UNIVERSIDAD POLITÉCNICA DE TULANCINGO

Aplicación móvil para la descripción de piezas de la zona arqueológica de Tula

Tesis

para obtener el título de

Maestro en desarrollo de sotfware

presenta

IVAN MAC GREGOR OVIEDO DORANTES

Asesores

Dr. César Joel Camacho Bello

Mtro. Jorge Alberto Hernández Tapia

Dedicatoria

El presente trabajo de investigación lo dedico a mis padres, por su amor, trabajo y sacrificio en todos estos años, gracias a ustedes he logrado llegar hasta aquí y convertirme en lo que soy. Ha sido el orgullo y un privilegio de ser su hijo, son los mejores padres.

A mi hermano y hermana por estar siempre presentes, acompañándome y por el apoyo moral, que me brindan a lo largo de esta etapa de mi vida.

Agradecemos a los docentes de la Universidad Politécnica de Tulancingo, por haber compartido sus conocimientos a lo largo de la preparación de nuestra esta maestría, de manera especial, al Dr. César Joel Camacho Bello y Jorge Alberto Hernández Tapia tutores de este proyecto de investigación quien me ha guiado con su paciencia, y su rectitud como docentes.

Índice general

Capítulo 1

Propósito y organización

Actualmente, las nuevas tecnologías han ayudado a que el acervo cultural esté al alcance de todos; ya sea para observar obras en imágenes de alta definición, ver vídeos en 360 grados de sitios arqueológicos, museos o galerías. También proporciona al usuario, la información sobre la obra y datos relevantes como la información del autor, fecha de creación, materiales utilizados, donde se ha expuesto la obra, a que cultura pertenecía y hasta el precio actual de la pieza.

1.1. Antecedentes

La forma mas sencilla de acceder a la información en la actualidad es por medio de los dispositivos móviles, de acuerdo a UIT [28] más de la mitad de la población mundial los usa. Por tal motivo, se realiza una inspección a las aplicaciones relacionadas con la difusión del acervo cultural. Entre las aplicaciones más utilizadas se encuentran:

Google Arts & Culture: Es un proyecto sin fines de lucro que en colaboración de museos de todo el mundo pretenden hacer más accesible el patrimonio cultural. Con la aplicación se puede ver vídeos en 360, visitas guiadas en realidad virtual y street view de lugares emblemáticos. Además, la aplicación sitúa museos cercanos. Desde su lanzamiento en 2011, año tras año se van sumando colecciones de diferentes museos. Desafortunadamen-

te, sólo está disponible en Estados Unidos y no parece que Google tenga intenciones de ampliarlo [22].

Magnus: proyecto realizado por el alemán Magnus Resch. Muestra una gama de galerías y museos principalmente de Estados Unidos y Europa. Por medio de una fotografía identifica el autor, el nombre de la obra y su valor; esta última tiene la característica de mostrar el costo en el mercado de la obra [39]. Su base de datos combina precios de galerías y subastas que datan desde 1900. Además, busca galerías y museos cercanos, así como horarios en que se encuentran abiertos. Otra de sus características es compartir obras por redes sociales, te permite crear una colección digital, buscar colecciones por artista y hacer zoom a las obras [1]. Esta aplicación tiene gran apoyo de los coleccionistas, quienes son sus principales inversionistas [64].

Smartify: reconoce las obras por medio de realidad aumentada, brinda información sobre el autor, referencias históricas, dimensiones. Además de información multimedia acerca de la misma, todo se limita a museos de Estados Unidos, Inglaterra, Francia, Italia y Rusia [45] [56].

Las aplicaciones mencionadas anteriormente presentan grandes avances en reconocimiento de imágenes e innovaciones tecnológicas. Todas estas nacen de la necesidad de difundir patrimonio cultural en cada una de las regiones en donde fueron desarrollados. La base de datos que utilizan son de museos o galerías fuera de México, por esta razón, no son capaces de reconocer gran parte del arte o patrimonio histórico de nuestro país. Otra de las limitaciones es el idioma, la mayoría de las aplicaciones son en inglés.

En este trabajo de tesis se pretende desarrollar una aplicación móvil de reconocimiento de patrones específicamente para realizar visitas guiadas en la zona arqueológica de Tula de Allende, es decir, por medio de un dispositivo móvil que brinde una reseña de las piezas expuestas y pirámides.

1.2. Justificación de la investigación

En este trabajo de tesis, se pretende acercar al turismo a la historia de la cultura tolteca, con esta aplicación los turistas conocerán las 5 piezas y pirámides más relevantes que hay en la Zona Arqueológica y Museo de la ciudad de Tula. Al captar con la cámara del dispositivo móvil la pieza muestra una reseña, esto hace que la aplicación sea muy fácil de usar.

1.3. Objetivo general

Desarrollar una aplicación móvil, robusta y versátil para el reconocimiento y descripción de monumentos históricos de la zona arqueológica de región de Tula de Allende Hidalgo.

1.4. Objetivos específicos

- Investigar sobre las piezas y pirámides que tiene el museo de las zona arqueológica de la región de Tula de Allende.
- Generar una base de datos de los monumentos que se encuentran en zona arqueológica de Tula de Allende.
- Revisión de conceptos de redes neuronales convolucionales para el reconocimiento de imágenes.
- Revisión de arquitecturas de LeNet y MobileNet.
- Revisión TensorFlow y Python
- Revisión de APIs de Android
- Revisión datos aumentados.
- Generar la aplicación en Android.

1.5. Alcances

- Realizar una aplicación en Android.
    - Realizar el reconocimiento de objetos prehispánicas del museo de Tula de Allende Hidalgo.
    - Dar una reseña del objeto prehispánico captado con la cámara del dispositivo móvil.
    - Mostrar información relevante; como horarios, costo y recomendaciones generales.

1.6. Organización de la tesis

Capítulo 1: En este capítulo se presenta una breve introducción al trabajo realizado, los motivos por el cual se realizo la investigación. Así como los objetivos que se pretenden alcanzar en la investigación.

Capítulo 2: Se presenta la teoría desde lo general como es el apréndase automático hasta llegar a las redes neuronales convoluciones. Conceptos fundamentales como red neuronal, red neuronal de una capa y multicapa. También, se analizan los fundamentos del aprendizaje profundo para clasificar monumentos históricos mediante imágenes.

Capítulo 3: Este capítulo esta dedicado a comparar las diferentes librerías relacionados con aprendizaje profundo, como plataformas en las que corren y lenguajes que se utilizan.

Capítulo 4: Este capítulo se explica la construcción de *DataSet*. Los plugin ocupados para la descarga masiva de imágenes, así como las librerías que se ocuparon para hacer el algoritmo de aumentó de datos.

Capítulo 5: En este capítulo se exponen los lenguajes de programación utilizados para la realización de la aplicación. Además, se exponen las APIs y como se ocupan en el construcción de la App.

Capítulo 2

Marco teórico

2.1. Fundamentos teóricos

En este capítulo se tratarán los conceptos fundamentales del aprendizaje profundo, red neuronal, red neuronal de una capa y multicapa. También, se analizan los fundamentos del aprendizaje profundo para clasificar monumentos históricos mediante imágenes.

El **Aprendizaje Automático** (AA) es una rama de la **Inteligencia Artificial** (IA) que permite que las computadoras aprendan de los datos sin estar explícitamente programadas, es decir, el objetivo del aprendizaje automático es diseñar métodos que automáticamente realicen el aprendizaje utilizando observaciones del mundo real llamados "datos de entrenamiento", todo esto sin una definición explícita de reglas o lógica por parte de los humanos. En ese sentido, el aprendizaje automático se puede considerar como programación mediante muestras de datos [35].

Existen, tres paradigmas de aprendizaje; **Aprendizaje Supervisado** (AS), **Aprendizaje no Supervisado** (ANS) y **Aprendizaje Reforzado** (AR) como se muestra en la Fig. 2.1 [55]. Sin embargo, el más usado es AS, debido a su rendimiento superior en comparación con otros aprendizajes [51]. En la Fig. 2.1 se muestra el universo de **IA** y la interacción con los diferentes tipos de aprendizaje.

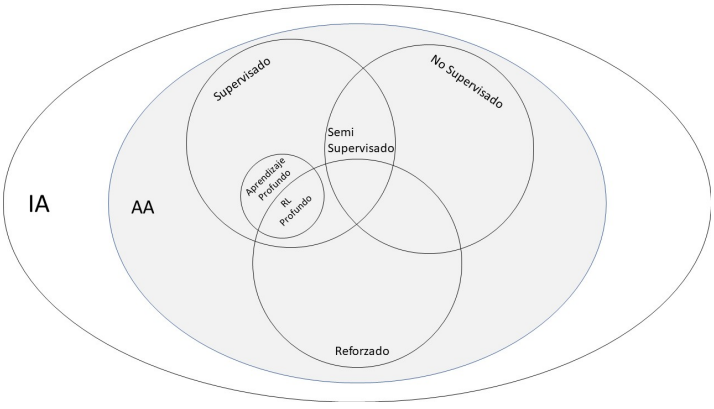


Figura 2.1: Clasificación de IA

El AS es muy similar al proceso de aprender de una persona, considerando que los humanos adquieren conocimientos al resolver problemas [35]. El conocimiento está dado por los datos de entrenamiento en donde toman la forma de una colección (x, y) donde x son los datos y y son las etiquetas. El objetivo es producir una predicción y en respuesta a una muestra de consulta x. La entrada x puede ser un vector de características o datos más complejos, como imágenes, documentos o gráficos. Del mismo modo, se han estudiado diferentes tipos de resultados de salida y, los cuales pueden ser una etiqueta binaria que se usa en un problema simple de clasificación binaria. También, se han realizado numerosos trabajos de investigación sobre problemas de la clasificación multiclase [51].

Existen diferentes formas de una colección (x, y) como son: los *árboles de decisión, bosques de decisión aleatoria* (RDF), *regresión logística* (LR), *máquinas de soporte vectorial* (SVM), *redes neuronales* (RN), *máquinas kernel* y *clasificadores bayesianos*. También, se ha propuesto una amplia gama de algoritmos de aprendizaje para estimar diferentes tipos de mapeos [51].

El ANS, es importante porque probablemente se asemeja más al cerebro humano. En este método, los datos de entrenamiento solo contienen entradas sin reglas correctas, se utiliza para fines de visualización, compresión o

eliminación de datos, o para comprender mejor las correlaciones presentes en los mismos [9].

Los métodos de aprendizaje semi-supervisados (ASS) se ubican entre AS y ANS. Estos métodos de aprendizaje se utilizan cuando hay una gran cantidad de datos de entrada disponibles y solo algunos de los datos están etiquetados. Un buen ejemplo es un archivo fotográfico donde solo algunas de las imágenes están etiquetadas (por ejemplo, perro, gato, persona) y la mayoría no están etiquetadas [51].

El AR es un enfoque de inteligencia artificial que enfatiza el aprendizaje del sistema a través de sus interacciones con el medio ambiente. Con el aprendizaje de refuerzo, el sistema adapta sus parámetros en función de los comentarios recibidos del entorno, que luego proporciona retroalimentación sobre las decisiones tomadas. Por ejemplo, un sistema que modela a un jugador de ajedrez, el cual usa el resultado de los pasos anteriores para mejorar su rendimiento como un sistema de aprendizaje de refuerzo. La investigación actual sobre aprendizaje con refuerzo es altamente interdisciplinaria e incluye investigadores especializados en algoritmos genéticos, redes neuronales, psicología e ingeniería de control [21].

Las Redes Neuronales Artificiales (RNA) son un subcampo del AA que a su vez es un subcampo de la IA, es decir, RNA en una implementación de los modelos de AA, se han desarrollado como generalizaciones de modelos matemáticos de las neuronas biológicas. Las RNAs tratan de emular el comportamiento del cerebro, caracterizado por el aprendizaje a través de la experiencia y la extracción de conocimiento genérico a partir de un conjunto de datos. Cada vez que aprendemos algo, nuestro cerebro almacena el conocimiento, análogamente, la computadora usa memoria para almacenar información. Aunque ambos almacenan información, los mecanismos son muy diferentes. La computadora almacena información en ubicaciones específicas de la memoria, mientras que el cerebro altera la asociación de neuronas. Una neurona artificial en sí no tiene capacidad de almacenamiento; simplemente transmite señales de una neurona a la otra, mientras que el cerebro es una red gigantesca de neuronas [44].

2.1.1. Conceptos de Redes Neuronales Artificiales

Las RNA (también llamadas perceptrones) fueron motivadas por sistemas biológicos, como la neurona McCulloch-Pitts et. al [69], la mayoría fueron diseñadas como simples procedimientos computacionales con poca relevancia biológica directa. La red neuronal artificial esta construida por un conjunto de conecciones de nodos, los cuales son elementos que corresponden a las neuronas de un cerebro. La Fig. 2.2 considera un nodo que recibe tres entradas, donde x_1, x_2, x_3 son las señales de entrada y w_1, w_2, w_3 son los peso de las conecciones. Por último, b es el sesgo, que es otro factor asociado con el almacenamiento de información. Además, el circulo y la flecha denota el nodo y el flujo de la señal respectivamente.

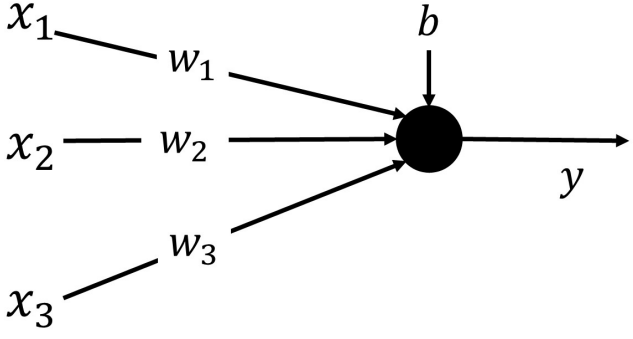


Figura 2.2: Red Neuronal Artificial

Para determinar la salida de una neurona se realizan tres tipos de operaciones; función de propagación, función de activación y función de salida como se muestra en la Fig. 2.3.

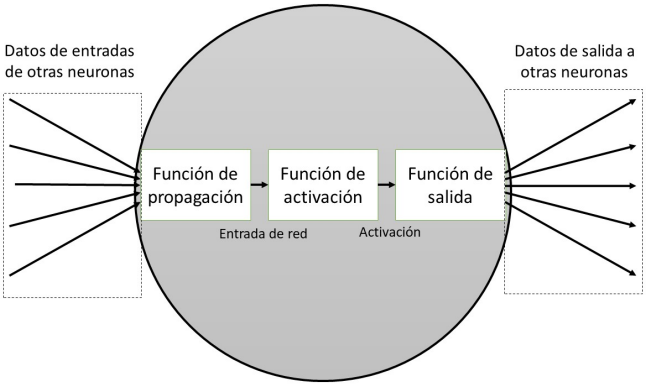


Figura 2.3: Procesamiento de los datos de una red neuronal

Función de Propagación

La función de propagación es la suma ponderada, la cual multiplica las señales de entrada x_j por los pesos w_j, mas b [35], expresada de la siguiente manera,

$$v = \sum_i (w_i x_i + b) \tag{2.1.1}$$

Por tanto, la suma ponderada para la Fig. 2.2 se calcula de la siguiente manera:

$$v = (w_1 \cdot x_1) + (w_2 \cdot x_2) + (w_3 \cdot x_3) + b$$

También, la Ec. 2.1.1 puede ser escrita de forma matricial,

$$v = w \cdot x + b \tag{2.1.2}$$

donde w y x se definen como,

$$w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$$

$$y = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Función de Activación

La función de activación determina el comportamiento del nodo [35], definida de la siguiente manera,

$$y = \varphi(v) = \varphi(w \cdot x + b). \tag{2.1.3}$$

donde $\varphi(\cdot)$ es la función de activación. Existen diferentes tipos de funciones de activación que pueden usarse en redes neuronales. Por lo general se utilizan funciones no lineales y continuamente diferenciables, como se muestra en la Tab. 2.1. Una función de activación no lineal hace posible que una red neuronal aprenda siempre que la red tenga suficientes neuronas y capas. La propiedad de diferenciabilidad también es importante ya que principalmente entrenamos una red neuronal utilizando el método de descenso del gradiente.

Función de Salida

La función de salida de una neurona j calcula los valores que se transfieren a las otras neuronas conectadas a j. Se define de la siguiente manera [35],

$$f_{out}(a_j) = o_j. \tag{2.1.4}$$

En general, la función de salida se define globalmente. A menudo, esta función es la identidad, es decir,

$$f_{out}(a_j) = a_j \ \text{ ó } o_j = a_j \tag{2.1.5}$$

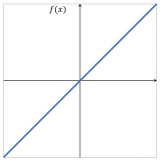
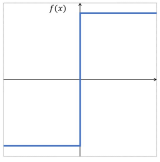
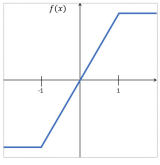
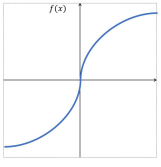
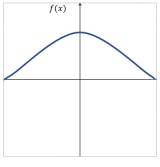
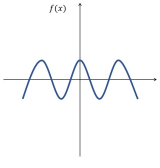
	Función	Gráfica
Identidad	$\varphi(x) = x$	
Escalón	$\varphi(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x \leq 0 \end{cases}$	
Lineal a Tramos	$\varphi(x) = \begin{cases} 1, & \text{si } x < 1 \\ x, & \text{si } -1 \leq x \leq 1 \\ -1, & \text{si } x < -1 \end{cases}$	
Sigmoidea	$\varphi(x) = \frac{1}{1+e^{-x}} = \tanh x$	
Gaussiana	$\varphi(x) = Ae^{-bx^2}$	
Sinusoidal	$\varphi(x) = A * \sin(wx + \varphi)$	

Tabla 2.1: Funciones de activación

2.1.2. Reglas de aprendizaje

Regla delta

La regla delta fue propuesta por Widrow et al. en 1960 [5], la cual actualiza los parámetros de red en función de la diferencia entre la salida de real y la salida predicha. Esta regla trata de aprender de los errores de la red neuronal durante la fase de entrenamiento [51]. Las unidades de salida son una función lineal de las entradas denotadas por x, es decir,

$$p_i = \sum_j (w_{ij} \cdot x_j) \tag{2.1.6}$$

donde w_{ij} son los peso de las conecciones, p_i denota la salida predicha y y_i la salida real. El error puede ser calculado de la siguiente manera,

$$E = \frac{1}{2} \sum_i (y_i - p_i)^2 \tag{2.1.7}$$

donde i es el número de categorías. La regla delta calcula el gradiente de la función de la Eq. 2.1.7 con respecto a los parámetros de la red. Por lo tanto, el gradiente de los pesos w se obtiene de la siguiente manera,

$$w_{ij}^{t+1} = w_{ij}^t + \eta \frac{\partial E}{\partial w_{ij}} \tag{2.1.8}$$

$$w_{ij}^{t+1} = w_{ij}^t + \eta \cdot (y_i - p_i) \cdot x_i \tag{2.1.9}$$

donde t denota la operación previa del proceso de aprendizaje, η es el tamaño de paso de la actualización del parámetro en dirección al gradiente. Los parámetros se actualizan de manera que las salidas predichas se acerquen más a las salidas reales. Después de varias iteraciones, se dice que el proceso de entrenamiento de la red converge cuando los parámetros ya no cambian como resultado de la actualización.

Regla delta generalizada

La regla delta generalizada fue propuesta por Rumelhart et al. en 1985 [51]. La regla delta solo calcula combinaciones lineales entre los pares

de entrada y salida. Para superar esta limitación, la regla delta generalizada utiliza funciones de activación no lineales en cada unidad de procesamiento para modelar relaciones no lineales entre los dominios de entrada y salida. También nos permite hacer uso de múltiples capas ocultas en la arquitectura de la red neuronal, un concepto que forma el corazón del aprendizaje profundo. A partir de la Ec. 2.1.7 el gradiente respecto a los parámetros en la capa de salida L para cada neurona i esta dada por,

$$\frac{\partial E}{\partial w_{ij}^{L}} = \delta_i^L x_j \tag{2.1.10}$$

$$\delta_i^L = (y_i - p_i) \cdot f_i'(a_i) \tag{2.1.11}$$

donde $a_i = \sum_j (w_{ij} x_j + b_j)$ es la función de propagación, los x_j son las salidas de las capas previas, $p_i = f(a_i)$ son las salidas de la neurona (predicción de la capa de salida) y $f(\cdot)$ denota a la función de activación no lineal y $f'(\cdot)$ la derivada de esta [51]. La función de activación decide si la neurona se activa o no, en respuesta a una activación de entrada determinada, dada de la siguiente manera

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta \cdot \delta_i^L \cdot x_j \tag{2.1.12}$$

donde t denota la operación previa del proceso de aprendizaje y η es el tamaño del descenso.

Ejemplos de la regla delta y regla delta generalizada en una red neuronal simple.

Consideremos una red neuronal simple de tres neuronas de entradas y una de salida como se muestra en la Fig. 2.4. La función de activación es la función Sigmoidea.

Algoritmo 1 Regla delta generalizada con función Sigmoidea

Entrada:

$x = \{x_1, x_2, \ldots, x_n\}$ donde x_i son los datos de entrada.

$w = \{w_1, w_2, \ldots, w_m\}$ donde w_j son los pesos de las conecciones.

$y = \{y_1, y_2, \ldots, y_n\}$ donde y_i son los valores correctos de cada x_i.

Salida:

$w = \{w_1, w_2, \ldots, w_k\}$ pesos de red entrenada.

1: **while** *iteraciones* **do**
2: **for** $i = 0$ hasta n **do**
3: $v = w \cdot x_i$
4: $p = \frac{1}{1+e^{-v}}$
5: $\delta = (y_i - p) \cdot p \cdot (1 - p)$
6: $w = w + \eta \cdot \delta \cdot x_i$
7: **end for**
8: **end while**

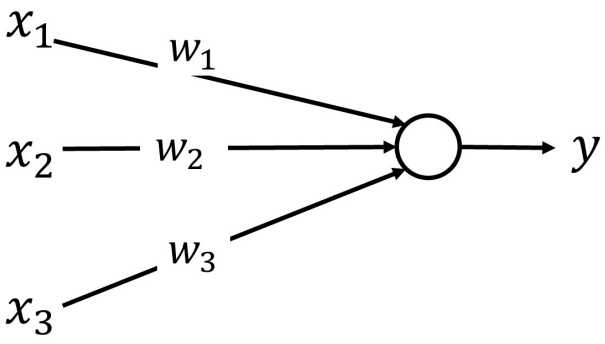


Figura 2.4: Red neuronal de 3 neuronas de entrada y una de salida

Sea x la matriz de los datos de entrada en donde las filas representan los datos de entrada, w representa el vector de pesos, y y el vector de salida con los valores correctos.

$$x = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, w = \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

iteración 1: Primero se obtiene a.

$$\begin{aligned} a &= \sum_j w_j \cdot x_{ij} \\ &= w \cdot x_{fila-uno} \\ &= \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= 0.5 \end{aligned}$$

posteriormente se calcula $p = f(a) = \frac{1}{1+e^{-a}}$

$$p = f(0.5) = 0.6225$$

obsérvese que $f^{'}(a) = f(a) \cdot (1 - f(a))$, por lo tanto, de la ecuación 2.1.11 se tiene,

$$\begin{aligned}
\delta &= (y_1 - p) \cdot f(a) \cdot (1 - f(a)) \\
&= (0 - 0.6225)(0.6225)(1 - 0.6225) \\
&= -0.1463
\end{aligned}$$

donde y_1 es la primer entrada de y. Luego entonces, w^{t+1} se calcula

$$\begin{aligned}
w^{t+1} &= w^t + \eta \cdot \delta \cdot x_{fila-uno} \\
&= \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix} + (0.9) \cdot (-0.1463) \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.5 & 0.5 & 0.3683 \end{bmatrix}
\end{aligned}$$

análogamente para las demás filas de x se tiene que

$$\begin{aligned}
a &= w \cdot x_{fila-dos} \\
&= \begin{bmatrix} 0.5 & 0.5 & 0.3683 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \\
&= 0.8683 \\
p &= f(0.8683) \\
&= 0.7044 \\
\delta &= (0 - 0.7044)(0.7044)(1 - 0.7044) \\
&= -0,1467 \\
w^{t+1} &= w^t + \eta \cdot \delta \cdot x_{fila-dos} \\
&= \begin{bmatrix} 0.5 & 0.5 & 0.3683 \end{bmatrix} + (0.9) \cdot (-0.1467) \cdot \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.5 & 0.3680 & 0.2363 \end{bmatrix}
\end{aligned}$$

$$
\begin{aligned}
a &= w \cdot x_{fila-tres} \\
&= \begin{bmatrix} 0.5 & 0.3680 & 0.2363 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \\
&= 0.7367 \\
p &= f(0.7367) \\
&= 0.6763 \\
\delta &= (0 - 0.6763)(0.6763)(1 - 0.6763) \\
&= 0.0709 \\
w^{t+1} &= w^t + \eta \cdot \delta \cdot x_{fila-tres} \\
&= \begin{bmatrix} 0.5 & 0.53680 & 0.2363 \end{bmatrix} + (0.9) \cdot (0.0709) \cdot \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.5638 & 0.3680 & 0.3002 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
a &= w \cdot x_{fila-cuatro} \\
&= \begin{bmatrix} 0.5638 & 0.3680 & 0.3002 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\
&= 0.8417 \\
p &= f(0.8417) \\
&= 0.6988 \\
\delta &= (0 - 0.6988)(0.6988)(1 - 0.6988) \\
&= 0.0634 \\
w^{t+1} &= w^t + \eta \cdot \delta \cdot x_{fila-cuatro} \\
&= \begin{bmatrix} 0.5638 & 0.3680 & 0.3002 \end{bmatrix} + (0.9) \cdot (-0.0395) \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.5993 & 0.4035 & 0.3357 \end{bmatrix}
\end{aligned}
$$

obteniendo como salida predicha y para la primera iteración como;

$$y = \begin{bmatrix} 0.5831 \\ 0.6768 \\ 0.7181 \\ 0.7923 \end{bmatrix}$$

iteración 2

$$y = \begin{bmatrix} 0.5464 \\ 0.6224 \\ 0.7102 \\ 0.7703 \end{bmatrix}$$

iteración 10

$$y = \begin{bmatrix} 0.3776 \\ 0.3636 \\ 0.7581 \\ 0.7469 \end{bmatrix}$$

iteración 1000

$$y = \begin{bmatrix} 0.0337 \\ 0.0272 \\ 0.9780 \\ 0.9727 \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Es posible observar que entre más iteraciones, la salida converge más al valor real.

Para el siguiente ejemplo se hará un pequeño cambio en los valores de salida del ejemplo anterior, llamado XOR. Sea x la siguiente matriz, donde las columnas son las neuronas de la capa de entrada, w el vector de pesos, y y el vector de salida de valores correctos.

$$x = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, w = \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

iteración 1:

$$y = \begin{bmatrix} 0.5831 \\ 0.6768 \\ 0.7181 \\ 0.7923 \end{bmatrix}$$

iteración 2:

$$y = \begin{bmatrix} 0.5637 \\ 0.6589 \\ 0.6572 \\ 0.7413 \end{bmatrix}$$

iteración 10:

$$y = \begin{bmatrix} 0.4684 \\ 0.5088 \\ 0.4973 \\ 0.5377 \end{bmatrix}$$

iteración 1000:

$$y = \begin{bmatrix} 0.5297 \\ 0.5000 \\ 0.4703 \\ 0.4409 \end{bmatrix} \quad \neq \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Al entrenar la red neuronal por más tiempo no existe diferencia, la Fig 2.5 ilustra el concepto del problema. Donde a los tres datos de entrada se representan como coordenadas XYZ. La coordenada Z se fija en 1, X y Y se visualiza en el plano como se muestra en la Fig 2.5

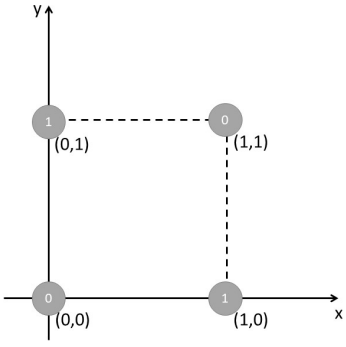


Figura 2.5: Interpretación de valores de los datos de entrada X Y y Z

Los valores 0 y 1 en los círculos son las salidas reales. La forma de dividir en regiones de ceros y unos con una línea recta no es posible. La división se realiza con una curva como se muestra en la fig. 2.6

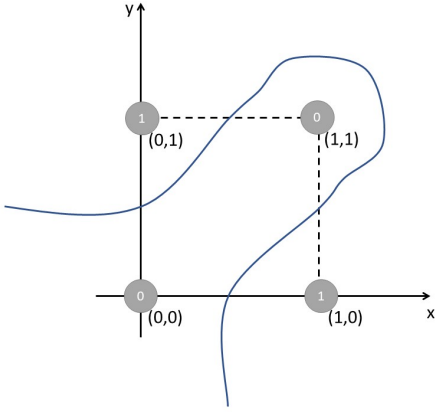


Figura 2.6: Separación de regiones de 0 y 1 por una curva

Para el ejemplo uno representamos a X Y en el plano como en la fig. 2.7

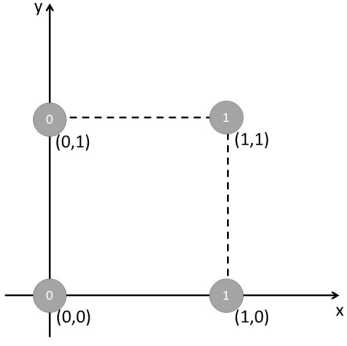


Figura 2.7: Datos de aprendizaje por regla delta

En este caso podemos hacer la división de las regiones de 0 y 1 muy fácil. Esto es un problema linealmente separable y lo mostramos en la Fig 2.8.

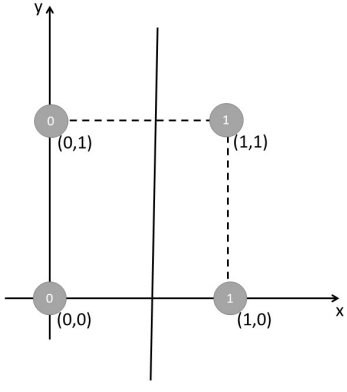


Figura 2.8: Datos de aprendizaje por regla delta

En conclusión, una red neuronal de una sola capa solo puede resolver problemas linealmente separables. Esto se debe a que la red neuronal de una sola capa es un modelo que divide linealmente las regiones de datos de entrada. Para superar esta limitación de la red neuronal de una sola capa, es necesario incrementar el número de capas en la red. Esta necesidad ha llevado a la aparición de la red neuronal multicapa, que puede lograr lo que la red neuronal de una sola capa no puede lograr. Se debe tener en cuenta que la red neuronal de una sola capa es aplicable para tipos de problemas específicos. La red neuronal multicapa no tiene tales limitaciones.

2.1.3. Redes Neuronales Multicapa

La red neuronal se ha desarrollado desde una arquitectura simple a una estructura cada vez más compleja. Inicialmente, las redes neuronales tenían una arquitectura muy simple con solo capas de entrada y salida, que se denominan redes neuronales de **una sola capa**. Cuando se agregan capas ocultas a una red neuronal de una sola capa, esto produce una **red neuronal multicapa**. Por lo tanto, la red neuronal multicapa consta de una capa de entrada, una capa o capas ocultas y una capa de salida. La red neuronal que tiene una sola capa oculta se llama **red neuronal superficial**. Una red neuronal multicapa que contiene dos o más capas ocultas se denomina **red neuronal profunda**. La mayoría de las redes neuronales contemporáneas utilizadas en aplicaciones prácticas son **redes neuronales profundas**. La figura 2.9 resume las ramas de la red neuronal en función de la arquitectura de capa [35].

Las redes neuronales con más de una capa de pesos entre las conexiones (w) son referidas como **redes neuronales multicapa** (RNM). Una RNA con n capas tiene exactamente n capas de pesos y $n+1$ capas de neuronas. [37]. En términos más simples, la red se puede tratar como una caja negra, que opera en un conjunto de entradas y genera salidas.

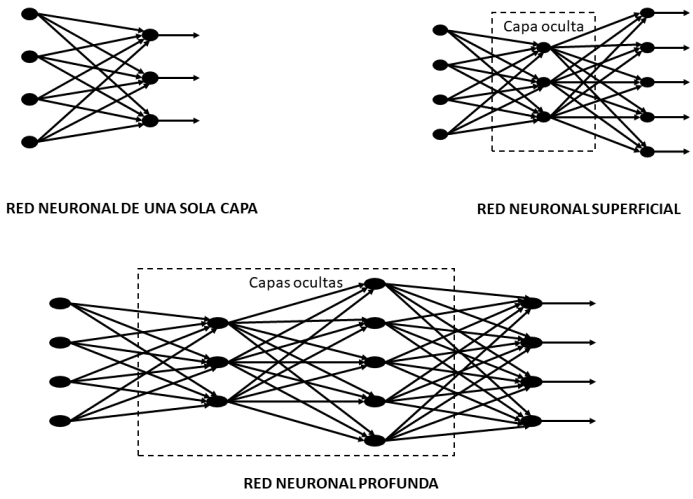


Figura 2.9: Clasificación de redes neuronales dependiendo de la arquitectura en las capas

Redes de propagación hacia atrás (backpropation)

El algoritmo de propagación hacia atrás (PA) ha sido el método principal para lograr el aprendizaje de los parámetros de una RNM. La idea central del algoritmo de PA para el aprendizaje de los pesos se basa en gradientes-decentes, para establecer una conexión entre la pérdida calculada en la capa de salida y los nodos ocultos. [66]. Esta conexión ayuda a retransmitir la pérdida final de la red a las capas anteriores en la red para que los pesos de esas capas puedan ajustarse proporcionalmente.

Se explica el algoritmo de propagación hacia atrás utilizando un ejemplo de la red neuronal multicapa simple [35]. Considere una red neuronal que consta de dos nodos para la entrada y la salida, una capa oculta que también tiene dos nodos y se omitirá el sesgo por conveniencia. El ejemplo de red neuronal se muestra en la Fig. 2.10, donde el superíndice describe el indicador de capa.

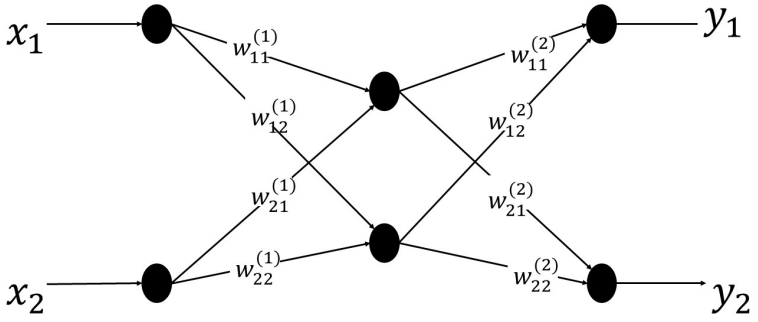


Figura 2.10: Red neuronal de dos nodos para la capa de entrada, la capa de salida y la capa oculta.

Es necesario la salida de la red neuronal de los datos de entrada para calcular la suma ponderada del nodo oculto como se muestra a continuación:

$$v^{(1)} = W_1 \cdot x \tag{2.1.13}$$

$$\begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Al calcular la suma ponderada, Ecu. 2.1.13, en la función de activación, se obtiene la salida de los nodos ocultos.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \end{bmatrix} = \begin{bmatrix} \varphi(v_1^{(1)}) \\ \varphi(v_2^{(1)}) \end{bmatrix}$$

donde $y_1^{(1)}$ y $y_2^{(1)}$ son las salidas de los nodos ocultos. De manera similar, la suma ponderada de los nodos de salida se calcula como:

$$v = W_2 \cdot y^{(1)} \tag{2.1.14}$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} \cdot \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \end{bmatrix}$$

cuando se coloca esta suma ponderada en la función de activación, la red neuronal genera el siguiente resultado.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \varphi(v_1) \\ \varphi(v_2) \end{bmatrix}$$

A partir que aquí se explica el análisis del algoritmo de propagación hacia atrás utilizando el ejemplo. En la Fig. 2.11. se ha rediseñado con las conexiones innecesarias para explicarlo.

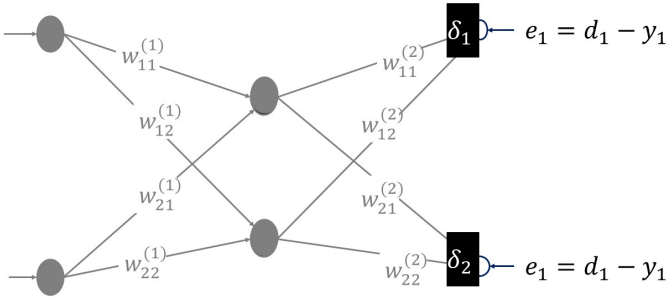


Figura 2.11: Algoritmo de propagación hacia atrás.

Sea

$$\begin{aligned} e_1 &= d_1 - y_1 \\ \delta_1 &= \varphi'(v_1) \cdot e_1 \\ \\ e_2 &= d_2 - y_2 \\ \delta_2 &= \varphi'(v_2) \cdot e_2 \end{aligned} \tag{2.1.15}$$

donde $\varphi'(\cdot)$ es la derivada de la función de activación del nodo de entrada, y_i es la salida del nodo de salida, d_i es la salida correcta de los datos de entrenamiento, y v_i es la suma ponderada de los correspondientes nodos.

Obteniendo los delta para cada nodo de salida, Se calcula el delta de los nodos ocultos como se muestra en la Fig. 2.12.

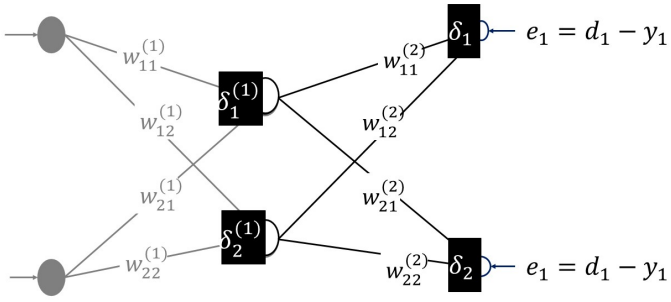


Figura 2.12: Calculo el delta de los nodos ocultos

En el algoritmo de propagación hacia atrás, el error del nodo se define como la suma ponderada de los deltas propagados hacia atrás desde la capa de la derecha inmediata en este caso la capa de salida.

$$\begin{aligned} e_1^{(1)} &= w_{11}^{(2)}\delta_1 + w_{21}^{(2)}\delta_2 \\ \delta_1^{(1)} &= \varphi'(v_1^{(1)}) \cdot e_1^{(1)} \\ \\ e_2^{(1)} &= w_{12}^{(2)}\delta_1 + w_{22}^{(2)}\delta_2 \\ \delta_2^{(1)} &= \varphi'(v_2^{(1)}) \cdot e_2^{(1)} \end{aligned} \tag{2.1.16}$$

donde $v_1^{(1)}$ y $v_2^{(1)}$ son la suma ponderada de los modos respectivos. En la ecuacion 2.1.16 los procesos hacia adelante y hacia atrás se aplican de manera idéntica a los nodos ocultos, así como a los nodos de salida. Esto implica que la salida y los nodos ocultos experimentan el mismo proceso hacia atrás. como se observa en la Fig. 2.13

Figura 2.13: Calculo del error

Si existen capas ocultas adicionales, se repite el proceso anterior para cada capa oculta y se calcula todos los deltas. Una vez que se hayan calculado todos los deltas, se utiliza la siguiente ecuación para ajustar los pesos de las capas respectivas.

$$\begin{aligned} \Delta w_{ij} &= \alpha \delta_i x_j \\ w_{ij}^k &\leftarrow w_{ij} + \Delta w_{ij} \end{aligned} \tag{2.1.17}$$

donde x_i es la señal de entrada para el peso correspondiente, el ajuste es para cada capa de la red.

2.1.4. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (ConvNets o RNC) son una categoría de redes neuronales que han demostrado su eficacia en áreas como el reconocimiento y la clasificación de imágenes. RNC ha tenido éxito en la identificación de rostros, objetos y señales de tráfico, además de potenciar la visión en robots y autos de conducción automática. De igual forma han sido empleadas en varias tareas de procesamiento del lenguaje natural como la clasificación de oraciones [32].

La Fig. 2.14 muestra como una RNC puede reconocer escenas y el sistema puede sugerir títulos relevantes como *un vendedor de paletas* mientras que la Fig. 2.15 muestra un ejemplo de RNC que se usa para reconocer objetos cotidianos, humanos y animales [32].

(a) Un vendedor de paletas (b) Personas trabajando

Figura 2.14: Reconocimiento de escenas

Figura 2.15: Reconocimiento de Objetos cotidianos; humanos, animales, automóviles, etc.

Arquitectura LeNet

La arquitectura LeNet de LeCun se introduce en la década de los 90s estaba diseñada para aprender a reconocer imágenes. Se han propuesto varias arquitecturas nuevas en los últimos años, que son mejoras con respecto a LeNet. LeNet se usaba principalmente para tareas de reconocimiento de caracteres como la lectura de códigos postales, dígitos, etc [14].

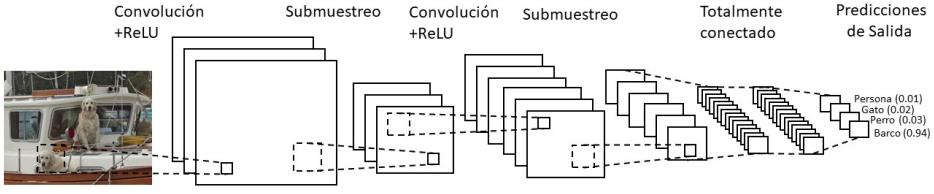


Figura 2.16: Arquitectura LeNet

Como se muestra en la Fig 2.16 la arquitectura LeNet es entrenada para clasifica una imagen que tiene como entrada cuatro categorías: perro, gato, barco o persona. LeNet recibe una imagen de un barco como entrada, la red asigna correctamente la mayor probabilidad al barco (0,94) entre las cuatro categorías. La suma de todas las probabilidades de las capas de salida debe ser uno [14].

LeNet tiene cuatro operaciones principales en la RNC que se muestran en la Fig 2.16

1. Convolución.
2. No linealidad (ReLU).
3. Agrupación o submuestreo.
4. Clasificación (Capa Totalmente Conectada).

Convolución.

La Convolución en RNC es extraer características de la imagen de entrada. La convolución preserva la relación espacial entre los píxeles al aprender las características de la imagen utilizando pequeños cuadrados de datos de entrada. Consideremos una imagen de 5 x 5 cuyos valores de píxel son solo 0 y 1 como se muestra en la Fig. 2.17, recordemos que una imagen puede considerarse como una matriz de valores de píxeles [32].

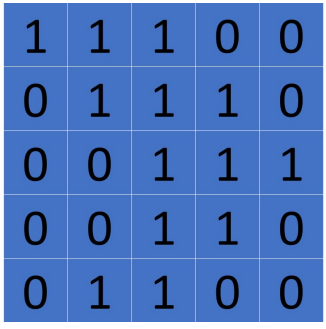


Figura 2.17: Imagen, matriz de 5x5

En la Fig. 2.18 se tiene una matriz 3 x 3 denomina filtro, kernel o detector de características.

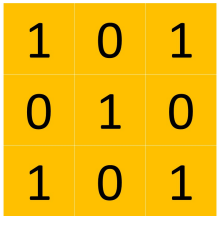


Figura 2.18: Filtro, matriz de 3x3

Para hacer el calculo de la convolución se hace una multiplicación de matrices, seccionando a la matriz de 5x5 a una de matriz 3x3 como se muestra en la Fig. 2.19, para poder realizar el producto matricial. A la matriz de salida se le denomina mapa de características [31].

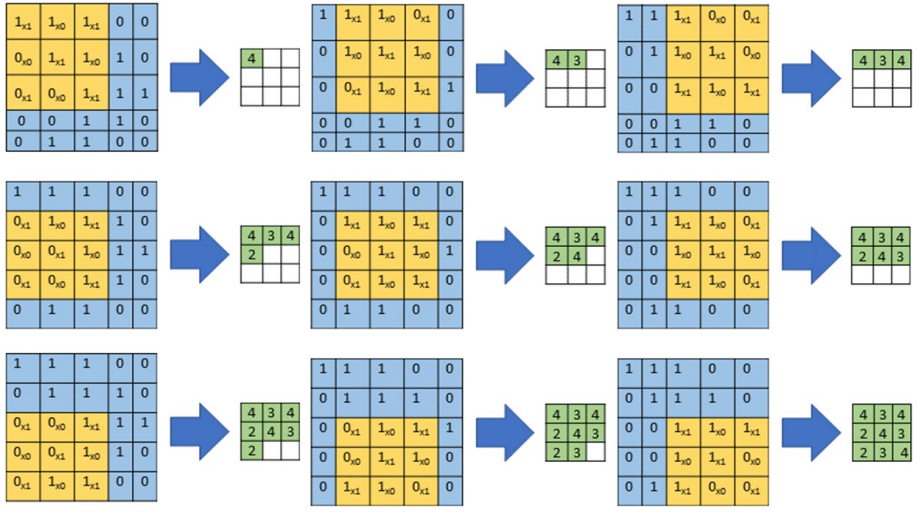


Figura 2.19: Convolución de la imagen

Es importante tener en cuenta que los filtros actúan como detectores de características de la imagen de entrada original, los efectos de la convolución se muestran el la Tab. 2.2.

Es posible realizar operaciones como detección de bordes, enfocar y difuminar simplemente cambiando los valores numéricos de la matriz de filtros antes de la operación de convolución; esto significa que diferentes filtros pueden detectar diferentes características de una imagen, por ejemplo, bordes, curvas, etc [32], como se muestra en la Tabla 2.2.

Las RNC aprenden los valores de estos filtros por sí mismas durante el proceso de entrenamiento con parámetros como el número de filtros, el tamaño del filtro, la arquitectura de la red, etc., antes del proceso de entrenamiento. Cuantos más filtros se tengan, más funciones de imagen se extraerán y mejor será la red para reconocer patrones en imágenes invisibles [32].

Operación	Filtro	Imagen Involucionada
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Detección de bordes	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ -1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Tabla 2.2: Funciones de activación

El tamaño del mapa de características se controla mediante tres parámetros que se debe decidir antes de realizar el paso de convolución [31]:

1. **Profundidad**: la profundidad corresponde al número de filtros que se utilizan para la operación de convolución. En la red de la Fig 2.16, se realiza la convolución de la imagen original del barco utilizando tres filtros distintos, produciendo así tres mapas de características diferentes. Se analizan a estos tres mapas de características como matrices 2d apiladas, por lo tanto, la profundidad del mapa de características sería tres.

2. **Stride**: es el número de píxeles por los que se desliza la matriz de filtro sobre la matriz de entrada. Cuando la zancada es 1, se trasladan los filtros un píxel a la vez. Cuando la zancada es 2, los filtros se trasladan 2 píxeles a la vez. Tener un paso más grande producirá mapas de características más pequeños.

3. **Relleno con cero**: es conveniente rellenar la matriz de entrada con ceros alrededor del borde, de modo que se pueda aplicar el filtro a los elementos que delimitan a la matriz de imagen de entrada. Una buena característica del relleno cero es controlar el tamaño de los mapas de características. Agregar relleno cero también se llama convolución amplia, y no usar relleno cero sería una convolución estrecha.

No linealidad (ReLU).

ReLU significa Unidad lineal rectificada y es una operación no lineal como se muestra en la Fig 2.20. ReLU es una operación inteligente de elementos aplicada a los píxel y reemplaza todos los valores de píxeles negativos en el mapa de características por cero. El propósito de ReLU es introducir la no linealidad en la RNC, ya que la mayoría de los datos del mundo real no son lineales [31].

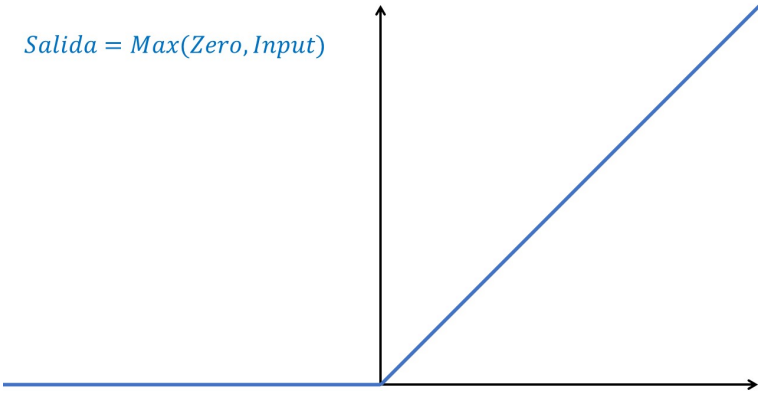
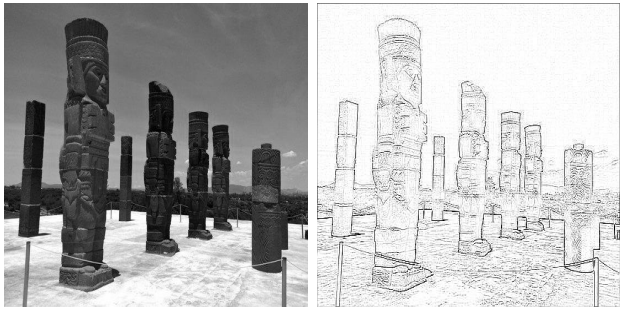


Figura 2.20: Operacion ReLU

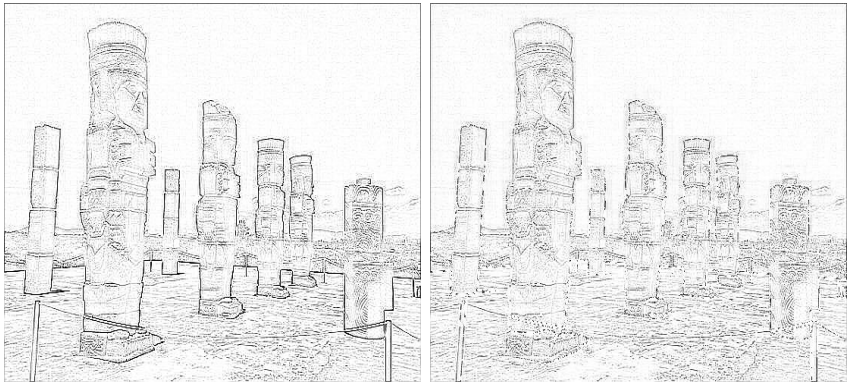
La operación ReLU se suestra en la siguiente Fig. 2.21, donde se muestra la operación ReLU aplicada a uno de los mapas de características obtenidos en la Tab. 2.2. El mapa de características de salida aquí también se conoce como el mapa de características *Rectificado* [31].



(a) Imagen de entrada (b) Convolución

Figura 2.21: Convolución en detección de bordes

La Fig 2.22 muestra el efecto de la agrupación en el mapa de características rectificado que se obtuvo después de la operación ReLU en la Fig 2.22 anterior.



(a) Negros negativo y blanco positivos (b) No negativo

Figura 2.22: Operación ReLU

Otras funciones no lineales tales como *tanh* o *sigmoide* también se pueden utilizar en lugar de ReLU, pero ReLU obtiene mejores resultados en la mayoría de las situaciones.

Agrupación o submuestreo.

La agrupación o submuestreo reduce la dimensión de cada mapa de características, pero conserva la información más importante. Los tipos de submuestreo son: Máx, Promedio, Suma, etc. también llamada Pooling Step [32].

En el caso de Max Pooling, se define una región mapa de características y se toma el elemento más grande dentro de esa región. En lugar de tomar el elemento más grande, también se puede tomar el promedio o la suma de todos los elementos de la región. En la práctica, se ha demostrado que Max Pooling funciona mejor [31].

Un ejemplo de la operación de Max Pooling en el mapa de función rectificada el cual se obtiene después de la operación de convolución + ReLU. Deslizamos la región 2 x 2 por 2 celdas y se toma el valor máximo en cada

región. Como se muestra en la Fig 2.23, esto reduce la dimensión en el mapa de características [31].

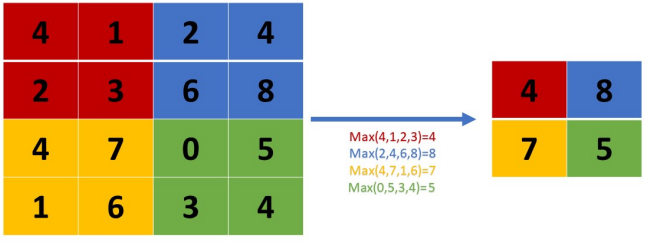


Figura 2.23: Max Pooling

En la red que se muestra en la Fig 2.24, la operación de agrupación se aplica por separado a cada mapa de características, debido a esto, obtenemos tres mapas de salida de tres mapas de entrada.

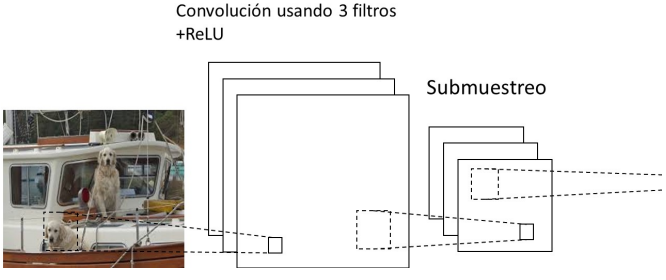


Figura 2.24: Agrupación aplicada a mapas de características rectificados

Clasificación.

La capa totalmente conectada es una neurona múlticapa que utiliza una función de activación de $softmax$ en la capa de salida . El término totalmente conectado implica que cada neurona en la capa anterior está conectada a cada neurona en la siguiente capa [32].

La salida de las capas convolucionales y de agrupación representa características de alto nivel de la imagen de entrada. El propósito de la capa totalmente conectada es usar estas funciones para clasificar la imagen de entrada en varias clases según el conjunto de datos de entrenamiento [32]. Por ejemplo, la tarea de clasificación de imágenes tiene cuatro salidas posibles, como se representa en la Fig 2.25.

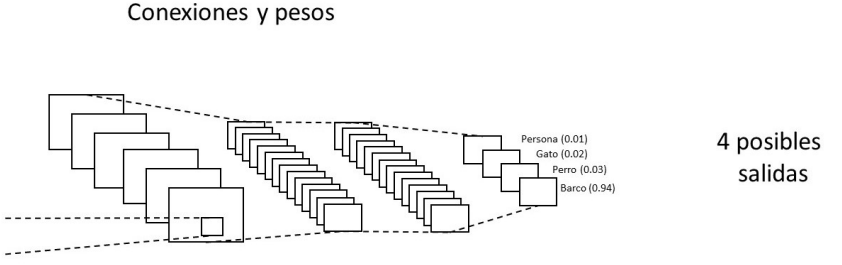


Figura 2.25: Capa totalmente conectada

Además de la clasificación, agregar una capa totalmente conectada es una forma fácil de aprender combinaciones no lineales de estas características. La mayoría de las características de las capas convolucionales y de agrupación pueden ser buenas para la tarea de clasificación, pero las combinaciones de esas características pueden ser incluso mejores [32].

La suma de las probabilidades de salida de la capa totalmente conectada es 1. Esto se garantiza mediante el uso de $softmax$ como la función de activación en la capa de salida de la capa totalmente conectada. La función $softmax$ toma un vector de puntuaciones arbitrarias de valores reales y lo aplasta en un vector de valores entre cero y uno que suma a uno [32].

2.2. Estado del arte

En años resientes el ser humano a tratado que las máquinas tengan inteligencia, conjuntamente con los avances tecnológicos se han desarrollado nuevos algoritmos y teorías. Actualmente las redes neuronales juegan un papel importante en el área de inteligencia artificial.

McCulloch y Pitts [69], desarrollaron los primeros modelos matemáticos de redes neuronales, los cuales describen el cálculo lógico de las actividades nerviosas. Sin embargo, no fue tecnológicamente posible desarrollar dichos modelos en esa época. Tiempo después, Donald Hebb [25] propuso un esquema de aprendizaje para actualizar las conexiones de las neuronas. Esta técnica de aprendizaje tuvo gran impacto en los desarrollos futuros en este campo. En la década de los 50, Rosenblatt [47] y sus colaboradores desarrollaron el perceptrón con bases biológicas y mostrando la capacidad de aprendizaje. En un principio no tuvo gran importancia y desagradó a muchos sectores, al pasar el tiempo fue desarrollada la teoría matemática útil en relación al perceptrón. Rosenblatt tenia un gran interés en el reconocimiento de patrones y máquinas de aprendizaje, capas de clasificar mediante un perceptrón.

Bernard Widrow y Marcian Hoff [6] desarrollaron el elemento lineal adaptativo (ADALINE por sus siglas en ingles), el cual utiliza reglas de aprendizaje diferente al perceptrón; usa la regla de aprendizaje LMS (Least Mean Square) o regla Widrow-Hofflearning. En la segunda mitad de la década, este campo sufre dos problemas; un punto de vista cualitativo y otro experimental. Esta ideología resulta en una falta de rigor y ocasiono que una gran parte de los investigadores de redes neuronales se dejan llevar por su entusiasmo de sus declaraciones y sus escritos fantasiosos. Un factor fuerte en la desacreditación de las redes neuronales fue el libro de Perceptrons de Minsky y Papert de 1969 [27] en el cual demuestra que el perceptrón simple solo puede resolver problemas lineales y muestra que el perceptrón no resuelve el problema XOR, con ello la investigación en redes neuronales se ve afectada los siguientes 15 años. Tiempo después, Teuvo Kohonen introduce la red neuronal artificial llamada un mapa o red Kohonen [36] en donde desarrolla redes de aprendizaje no supervisadas para el

mapeo de características en arreglos regulares de neuronas.

En los 70's Paul Werbos [38] propone el algoritmo de propagación hacia atrás, pero toma relevancia quince años después en el libro de Rumelhart, Hinton y Willams [48] donde muestran como entrenar redes neuronales multicapa por medio del algoritmo de propagación hacia atrás, realizando el cálculo de los pesos de la red por gradientes. Este algoritmo funciona mucho más rápido que los enfoques anteriores de aprendizaje, lo que hace posible el uso de redes neuronales para resolver problemas que antes eran irresolubles. Hoy en día es fundamental para aprendizaje en redes neuronales.

Años después, Sartori y Antsaklis proponen un método para encontrar el número de neuronas ocultas en redes neuronales multicapa para un conjunto de entrenamiento arbitrario [52]. Dos años más tarde Arai [4], propuso dos métodos de hiperplano paralelos para encontrar el número de neuronas ocultas. Afines de esta década, Fujita [18] propuso una estimación estadística del número de neuronas ocultas, el mérito de este método fue la velocidad de aprendizaje. L a cantidad de neuronas ocultas depende principalmente del error de salida.

El inicio de redes neuronales convolucionales (CNN) fue el modelo de Neocognitron propuesto por Kunihiko Fukushima [68], consistió en múltiples capas que aprendieron automáticamente una jerarquía de abstracciones de características para el reconocimiento de patrones. Una mejora importante sobre el Neocognitron fue el modelo de LeNet propuesto por LeCun [70], donde los parámetros del modelo se aprenden utilizando la propagación de errores por retroceso. Este modelo de CNN se aplicó con éxito para reconocer los dígitos escritos a mano.

Kulkarni y Li [65] trabajaron con imágenes del mismo objeto, pero con distintas poses, iluminaciones, escalas, fondos y diferentes parámetros de la cámara, los cuales clasifican la imagen de acuerdo a sus características, se definen como códigos dispersos afines que se aprenden de una gran colección de imágenes, tales características se utilizan luego para la categorización de imágenes. Por otra parte, David Eigen y Rob Fergus [15] utilizan tres

diferentes visiones en una red de convolución multiescala para cada imagen: estimulación profunda, estimación normal exterior y etiquetado semántico. Obteniendo una red de capas adaptables a cada tarea al reconocer satisfactoriamente las imágenes. En este mismos año Fayao Liu et al. [17] proponen el aprendizaje de profundidad en imágenes, basados en modelos CNN y el método de agrupación de superpíxeles, ayudando a mejorar su velocidad de aprendizaje. Estos experimentos demuestran que el método propuesto supera los enfoques de estimación de profundidad de vanguardia.

Capítulo 3

Comparación de librerías de aprendizaje profundo

3.1. Librerías/Framework de aprendizaje profundo

Desde el 2016 la IA está creciendo rápidamente. Por ejemplo, el 20 % de las empresas en los Estados Unidos [30] utilizan IA en sus negocios. Por tal motivo es importante conocer los mejores lenguajes o bibliotecas para implementar algoritmos de IA. En esta sección, cubriremos las diferentes bibliotecas.

3.1.1. Torch

Fue desarrollado en 2002 por Collober et al [46]. inicialmente escrito en el lenguaje LUA, pero tiene una implementación en C. Torch admite una amplia biblioteca para algoritmos de aprendizaje automático, incluido el aprendizaje profundo. Además es compatible con la implementación de CUDA para computación paralela [46], sus características principales incluyen [63]:

- Una poderosa matriz N-dimensional.
- Infinidad de rutinas para indexar, recortar, transponer, entre otras.

- Interfaz a C, a través de LuaJIT.
- Rutinas de álgebra lineal.
- Red neuronal.
- Rutinas de optimización numérica.
- Soporte GPU
- Embebible con puertos para backends de iOS y Android

Torch es utilizado por la mayoría de los laboratorios líderes como Facebook, Google, Twitter, Nvidia. Tiene la biblioteca Pytorch en Python [46]. El modelo de este framework se basa en un script, el cual puede crear una red neuronal compleja con pocas lineas de código. Sin embargo la portabilidad del código a un framework diferente es cuestionable [43].

3.1.2. Keras

Es un framework de aprendizaje profundo en Python capaz de ejecutarse sobre TensorFlow , CNTK o Theano. Además, el entorno de codificación permite el entrenamiento de algoritmos de redes convolucionales y redes recurrentes o una combinación de ambos. Funciona a la perfección en CPU y GPU [34].

Keras ha sido desarrollado por François Chollet, investigador de Google. Se utiliza en organizaciones destacadas como CERN, Yelp, Square o Google, Netflix y Uber [49]. Características clave:

- Facilidad de uso del usuario.
- Modularidad.
- Capacidad de ampliación sencilla.

3.1.3. Theano

Es una biblioteca de aprendizaje profundo desarrollada en 2007. Ofrece un cómputo rápido y puede ejecutarse tanto en la CPU como en GPU. Theano ha sido desarrollado para entrenar algoritmos de redes neuronales profundas. Sus características son [62]:

- Integración estrecha con NumPy.
- Uso transparente de una GPU.
- Diferenciación simbólica eficiente.
- Optimizaciones de velocidad y estabilidad.
- Generación dinámica de código C.
- Pruebas unitarias extensivas y auto-verificación.

3.1.4. El kit de herramientas cognitivas de Microsoft (CNTK)

El kit de herramientas de Microsoft, anteriormente conocido como CNTK, es una biblioteca de aprendizaje profundo desarrollada por Microsoft. Según Microsoft, la biblioteca se encuentra entre las más rápidas del mercado. Es una biblioteca de código abierto, utilizado en productos como Skype, Cortana, Bing y Xbox [49], Es compatible con sistemas operativos Linux de 64 bits o Windows de 64 bits [12]. El kit de herramientas está disponible tanto en Python como en C, al igual que Torch el modelo utiliza scripts. El framework de Microsoft describe las redes neuronales como una serie de pasos computacionales a través de un gráfico dirigido. Además, permite al usuario realizar fácilmente y combinar tipos de modelos populares, tales como redes neuronales dinámicas, redes neuronales convolucionales y redes neuronales recurrentes. Implementa el aprendizaje de descenso de gradiente estocástico del error de propagación hacia atrás con diferenciación automática y paralelización en múltiples GPU y servidores [12]. Sus características son [43]:

- Velocidad y capacidad de ampliación.
- Calidad de tipo comercial.

- Compatibilidad.
- Scripts

3.1.5. MXNet

MXnet es una biblioteca de aprendizaje profundo, su principal usuario es Amazon. Es accesible a múltiples lenguajes de programación, como C ++, Julia, Python y R. Además, puede configurarse para funcionar tanto en la CPU como en la GPU. MXNet incluye una arquitectura de aprendizaje profundo, como redes neuronales convolucionales y la memoria a largo plazo. Está construido para funcionar en armonía con la infraestructura de nube dinámica [49]. Tambien incluye la interfaz de Gluon que permite a los desarrolladores usar el aprendizaje profundo en la nube, en dispositivos de borde y en aplicaciones para dispositivos móviles. En tan solo unas líneas de código de Gluon, es posible crear regresión lineal, redes convolucionales y LSTM recurrentes para la detección de objetos, el reconocimiento de voz [40]. Sus características clave son:

- Compatibilidad.
- Es portátil a través de plataformas.
- Escalabilidad.
- Scripts.
- Facilidad de uso con GLUON.
- Mayor rendimiento.

3.1.6. Caffe

Caffe es una librería desarrollada por Yangqing Jia cuando era estudiante de doctorado en Berkeley. Caffe está escrito en C ++ y puede realizar cálculos tanto en la CPU como en la GPU. Los principales usos de Caffe es la red neuronal convolucional. Aunque, en 2017, Facebook amplió Caffe con una arquitectura de aprendizaje más profunda, incluida la red neuronal recurrente [49]. También, es utilizado por académicos, startups y por algunas

grandes empresas como Yahoo!. Sus principales características son la arquitectura expresiva, el código extensible, la velocidad y su comunidad [8]. Caffe se define en un modelo de archivos de configuración, es bueno para la portabilidad del modelo, pero son difíciles de usar cuando se construye una arquitectura de red neuronal compleja. Sus principales características [43]:

- Arquitectura expresiva.
- Código extensible.
- Velocidad.
- Proyectos de investigación académica.

TensorFlow

Es una biblioteca de software de código abierto para cálculos numéricos que utilizan gráficos de flujo de datos. Los nodos en el gráfico representan operaciones matemáticas, mientras que los vértices del gráfico representan las matrices de datos multidimensionales (tensores) comunicadas entre ellos. Su arquitectura flexible le permite ser implementado en una o más CPU o GPU, en una computadora de escritorio, servidor o dispositivo móvil con una sola API. TensorFlow fue desarrollado originalmente por investigadores e ingenieros que trabajan en Google Brain Team dentro de la organización de investigación Machine Intelligence de Google con el propósito de llevar a cabo el aprendizaje automático y la investigación en redes neuronales profundas, pero el sistema es lo suficientemente general como para ser aplicable en una amplia variedad de proyectos [2].

La Tab. 3.1 muestra una comparación a las principales características de los Frameworks de aprendizaje profundo.

Framework	Plataforma	Lenguajes	APIs	Soporta CUDA	RNC	Dispositivos Móviles	Scripts
TensorFlow	Linux, MacOS, Windows, Android, iOS	Python, c++, CUDA	Python, C++, Java, Go	SI	SI	SI	SI
Keras	Linux, MacOS, Windows	Python	Python, R	SI	SI	SI	SI
Caffe	Linux, MacOS, Windows	C++	Python, MATLAB	SI	SI	SI	NO
MXNet	Linux, MacOS, Windows, iOS	C++	Python, Scala, R, JavaScript, Julia, Perl, MATLAB, Go,C++,	SI	SI	SI	SI
Theano	Linux, MacOS, Windows	Python	Python	SI	SI	NO	SI
CNTK	Linux, Mac Docker, Windows	C++	Python, C++, C#, Java	SI	SI	NO	SI
Torch	Linux, MacOS, Windows	Lua	C++, Lua, LuaJIT	SI	SI	SI	SI

Tabla 3.1: Comparación de Frameworks

Framework	Oferta de Trabajo	Encuesta de Uso	Actividad de GitHub	Volumen de busqueda	Articulos ArXiv	Libros de Amazon	Artículos Medianos	Puntaje Total
TensorFlow	30.00	20.00	10.00	10.00	10.00	10.00	6.77	96.77
Keras	8.94	15.17	2.53	7.26	3.74	3.91	10.00	51.55
PyTorch	7.34	4.14	1.66	2.60	4.20	0.89	1.89	22.72
Caffe	8.94	0.69	1.80	0.55	3.66	0.69	0.82	17.15
Theano	5.80	2.76	0.49	0.00	1.74	0.84	0.40	12.02
MXNet	3.26	0.69	1.39	0.27	0.67	1.58	0.50	8.35
CNTK	1.69	2.07	0.71	0.00	0.21	0.05	0.17	4.89

Tabla 3.2: Score de Framework

De acuerdo al estudio realizado por Jeff Hale [24] TensorFlow casi alcanzó un 100 % como se muestra en la Tab. 3.2, lo cual no fue sorprendente después de verlo en la parte superior de cada categoría evaluada. Keras quedó un segundo claro. En la tabla muestra la comparacion de los frameworks por su versatilidad en plataformas como en APIs [43].

3.2. El aprendizaje automático como un servicio

Los proveedores de servicios proporciona a los usuarios una gama de herramientas como parte de un servicio de computación en la nube. Esto puede incluir herramientas para la visualización de datos, reconocimiento facial, procesamiento de lenguaje natural, reconocimiento de imágenes, análisis predictivo y aprendizaje profundo. El proveedor maneja los cálculos reales en sus propios centros de datos, los clientes no tienen que instalar su propio software o ejecutar sus propios servidores.

3.2.1. Google Cloud ML

Google proporciona un modelo pre-entrenado disponible en Cloud AutoML. Esta solución existe para un desarrollador sin una sólida formación en aprendizaje automático. Los desarrolladores pueden utilizar el modelo pre-entrenado de Google de vanguardia en sus datos. Además, permite a cualquier desarrollador entrenar y evaluar cualquier modelo en solo unos minutos [49].

Google Cloud, puede entrenar un marco de aprendizaje automático basado en TensorFlow, Scikit-learn, XGBoost o Keras. El aprendizaje automático de Google Cloud capacitará a los modelos en toda la nube. Por otra parte, los servicios IA de Cloud son rápidos, escalables y fáciles de usar. Google proporciona una API REST para visión de computadora, reconocimiento de voz, reconocimiento de texto, análisis de vídeo, traducción y PNL [11].

La ventaja de usar Google Cloud Computing es la simplicidad de implementar el aprendizaje automático en la producción. No hay necesidad

de configurar el contenedor Docker. Además, la nube se ocupa de la infraestructura. Sabe cómo asignar recursos con CPU, GPU y TPU [49].

3.2.2. AWS SageMaker

Amazon ha desarrollado Amazon SageMaker para permitir que los científicos y desarrolladores de datos construyan, entrenen y pongan en producción cualquier modelo de aprendizaje automático [50].

Está disponible en Jupyter Notebook e incluye la librerías de aprendizaje automático más utilizada, TensorFlow, MXNet, Scikit-learn, entre otros. Los programas escritos con SageMaker se ejecutan automáticamente en los contenedores de Docker. Amazon maneja la asignación de recursos para optimizar la capacitación y el despliegue [49]. También, Amazon proporciona API a los desarrolladores para agregar inteligencia a sus aplicaciones. En alguna ocasión, no es necesario construir desde cero nuevos modelos mientras haya en la nube modelos poderosos pre-entrenados. Amazon proporciona servicios API para visión por computadora, chatbots conversacionales y servicios de idiomas [49].

Amazon SageMaker incluye los siguientes módulos: El módulo de construcción es un entorno alojado para trabajar con sus datos, experimentar con algoritmos y visualizar sus salidas. El módulo de implementación ofrece un entorno administrado para que aloje y pruebe los modelos para inferencia de forma sencilla, segura y con baja latencia [50]. Las tres principales APIs disponibles son:

- Amazon Rekognition: proporciona reconocimiento de imagen y video a una aplicación.

- Amazon Comprender: realizar minería de texto y procesamiento de lenguaje neural para automatizar el proceso de verificación de la legalidad de un documento financiero.

- Amazon Lex: Agrega chatbot a una aplicación.

3.2.3. Azure Machine Learning Studio

Probablemente uno de los enfoques más amigables para el aprendizaje automático. La ventaja significativa de esta solución es que no se requieren conocimientos previos de programación. Microsoft Azure Machine Learning Studio es una herramienta de colaboración de arrastrar y soltar para crear, entrenar, evaluar e implementar una solución de aprendizaje automático. El modelo se puede implementar de manera eficiente como servicios web y se puede utilizar en varias aplicaciones como Excel. La interfaz de aprendizaje es interactiva, lo que permite al usuario construir un modelo simplemente arrastrando y soltando elementos rápidamente. Cuando el modelo esté listo, el desarrollador puede guardarlo y enviarlo a Azure Gallery o Azure Marketplace. También, puede integrarse en R o Python en su paquete personalizado.

3.2.4. IBM Watson ML

Watson Studio proporciona el entorno y las herramientas para resolver sus problemas comerciales al trabajar en colaboración con los datos. Puede elegir las herramientas que necesita para analizar y visualizar datos, para limpiar y dar forma a los datos, para ingerir datos de transmisión, o para crear y entrenar modelos de aprendizaje automático [58]. También es fácil de usar con un código de arrastrar y soltar.

Watson studio admite algunos de los frameworks más populares como Tensorflow, Keras, Pytorch, Caffe y puede implementar un algoritmo de aprendizaje profundo en las GPU más recientes de Nvidia para ayudar a acelerar el modelado [58].

3.3. Conclusiones

Los servicios en la nube es una gran alternativa debido a que no requieren conocimientos previos de programación o aprendizaje profundo y su migración a otros framework es relativamente fácil, pero el objetivo de esta tesis es entender el funcionamiento de las RNC y los frameworks son la mejor opción. De acuerdo con el estudio realizado por Jeff Hale, Tensor-

Flow es el mejor framework, porque está diseñado para ser accesible para todos. El framework incorpora diferentes APIs para construir a escala la arquitectura de aprendizaje profundo como CNN o RNN. Se basa en el cálculo grafos el cual permite al desarrollador visualizar la construcción de la red neuronal con Tensorboad. Esta herramienta es útil para depurar el programa. Finalmente, Tensorflow está diseñado para ser implementado a escala y se ejecuta en CPU y GPU.

Es compatible con TensorFlow Serving y el modelo entrenado se puede convertir en un servicio web. Además, TensorFlow Mobile admite la compresión de modelos de forma inmediata para dispositivos móviles.

Capítulo 4

Imágenes de entrenamiento

4.1. Introducción

Este capítulo presenta la selección de las piezas para ser reconocidas por la aplicación y las reseñas de las mismas. Así como las técnicas aumento de datos usadas. El capítulo 2 se explico como es necesario un gran número de imágenes para poder entrenar a una red neuronal convolucional. Por lo tanto, se descargan imágenes por medio búsquedas en internet y con ayuda de plugins de Google Chrome; Image Downloader e I'm a Gentleman. Presentando el inconveniente de descargar más imágenes de un objeto que otro, para igualar el numero de imágenes se usaron técnicas de aumento de datos mostradas en este capítulo.

4.2. Selección de muestra

La muestra consta de 5 objetos debido a la incertidumbre de las piezas que mostrara el museo después de su remodelación, en cuanto el museo proporcione la lista se agregara la totalidad de las piezas en la aplicación.

Cariátide. Fue encontrado por Jorge R. Acosta en el lado norte de el edificio B y mide 1.04 m largo, 4.58 m alto y 1.07 m de ancho. Personaje de pie con los brazos extendidos junto al cuerpo. Usa un penacho, orejeras rectangulares, pechero, pectoral de mariposa estilizada, disco dorsal, rodi-

lleras, ajorcas y sandalias. La mano derecha sujeta una lanza dardos y la izquierda sostiene tres objetos: un calabazo decorado con grecas, un arma curva y cuatro largos dardos. En el hombro izquierdo lleva un brazalete que sostiene un cuchillo [19].

Chac mool. Esta figura fue encontrada en Edificio 3 del Palacio Quemado, lado Este de la Sala 2 por el arqueólogo Jorge R. Acosta en 1953. Las medidas son 1.10 m de largo, 81.5 cms de alto y 52 cms de ancho. Lleva una diadema triangular que se amarra en la parte posterior de la cabeza. Usa narigueras de botón dobles y orejeras rectangulares a semejanza de las cariátides. Sobre el pecho lleva un pectoral de mariposa atado al cuello. Las manos, decoradas con pulseras, sostienen un objeto cuadrangular plano que descansa en su vientre. Como prenda lleva un delantal triangular atado a la cintura y usa sandalias sencillas.

Coatepantli. Este muro se encuentra a lado Este del Edificio B, hallado por Jorge R. Acosta, la parte que se conserva mide 18.15 m de largo y donde se encuentran los relieves tiene 1.50 m de alto. Este muro es el prototipo de los muros que construían los pueblos mesoamericanos alrededor de sus plazas y templos ceremoniales y según la cosmogonía de éstos pueblos, separaba o delimitaba el espacio exterior de su recinto sagrado [19].

El Palacio Quemado o edificio 3. Palacio Quemado, se nombra así por el incendio localizada en su interior. El edificio comprende tres grandes salas en su interior un patio interior abierto, funcionó como recolector de agua pluvial, además de una entrada de luz y ventilación. Cada sala contaba con un gran número de columnas y pilastras para sostener los techos [10].

Edificio B. Edificio B construida en honor a Quetzalcóatl, Señor del Lugar de la Casa de Venus como Estrella Matutina, y como Ehécatl, Señor del Viento y la fertilidad agrícola. Se trata de una plataforma compuesta por cinco cuerpos trunco-piramidales, en cuya cima se encuentran los llamados atlantes de Tula. La mayor parte de las esculturas que coronan este edificio fueron halladas durante la temporada de investigaciones encabezada por Jorge Acosta y su equipo en 1941. Se trata de un basamento escalonado

de cinco cuerpos en talud poco inclinado, con base cuadrada de 36 m por lado y una altura total de 9.5 m [13].

4.3. Recolección de datos

El conjunto de imágenes para entrenar al modelo para reconocer las piezas del museo de Tula, fueron obtenidas de la web y descargadas por el plugin de Image Downloader e I'm a Gentleman. Obteniendo una base fotográfica de 1073 Cariatide, 171 de Chac Mool, 110 de Coatepantli, 214 del Palacio Quemado y 328 de la Pirámide B Fig 4.3.

Image Downloader plugin gratuito, descarga imágenes de forma masiva desde una página web, con esta extensión puedes;

- Ver imágenes que contiene la página y el enlaces.
- Filtrarlos por ancho, alto y URL.
- Opcionalmente mostrar solo imágenes de enlaces.
- Selecciona las imágenes para descargar haciendo clic en la imagen.
- Use botones dedicados para descargar o abrir imágenes individuales en nuevas pestañas
- Personalice el ancho de visualización de la imagen, las columnas, el tamaño del borde y el color.
- Ocultar filtros, botones y notificaciones que no necesites.

I'm a Gentleman Este plugin es también gratuito, descarga todas las imágenes de la página usando el botón de la extensión. Es destacado en The Next Web y Gizmodo, como una de las mejores extensiones de Chrome disponibles en el webstore.

Figura 4.1: Logo Image Downloader

Figura 4.2: Logo I'm a Gentleman

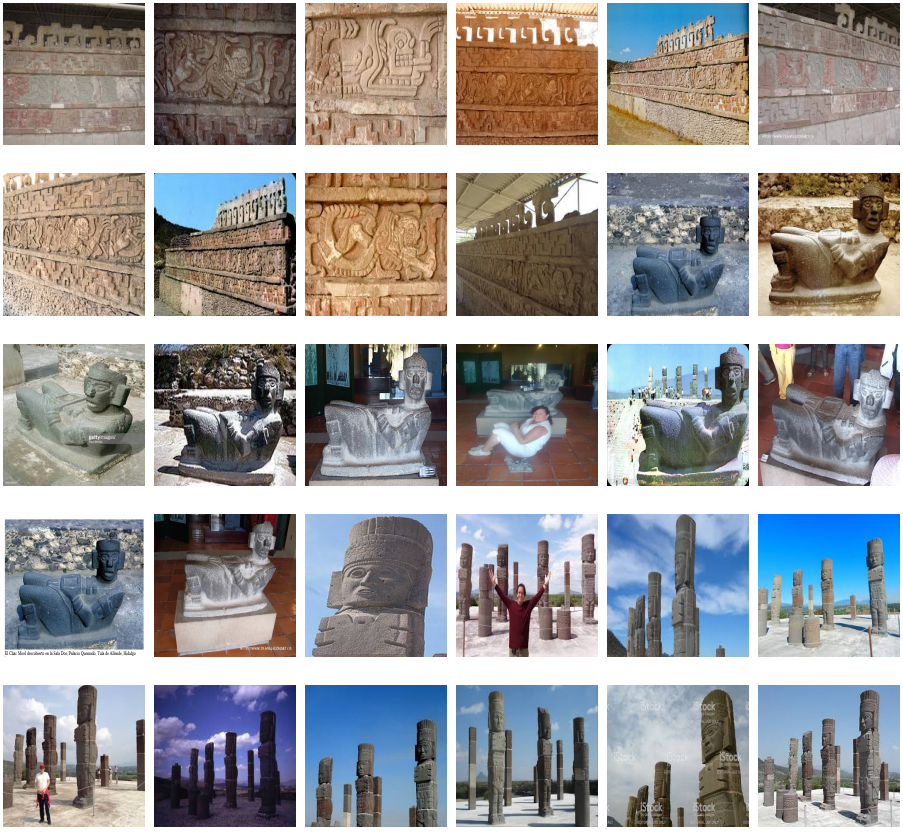


Figura 4.3: Imágenes Descargadas

Find.Same.Images.OK es un software gratuito para Windows que ayuda a encontrar imágenes iguales, duplicadas o similares, teniendo en cuenta que hayan pasado por procesos de recorte, de ampliación, e incluso llevadas al negativo, esta software es capaz de dar con ellas.

Cuenta con una interfaz de usuario simple con dos imágenes. Para encontrar fotos duplicadas, se selecciona la carpeta a escanear por el software. Al correr el programa aparece una lista de fotos duplicadas.

Al seleccionar una imagen de la lista aparece en ambas ventanas de la interfaz. La ventana de vista previa muestra ambas versiones de una imagen y despliega el porcentaje de coincidencia entre las fotos.

Debajo de las imágenes, los usuarios pueden elegir rotar o voltear la imagen, así como determinar el balance de calidad de la foto.

Para eliminar la fotografía, con el botón derecho del mouse y elija Papelera de reciclaje. Incluso te permite decidir dónde mover o copiar una imagen.

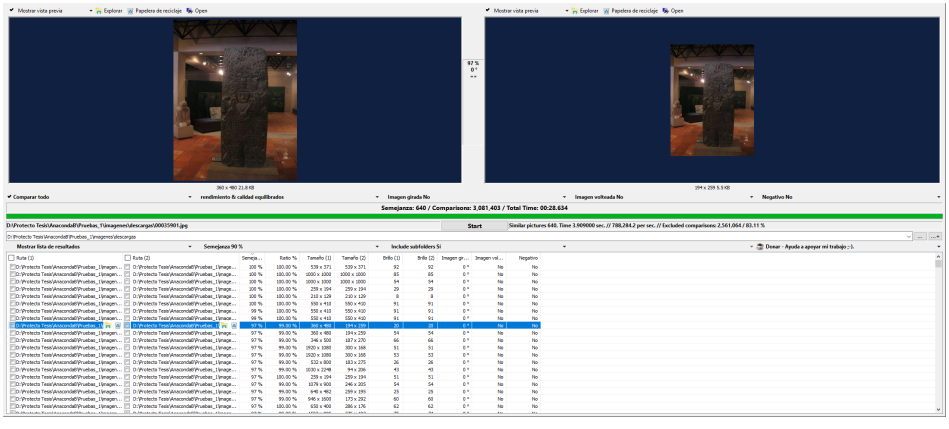


Figura 4.4: Find.Same.Images.OK

4.4. Aumento de datos

El aumento de datos de las imágenes es una técnica que se puede utilizar para ampliar artificialmente el tamaño de un conjunto de datos de entrenamiento mediante la creación de versiones modificadas de imágenes en el conjunto de datos. Esta técnica es útil en escenarios cuando el conjunto de datos es pequeño y se puede combinar y utilizar con otras técnicas. Hay varias formas de aumentar las imágenes [54]:

- **Filtros**: Se puede filtrar directamente sobre los píxeles de la imagen, o en el dominio de la frecuencia, donde las operaciones se llevan a cabo en la transformada de Fourier de la imagen.
- **Reflejar o Voltear**: La imagen se refleja o voltea en dirección horizontal o vertical.
- **Recorte aleatorio**: Las imágenes se recortan de forma aleatoria.
- **Intensidad de corte**: El ángulo de corte se realiza en sentido antihorario en grados.
- **Zoom**: Las partes ampliadas de las imágenes están entrenadas para tratar con diferentes escalas de imágenes.
- **Rotación**: Los objetos se rotan para tratar diversos grados de cambio en los objetos.
- **Blanqueo**: El blanqueo se realiza mediante un análisis de componentes principales que conserva solo los datos importantes.
- **Normalización**: Normaliza los píxeles al estandarizar la media y la varianza.
- **Cambio de canal**: Los canales de color se desplazan para hacer que el modelo sea robusto a los cambios de color causados por diversos artefactos.

4.4.1. Filtros

Son métodos para resaltar o suprimir, información contenida en una imagen a diferentes escalas, resaltando determinados elementos de la imagen. Para mostrar los tipo de filtros utilizaremos la librería Pillow, esta librería nos ayuda a la manipulación de imágenes en Python Fig. 4.5.

```
from PIL import ImageFilter
#Difuminar
imagenFilter = imagen.filter(ImageFilter.BLUR)
#Contorno
imagenFilter = imagen.filter(ImageFilter.CONTOUR)
#Detalle
imagenFilter = imagen.filter(ImageFilter.DETAIL)
#Resaltar borde
imagenFilter = imagen.filter(ImageFilter.EDGE_ENHANCE)
#Resaltar mas el borde
imagenFilter = imagen.filter(ImageFilter.EDGE_ENHANCE_MORE)
#Estampar en relieve
imagenFilter = imagen.filter(ImageFilter.EMBOSS)
#Mostrar borde
imagenFilter = imagen.filter(ImageFilter.FIND_EDGES)
 #Suavizar
imagenFilter = imagen.filter(ImageFilter.SMOOTH)
```

Ajuste el balance de color en la imagen. Esta clase se puede usar para ajustar el balance de color de una imagen, de manera similar a los controles de un televisor en color [34]. Un factor de mejora de 0.0 da una imagen en blanco y negro. Un factor de 1.0 da la imagen original [41] como se muestra en la Fig 4.6(b).

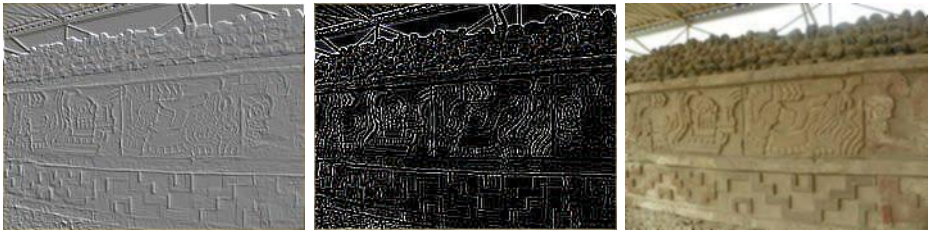
```
    imagenFiltrada = ImageEnhance.Color(imagen)
```

Ajustar el contraste de la imagen. Esta clase se puede usar para controlar el contraste de una imagen, similar al control de contraste en un televisor. Un factor de mejora de 0.0 da una imagen gris sólida [41]. Un factor de 1.0 da la imagen original Fig. 4.6(c).

```
    imagenFiltrada = ImageEnhance.Contrast(imagen)
```

(a) Original (b) Difuminar (c) Contorno

(d) Detalle (e) Resaltar borde (f) Resaltar mas el borde



(g) Estampar en relieve (h) Mostrar borde (i) Suavizar

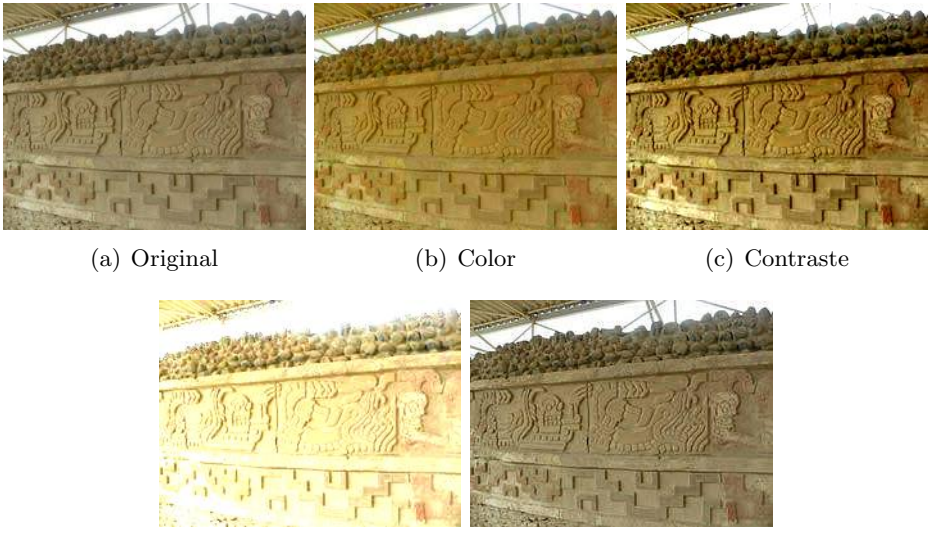
Figura 4.5: ImageFilter

Ajustar el brillo de la imagen. Esta clase se puede utilizar para controlar el brillo de una imagen [41]. Un factor de mejora de 0.0 da una imagen negra. Un factor de 1.0 da la imagen original Fig. 4.6(d).

```
imagenFiltrada = ImageEnhance.Brightness(imagen)
```

Ajustar la nitidez de la imagen. Esta clase se puede utilizar para ajustar la nitidez de una imagen [41]. Un factor de mejora de 0.0 da una imagen borrosa, un factor de 1.0 da la imagen original y un factor de 2.0 da una imagen más nítida Fig. 4.6(e).

```
imagenFiltrada = ImageEnhance.Sharpness(imagen)
```



(a) Original (b) Color (c) Contraste

(d) Brillo (e) Nitidez

Figura 4.6: Filtros ImageEnhance

4.4.2. Reflejar o Voltear

Reflejar o Voltear la imagen significa invertir las filas o columnas de píxeles en el caso de un giro vertical u horizontal, además de que el giro es aleatorio [34]. *Flip* es el argumento de la clase *ImageDataGenerator* este argumento es booleano *horizontal_flip* o *vertical_flip*, se muestra en la Fig. 4.7(b) y Fig 4.7(c) respectivamente.

```
datagen = ImageDataGenerator(horizontal_flip=True)
```

4.4.3. Intensidad de corte

Intensidad de corte el ángulo de corte es en sentido antihorario y en grados como se muetra en Fig 4.7(d)).

```
datagen = ImageDataGenerator(shear_range)
```

4.4.4. Zoom

Un aumento de zoom amplía aleatoriamente la imagen y agrega nuevos valores de píxeles alrededor de la imagen o interpola valores de píxeles respectivamente. La imagen de zoom puede ser configurado por el *zoom_range* es el argumento del constructor *ImageDataGenerator* . Puede especificar el porcentaje del zoom como un único flotante o un rango como una matriz o tupla. La cantidad de zoom se muestrea de manera uniforme y aleatoria desde la región de zoom para cada dimensión (ancho, altura) por separado como se muestra en el Fig 4.7(e). El zoom puede no ser intuitivo. Tenga en cuenta que los valores de zoom inferiores a 1.0 harán que la imagen se acerque, por ejemplo, [0.5,0.5] hace que el objeto en la imagen sea 50 % más grande o más cercano, y los valores mayores a 1.0 alejarán la imagen en un 50 %, por ejemplo, $[1.5, 1.5]$ hace que el objeto en la imagen sea más pequeño o más lejano. Un zoom de $[1.0, 1.0]$ no tiene efecto [7].

```
datagen = ImageDataGenerator(zoom_range=[0.5,1.5])
```

4.4.5. Rotación

Un aumento de rotación gira aleatoriamente la imagen en el sentido de las agujas del reloj en un número dado de grados de 0 a 360 como se muestra en el Fig 4.7(f). Es probable que la rotación gire los píxeles fuera del marco de la imagen y deje áreas del marco sin datos de píxeles que se deben completar. El siguiente ejemplo muestra rotaciones aleatorias a través del argumento *rotation_range*, con rotaciones a la imagen entre 0 y 90 grados [7].

```
datagen = ImageDataGenerator(
            rotation_range=random.randrange(0, 360))
```

4.4.6. Blanqueamiento

En este proceso transforma los datos para tener una matriz de covarianza que sea la matriz de identidad: 1 en la diagonal y 0 para las otras celdas. Se llama blanqueamiento en referencia al ruido blanco [7] este proceso se muetra en la Fig. 4.7(g).

```
datagen = ImageDataGenerator(zca_whitening = True)
```

4.4.7. Normalizar

Keras nos permite dos tipos de normalización: normalización estándar unitario (*samplewise_std_normalization*) esta permite dividir cada entrada por su estándar, como el final es un vector unitario o simplemente suman 1 y (featurewise_std_normalization). Divide las entradas por el estándar del conjunto de datos, en función de la función.

```
datagen = ImageDataGenerator(
                samplewise_std_normalization  = True)
```

```
datagen = ImageDataGenerator(
                featurewise_std_normalization = True)
```

(a) Original, (b) Reflejar (c) Voltear

(d) Intensidad de corte (e) Zoom (f) Rotación

(g) Blanqueamiento (h) Normalizar

Figura 4.7: Filtros ImageDataGenerator

4.5. Algoritmo para Aumento de datos

En esta sección se muestra el algoritmo que se ocupo para aumentar la base de fotográfica. Inicialmente como se menciono en la sección 4.3 se contaba con 1073 en imágenes de Cariátide (Atlante), 171 de Chac-mool, 110 de Coatepantli, 214 del Palacio Quemado y 328 de la Pirámide B. La primera desventaja es la desigualdad en el numero de imágenes que hay entre cada una de las piezas, la otra desventaja es que el numero de fotografías, son muy pocas. Por tanto, necesitamos un algoritmo para cubrir estas desventajas, propongo alterar las fotos originales de dos formas agregar ruido a las imágenes y cambiar de posición las imágenes. La primera parte lo haremos con ayuda de ImageFilter y ImageEnhance de Pill Alg. A.2 y para cambiar su posición ImageDataGenerator de Keras Alg. A.1. Obteniendo con ello 2,107 imágenes de Cariatide, 1,867 de Chac Mool, 2,005 de Coatepantli, 2,095 del Palacio Quemado y 1,927 de la Pirámide B. Consulta el codigo en el Anexo A.

4.6. Conclusión

En el capitulo se mostraron las herramientas necesarias para la construcción del DataSet, en la Fig. 4.8 muestra un diagrama donde explica su construcción. Inicia por hacer una búsqueda en el navegador Google Chrome con palabras relacionadas al museo como Atlantes de Tula, museo de tula, Museo Arqueológico de Tula, Museo Jorge R. Acosta, Museo del sitio de Tula, Toltecas, Cariátide, Chac mool, Coatepantli, Palacio Quemado, Edificio de tula, Edificio 3 de tula, etc. Por cada palabra(s) se realiza una descarga de las imágenes por medio de los plugins Image Downloader ó I'm a Gentleman hasta lograr las imágenes suficientes (mayor a 3000).

Con ayuda del software *Find.Same.Images.OK* se eliminar todas las imágenes repetidas. Posteriormente se etiquetan las imágenes, es decir, se crean carpetas con el nombre del objetó a clasificar Apéndice C.6. Por ultimo se revisa cada una de las imágenes en cada una de las etiquetas, si la fotografía no cumple con el objetivo de la etiqueta se elimina.

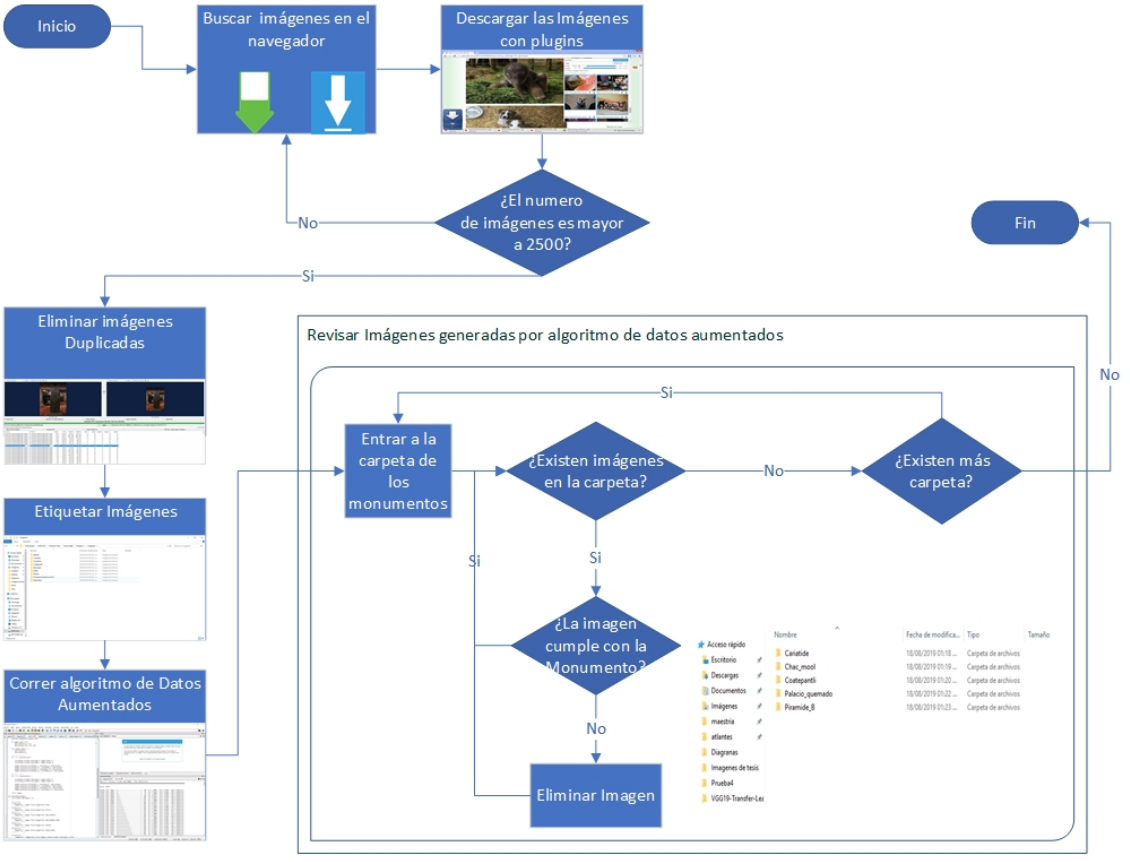


Figura 4.8: Diagrama de flujo para generar el DataSet.

Capítulo 5

Clasificación de piezas arqueológicas con aprendizaje profundo

5.1. Introducción

Este capítulo se divide en dos secciones, la primera se relaciona a el entrenamiento del modelo MobileNet como es el ambiente de desarrollo, arquitectura del modelo, código de entrenamiento y resultados. La segunda sección lo referente a la aplicación en Android como una explicación de la arquitectura TensorFlow Lite, configuración de TensorFlow Lite en Android y los layout de la aplicación.

5.2. Ambiente de Trabajo de TensorFlow

Anaconda es una distribución libre y Open Source de los lenguajes de programación Python y R muy usada en computación científica (Data ScienceData Science, Machine Learning, Ciencia, Ingeniería, analítica predictiva, Big Data, etc). Los paquetes científicos requieren una versión específica de Python para ejecutarse. Es difícil mantener varias instalaciones de Python en una computadora para que no interactúen y se rompan, y

es más difícil mantenerlas actualizadas. Anaconda hace que la administración de múltiples versiones de Python en una computadora sea más fácil, y proporciona una gran colección de bibliotecas de ciencia de datos altamente optimizadas y de uso común para que pueda comenzar más rápido [53].

La distribución de Anaconda incluye a **Anaconda Navigator** que es una interfaz gráfica de usuario (GUI) de escritorio. Permite iniciar aplicaciones y administrar fácilmente los paquetes, entornos y canales de Conda sin usar comandos de la línea de comandos. El navegador puede buscar paquetes en Anaconda Cloud o en un repositorio local de Anaconda. Está disponible para Windows, MacOS y Linux [53]. Las aplicaciones disponibles de forma predeterminada en Navigator son JupyterLab, Jupyter Notebook, QTConsole, Spyder, VSCode, Glueviz, Orange 3 App, Rodeo, RStudio como se muestra en la Fig. 5.1.

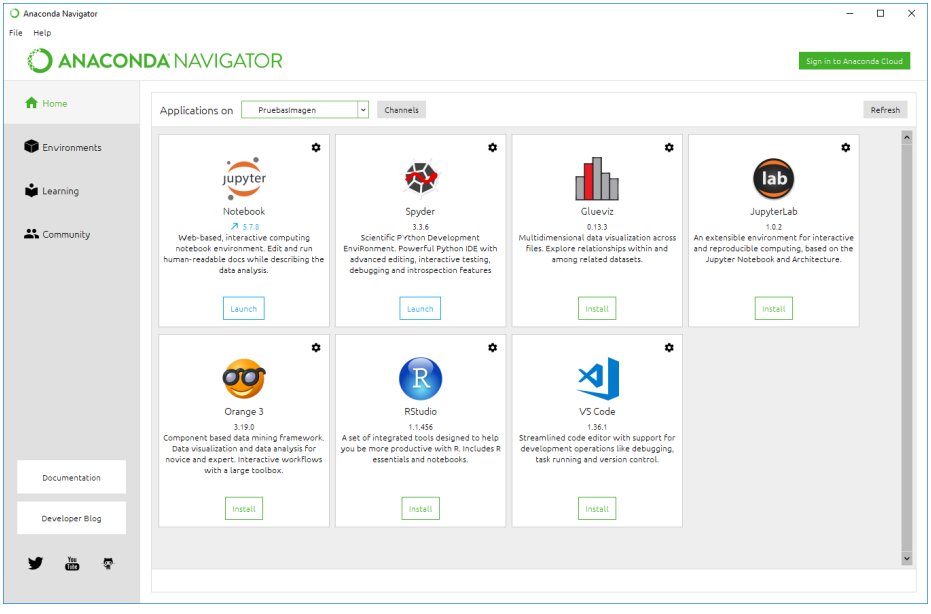


Figura 5.1: Anaconda Navigator

Spyder.

Spyder es entorno científico escrito para Python, diseñado para ingenieros y analistas de datos. Cuenta con una combinación de funciones de edición, análisis, depuración y creación de perfiles, esta es una herramienta de desarrollo integral con exploración de datos, ejecución interactiva, inspección profunda y capacidades de visualización de paquetes. Además, Spyder ofrece integración de paquetes, como NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy y más [57].

Las capacidades de Spyder se pueden ampliar aún más a través de su sistema de plugin y API. Spyder también se puede usar como una biblioteca de extensión PyQt5, lo que le permite desarrollar su funcionalidad e incrustar sus componentes, como la consola interactiva, en su propio software [57]. Los principales componentes son el editor, consola IPython, Explorador de variables, perfilador, depurador y ayuda, como se muestra en la siguiente Fig. 5.2.

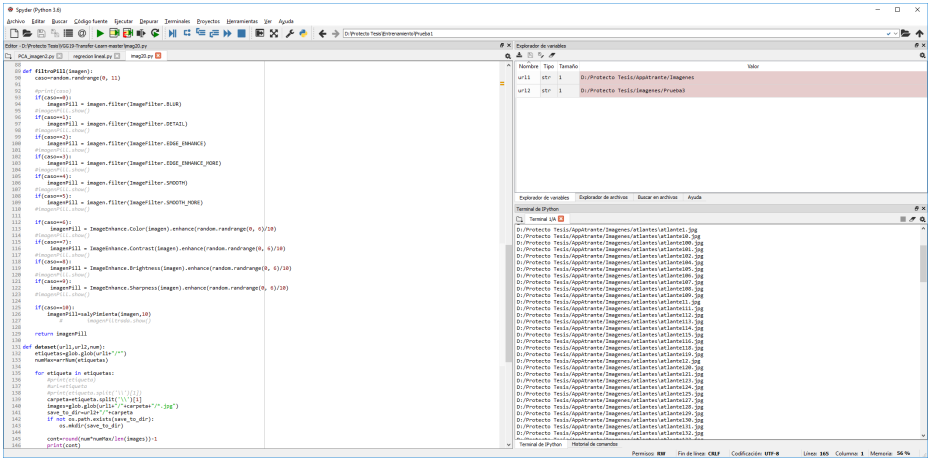


Figura 5.2: Spyder

Editor de Spyder integra una serie de herramientas de edición eficiente y fácil de usar. Las características clave del editor incluyen resaltado de sintaxis, análisis de código, estilo en tiempo real, características de funciones y un navegador de funciones / clases [57].

Consola de IPython permite ejecutar comandos e ingresar, interactuar y visualizar datos dentro de un número de intérpretes de IPython con todas las funciones . Cada consola se ejecuta en un proceso separado, lo que le permite ejecutar scripts, interrumpir la ejecución y reiniciar o terminar un shell sin afectar a los demás o al propio Spyder, y probar fácilmente su código en un entorno limpio sin interrumpir su sesión principal [57].

Explorador de variables muestra el contenido de todas las variables por medio de sus nombres, todas las referencias de objetos globales, como variables, funciones y módulos. La sesión de la Consola IPython seleccionada actualmente permite interactuar con ellos a través de una variedad de editores [57].

Panel del Perfilador recursivamente determina el tiempo de ejecución y el número de llamadas para cada función y método llamado del archivo, ya sea directa o indirectamente, desglosando cada procedimiento en sus unidades individuales más pequeñas. Esto le permite identificar fácilmente los cuellos de botella en su código, lo dirige hacia las declaraciones exactas más críticas para la optimización y mide el delta de rendimiento después de los cambios de seguimiento [57].

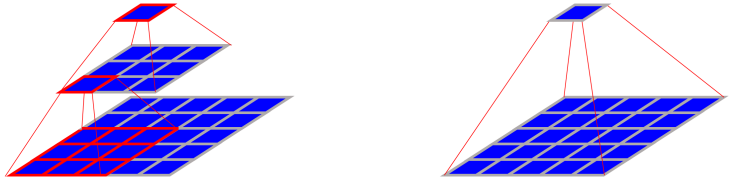
Depuración de la Consola de IPython permite que los puntos de interrupción y de flujo de ejecución se puedan ver y controlar directamente desde la GUI de Spyder, así como con todos los comandos de consola familiares de IPython [57].

5.3. Arquitecturas CNN

La arquitectura de una red neuronal artificial es la forma como se organizan las neuronas en su interior y está estrechamente ligada al algoritmo de aprendizaje usado para entrenar una red neuronal. En esta sección se presenta una breve revisión de las arquitecturas *Inception V3*, *NASNet*, y *MobileNet* con la finalidad de comprender su funcionamiento para poder implementarlas a la clasificación de piezas prehispánicas de la zona arqueológicas de Tula.

5.3.1. Inception V3

Inception de GoogLeNet fue diseñada para funcionar con bajas restricciones en memoria y recursos computacionales. El costo computacional de Inception es más bajo que sus sucesores de mayor rendimiento. Esto hace posible utilizar redes con una gran cantidad de imágenes, donde es necesario procesar una gran cantidad de datos a un costo razonable, donde la memoria o la capacidad computacional son limitadas. Ciertamente es posible mitigar parte de este problema mediante la aplicación de soluciones especializadas, a través de la optimización de ciertas operaciones [59].



(a) Convoluciones 3×3 apiladas (b) Convolución 5×5

Figura 5.3: Convoluciones

El modelo se compone de una unidad básica denominada *Inception cell* en la que realiza una serie de convoluciones a diferentes escalas. Para minimizar los cálculo, se utilizan convoluciones 1×1 para reducir la profundidad del canal de entrada. Por otra parte, cada celda aplica un conjunto de filtros

1×1, 3×3 y 5×5 que pueden aprender a extraer características a diferentes escalas de la entrada. La función *Maxpooling* se usa para preservar las dimensiones de modo que la salida se pueda concatenar adecuadamente.

Para esta arquitectura, las convoluciones con filtros espacialmente grandes como 5×5 o 7×7, se benefician en términos de su capacidad para extraer características a una escala mayor; sin embargo, el cálculo es desproporcionadamente costoso. Los investigadores señalaron que una convolución de 5×5 puede ser representada de manera más económica por dos filtros de 3×3 apilados [59]. También, se demostró que las convoluciones 3×3 podrían construirse en convoluciones 3×1 y 1×3 apiladas una en otra como se muestra en la Fig. 5.4, mejoran el rendimiento general de la red al agregar dos salidas auxiliares en toda la red para mejorar el rendimiento final del modelo.

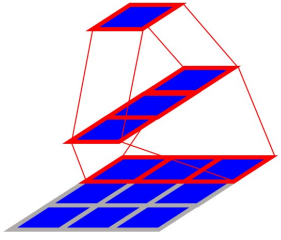


Figura 5.4: convoluciones 3×1 y 1×3.

La Fig. 5.5 muestra una versión revisada y más profunda de la red *Inception V3* que aprovecha las celdas *Inception* más eficientes.

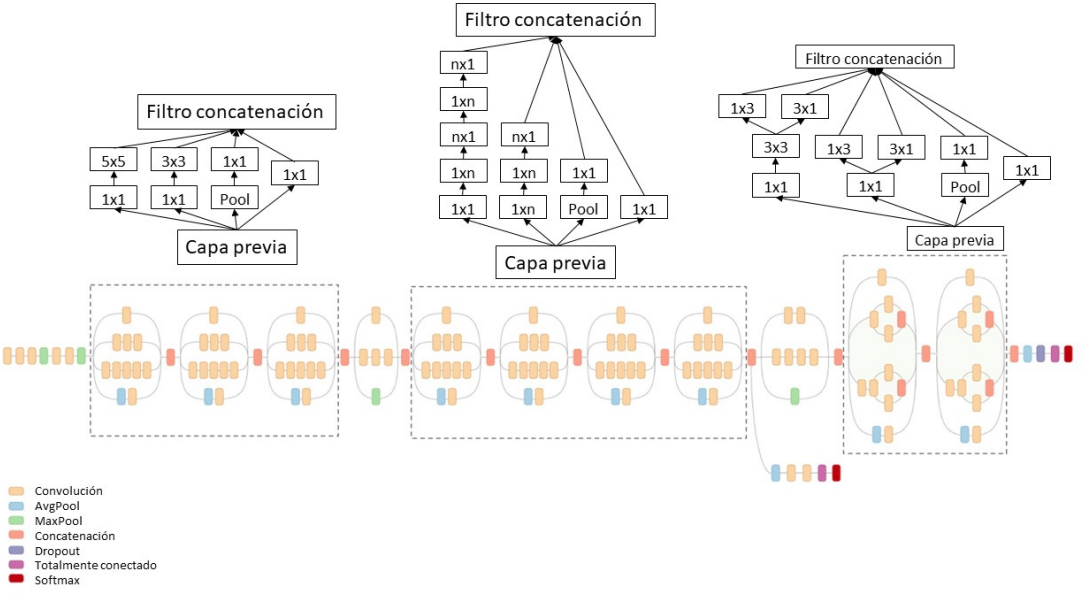


Figura 5.5: Aequitectura Inception.

5.3.2. NASNet

El enfoque de *NASNet* se inspira en el marco de búsqueda de la arquitectura neuronal (NAS), que utiliza un método de búsqueda de aprendizaje por refuerzo para optimizar las configuraciones de la arquitectura. Sin embargo, aplicar NAS, o cualquier otro método de búsqueda, directamente a un gran conjunto de datos, es computacionalmente costoso. Por lo tanto, se busca un conjunto de datos más pequeño, y luego transfiere la arquitectura aprendida a ImageNet. Se logra esta capacidad de transferencia mediante el diseño de un espacio de búsqueda la cual la llaman espacio de búsqueda NASNet [72].

En NASNet la arquitectura general está predefinida como se muestra en la Fig. 5.6, los bloques o celdas no están predefinidos por los autores. Se buscan mediante el método de búsqueda de aprendizaje por refuerzo, es decir, el número de repeticiones N y el número de filtros convolucionales iniciales son parámetros libres y se usan para la escalar [72].

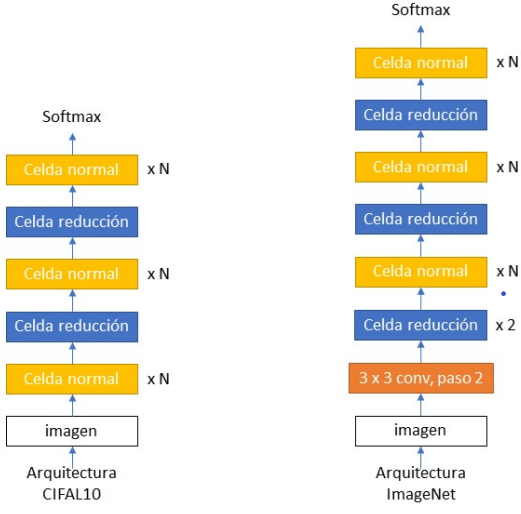


Figura 5.6: Arquitecturas escalables para CIFAR-10 e ImageNet

Se necesitan dos tipos de celdas convolucionales para cumplir dos funciones principales cuando se toma un mapa de características como entrada: las celdas convolucionales que devuelven un mapa de características de la misma dimensión, y las celdas convolucionales que devuelven un mapa de características donde la altura y el ancho del mapa de características se reducen en un factor de dos, se le llaman convolucionales Celda normal y Celda de reducción respectivamente [72].

Las predicciones del controlador para cada celda se agrupan en bloques B como se muestra en la Fig. 5.7, donde cada bloque tiene 5 pasos de predicción realizados por 5 clasificadores softmax distintos que corresponden a elecciones discretas de los elementos de un bloque:

Paso 1 : Seleccione un estado oculto desde h_i , h_{i-1} o del conjunto de estados ocultos creados en los bloques anteriores.
Paso 2 : Seleccionar un segundo estado oculto a las mismas opciones que en el Paso 1.
Paso 3 : Seleccione una operación para aplicar al estado oculto seleccionado en el Paso 1.
Paso 4 : Seleccione una operación para aplicar al estado oculto seleccionado en el Paso 2.
Paso 5 : Seleccione un método para combinar las salidas de los pasos 3 y 4 para crear un nuevo estado oculto.

El conjunto de operaciones para seleccionar:

- identity
- 1×3 then 3×1 convolution
- 1×7 then 7×1 convolution
- 3×3dilated convolution
- 3×3 average pooling
- 3×3 max pooling
- 5×5 max pooling
- 7×7 max pooling
- 1×1 convolution
- 3×3 convolution
- 3×3 depthwise-separable conv
- 5×5 depthwise-seperable conv
- 7×7 depthwise-separable conv

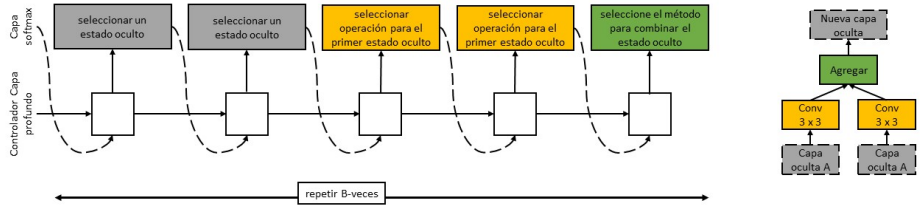
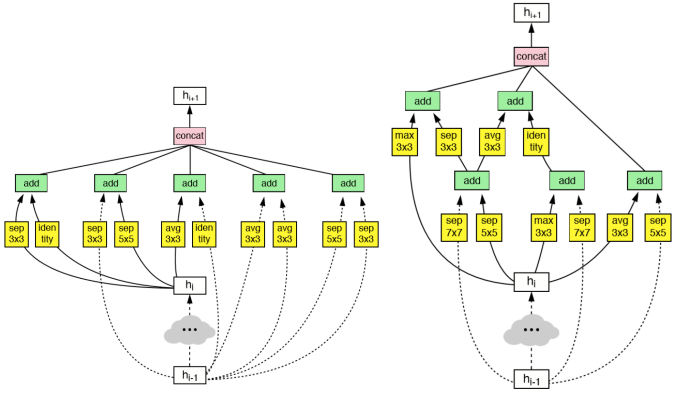


Figura 5.7: Arquitectura del controlador para construir recursivamente un bloque de una celda convolucional

La Fig 5.8 muestra un diagrama de la celda normal y la celda de reducción de mayor rendimiento. Note que la prevalencia de convoluciones separables y el número de ramas en comparación.



(a) celda normal (b) celda de reducción

Figura 5.8: Arquitectura celdas convolucionales.

5.3.3. MobileNet

Desde que AlexNet propone a las redes neuronales convolucionales profundas para el desarrollo en la visión por computadora al ganar el Desafío ImageNet. La tendencia ha sido hacer redes más profundas y complicadas

para lograr una mayor precisión. Sin embargo, estos avances para mejorar la precisión no necesariamente hacen que las redes sean más eficientes con respecto al tamaño y la velocidad. En muchas aplicaciones del mundo real, como la robótica, el auto-manejo y la realidad aumentada.

MobileNet fue diseñado para maximizar la precisión de manera efectiva, teniendo en cuenta los recursos restringidos para una aplicación o un dispositivo. MobileNets son modelos pequeños, de baja latencia y baja potencia para cumplir con las limitaciones de recursos de una variedad de casos de uso. Se pueden construir para la clasificación, detección y segmentación de forma similar a cómo se utilizan otros modelos populares a gran escala, como Inception.

MobileNet se basa en cuatro capas centrales, que son convolución separable profunda, la estructura de red y entrenamiento, por ultimo la descripción de los dos modelos de reducción de hiperparámetros multiplicador de amplitud y el multiplicador de resolución.

Convolución separable en profundidad

El modelo de MobileNet se basa en convoluciones separables en profundidad que es una forma de convoluciones factorizadas que factorizan una convolución estándar en una convolución en profundidad y una convolución 1×1 convalidada a una circunvolución en punto. Para MobileNets, la convolución en profundidad aplica un único filtro a cada canal de entrada. La convolución puntual a la que se aplica una convolución 1×1 se combina con la salida de la convolución profunda. Esta factorización tiene el efecto de reducir drásticamente el cálculo y el tamaño del modelo. La Fig. 5.9 muestra cómo una convolución se factoriza en una convolución profunda y una convolución puntual [26].

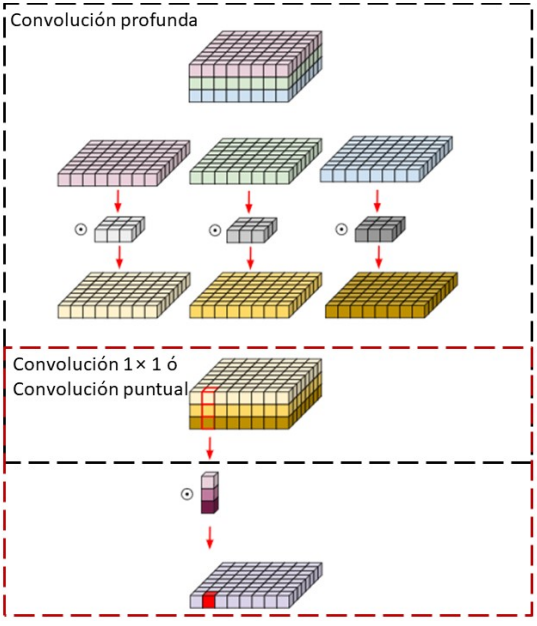


Figura 5.9: Convoluciones separables en profundidad

Estructura de red y entrenamiento

Como se menciona en la sección anterior, la estructura de MobileNet se basa en convoluciones separables en profundidad, excepto la primera capa, que es una convolución completa. Todas las capas son seguidas por una batchnorm y una no linealidad ReLU, con la excepción de la capa final completamente conectada que no tiene linealidad y alimenta a una capa softmax para su clasificación. La Fig. 5.10 contrasta una capa con convoluciones, la batchnorm y no linealidad ReLU con la capa factorizada con convolución profunda, convolución puntual 1 x 1, así como la batchnorm y ReLU después de cada capa convolucional. El pooling promedio reduce la resolución espacial a 1 antes de la capa completamente conectada. Contando las convoluciones profundas y puntuales como capas separadas, MobileNet tiene 28 capas [26].

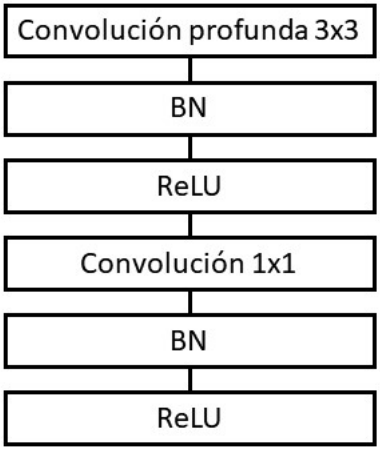


Figura 5.10: Depthwise Separable convolutions

Al contrario de entrenar modelos grandes, se usan técnicas de regularización y aumento de datos porque los modelos pequeños tienen menos problemas con el sobreajuste. Al entrenar MobileNets, se reduce la cantidad de imágenes de distorsiones al limitar el tamaño de las muestras que se usan en el entrenamiento inicial. Además, es importante poner muy poca pérdida de peso en los filtros profundos ya que hay muy pocos parámetros en ellos [26].

Multiplicador de amplitud y de resolución

La arquitectura básica de MobileNet es pequeña y de baja latencia, el rol del multiplicador de amplitud es para adelgazar una red de manera uniforme en cada capa. El multiplicador de amplitud tiene el efecto de reducir el costo computacional y el número de parámetros. También, se utiliza para definir una nueva estructura reducida que necesita ser entrenada desde cero. El segundo hiperparámetro reduce el costo computacional de una red neuronal y controla la resolución de la imagen de entrada de la red.

Código de MobileNet.

Para declarar las variables de entorno en Anaconda es de la siguiente forma Alg. 5.1. Donde $IMAGE\_SIZE$ es la resolución de imagen de entrada con valores 128, 160, 192, o 224 píxeles. A mayor resolución es más tiempo de procesamiento, pero mejora la precisión de clasificación. $ARCHITECTURE$ porcentaje del modelo de MobileNet con valores 1.0, 0.75, 0.50 o 0.25 [23].

```
set IMAGE_SIZE=224
set ARCHITECTURE="␣mobilenet_0.50_␣%IMAGE_SIZE␣%"
```

Algoritmo 5.1: Variables de entorno.

El modelo se entrena con el Alg. 5.2. Donde, *scripts.retrain* es el scripts del modelo. El modelo requiere los siguientes parámetros: *bottleneck_dir*, *model_dir* y *summaries_dir* son rutas de almacenamiento de bottleneck, el modelo y reportes de entrenamiento, *how_many_training_steps* numero de iteraciones del model, *output_graph* y *output_labels* rutas de almacenamiento de la gráfica y las etiquetas, por ultimo *image_dir* directorio donde se encuentra el dataset [23].

```
python -m scripts.retrain
--bottleneck_dir=tf_files\bottlenecks
--how_many_training_steps=5000
--model_dir=tf_files\models\
--summaries_dir=tf_files\training_summaries\" %
    ARCHITECTURE% %
--output_graph=tf_files\retrained_graph.pb
--output_labels=tf_files\retrained_labels.txt
--architecture=" %ARCHITECTURE% %
--image_dir=tf_files\photos
```

Algoritmo 5.2: Entrenamiento de MobileNet.

La forma de probar el modelos se muestra con el siguiente Alg. 5.3. Donde *graph* y *image* es la ruta de la gráfica y de la imagen de prueba.

```
python -m scripts.label_image
--graph=tf_files\retrained_graph.pb
--image=tf_files\photos\Cariatide\Cariatide_1_0.jpg
```

Algoritmo 5.3: Prueba de Entrenamiento.

Entrenamiento de MobileNet

En la sección anterior se explica como el modelo depende de dos variables, en la Tabla 5.1 se muestra una evaluación comparativa dependiendo del valor de la resolución de la imagen y el porcentaje del modelo.

Resolución de imagen	Porcentaje del modelo	Nivel de confianza de entrenamiento	Cariatide	Chac_mool	Coatepantli	Palacio_quemado	Piramide_B
224	0.5	79.7 %	Cariatide (score=0.99995) Chac_mool (score=0.00001) Coatepantli (score=0.00004) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00004) Chac_mool (score=0.99996) Coatepantli (score=0.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00000) Chac_mool (score=0.00000) Coatepantli (score=1.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.02096) Chac_mool (score=0.00098) Coatepantli (score=0.00000) Palacio_quemado (score=0.97798) Piramide_B (score=0.00008)	Cariatide (score=0.00100) Chac_mool (score=0.01023) Coatepantli (score=0.00021) Palacio_quemado (score=0.00001) Piramide_B (score=0.98855)
224	0.75	79.6 %	Cariatide (score=0.99842) Chac_mool (score=0.00078) Coatepantli (score=0.00081) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00000) Chac_mool (score=1.00000) Coatepantli (score=0.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00000) Chac_mool (score=0.00000) Coatepantli (score=1.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00001)	Cariatide (score=0.00108) Chac_mool (score=0.10685) Coatepantli (score=0.00001) Palacio_quemado (score=0.76311) Piramide_B (score=0.12895)	Cariatide (score=0.02096) Chac_mool (score=0.00098) Coatepantli (score=0.00548) Palacio_quemado (score=0.00008) Piramide_B (score=0.97798)
224	1.0	80.8 %	Cariatide (score=0.94814) Chac_mool (score=0.00561) Coatepantli (score=0.04066) Palacio_quemado (score=0.00432) Piramide_B (score=0.00126)	Cariatide (score=0.00007) Chac_mool (score=0.99993) Coatepantli (score=0.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00000) Chac_mool (score=0.00000) Coatepantli (score=1.00000) Palacio_quemado (score=0.00000) Piramide_B (score=0.00000)	Cariatide (score=0.00021) Chac_mool (score=0.00047) Coatepantli (score=0.00000) Palacio_quemado (score=0.62741) Piramide_B (score=0.37191)	Cariatide (score=0.00070) Chac_mool (score=0.00056) Coatepantli (score=0.00000) Palacio_quemado (score=0.14438) Piramide_B (score=0.85436)

Tabla 5.1: Tabla comparativa

5.4. Entrenamiento de la Red Neuronal Convolucional para la clasificación de piezas Prehispanicas

En este trabajo de tesis se realiza una comparación de las arquitecturas *Inception V3*, *NASNet*, y *MobileNet*. El procedimiento que se utilizó para el entrenamiento de las arquitecturas se muestra en la Fig. 5.11. Primero se genera el dataSet que se desarrolla en la Sec. 4.2. El siguiente paso es el entrenamiento con cada una de las arquitecturas, si el porcentaje de clasificación es mayor al 80 % se acepta el modelo, en caso contrario se modifican los parámetros dependiendo de cada arquitectura. Si a pesar de cambiar los parámetros, la arquitectura no permite generar una mejor clasificación, se tiene que proponer un nuevo dataSet.

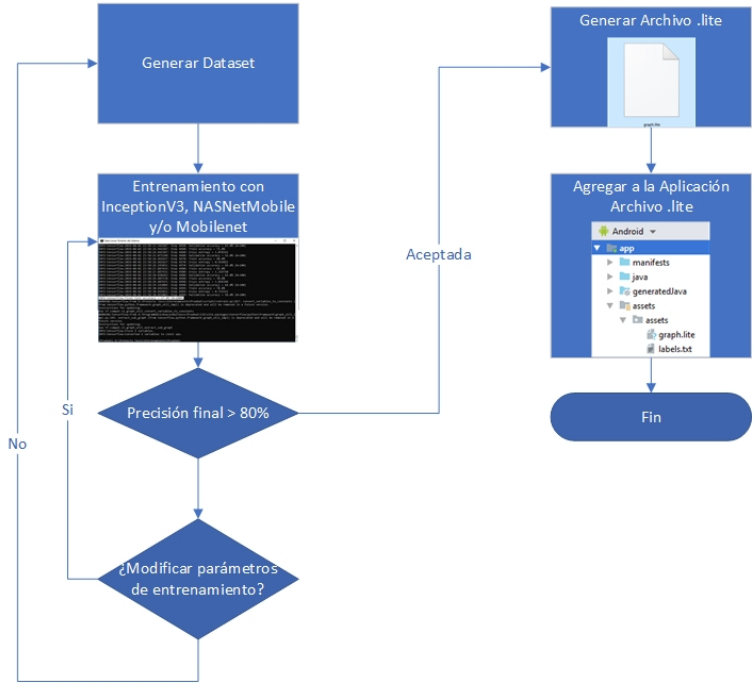


Figura 5.11: Diagrama de flujo de entrenamiento del modelo

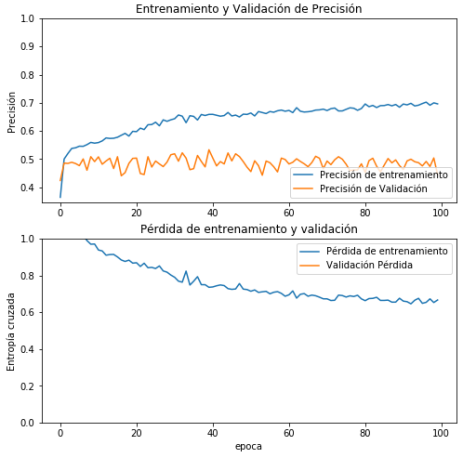
5.4.1. Comparación de Arquitecturas.

Para comparar los entrenamientos se toman las arquitecturas con un número de épocas igual a 100. Como referencia se considera el porcentaje y que tan rápido converge, tanto en la precisión del entrenamiento como en función de la perdida, además del tiempo que se tardan en entrenar. Los datos del entrenamiento se muestran en la Tabla 5.2.

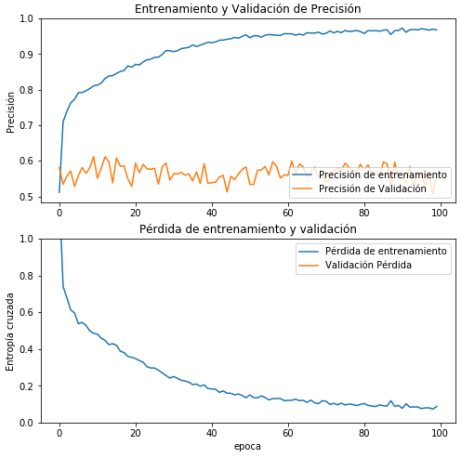
Arquirectura	Tiempo de entrenamiento en segundos	Porcentaje final de precisión	Porcentaje final de perdida
InceptionV3	31326	69.61 %	66.72 %
NASNetMobile	86958	96.74 %	8.74 %
Mobilenet	60626	95.95 %	10.61 %

Tabla 5.2: Tabla comparativa de las Arquitecturas.

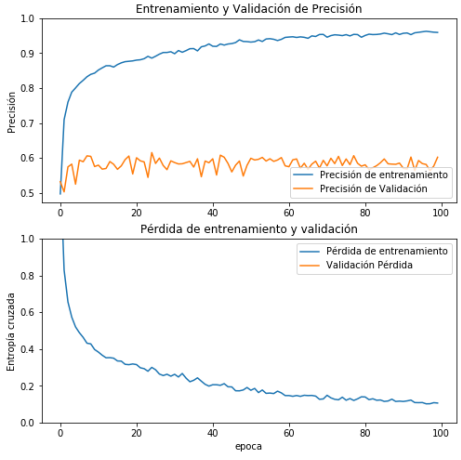
InceptionV3 en comparación con las demás arquitecturas el tiempo de entrenamiento es menor; sin embargo, tiene una precisión muy baja. En cuanto a NASNetMobile, tiene el mejor porcentaje de precisión y perdida, su desventaja es el tiempo de cómputo; es el más costoso en cuanto a recursos de los tres. Por último, Mobilenet de acuerdo a las Fig. 5.12(c) converge mas rápido que las otra dos arquitecturas y tiene mejor tiempo de cómputo que NASNetMobile.



(a)



(b) NASNetMobile



(c) Mobilenet

Figura 5.12: Gráficas de resultados de entrenar InceptionV3, NASNetMobile y Mobilenet.

Mobilenet y NASNetMobile tiene valores muy similares en cuanto a la precisión, perdida y convergencia. Pero NASNetMobile requiere mas recurso computacionales lo que se refleja en el tiempo de entrenamiento.

5.5. TensorFlow Lite y Android Studio

TensorFlow Lite es la solución ligera de TensorFlow para dispositivos móviles y sistema embebido. Permite la inferencia de aprendizaje automático en el dispositivo con baja latencia y un tamaño binario pequeño. También, admite la aceleración de hardware con la API de redes neuronales de Android, usa técnicas como la optimización de los kernels para aplicaciones móviles, kernels cuantificados que permiten modelos más pequeños y más rápidos [61].

TensorFlow Lite define un nuevo formato de archivo de modelo, basado en FlatBuffers. FlatBuffers es una eficiente biblioteca de serialización multiplataforma de código abierto. Es similar a los búferes de protocolo, pero la principal diferencia es que FlatBuffers no necesita un paso de análisis o desempaquetado en una petición indirecta antes de poder acceder a los datos, a menudo junto con la asignación de memoria por objeto. Además, la huella del código de FlatBuffers es un orden de magnitud más pequeño que los buffers de protocolo [61]. También, tienen un nuevo intérprete optimizado para dispositivos móviles, que tiene como objetivos clave mantener las aplicaciones ágiles y rápidas. El intérprete utiliza un ordenamiento gráfico estático y un asignador de memoria personalizado (menos dinámico) para garantizar la carga mínima, la inicialización y la latencia de ejecución. TensorFlow Lite proporciona una interfaz para aprovechar la aceleración de hardware, si está disponible en el dispositivo, lo hace a través de la API de redes neuronales de Android , disponible en Android 8.1 (nivel de API 27) y superior.

Arquitectura La arquitectura de TensorFlow Lite se divide en varias capas la primera comienza con el modelo entrenado de TensorFlow, después se convierte ese modelo al formato de archivo TensorFlow Lite (.tflite) usando el convertidor TensorFlow Lite. Por ultimo, se usa el archivo convertido en la aplicación móvil Fig. 5.13.

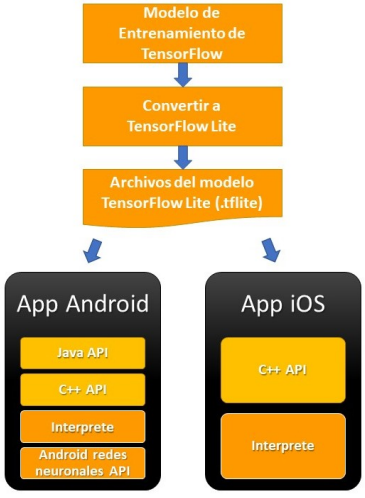


Figura 5.13: Diseño arquitectónico de TensorFlow Lite

La implementación del archivo de modelo TensorFlow Lite utiliza:

- API de Java: encapsula alrededor de la API de C ++ en Android.
- API de C ++: carga el archivo del modelo de TensorFlow Lite e invoca al intérprete.
- Intérprete: ejecuta el modelo utilizando un conjunto de kernels. El intérprete soporta la carga selectiva del kernel.
- API de redes neuronales de Android es usado por el interprete para la aceleración de hardware de manera predeterminada o la ejecución de la CPU.

Linea de comando Los dispositivos móviles tienen limitaciones importantes, por lo que vale la pena considerar cualquier preprocesamiento que se pueda hacer para reducir la carga de una aplicación. Con TFLite se incluye un convertidor de gráficos. Este programa se llama *TensorFlow Lite Optimizing Converter* o *tflite_convert*. A partir de TensorFlow 1.9, la herramienta de línea de comandos *tflite_convert* se instala como parte del

paquete Python [61]. Descuerdo al capítulo anterior se obtiene el archivo *retrained_graph.pb* como producto del entrenamiento de la RNC y el archivo *optimized_graph.lite* se obtiene al hacer la convección tensor de TensorFlow lite, como se muestra a continuación Alg: 5.4.

```
IMAGE_SIZE=224
tflite_convert \
  --graph_def_file=tf_files/retrained_graph.pb \
  --output_file=tf_files/optimized_graph.lite \
  --input_format=TENSORFLOW_GRAPHDEF \
  --output_format=TFLITE \
  --input_shape=1,\${IMAGE_SIZE},\${IMAGE_SIZE},3 \
  --input_array=input \
  --output_array=final_result \
  --inference_type=FLOAT \
  --input_data_type=FLOAT
```

Algoritmo 5.4: TensorFlow Lite Optimizing Converter

5.6. Configuracion de Android Studio

Para poder mandar llamar la librería en Android Studio, se utiliza un archivo precompilado TFLite Android Archive (AAR) alojado en JCenter. Esta configuración se hace en archivo *build.gradle* de la siguiente manera:

```
dependencies {
    compile 'org.tensorflow:tensorflow-lite:+'
}
```

Lo siguiente es instruir el Android Packaging Tool, que el archivo *.lite* o *.tflite* no debe ser comprimido. Esto es importante ya que el archivo *.lite* se asignará a la memoria y no funcionará cuando se comprima el archivo. La configuración es en archivo *build.gradle* como se muestra a continuación:

```
android {
    aaptOptions {
        noCompress "tflite"
        noCompress "lite"
    }
}
```

Lo siguiente es el constructor de la clase en java, se crea una instancia del intérprete de TFLite. El intérprete hace el trabajo de tf.Session. Se pasa el intérprete a $MappedByteBuffer$ el cual contiene el modelo. La función local $loadModelFile$ crea un archivo de activos de $MappedByteBuffer$ la actividad que contiene *graph.lite*.

```
tflite = new Interpreter(loadModelFile(activity));
```

En el constructor crear el búfer de datos de entrada. Este búfer de bytes contiene los datos de la imagen una vez convertidos a flotante. El intérprete puede aceptar matrices flotantes directamente como entrada, pero $ByteBufferes$ es más eficiente ya que evita copias adicionales en el intérprete.

```
imgData = ByteBuffer.allocateDirect(
    4 * DIM_BATCH_SIZE * DIM_IMG_SIZE_X * DIM_IMG_SIZE_Y *
        DIM_PIXEL_SIZE
);
```

Algoritmo 5.5: búfer de datos de entrada

Las siguientes líneas es cargan la lista de etiquetas y crean el búfer de salida. El búfer de salida es una matriz flotante con un elemento para cada etiqueta donde el modelo escribirá las probabilidades de salida.

```
labelList = loadLabelList(activity);
//...
labelProbArray = new float[1][labelList.size()];
```

Algoritmo 5.6: búfer de datos de salida

El método *classifyFrame* toma *Bitmap* como entrada, ejecuta el modelo y devuelve el texto para imprimir en la aplicación. Primero convierte y copia la entrada *Bitmap* a la *imgData ByteBuffer* entrada para el modelo. Luego llama al método de ejecución del intérprete, pasando el búfer de entrada y la matriz de salida como argumentos. El intérprete establece los valores en la matriz de salida a la probabilidad calculada para cada clase. Los nodos de entrada y salida están definidos por los argumentos para el *toco* paso de conversión que creó el *.lite*.

```
String classifyFrame(Bitmap bitmap) {
// ...
convertBitmapToByteBuffer(bitmap);
// ...
tflite.run(imgData, labelProbArray);
// ...
String textToShow = printTopKLabels();
// ...
}
```

Algoritmo 5.7: método classifyFrame

En cuanto al código en java esta pensado a futuro, cuando el museo se encuentre renovado y se hayan definido las piezas que se mostraran al publico solo habrá que agregar las nuevas reseñas y nuevo modelo entrenado. El código completo se muestra en el apéndice B para su análisis.

5.7. Layout de Android Studio

La aplicación móvil cuenta con Layout específicos para cada tarea: inicio, reconocimiento de piezas, reseñas, costos y horarios. Con ello el turista que visite la zona tendrá acceso a la información de manera fácil y rápida Fig 5.14. El código de los Layout se encuentra en el Ape. B.

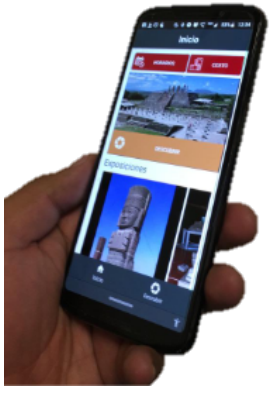


Figura 5.14: Aplicación

La vista inicio es la puerta de acceso a toda la aplicación y se divide en tres secciones; comienza por la sección de información general, describe los horarios y los costos de la zona arqueológica, la siguiente sección es el acceso al visor de reconocimiento y por último un scroll el cual muestra las piezas expuestas en el museo. Fig 5.15(a).

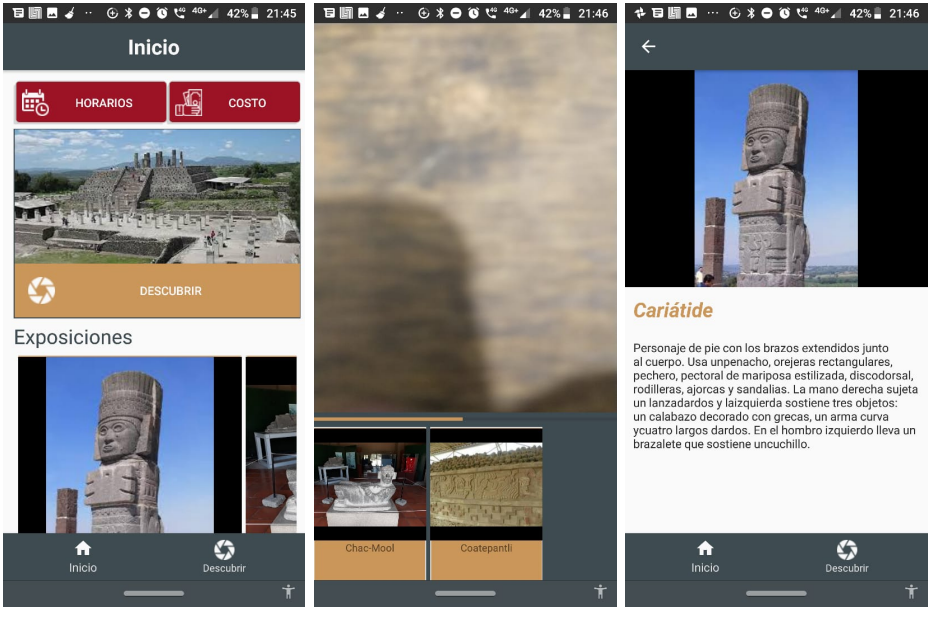
La vista de reconocimiento de piezas engloba prácticamente todo el trabajo visto en los capítulos de la tesis. Se divide en dos secciones como se muestra en la Fig. 5.15(b), la primera sección muestra la imagen captada por la cámara, esta imagen es enviada a el modelo previamente entrenado en TensorFlow. La siguiente sección es un scroll de imágenes ubicado en la parte inferior de la aplicación, el cual despliega las piezas con mayor similitud al imagen captada, ubicando la más semejante a la izquierda hasta las menos semejante a la derecha.

La vista reseña muestra una imagen y la descripción del objeto como se describe en la Fig. 5.15(c), presenta la información oficial de la pieza, por ejemplo, el material que esta echo, el arqueólogo que la encontró o que la clasifico, cual era el uso que tenia en la cultura Tolteca, etc.

La siguiente vista Fig. 5.15(c) muestra los costos actualizados de la entrada, descuentos y/o entrada libre por día de la semana. Ayudando al

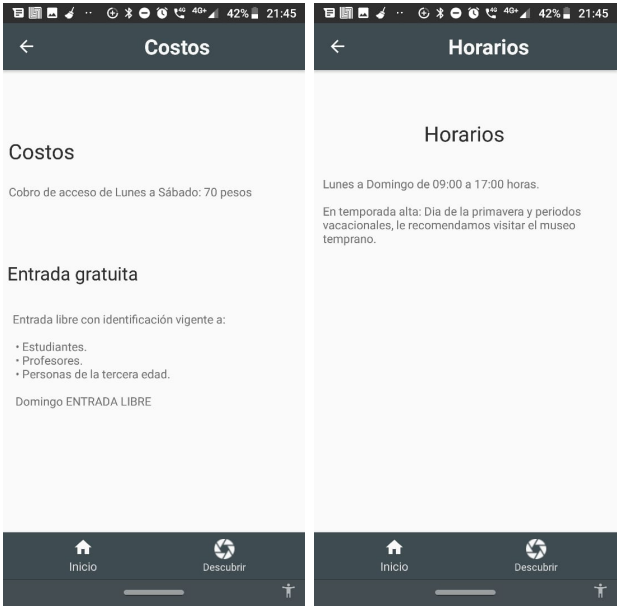
turista a tener una mejor organización al visitar el sitio.

Por ultimo los horarios de la entrada por día de la semana e información de los días con mayor afluencia de visitantes como en la Fig 5.16(b). El propósito es ayudar al turista a tener una mejor planeación en su visita.



(a) Inicio (b) Reconocimiento (c) Reseña

Figura 5.15: Aplicación



(a) Costos (b) Horario

Figura 5.16: Aplicación

Capítulo 6

Conclusiones

6.1. Conclusiones

En este trabajo de tesis se realizó la revisión bibliográfica de los conceptos básicos de inteligencia artificial hasta llegar a redes neuronales convolucionales. Se revisaron conceptos básicos, tales como: neurona, redes neuronales de una capa o multicapa, el procesamiento de los datos de una red neuronal como la funciones de propagación, activación y salida, regla delta, propagación hacia atrás. Además, de la convolución de imágenes y sus características más importantes para poder entender las redes neuronales convolucionales, entre las arquitecturas más comunes se revisaron LeNet y MobileNet. Aunque no usamos el algoritmo de LeNet es importante estudiarlo por ser el primero y la base de las demás arquitecturas. Por otra parte, MobileNet es un modelo para móviles, que utiliza convoluciones separables en profundidad para construir redes neuronales profundas y livianas.

También, en las Secciones 4.4 y 5.4 se realizaron una revisión de las principales herramientas para el desarrollo de aplicaciones de inteligencia artificial, tales como TensorFlow, Keras y NumPy, las cuales facilitan el diseño de sistemas de visión por computadora basados en redes neuronales.

De igual manera, se crearon distintos ambientes en Anaconda para simplificar el despliegue y administración de los paquetes de software utilizado en ciencia de datos, y aprendizaje automático; ya sea para entrenamiento

del modelo con diferentes parámetros de entrada, generar el conjunto de imágenes o la paquetería de TensorFlow lite. Además, cuenta con aplicaciones muy útiles como el editor Spyder que fue utilizado en este trabajo de tesis para generar los algoritmos de aumento de datos.

A pesar de que la Zona Arqueológica de Tula es de gran importancia en el estado de Hidalgo, no cuenta con un banco de imágenes amplio de las piezas arqueológicas, por lo cual se realizo uno a partir de imágenes de internet con los plugin Image Downloader e I'm a Gentleman. Aunque, estos dos plugins fueron bastante útiles al descargar de forma masiva las imágenes, no fue fácil debido a las múltiples búsquedas que se tuvieron que hacer para obtener el mayor número de imágenes de cada objeto sin que estuvieran repetidas y clasificar las imágenes de acuerdo al objeto.

Por otra parte, el proceso de aprendizaje de una red neuronal convolucional, fue necesario repetir en múltiples ocasiones el entrenamiento cambiando el tamaño de resolución de la imagen, el tamaño del modelo o modificar el dataset, hasta dar con los ajustes adecuados para que el entrenamiento obtuviera resultados razonablemente satisfactorios.

Por último, se desarrollo una aplicación que cubre los objetivos de la tesis, presenta una vista agradable al usuario, información útil para visitar la zona arqueológica como los costos y horarios de entrada. La aplicación consta de una galería de 5 piezas que se encuentran en la zona y una reseña de ellas. Si te encuentras en la Zona Arqueológica de Tula puedes captar con la cámara alguna de las 5 piezas y tener su información.

6.2. Producto derivados del trabajo de tesis

Con apoyo coordinación de la Maestría en Desarrollo de Software se gestiono en Indautor los de autor y el registro a nombre de Dr. César Joel Camacho Bello, Mtro. Jorge Alberto Hernández Tapia y Ivan Mac Gregor Oviedo Dorantes.

6.3. Trabajo a futuro

Quiero destacar que esta aplicación tiene múltiples posibilidades de mejora. Algunas de las mejoras que se podrían plantear son:

- Completar la base de datos de todas las piezas y pirámides de la zona arqueológica para poder ofrecer una aplicación lo bastante completa y útil a los usuarios.
- Investigar sobre metodología para no generar un sobre ajuste.
- Agregar geolocalización a la aplicación para ayudar al usuario a localizar el sitios de las piezas o pirámides.
- Agregar realidad aumentada para hacer la aplicación más amigable.
- Hacer la aplicación para iOs.
- Comparar con otras arquitecturas de reconocimiento de objetos.

Apéndice A

Algoritmos de Aumento de datos

```
# -*- coding: utf-8 -*-
"""

@author: iMac
"""
import random
from tensorflow.python.keras.preprocessing.image import
    ImageDataGenerator, img_to_array
from numpy import expand_dims

def filtroImageDataGenerator(imagen,save_to_dir,save_prefix):

    datagen = ImageDataGenerator(
            rotation_range=random.randrange(0, 360),
            zoom_range=random.randrange(0, 10)/10,
            horizontal_flip=True,
            fill_mode='nearest')

    data = img_to_array(imagen)
    x = expand_dims(data, 0)
    datagen.flow(x, batch_size=1, save_to_dir=save_to_dir,
        save_prefix=save_prefix, save_format='jpg').next()
    return ''
```

Algoritmo A.1: Filtros ImageDataGenerator

```
# -*- coding: utf-8 -*-
"""

@author: iMac
"""
import random
from PIL import ImageFilter, ImageEnhance
from SalyPimienta import salyPimienta

def filtroPill(imagen):
    caso=random.randrange(0, 11)

    if(caso==0):
        imagenPill = imagen.filter(ImageFilter.BLUR)
    if(caso==1):
        imagenPill = imagen.filter(ImageFilter.DETAIL)
    if(caso==2):
        imagenPill = imagen.filter(ImageFilter.EDGE_ENHANCE)
    if(caso==3):
        imagenPill = imagen.filter(ImageFilter.
            EDGE_ENHANCE_MORE)
    if(caso==4):
        imagenPill = imagen.filter(ImageFilter.SMOOTH)
    if(caso==5):
        imagenPill = imagen.filter(ImageFilter.SMOOTH_MORE)
    if(caso==6):
         imagenPill = ImageEnhance.Color(imagen).enhance(random
             .randrange(0, 6)/10)
    if(caso==7):
         imagenPill = ImageEnhance.Contrast(imagen).enhance(
             random.randrange(0, 6)/10)
    if(caso==8):
         imagenPill = ImageEnhance.Brightness(imagen).enhance(
             random.randrange(0, 6)/10)
    if(caso==9):
         imagenPill = ImageEnhance.Sharpness(imagen).enhance(
             random.randrange(0, 6)/10)
    if(caso==10):
        imagenPill=salyPimienta(imagen,10)

    return imagenPill
```

Algoritmo A.2: Filtros Pill

```
# -*- coding: utf-8 -*-
"""

@author: iMac
"""
import glob
import os
from PIL import Image
from FiltroPill import filtroPill
from FiltroImageDataGenerator import filtroImageDataGenerator

def dataset(url1,url2,num):
    etiquetas=glob.glob(url1+"/*")
    numMax=arrNum(etiquetas, url1)

    for etiqueta in etiquetas:
        carpeta=etiqueta.split('\\')[1]
        images=glob.glob(url1+"/"+carpeta+"/*.jpg")
        save_to_dir=url2+"/"+carpeta
        if not os.path.exists(save_to_dir):
            os.mkdir(save_to_dir)

        cont=round(num*numMax/len(images))-1
        print(cont)
        i=0
        for image in images:
            print(image)

            imagen = Image.open(image)
            imagen = imagen.convert("RGB")
            imagen.mode    # RGB
            imagen.save(save_to_dir+"/"+carpeta+"_"+str(i)+"_0.
                jpg")
            for j in range(cont):
                imagenFP=filtroPill(imagen)
                filtroImageDataGenerator(imagenFP,save_to_dir,
                    carpeta+"_"+str(i))
            i=i+1
    return ''

def arrNum(etiquetas, url1):
    lista = []
    mayor=0
    for etiqueta in etiquetas:
```

```
        carpeta=etiqueta.split('\\')[1]
        images=glob.glob(url1+"/"+carpeta+"/*.jpg")
        lista.append(len(images))
        if (len(images)>mayor):
            mayor=len(images)
    return mayor
```

Algoritmo A.3: DataSet

Apéndice B

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="
    http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MenuActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toTopOf="@+id/navigation"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
```

```
            android:layout_marginLeft="5dp"
            android:layout_marginTop="5dp"
            android:layout_marginRight="5dp"
            android:orientation="horizontal">

            <Button
                android:id="@+id/btnHorario"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/
                    button_selector"
                android:drawableLeft="@drawable/horario"
                android:text="Horarios"
                android:textColor="@color/textColorPrimary"
                    />

            <Button
                android:id="@+id/btnCosto"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/
                    button_selector"
                android:drawableLeft="@drawable/costo"
                android:text="Costo"
                android:textColor="@color/textColorPrimary"
                    />

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_margin="5dp"
            android:background="@drawable/border_shadow"
            android:orientation="vertical"
            android:padding="1dp">

            <ImageView
                android:id="@+id/ivDescubrir"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
```

```
            app:srcCompat="@drawable/imag_reco" />

        <Button
            android:id="@+id/btnDescubrir"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorDorado"
            android:drawableLeft="@drawable/camara"
            android:paddingLeft="@dimen/
                activity_horizontal_margin"
            android:paddingTop="@dimen/
                activity_vertical_margin"
            android:paddingRight="@dimen/
                activity_horizontal_margin"
            android:paddingBottom="@dimen/
                activity_vertical_margin"
            android:text="Descubrir"
            android:textColor="@color/textColorPrimary"
                />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvExposiciones"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Exposiciones"
            android:textColor="@color/colorPrimary"
            android:textSize="24sp" />

        <HorizontalScrollView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp">

            <LinearLayout
                android:id="@+id/llExpo"
```

```
                        android:layout_width="wrap_content"
                        android:layout_height="800dp"
                        android:orientation="horizontal" />
                </HorizontalScrollView>

            </LinearLayout>

        </LinearLayout>
    </ScrollView>

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginEnd="0dp"
        android:background="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:itemIconTint="@color/textColorPrimary"
        app:itemTextColor="@color/textColorPrimaryDark"
        app:menu="@menu/navigation" />

</android.support.constraint.ConstraintLayout>
```

Algoritmo B.1: Layout Inicio

```
<?xml version="1.0" encoding="utf-8"?><
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"
    tools:context="mx.edu.upt.com.imac.proyectov002.
        CameraActivity" />
```

Algoritmo B.2: Layout camara

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="
    http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HorariosActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="336dp"
        android:layout_height="85dp"
        android:gravity="center"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="Horarios"
        android:textColor="@color/colorPrimaryDark"
        android:textSize="25sp"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.338"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="335dp"
        android:layout_height="394dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="Lunes a Domingo de 09:00 a 17:00 horas. \
            n\nEn temporada alta: Dia de la primavera y periodos
            vacacionales, le recomendamos visitar el museo
            temprano."
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.327"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.591" />
```

```
    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginEnd="0dp"
        android:background="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:itemIconTint="@color/textColorPrimary"
        app:itemTextColor="@color/textColorPrimaryDark"
        app:menu="@menu/navigation" />

</android.support.constraint.ConstraintLayout>
```

Algoritmo B.3: Layout horario

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="
    http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CostoActivity">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="347dp"
        android:layout_height="42dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="Costos"
        android:textColor="@color/colorPrimaryDark"
        android:textSize="25sp"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.338"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="346dp"
        android:layout_height="67dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="136dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="Cobro de acceso de Lunes a S bado: 70
            pesos"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.408"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="351dp"
        android:layout_height="40dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="228dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="Entrada gratuita"
        android:textColor="@color/colorPrimaryDark"
        android:textSize="22sp"
        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.338"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="335dp"
        android:layout_height="150dp"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="288dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="Entrada libre con identificaci n vigente
            a: \n \n &#8226; Estudiantes. \n &#8226; Profesores
            . \n &#8226; Personas de la tercera edad. \n \n
            Domingo ENTRADA LIBRE"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.408"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginEnd="0dp"
        android:background="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:itemIconTint="@color/textColorPrimary"
        app:itemTextColor="@color/textColorPrimaryDark"
        app:menu="@menu/navigation" />

</android.support.constraint.ConstraintLayout>
```

Algoritmo B.4: Layout costo

Apéndice C

Java

```
package mx.edu.upt.com.imac.proyectov002;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.Matrix;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import mx.edu.upt.com.imac.proyectov002.dao.ListaExpo;
import mx.edu.upt.com.imac.proyectov002.pojos.ExpoPojo;

public class MenuActivity extends AppCompatActivity {


    private Button btnDescubrir, btnHorario, btnCosto;
    private LinearLayout iiExpo;
```

```
ImageView ivDescubrir ;
private BottomNavigationView.
    OnNavigationItemSelectedListener
    mOnNavigationItemSelectedListener
        = new BottomNavigationView.
            OnNavigationItemSelectedListener() {

    @Override
    public boolean onNavigationItemSelected(@NonNull
        MenuItem item) {
        switch (item.getItemId()) {
            case R.id.navigation_home:
                return true;
            case R.id.navigation_dashboard:
                Intent intent = new Intent(MenuActivity.
                    this, CameraActivity.class);
                startActivity(intent);
                return true;
        }
        return false;
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_menu);

    getSupportActionBar().setDisplayOptions(ActionBar.
        DISPLAY_SHOW_CUSTOM);
    getSupportActionBar().setCustomView(R.layout.
        bar_layout_inicio);

    BottomNavigationView navigation = (BottomNavigationView
        ) findViewById(R.id.navigation);
    navigation.setOnNavigationItemSelectedListener(
        mOnNavigationItemSelectedListener);

    btnDescubrir=(Button)findViewById(R.id.btnDescubrir);
    btnHorario=(Button) findViewById(R.id.btnHorario);
    btnCosto=(Button) findViewById(R.id.btnCosto);

    iiExpo=(LinearLayout)findViewById(R.id.llExpo);
```

```
        ivDescubrir = (ImageView) findViewById(R.id.ivDescubrir
            );

        btnHorario.setOnClickListener(new View.OnClickListener
            () {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
                    HorariosActivity.class);
                startActivity(intent);
            }
        });
        btnCosto.setOnClickListener(new View.OnClickListener()
            {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
                    CostoActivity.class);
                startActivity(intent);
            }
        });
        btnDescubrir.setOnClickListener(new View.
            OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
                    CameraActivity.class);
                startActivity(intent);
            }
        });
        ivDescubrir.setOnClickListener(new View.OnClickListener
            () {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
                    CameraActivity.class);
                startActivity(intent);
            }
        });
        listExpo();
    }

    private void listExpo(){
        ListaExpo listaExpo = new ListaExpo();
```

```
        for (ExpoPojo pojo : listaExpo.getListExpo()){

            LinearLayout layout = new LinearLayout(this);
            layout.setOrientation(LinearLayout.VERTICAL);
            LinearLayout.LayoutParams layoutParams = new
                LinearLayout.LayoutParams(LinearLayout.
                LayoutParams.MATCH_PARENT, LinearLayout.
                LayoutParams.WRAP_CONTENT);
            layoutParams.setMargins(0,5,15,5);
            layout.setLayoutParams(layoutParams);
            layout.setBackgroundResource(R.drawable.border_expo
                );

            ImageView imageView= new ImageView(this);
            imageView.setLayoutParams(new LinearLayout.
                LayoutParams(800,800));
            imageView.setImageResource(pojo.getImagen());
            imageView.setBackgroundColor(Color.BLACK);
            layout.addView(imageView);

            imageView.setOnClickListener(new View.
                OnClickListener() {
                @Override
                public void onClick(View v) {
                    startActivity(new Intent(MenuActivity.this,
                            ObjetosActivity.class).putExtra("label"
                        , pojo.getLabel()));
                }
            });

            TextView valueTV = new TextView(MenuActivity.this);
            valueTV.setText(pojo.getNombre());
            valueTV.setTextSize(18);
            valueTV.setGravity(Gravity.CENTER_VERTICAL |
                Gravity.CENTER_HORIZONTAL);
            layout.addView(valueTV);

            valueTV.setOnClickListener(new View.OnClickListener
                () {
                @Override
                public void onClick(View v) {
                    startActivity(new Intent(MenuActivity.this,
                            ObjetosActivity.class).putExtra("label"
                        , pojo.getLabel()));
```

```
                }
            });

            iiExpo.addView(layout);
        }
    }

    public Bitmap redimensionarImagenMaximo(int imagen, float
        newWidth, float newHeigth){
        Bitmap mBitmap = BitmapFactory.decodeResource(this.
            getResources(),imagen);

        int width = mBitmap.getWidth();
        int height = mBitmap.getHeight();
        float scaleWidth = ((float) newWidth) / width;
        float scaleHeight = ((float) newHeigth) / height;
        Matrix matrix = new Matrix();
        matrix.postScale(scaleWidth, scaleHeight);
        return Bitmap.createBitmap(mBitmap, 0, 0, width, height
            , matrix, false);
    }

}
```

Algoritmo C.1: Clase Menú

```
package mx.edu.upt.com.imac.proyectov002;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.os.Bundle;

public class CameraActivity extends Activity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    setRequestedOrientation(ActivityInfo.
        SCREEN_ORIENTATION_PORTRAIT);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera);
    if (null == savedInstanceState) {
      getFragmentManager()
```

```
            .beginTransaction()
            .replace(R.id.container, Camera2BasicFragment.
                newInstance())
            .commit();
    }
  }
}
```

Algoritmo C.2: Clase camara

```
package mx.edu.upt.com.imac.proyectov002;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.widget.ImageView;
import android.widget.TextView;

import mx.edu.upt.com.imac.proyectov002.dao.ListaExpo;
import mx.edu.upt.com.imac.proyectov002.pojos.ExpoPojo;

public class ObjetosActivity extends AppCompatActivity {
    private Intent intent;
    private String label;
    private TextView tvNombre, tvDescripcion ;
    private ImageView ivImagen;

    private BottomNavigationView.
        OnNavigationItemSelectedListener
        mOnNavigationItemSelectedListener
            = new BottomNavigationView.
                OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull
            MenuItem item) {
            switch (item.getItemId()) {
                case R.id.navigation_home:
                    intent = new Intent(ObjetosActivity.this,
                        MenuActivity.class);
```

```
                startActivity(intent);
                return true;
            case R.id.navigation_dashboard:
                intent = new Intent(ObjetosActivity.this,
                    CameraActivity.class);
                startActivity(intent);
                return true;
        }
        return false;
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_objetos);
    getSupportActionBar().setDisplayOptions(ActionBar.
        DISPLAY_SHOW_CUSTOM);
    getSupportActionBar().setCustomView(R.layout.
        bar_layout_piezas);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);


    BottomNavigationView navigation = (BottomNavigationView
        ) findViewById(R.id.navigation);
    navigation.setOnNavigationItemSelectedListener(
        mOnNavigationItemSelectedListener);

    tvNombre=(TextView)findViewById(R.id.tvObjNombre);
    tvDescripcion=(TextView)findViewById(R.id.
        tvObjDescripcion);
    ivImagen=(ImageView) findViewById(R.id.ivObjImagen);

    getSupportActionBar().setTitle(titleActionBar());
    llenarLayout();

}
@Override
public boolean onSupportNavigateUp() {
    intent = new Intent(ObjetosActivity.this, MenuActivity.
        class);
    startActivity(intent);
    return false;
}
```

```
    private void llenarLayout(){
        label=getIntent().getExtras().getString("label");
        ListaExpo listaExpo=new ListaExpo();
        int i=0;
        boolean van=true;
        do{
            if(i<listaExpo.getListExpo().size()){
                if(listaExpo.getListExpo().get(i).getLabel().
                    contentEquals(label)){
                    Log.d("label",label);
                    ivImagen.setImageResource(listaExpo.
                        getListExpo().get(i).getImagen());
                    tvNombre.setText(listaExpo.getListExpo().
                        get(i).getNombre());
                    tvDescripcion.setText(listaExpo.getListExpo
                        ().get(i).getDescripcion());
                    van=false;
                }
                i++;
            }else
                van=false;
        }while (van);
    }

    private String titleActionBar(){
        label=getIntent().getExtras().getString("label");
        ListaExpo listaExpo=new ListaExpo();
        int i=0;
        boolean van=true;
        do{
            if(i<listaExpo.getListExpo().size()) {
                if (listaExpo.getListExpo().get(i).getLabel().
                    contentEquals(label)) {
                    Log.d("title",label);
                    return listaExpo.getListExpo().get(i).
                        getNombre();
                }
                i++;
            }
        }while (i<listaExpo.getListExpo().size());
        return "";
    }
}
```

Algoritmo C.3: Clase objetos

```
package mx.edu.upt.com.imac.proyectov002;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;

public class HorariosActivity extends AppCompatActivity {
    Intent intent;
    private BottomNavigationView.
        OnNavigationItemSelectedListener
        mOnNavigationItemSelectedListener
            = new BottomNavigationView.
                OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull
            MenuItem item) {
            switch (item.getItemId()) {
                case R.id.navigation_home:
                    intent = new Intent(HorariosActivity.this,
                        MenuActivity.class);
                    startActivity(intent);
                    return true;
                case R.id.navigation_dashboard:
                    intent = new Intent(HorariosActivity.this,
                        CameraActivity.class);
                    startActivity(intent);
                    return true;
            }
            return false;
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_horarios);
```

```
        getSupportActionBar().setDisplayOptions(ActionBar.
            DISPLAY_SHOW_CUSTOM);
        getSupportActionBar().setCustomView(R.layout.
            bar_layout_horario);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        BottomNavigationView navigation = (BottomNavigationView
            ) findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(
            mOnNavigationItemSelectedListener);
    }

    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return false;
    }
}
```

Algoritmo C.4: Clase horario

```
package mx.edu.upt.com.imac.proyectov002;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;

public class CostoActivity extends AppCompatActivity {
    Intent intent;
    private BottomNavigationView.
        OnNavigationItemSelectedListener
        mOnNavigationItemSelectedListener
            = new BottomNavigationView.
                OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull
            MenuItem item) {
            switch (item.getItemId()) {
                case R.id.navigation_home:
```

```
                    intent = new Intent(CostoActivity.this,
                        MenuActivity.class);
                    startActivity(intent);
                    return true;
                case R.id.navigation_dashboard:
                    intent = new Intent(CostoActivity.this,
                        CameraActivity.class);
                    startActivity(intent);
                    return true;
            }
            return false;
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_costo);

        getSupportActionBar().setDisplayOptions(ActionBar.
            DISPLAY_SHOW_CUSTOM);
        getSupportActionBar().setCustomView(R.layout.
            bar_layout_costo);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        BottomNavigationView navigation = (BottomNavigationView
            ) findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(
            mOnNavigationItemSelectedListener);
    }

    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return false;
    }
}
```

Algoritmo C.5: Clase costo

```
package creararchivosimagen;

import java.awt.Image;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.channels.FileChannel;
import javax.swing.Icon;
import javax.swing.ImageIcon;

public class Imagen extends javax.swing.JFrame {

    private String[] ficheros;
    private int contImagen = 0;
    private String ruta;

    public Imagen() {
        initComponents();
        btnBuscar.doClick();

    }

    private void cargarImagenes(String ruta) {
        File dir = new File(ruta);
        ficheros = dir.list();
        if (ficheros == null) {
            System.out.println("No hay ficheros en el
                directorio especificado");
        } else {
            for (int x = 0; x < ficheros.length; x++) {
                System.out.println(ficheros[x]);
            }
        }
    }


    @SuppressWarnings("unchecked")
    private void initComponents() {

        txtRuta = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        btnBuscar = new javax.swing.JButton();
        btnAntes = new javax.swing.JButton();
        btnDespues = new javax.swing.JButton();
        jCheckBox1 = new javax.swing.JCheckBox();
        txtArchivo1 = new javax.swing.JTextField();
        txtRutaAGuardar = new javax.swing.JTextField();
        jCheckBox2 = new javax.swing.JCheckBox();
```

```
        txtArchivo2 = new javax.swing.JTextField();
        jCheckBox3 = new javax.swing.JCheckBox();
        txtArchivo3 = new javax.swing.JTextField();
        jCheckBox4 = new javax.swing.JCheckBox();
        txtArchivo4 = new javax.swing.JTextField();
        jCheckBox5 = new javax.swing.JCheckBox();
        txtArchivo5 = new javax.swing.JTextField();
        jCheckBox6 = new javax.swing.JCheckBox();
        txtArchivo6 = new javax.swing.JTextField();
        jCheckBox7 = new javax.swing.JCheckBox();
        txtArchivo7 = new javax.swing.JTextField();
        jCheckBox8 = new javax.swing.JCheckBox();
        txtArchivo8 = new javax.swing.JTextField();
        jCheckBox9 = new javax.swing.JCheckBox();
        txtArchivo9 = new javax.swing.JTextField();
        jCheckBox10 = new javax.swing.JCheckBox();
        txtArchivo10 = new javax.swing.JTextField();
        jCheckBox11 = new javax.swing.JCheckBox();
        txtArchivo11 = new javax.swing.JTextField();
        jCheckBox12 = new javax.swing.JCheckBox();
        txtArchivo12 = new javax.swing.JTextField();
        btnGuardar = new javax.swing.JButton();
        btnEliminar = new javax.swing.JButton();
        labNum = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        labNombreArchivo = new javax.swing.JLabel();
        txtCambiar = new java.awt.TextField();
        btnCambiar = new javax.swing.JButton();
        labImagen = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.
            EXIT_ON_CLOSE);
        setAlwaysOnTop(true);
        setCursor(new java.awt.Cursor(java.awt.Cursor.
            DEFAULT_CURSOR));

        txtRuta.setText("D:\\Protecto Tesis\\AnacondaB\\
            Pruebas_1\\imagenes\\descargas");

        jLabel1.setText("Archivo a Buscar:");

        btnBuscar.setText("Buscar");
        btnBuscar.addActionListener(new java.awt.event.
```

```
        ActionListener() {
        public void actionPerformed(java.awt.event.
            ActionEvent evt) {
            btnBuscarActionPerformed(evt);
        }
    });

    btnAntes.setText("<");
    btnAntes.addActionListener(new java.awt.event.
        ActionListener() {
        public void actionPerformed(java.awt.event.
            ActionEvent evt) {
            btnAntesActionPerformed(evt);
        }
    });

    btnDespues.setText(">");
    btnDespues.addActionListener(new java.awt.event.
        ActionListener() {
        public void actionPerformed(java.awt.event.
            ActionEvent evt) {
            btnDespuesActionPerformed(evt);
        }
    });

    jCheckBox1.setText("Archivo 1");

    txtArchivo1.setText("Cariatide");

    txtRutaAGuardar.setText("D:\\Protecto Tesis\\
        Entrenamiento\\Imagenes");

    jCheckBox2.setText("Archivo 2");

    txtArchivo2.setText("Chac-mool");

    jCheckBox3.setText("Archivo 3");

    txtArchivo3.setText("Coatepantli");

    jCheckBox4.setText("Archivo 4");

    txtArchivo4.setText(" Fuste-pilastra");
```

```
        jCheckBox5.setText("Archivo 5");

        txtArchivo5.setText(" Fustes serpentina");

        jCheckBox6.setText("Archivo 6");

        txtArchivo6.setText("Estela");

        jCheckBox7.setText("Archivo 7");

        txtArchivo7.setText("Almena");

        jCheckBox8.setText("Archivo 8");

        txtArchivo8.setText("Palacio quemado");

        jCheckBox9.setText("Archivo  9");

        txtArchivo9.setText("Felino");

        jCheckBox10.setText("Archivo 10");

        txtArchivo10.setText("Piramide B");
        txtArchivo10.addActionListener(new java.awt.event.
            ActionListener() {
            public void actionPerformed(java.awt.event.
                ActionEvent evt) {
                txtArchivo10ActionPerformed(evt);
            }
        });

        jCheckBox11.setText("Archivo 1");

        txtArchivo11.setText("Piramide C");

        jCheckBox12.setText("Archivo 1");

        txtArchivo12.setText("Atlante");

        btnGuardar.setText("Guardar");
        btnGuardar.addActionListener(new java.awt.event.
            ActionListener() {
            public void actionPerformed(java.awt.event.
                ActionEvent evt) {
```

```
            btnGuardarActionPerformed(evt);
        }
    });

    btnEliminar.setText("Elininar");

    jLabel2.setText("Ruta");

    labNombreArchivo.setText("jLabel3");

    btnCambiar.setText("Cambiar");
    btnCambiar.addActionListener(new java.awt.event.
        ActionListener() {
        public void actionPerformed(java.awt.event.
            ActionEvent evt) {
            btnCambiarActionPerformed(evt);
        }
    });

    labImagen.setHorizontalAlignment(javax.swing.
        SwingConstants.CENTER);
    labImagen.setIcon(new javax.swing.ImageIcon("D:\\
        Protecto_Tesis\\Imagenes\\Prueba3\\atlantes\\
        atlantes_0_0.jpg"));
    labImagen.setToolTipText("");
    labImagen.setAutoscrolls(true);
    labImagen.setFocusable(false);
    labImagen.setPreferredSize(new java.awt.Dimension(275,
        275));

    jButton1.setText("Abrir");

    javax.swing.GroupLayout layout = new javax.swing.
        GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.
                DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.
                swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
```

```
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addComponent(txtRuta)
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addComponent(btnBuscar)
                .addGap(110, 110, 110))
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    LEADING, false)
                    .addGroup(layout.
                        createSequentialGroup()
                        .addComponent(btnAntes)
                        .addGap(275, 275, 275)
                        .addGroup(layout.
                            createParallelGroup(javax.
                            swing.GroupLayout.Alignment.
                            LEADING)
                            .addComponent(
                                labNombreArchivo)
                            .addGroup(layout.
                                createSequentialGroup()
                                .addComponent(labNum)
                                .addGap(309, 309, 309)
                                .addComponent(
                                    btnDespues))))
                    .addComponent(labImagen, javax.
                        swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.
                        DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED, javax.swing.GroupLayout.
                    DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    LEADING)
                    .addGroup(layout.
                        createSequentialGroup()
                        .addGroup(layout.
```

```
                createParallelGroup(javax.
                swing.GroupLayout.Alignment.
                LEADING)
                .addComponent(jCheckBox1,
                    javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.
                    swing.GroupLayout.
                    DEFAULT_SIZE, Short.
                    MAX_VALUE)
                .addComponent(jCheckBox2)
                .addComponent(jCheckBox3)
                .addComponent(jCheckBox4)
                .addComponent(jCheckBox5)
                .addComponent(jCheckBox6)
                .addComponent(jCheckBox7)
                .addComponent(jCheckBox8)
                .addComponent(jCheckBox9)
                .addComponent(jCheckBox10)
                .addComponent(jCheckBox11)
                .addComponent(jCheckBox12))
            .addPreferredGap(javax.swing.
                LayoutStyle.
                ComponentPlacement.RELATED,
                javax.swing.GroupLayout.
                DEFAULT_SIZE, Short.
                MAX_VALUE))
        .addGroup(layout.
            createSequentialGroup()
            .addComponent(txtCambiar, javax
                .swing.GroupLayout.
                PREFERRED_SIZE, 110, javax.
                swing.GroupLayout.
                PREFERRED_SIZE)
            .addPreferredGap(javax.swing.
                LayoutStyle.
                ComponentPlacement.RELATED,
                javax.swing.GroupLayout.
                DEFAULT_SIZE, Short.
                MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.
            Alignment.TRAILING, layout.
            createSequentialGroup()
            .addComponent(jLabel2, javax.
                swing.GroupLayout.
```

```
                    DEFAULT_SIZE, javax.swing.
                    GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
                .addGap(14, 14, 14)))
        .addGroup(layout.createParallelGroup(
            javax.swing.GroupLayout.Alignment.
            LEADING)
            .addGroup(layout.
                createSequentialGroup()
                .addComponent(btnCambiar)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addGap(31, 31, 31)
                .addComponent(btnEliminar)
                .addPreferredGap(javax.swing.
                    LayoutStyle.
                    ComponentPlacement.RELATED)
                .addComponent(btnGuardar)
                .addGap(10, 10, 10))
            .addGroup(layout.
                createSequentialGroup()
                .addGroup(layout.
                    createParallelGroup(javax.
                    swing.GroupLayout.Alignment.
                    LEADING)
                    .addComponent(txtArchivo1)
                    .addComponent(txtArchivo2)
                    .addComponent(txtArchivo3)
                    .addComponent(txtArchivo4)
                    .addComponent(txtArchivo5)
                    .addComponent(txtArchivo6)
                    .addComponent(txtArchivo7)
                    .addComponent(txtArchivo8)
                    .addComponent(txtArchivo9)
                    .addComponent(txtArchivo10,
                        javax.swing.GroupLayout
                        .PREFERRED_SIZE, 428,
                        javax.swing.GroupLayout.
                        PREFERRED_SIZE)
                    .addComponent(txtArchivo11)
                    .addComponent(txtArchivo12)
                    .addGroup(layout.
                        createSequentialGroup()
                        .addComponent(
```

```
                                        txtRutaAGuardar)
                                    .addGap(0, 0, Short.
                                        MAX_VALUE)))
                            .addContainerGap())))))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.
            DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.
            swing.GroupLayout.Alignment.BASELINE)
            .addComponent(txtRuta, javax.swing.
                GroupLayout.PREFERRED_SIZE, javax.swing.
                GroupLayout.DEFAULT_SIZE, javax.swing.
                GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel1)
            .addComponent(btnBuscar))
        .addPreferredGap(javax.swing.LayoutStyle.
            ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.
            swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(txtRutaAGuardar,
                        javax.swing.GroupLayout.
                        PREFERRED_SIZE, javax.swing.
                        GroupLayout.DEFAULT_SIZE, javax.
                        swing.GroupLayout.PREFERRED_SIZE
                        )
                    .addComponent(jLabel2))
                .addGap(6, 6, 6)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jCheckBox1)
                    .addComponent(txtArchivo1, javax.
                        swing.GroupLayout.PREFERRED_SIZE
                        , javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
```

```
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jCheckBox2)
                    .addComponent(txtArchivo2, javax.
                        swing.GroupLayout.PREFERRED_SIZE
                        , javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jCheckBox3)
                    .addComponent(txtArchivo3, javax.
                        swing.GroupLayout.PREFERRED_SIZE
                        , javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jCheckBox4)
                    .addComponent(txtArchivo4, javax.
                        swing.GroupLayout.PREFERRED_SIZE
                        , javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.
                    LayoutStyle.ComponentPlacement.
                    RELATED)
                .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jCheckBox5)
                    .addComponent(txtArchivo5, javax.
```

```
                swing.GroupLayout.PREFERRED_SIZE
                , javax.swing.GroupLayout.
                DEFAULT_SIZE, javax.swing.
                GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.
                LayoutStyle.ComponentPlacement.
                RELATED)
            .addGroup(layout.createParallelGroup(
                javax.swing.GroupLayout.Alignment.
                BASELINE)
                .addComponent(jCheckBox6)
                .addComponent(txtArchivo6, javax.
                    swing.GroupLayout.PREFERRED_SIZE
                    , javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.
                    GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.
                LayoutStyle.ComponentPlacement.
                RELATED)
            .addGroup(layout.createParallelGroup(
                javax.swing.GroupLayout.Alignment.
                BASELINE)
                .addComponent(jCheckBox7)
                .addComponent(txtArchivo7, javax.
                    swing.GroupLayout.PREFERRED_SIZE
                    , javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.
                    GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.
                LayoutStyle.ComponentPlacement.
                RELATED)
            .addGroup(layout.createParallelGroup(
                javax.swing.GroupLayout.Alignment.
                BASELINE)
                .addComponent(jCheckBox8)
                .addComponent(txtArchivo8, javax.
                    swing.GroupLayout.PREFERRED_SIZE
                    , javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.
                    GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.
                LayoutStyle.ComponentPlacement.
                RELATED)
            .addGroup(layout.createParallelGroup(
```

```
                javax.swing.GroupLayout.Alignment.
                BASELINE)
                 .addComponent(jCheckBox9)
                 .addComponent(txtArchivo9, javax.
                     swing.GroupLayout.PREFERRED_SIZE
                     , javax.swing.GroupLayout.
                     DEFAULT_SIZE, javax.swing.
                     GroupLayout.PREFERRED_SIZE))
             .addPreferredGap(javax.swing.
                 LayoutStyle.ComponentPlacement.
                 RELATED)
             .addGroup(layout.createParallelGroup(
                 javax.swing.GroupLayout.Alignment.
                 BASELINE)
                 .addComponent(jCheckBox10)
                 .addComponent(txtArchivo10, javax.
                     swing.GroupLayout.PREFERRED_SIZE
                     , javax.swing.GroupLayout.
                     DEFAULT_SIZE, javax.swing.
                     GroupLayout.PREFERRED_SIZE))
             .addGap(6, 6, 6)
             .addGroup(layout.createParallelGroup(
                 javax.swing.GroupLayout.Alignment.
                 BASELINE)
                 .addComponent(jCheckBox11)
                 .addComponent(txtArchivo11, javax.
                     swing.GroupLayout.PREFERRED_SIZE
                     , javax.swing.GroupLayout.
                     DEFAULT_SIZE, javax.swing.
                     GroupLayout.PREFERRED_SIZE))
             .addPreferredGap(javax.swing.
                 LayoutStyle.ComponentPlacement.
                 RELATED)
             .addGroup(layout.createParallelGroup(
                 javax.swing.GroupLayout.Alignment.
                 BASELINE)
                 .addComponent(jCheckBox12)
                 .addComponent(txtArchivo12, javax.
                     swing.GroupLayout.PREFERRED_SIZE
                     , javax.swing.GroupLayout.
                     DEFAULT_SIZE, javax.swing.
                     GroupLayout.PREFERRED_SIZE)))
         .addComponent(labImagen, javax.swing.
             GroupLayout.PREFERRED_SIZE, 345, javax.
```

```
                        swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED, javax.swing.
                    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.
                    swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createParallelGroup(javax.
                        swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(btnDespues)
                        .addComponent(btnAntes)
                        .addComponent(btnGuardar)
                        .addComponent(btnEliminar)
                        .addComponent(btnCambiar)
                        .addComponent(jButton1))
                    .addComponent(labNum)
                    .addComponent(txtCambiar, javax.swing.
                        GroupLayout.Alignment.TRAILING, javax.
                        swing.GroupLayout.PREFERRED_SIZE, javax.
                        swing.GroupLayout.DEFAULT_SIZE, javax.
                        swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(labNombreArchivo)
                .addContainerGap(javax.swing.GroupLayout.
                    DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }

    private void btnBuscarActionPerformed(java.awt.event.
        ActionEvent evt) {
        ruta = txtRuta.getText();
        cargarImagenes(ruta);
        ImageIcon icon = new ImageIcon(ruta + "\\" + ficheros[
            contImagen]);
        labImagen.setIcon(icon);
        labNum.setText(contImagen + "␣/␣" + ficheros.length);
        labNombreArchivo.setText(ficheros[contImagen]);
        this.show();
    }

    private void btnDespuesActionPerformed(java.awt.event.
        ActionEvent evt) {
        if (contImagen < ficheros.length - 1) {
```

```
            contImagen++;
            cargarImagen();
        }
    }

    private void btnAntesActionPerformed(java.awt.event.
        ActionEvent evt) {
        if (contImagen > 0) {
            contImagen--;
            cargarImagen();
        }
    }
    private void btnGuardarActionPerformed(java.awt.event.
        ActionEvent evt) {
        String inFile = ruta + "\\" + ficheros[contImagen];

        if (jCheckBox1.isSelected()) {
            String archivo = txtArchivo1.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox2.isSelected()) {
```

```
            String archivo = txtArchivo2.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox3.isSelected()) {
            String archivo = txtArchivo3.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
```

```
            FileChannel outChannel = fos.getChannel();
            inChannel.transferTo(0, inChannel.size(),
                outChannel);
            fis.close();
            fos.close();
        } catch (IOException ioe) {
            System.err.println("Error al Generar Copia");
        }
    }
    if (jCheckBox4.isSelected()) {
        String archivo = txtArchivo4.getText();
        try {
            File directorio = new File(txtRutaAGuardar.
                getText() + "\\" + archivo);

            if (!directorio.exists()) {
                directorio.mkdirs();
            }

            String outFile = txtRutaAGuardar.getText() + "
                \\" + archivo + "\\" + archivo + "_" +
                contImagen + ".jpg";
            FileInputStream fis = new FileInputStream(
                inFile); //inFile -> Archivo a copiar
            FileOutputStream fos = new FileOutputStream(
                outFile); //outFile -> Copia del archivo
            FileChannel inChannel = fis.getChannel();
            FileChannel outChannel = fos.getChannel();
            inChannel.transferTo(0, inChannel.size(),
                outChannel);
            fis.close();
            fos.close();
        } catch (IOException ioe) {
            System.err.println("Error al Generar Copia");
        }
    }
    if (jCheckBox5.isSelected()) {
        String archivo = txtArchivo5.getText();
        try {
            File directorio = new File(txtRutaAGuardar.
                getText() + "\\" + archivo);

            if (!directorio.exists()) {
                directorio.mkdirs();
```

```
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox6.isSelected()) {
            String archivo = txtArchivo6.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
```

```
            }
        }
        if (jCheckBox7.isSelected()) {
            String archivo = txtArchivo7.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox8.isSelected()) {
            String archivo = txtArchivo8.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
```

```
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox9.isSelected()) {
            String archivo = txtArchivo9.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox10.isSelected()) {
            String archivo = txtArchivo10.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);
```

```
            if (!directorio.exists()) {
                directorio.mkdirs();
            }

            String outFile = txtRutaAGuardar.getText() + "
                \\" + archivo + "\\" + archivo + "_" +
                contImagen + ".jpg";
            FileInputStream fis = new FileInputStream(
                inFile); //inFile -> Archivo a copiar
            FileOutputStream fos = new FileOutputStream(
                outFile); //outFile -> Copia del archivo
            FileChannel inChannel = fis.getChannel();
            FileChannel outChannel = fos.getChannel();
            inChannel.transferTo(0, inChannel.size(),
                outChannel);
            fis.close();
            fos.close();
        } catch (IOException ioe) {
            System.err.println("Error al Generar Copia");
        }
    }
    if (jCheckBox11.isSelected()) {
        String archivo = txtArchivo11.getText();
        try {
            File directorio = new File(txtRutaAGuardar.
                getText() + "\\" + archivo);

            if (!directorio.exists()) {
                directorio.mkdirs();
            }

            String outFile = txtRutaAGuardar.getText() + "
                \\" + archivo + "\\" + archivo + "_" +
                contImagen + ".jpg";
            FileInputStream fis = new FileInputStream(
                inFile); //inFile -> Archivo a copiar
            FileOutputStream fos = new FileOutputStream(
                outFile); //outFile -> Copia del archivo
            FileChannel inChannel = fis.getChannel();
            FileChannel outChannel = fos.getChannel();
            inChannel.transferTo(0, inChannel.size(),
                outChannel);
            fis.close();
```

```
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        if (jCheckBox12.isSelected()) {
            String archivo = txtArchivo12.getText();
            try {
                File directorio = new File(txtRutaAGuardar.
                    getText() + "\\" + archivo);

                if (!directorio.exists()) {
                    directorio.mkdirs();
                }

                String outFile = txtRutaAGuardar.getText() + "
                    \\" + archivo + "\\" + archivo + "_" +
                    contImagen + ".jpg";
                FileInputStream fis = new FileInputStream(
                    inFile); //inFile -> Archivo a copiar
                FileOutputStream fos = new FileOutputStream(
                    outFile); //outFile -> Copia del archivo
                FileChannel inChannel = fis.getChannel();
                FileChannel outChannel = fos.getChannel();
                inChannel.transferTo(0, inChannel.size(),
                    outChannel);
                fis.close();
                fos.close();
            } catch (IOException ioe) {
                System.err.println("Error al Generar Copia");
            }
        }
        btnDespues.doClick();
    }

    private void btnCambiarActionPerformed(java.awt.event.
        ActionEvent evt) {
         contImagen = Integer.parseInt(txtCambiar.getText());
        cargarImagen();
    }
    private void txtArchivo10ActionPerformed(java.awt.event.
        ActionEvent evt) {

    }
```

```
    private void cargarImagen() {

        Image fotoherr = getToolkit().getImage(ruta + "\\" +
            ficheros[contImagen]);
        fotoherr = fotoherr.getScaledInstance(900, 600, Image.
            SCALE_DEFAULT);
        labImagen.setIcon(new ImageIcon(fotoherr));

        labNum.setText(contImagen + " / " + ficheros.length);
        labNombreArchivo.setText(ficheros[contImagen]);
        System.out.println(ficheros[contImagen]);
        this.repaint();
    }

      public static void main(String args[]) {
       try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()
                ) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.
                        getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Imagen.class.
                getName()).log(java.util.logging.Level.SEVERE,
                null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Imagen.class.
                getName()).log(java.util.logging.Level.SEVERE,
                null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Imagen.class.
                getName()).log(java.util.logging.Level.SEVERE,
                null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex
            ) {
            java.util.logging.Logger.getLogger(Imagen.class.
                getName()).log(java.util.logging.Level.SEVERE,
                null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
            public void run() {
                new Imagen().setVisible(true);
            }
        });
    }

    private javax.swing.JButton btnAntes;
    private javax.swing.JButton btnBuscar;
    private javax.swing.JButton btnCambiar;
    private javax.swing.JButton btnDespues;
    private javax.swing.JButton btnEliminar;
    private javax.swing.JButton btnGuardar;
    private javax.swing.JButton jButton1;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JCheckBox jCheckBox10;
    private javax.swing.JCheckBox jCheckBox11;
    private javax.swing.JCheckBox jCheckBox12;
    private javax.swing.JCheckBox jCheckBox2;
    private javax.swing.JCheckBox jCheckBox3;
    private javax.swing.JCheckBox jCheckBox4;
    private javax.swing.JCheckBox jCheckBox5;
    private javax.swing.JCheckBox jCheckBox6;
    private javax.swing.JCheckBox jCheckBox7;
    private javax.swing.JCheckBox jCheckBox8;
    private javax.swing.JCheckBox jCheckBox9;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel labImagen;
    private javax.swing.JLabel labNombreArchivo;
    private javax.swing.JLabel labNum;
    private javax.swing.JTextField txtArchivo1;
    private javax.swing.JTextField txtArchivo10;
    private javax.swing.JTextField txtArchivo11;
    private javax.swing.JTextField txtArchivo12;
    private javax.swing.JTextField txtArchivo2;
    private javax.swing.JTextField txtArchivo3;
    private javax.swing.JTextField txtArchivo4;
    private javax.swing.JTextField txtArchivo5;
    private javax.swing.JTextField txtArchivo6;
    private javax.swing.JTextField txtArchivo7;
    private javax.swing.JTextField txtArchivo8;
    private javax.swing.JTextField txtArchivo9;
    private java.awt.TextField txtCambiar;
    private javax.swing.JTextField txtRuta;
```

```
    private javax.swing.JTextField txtRutaAGuardar;
}
```

Algoritmo C.6: Imagen

Bibliografía

[1] App store. https://itunes.apple.com/us/app/magnus-scan-and-discover-art/id1061967005?mt=8.

[2] Tensorflow. www.tensorflow.org.

[3] Accord. Accord. https://accord-framework.net/.

[4] MASAHIKO ARAI. Bounds on the number of hidden units in binary-valued three-layer neural networks. 1992.

[5] et al Bernard Widrow, Marcian E. Hoff. *Adaptive Switching Circuits.* IRE, New York, 1960.

[6] Marcian Hoff Bernard Widrow. Adaptive switching circuits. *Institute of Radio Engineers, Western Electronics Show and Convention*, Part 4:96–104, 1960.

[7] Jason Brownlee. How to configure image data augmentation when training deep learning neural networks. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/.

[8] Caffe. Caffe. https://caffe.berkeleyvision.org/.

[9] François Chollet. *Deep Learning with Python.* Manning Publications Co., United States of America, 2018.

[10] Miguel Guevara Chumacero. El edificio 3 de tula ¿historia de un palacio? *Escuela Nacional de Antropología e Historia.*, 2003.

[11] Google Cloud. Google cloud. https://cloud.google.com/products/machine-learning.

[12] CNTK. Cntk. https://docs.microsoft.com/en-us/cognitive-toolkit.

[13] TAMARA CRUZ Y CRUZ. *CONSTRUYENDO TOLLÁN*. INAH, INAH, 2007.

[14] Eugenio Culurciello. Neural network architectures. https://towardsdatascience.com/neural-network-architectures-156e5bad51ba, 2017.

[15] Rob Fergus David Eigen. Predicting depth, surface normals and semantic labels. *Predicting Depth, Surface Normals and Semantic Labels*, 2015.

[16] Guosheng Lin Fayao Liu, Chunhua Shen. Deep convolutional neural fields for depth estimation from a single image. *Journal of Archaeological Method and Theory*, 2016.

[17] Guosheng Lin Ian Reid Fayao Liu, Chunhua. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[18] Osamu Fujita. Statistical estimation of the number of hidden units for feedforward neural networks. *Elsevier Science*, 1998.

[19] Esperanza Elizabeth Jiménez García. Catálogo escultórico-iconográfico de tula, hidalgo: Sus imágenes en piedra. *FAMSI*, 2008.

[20] Adam Geitgey. Machine learning is fun! part 3: Deep learning and convolutional neural networks. https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721, 2016.

[21] Ahmed Menshawy Giancarlo Zaccone, Md. Rezaul Karim. *Deep Learning with TensorFlow*. Packt Publishing, Birmingham, 2017.

[22] Google. Google play. https://play.google.com/store/apps/details?id=com.google.android.apps.cultural.

[23] Google. Tensorflow for poets. https://codelabs.developers.google.com/codelabs/tensorflow-for-poets.

[24] Jeff Hale. Deep learning framework power scores 2018. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a.

[25] D. Hebb. *The Organization of Behavior*. Wiley, New York, 1949.

[26] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[27] Yanbo Huang. Advances in artificial neural networks - methodological development and application. *Algorithms*, 2(3):973–1007, 2009.

[28] Octavio Islas. Estudio global digital 2019. https://www.eluniversal.com.mx/columna/octavio-islas/techbit/estudio-global-digital-2019.

[29] Shannon Dugan Iverson. The enduring toltecs: History and truthduring the aztec-to-colonial transition at tula, hidalgo. *Rice University*, 2017.

[30] Sree Ramaswamy Michael Chui Tera Allas Peter Dahlström Nicolaus Henke Monica Trench Jacques Bughin, Eric Hazan. Artificial intelligencethe next digital frontier? *McKinsey and Company*, 2017.

[31] Justin Johnson. Convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/.

[32] Ujjwal Karn. An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets, 2017.

[33] kera. Image preprocessing. https://keras.io/preprocessing/image/.

[34] Keras. Keras. https://keras.io.

[35] Phil Kim. *MATLAB Deep Learning; With Machine Learning, Neural Networks and Artificial Intelligence*. Apress, Korea, 2017.

[36] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Department of Technical Physics, Helsinki University of Technology, Espoo, Finland*, 1982.

[37] David Kriesel. *A Brief Introduction to Neural Networks*. Zeta2 edition, 2007.

[38] Bohdan Macukow. Neural networks – state of art, brief history, basic models and architecture. *Faculty of Applied Mathematics and Information Science*, 2016.

[39] Magnus. Magnus. http://www.magnus.net.

[40] mxnet. mxnet. https://aws.amazon.com/es/mxnet.

[41] Pillow. Imageenhance. https://pillow.readthedocs.io/en/3.0.x /reference/ImageEnhance.html.

[42] Pillow. Imagefilter. https://pillow.readthedocs.io/en/5.1.x /reference/ImageFilter.html.

[43] Intel Developer Program. Intel developer program. https://software.intel.com/es-es/articles/hands-on-ai-part-5-select-a-deep-learning-framework.

[44] José Miguel Fernández Fernández Raquel Flórez López. *Las Redes Neuronales Artificiales*. netbiblo, Birmingham, 2008.

[45] Matt Reynolds. Newscientist. https://www.newscientist.com/article /2123373-image-recognition-app-scans-paintings-to-act-like-shazam-for-art.

[46] Samy Bengio y Johanny Mariéthoz Ronan Collobert. Torch: a modular machine learning software library. *Technical Report IDIAP-RR 02-46*, 2002.

[47] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[48] Hinton y Williams Rumelhart. Learning internal representations by error propagation. *Explorations in the Microstructure of Cognition*, 1986.

[49] Krishna Rungta. *TensorFlow in 1 Day: Make your own Neural Network*. Krishna Rungta, 2018.

[50] Amazon SageMaker. Amazon sagemaker. https://aws.amazon.com/es.

[51] Syed Afaq Ali Shah Mohammed Bennamoun Salman Khan, Hossein Rahmani. *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan and Claypool Publishers, California, 2018.

[52] Antsaklis y J. Panos. Sartori, A. Michael. Neural network training via quadratic optimization. *Dept. of Electrical and Computer Engineering*, 1991.

[53] Data Science. Anaconda navigator. https://docs.anaconda.com.

[54] Rajalingappaa Shanmugamani. *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*. Packt Publishing - ebooks Account, 2018.

[55] Sandro Skansi. *Introduction to Deep Learning (From Logical Calculus to Artificial Intelligence)*. Springer International Publishing AG, Zagreb, 2018.

[56] Smartify. Smartify. https://smartify.org/.

[57] Spyder. Spyder docs. https://docs.spyder-ide.org/.

[58] Watson Studio. Watson studio. https://dataplatform.cloud.ibm.com/docs/content.

[59] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[60] Tensorflow. Curso de tensorflow. https://developers.google.com/machine-learning/crash-course/.

[61] TensorFlow. Tensorflow lite. https://www.tensorflow.org/lite/overview.

[62] Theano. Theano. http://deeplearning.net/software/theano.

[63] Torch. Torch. https://torch.ch/.

[64] Hilary Weaver. Revista vanity fair. http://www.revistavanityfair.es/actualidad/articulos/leonardo-dicaprio-inversor-app-magnus-shazam-arte/28930.

[65] Naveen Kulkarni y Baoxin Li. Discriminative affine sparse codes for image classification. *Proc. of the IEEE*, 1998.

[66] Ragav Venkatesan y Baoxin Li. *Convolutional Neural Networks in Visual Computing A Concise Guide*. Taylor and Francis Group, London, 2018.

[67] Andrew G. Howard y Menglong Zhu. Mobilenets: modelos de código abierto para una visión eficiente en el dispositivo. https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html.

[68] Kunihiko Fukushima y Sei Miyake. Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. *The Journal of Physiology*, 1981.

[69] Warren S Mcculloch y Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *BULLETIN OF MATHEMATICAL BIOPHYSICS*, 5:19, 1943.

[70] Yoshua Bengio Patrick Haffner Yann LeCun, Léon Bottou. Gradient-based learning applied. *Proc. of the IEEE*, 1998.

[71] Jan Zacharias, Michael Barz, and Daniel Sonntag. A survey on deep learning toolkits and libraries for intelligent user interfaces. *CoRR*, abs/1803.04818, 2018.

[72] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

Índice de figuras

Índice de tablas