 %avg3 = averagePooling2dLayer(2,'Stride',2);
 drop = dropoutLayer(0.4);
 fc1 = fullyConnectedLayer(num_categories,'BiasLearnRateFactor',1);
 sm1 = softmaxLayer();
 class = classificationLayer();
```

```
 layers = [
    imageInputLayer([128 128 3])
    conv1
    relu
    maxPool
    conv2
    relu2
    maxPool2
    %conv3
    %relu3
    %avg3
    drop
    fc1
    sm1
    class];

% OPCIONES PARA EL ENTRENAMIENTO DE LA RED NEURONAL

    opts = trainingOptions('sgdm', ... % Utiliza el descenso de gradiente
estocastico conimpulso para el entrenamiento de la red
    'InitialLearnRate', 0.0001, ... % Rango inicial de aprendizaje
    'ValidationData',augimgTrain, ...
    'ValidationFrequency',1, ...
    'MaxEpochs', 20, ...    % Cantidad maxima de epoca
    'MiniBatchSize', 64, ...    % Mini lote con [x] observaciones en cada
iteracion
    'L2Regularization', 0.003, ...
    'Plot', 'training-progress','Verbose', true);

   % 'LearnRateSchedule', 'piecewise', ...  % Aprende la frecuencia de
aciertos y el input correcto

disp('ENTRENANDO RED...')

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

disp('RED ENTRENADA.')

% ------------ CONSULTAR SI DESEA TOMAR FOTO ------------

consulta = 'Desea tomar una foto para el testing? (s/n) : ';
respuesta = input(consulta, 's');
clearvars cam
if strcmp(respuesta, 's')

    % TOMAR FOTO CON LA WEBCAM Y GUARDARLA PARA EL TESTING
    cam=webcam;
    disp('LA FOTO SE TOMARA EN: ')
    for i=3:-1:1
        pause(0.5)
        disp(['     ', num2str(i), '...'])
    end
    img=snapshot(cam);
    imwrite( img , strcat('INPUT.jpg'));

    disp('FOTO TOMADA CON EXITO.')

    %TESTEAR LA IMAGEN
    fotoTest = imageDatastore(fullfile('INPUT.jpg'),...
```

```
    'LabelSource','foldernames');
    augfotoTest = augmentedImageDatastore(inputSize,fotoTest);
    [YPred,probs] = classify(net,augfotoTest);

    subplot(1,1,1)
    I = readimage(fotoTest,1);
    imshow(I)
    label = YPred(1);
    clearvars max
    [maximo,pos] = max(probs);

    % ASIGNACION DE COLOR AL TITULO DE LA IMAGEN SEGUN EL ACIERTO
    if  maximo >= 0.999
        colorText = 'g';
        title( string(label) + ", " + num2str((100*maximo),3) + "%",
'Color', colorText);
    else
        colorText = 'r';
        title("IMAGEN NO RECONOCIDA" , 'Color', colorText);
    end
    clearvars cam;
else

    % USO DEL DATA STORE DE TESTEO

    % Clasificacion de imagenes prueba y calculo de precision de la
clasificacion
    augimgTest = augmentedImageDatastore(inputSize,imgTest);
    [YPred,probs] = classify(net,augimgTest);

    % MUESTRA LA CANTIDAD DE IMAGENES DEL TESTEO
    [a, b] = size(imgTest.Files);
    for i = 1:(a)
        subplot(6,5,i)
        I = readimage(imgTest,i);
        imshow(I)
        title('Persona: N° ' + string(i));
    end
    %%%%%

    % Ingreso de imagen a probar
    prompt = 'Ingrese el número de imagen que desea probar: ';
    x = input(prompt);
    x = double(x);

    while( x > size(imgTest.Files) & x <= 0)
        x = input(prompt);
        x = double(x);
    end

    subplot(1,1,1)
    I = readimage(imgTest,x);
    imshow(I)
    label = YPred(x);
    clearvars max
    [maximo2,pos2] = max(probs(x,:));


    % ASIGNACION DE COLOR AL TITULO DE LA IMAGEN SEGUN EL ACIERTO
    % if YPred(x) == imgTest.Labels(x)
```

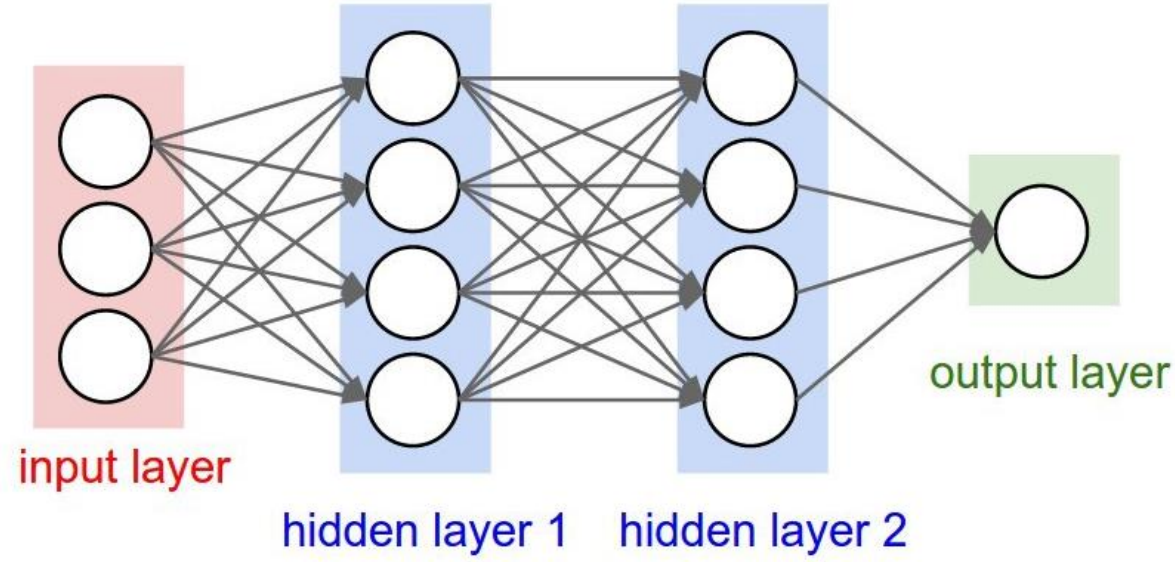
```
    if  maximo2 >= 0.999
        colorText = 'g';
        title(string(label) + ", " + num2str((100*maximo2),3) + "%",
'Color', colorText);
    else
        colorText = 'r';
        title("IMAGEN NO RECONOCIDA ", 'Color', colorText);
    end
end

% --------------------------------------------------------------
```

Redes Neuronales de Aprendizaje Profundo para principiantes

https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/

Veamos el cerebro de nuestra IA.

Como los animales, el cerebro de nuestro estimador AI tiene neuronas. Están representados por círculos. Estas neuronas están interconectadas.



Fuente: CS231n

Las neuronas se agrupan en tres tipos diferentes de capas:

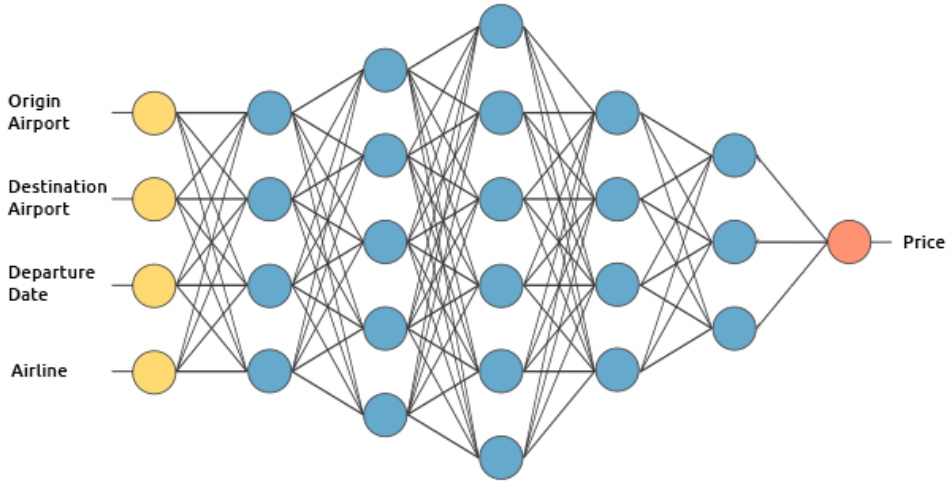
- Capa de entrada
- Capas ocultas
- Capa de salida

La **capa de entrada** recibe datos de entrada. En nuestro caso, tenemos cuatro neuronas en la capa de entrada: aeropuerto de origen, aeropuerto de destino, fecha de salida y aerolínea. La capa de entrada pasa las entradas a la primera capa oculta.

Las **capas ocultas** realizan cálculos matemáticos en nuestras entradas. Uno de los desafíos en la creación de redes neuronales es decidir la cantidad de capas ocultas, así como la cantidad de neuronas para cada capa.

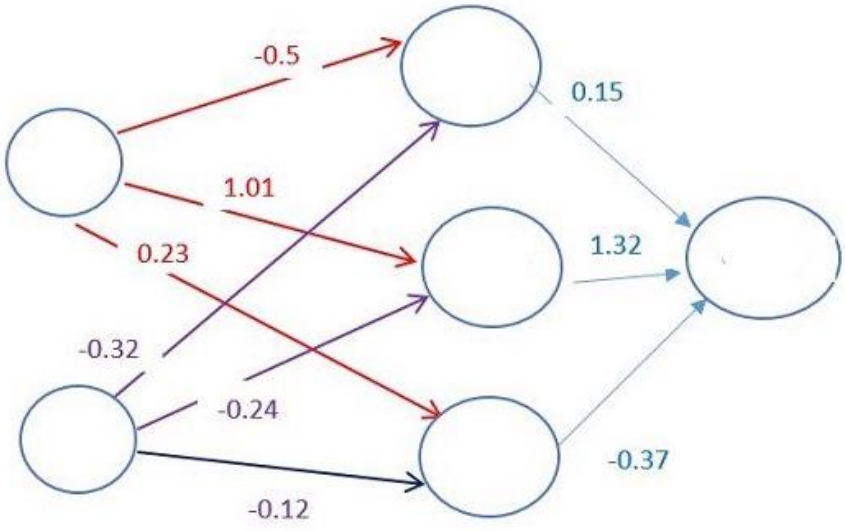
El "Deep" en Deep Learning se refiere a tener más de una capa oculta.

La **capa de salida** devuelve los datos de salida. En nuestro caso, nos da la predicción del precio.



Entonces, ¿cómo calcula la predicción de precios?
Aquí es donde comienza la magia del aprendizaje profundo.

Cada conexión entre las neuronas está asociada con un peso. Este peso dicta la importancia del valor de entrada. Los pesos iniciales se establecen al azar.
Al predecir el precio de un boleto de avión, la fecha de salida es uno de los factores más importantes. Por lo tanto, la fecha de salida de las conexiones neuronales tendrá un gran peso.



Fuente: CodeProject

Cada neurona tiene una función de activación. Estas funciones son difíciles de entender sin razonamiento matemático. En pocas palabras, uno de sus propósitos es "estandarizar" la salida de la neurona.
Una vez que un conjunto de datos de entrada ha pasado a través de todas las capas de la red neuronal, devuelve los datos de salida a través de la capa de salida.
Nada complicado, ¿verdad?

Entrenando un Red Neuronal

Entrenar a la IA es la parte más difícil de Deep Learning. ¿Por qué?

- Necesita un gran conjunto de datos.
- Necesita una gran cantidad de poder computacional.

Para nuestro estimador de precios de boletos de avión, necesitamos encontrar datos históricos de precios de boletos. Y debido a la gran cantidad de posibles aeropuertos y combinaciones de fechas de salida, necesitamos una lista muy grande de precios de boletos.

Para entrenar a la IA, necesitamos darle las entradas de nuestro conjunto de datos y comparar sus salidas con las salidas del conjunto de datos. Como la IA aún no está entrenada, sus resultados serán incorrectos.

Una vez que revisamos todo el conjunto de datos, podemos crear una función que nos muestre cuán equivocados estaban los resultados de la IA de los resultados reales. Esta función se llama la función de costo.

Idealmente, queremos que nuestra función de costo sea cero. Es entonces cuando las salidas de nuestra IA son las mismas que las salidas del conjunto de datos.

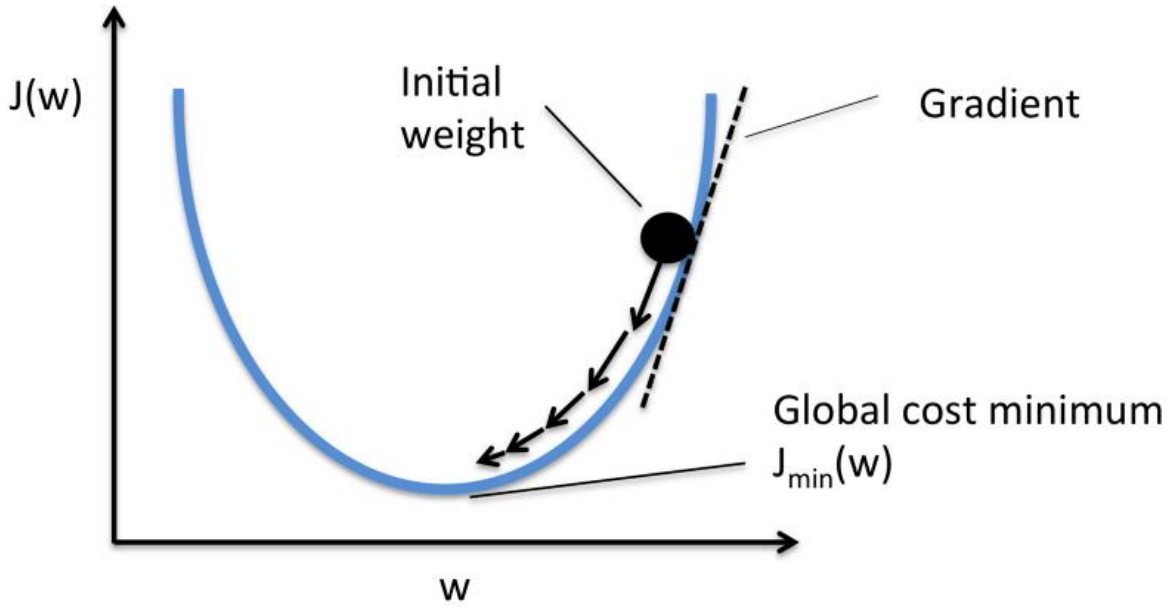
¿Cómo podemos reducir la función de costo?

Cambiamos los pesos entre las neuronas. Podríamos cambiarlos al azar hasta que nuestra función de costo sea baja, pero eso no es muy eficiente.

En su lugar, utilizaremos una técnica llamada Descenso de gradiente.

La pendiente de gradiente es una técnica que nos permite encontrar el mínimo de una función. En nuestro caso, estamos buscando el mínimo de la función de costo.

Funciona cambiando los pesos en pequeños incrementos después de cada iteración del conjunto de datos. Al calcular la derivada (o gradiente) de la función de costo en un determinado conjunto de peso, podemos ver en qué dirección es el mínimo.



Fuente: Sebastian Raschka

Para minimizar la función de costo, debe iterar a través de su conjunto de datos muchas veces. Es por eso que necesita una gran cantidad de potencia computacional.

La actualización de los pesos mediante el descenso de gradiente se realiza automáticamente. ¡Esa es la magia del aprendizaje profundo!

Una vez que hayamos capacitado a nuestro estimador de precios de boletos de avión AI, podemos usarlo para predecir precios futuros.

En resumen…

- Deep Learning utiliza una red neuronal para imitar la inteligencia animal.
- Hay tres tipos de capas de neuronas en una red neuronal: la capa de entrada, la (s) capa (s) oculta (s) y la capa de salida.
- Las conexiones entre las neuronas están asociadas con un peso, dictando la importancia del valor de entrada.
- Las neuronas aplican una función de activación en los datos para "estandarizar" la salida que sale de la neurona.
- Para entrenar una red neuronal, necesita un gran conjunto de datos.
- La iteración a través del conjunto de datos y la comparación de las salidas producirá una función de costo, que indica cuánto está apagada la IA de las salidas reales.
- Después de cada iteración a través del conjunto de datos, los pesos entre las neuronas se ajustan utilizando el Descenso de gradiente para reducir la función de costo.