Clasificación de Lenguaje de Señas con CNN

Jorge Martínez López
Tecnológico de Monterrey campus Querétaro
Querétaro, México
A01704518@tec.mx
jorgemartinez2555@hotmail.com

***Abstract*— Este documento es un ejemplo de formato apegado a las normas de IEEE para escribir artículos representativos de un proyecto realizado. Los autores deben seguir las instrucciones, incluyendo formato y tamaño de papel para mantener el estándar de publicación. Este documento puede interpretarse como un set de instrucciones para escribir su artículo o como una plantilla para hacerlo.**

I. INTRODUCCIÓN

La comunicación a través del lenguaje de señas es esencial para millones de personas en el mundo. Sin embargo, dentro de sectores de la sociedad, existen limitantes en la comunicación e interacción afectiva con personas con discapacidad auditiva, limitando su acceso a servicios cotidianos.

En este contexto, el avance en técnicas de visión por computadora, específicamente las redes neuronales convolucionales (CNN), abre nuevas posibilidades para desarrollar sistemas de reconocimiento automático del lenguaje de señas que traduzcan señales manuales y gestos en tiempo real.

Esta tecnología tiene el potencial de mejorar significativamente la inclusión social y la accesibilidad en diversos sectores, facilitando la comunicación entre personas sordas y oyentes.

Este proyecto busca explorar cómo el diseño e implementación de modelos basados en CNN pueden contribuir a superar las barreras de comunicación y acercar las oportunidades de interacción con todas las personas.

II. HISTORIA DL

Los orígenes de la inteligencia artificial surgen entre 1943-1955, parte de tres fuentes de conocimiento de la época, las cuales radica en la filosofía básica y el funcionamiento de las neuronas, el análisis de la Lógica proporcional de Russell y la Teoría Computacional de Turing, estos trabajos fueron los cimientos de lo que se conoce hoy como redes neuronales. Sin embargo, hasta el 2012 fue que empezó a tener avances significativos las redes neuronales gracias a su desempeño en competencias, ya que la precisión para clasificar imágenes aumentaba a la medida de que se usaban diferentes arquitecturas y dataset para entrenar los modelos neuronales, de tal manera que su asertividad no bajaba del 80% de precisión, algo que con los algoritmos de Machine Learning difícilmente es alcanzable (*Artificial Intelligence: A Modern Approach, 4th Global Ed.*, 2022).

A. Objetivo

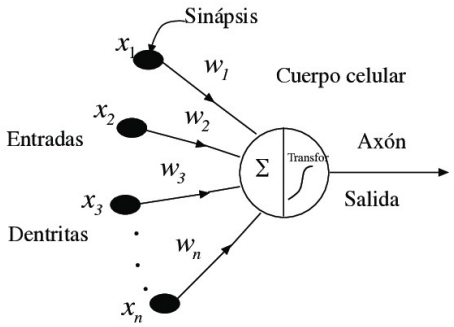
Generar una Red Neuronal Convolucional para detección de lenguaje de señas, con el fin de determinar su efectividad en su implementación y en su desenvolvimiento de su tarea.

III. MODELOS REDES NEURONALES

Los modelos de redes neurales cuentan con una estructura base. Primeramente, debemos entender que una red está consistida con una red de perceptrones, un perceptrón consiste un modelo de regresión logística que puede ser 0 o 1, a su vez se dice que una red neuronal densa consiste en una red de regresiones logísticas que se interconectan entre sí, solo que podemos cambiar las funciones de activación de cada perceptrón para mejorar su rendimiento.

A. Perceptrón

Como lo he mencionado anteriormente un perceptrón es la adaptación de un modelo de regresión logística con la posibilidad de cambiar las funciones de activación, esta es la única diferencia entre perceptrón y un modelo de regresión logística.



Img 1. Estructura de un perceptrón en diagrama.

Un perceptrón matemático se puede ver de la siguiente manera, siempre y cuando sea sigmoidal la función de activación.

$$Y = \frac{1}{1 + e^{-(X_n w_n + B)}}$$

Y es la salida de la neurona, X es la entrada de la neurona, W es el coeficiente o peso del modelo y B es el sesgo o el error del modelo.

La gran ventaja de un perceptrón es que es una regresión lineal por su función de activación y existen varias funciones de activación, y cada una tiene su propósito:

- Sigmoidal

- ReLU (Rectified Linear Unit)
- Tanh (Tangente hiperbólica)
- Leaky ReLU
- ELU (Exponential Linear Unit)
- Softmax (Normalización)
- Swish
- GELU (Gaussian Error Linear Unit)

Función	Rango de salida	No linealidad	Vanishing Gradient	Uso común
ReLU	[0, ∞)	Sí	No	Capas ocultas de CNNs, redes profundas
Sigmoide	(0, 1)	Sí	Sí	Clasificación binaria
Tanh	(-1, 1)	Sí	Sí	Capas ocultas de redes recurrentes
Leaky ReLU	(-∞, ∞)	Sí	No	Mitigar neuronas muertas
ELU	(-α, ∞)	Sí	No	Alternativa a ReLU
Softmax	(0, 1)	No (normaliza)	No	Clasificación multiclase
Swish	(-∞, ∞)	Sí	No	Redes profundas, variantes de ReLU
GELU	(-∞, ∞)	Sí	No	Transformers, redes profundas

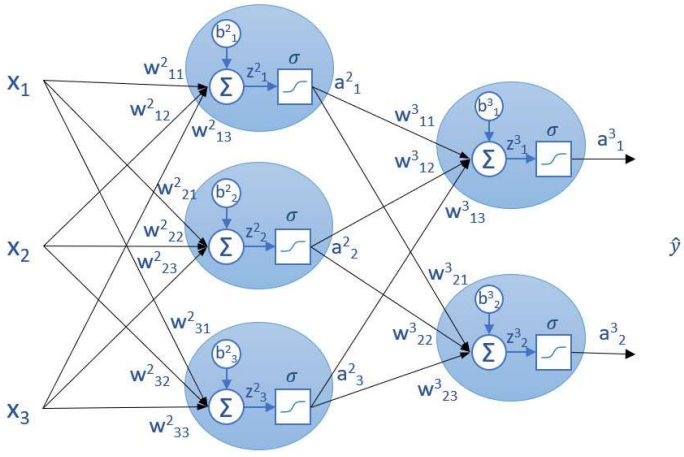
Img 2. Tabla de las funciones de activación.

Como se puede ver, existen diferentes variantes de funciones de activación y cada una funciona mejor que otra, ya que dependerá de la arquitectura que se esté manejando.

B. Red Neuronal Densa

Una Red Neuronal Densa está compuesta por capas de perceptrones, y un modelo neuronal puede ocupar desde N capas y N neuronas por capa, y la definición de la arquitectura dependerá de que tan complejo es el problema y que tan rápido quieres que un modelo se entrene.

Por otro lado, se puede hacer la analogía que una red neuronal es una matriz de regresiones logística con una salida, y lo mencionado anteriormente lo describe la siguiente imagen.



Img 3. Diagrama de una Red Neuronal, de dos capas, de tres neuronas de entrada y dos salidas.

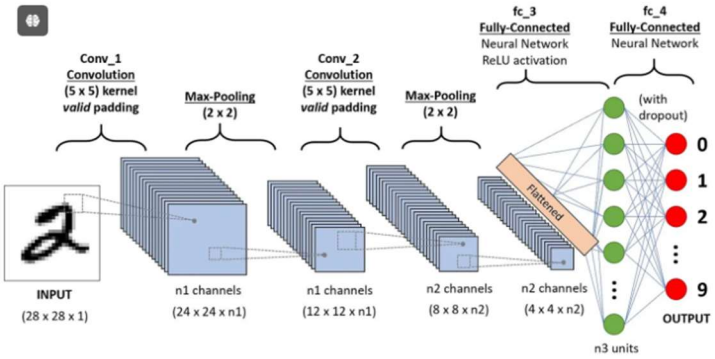
Al tener un diagrama que muestre la interconectividad de las neuronas, es más simple interpretar su funcionamiento interno de las redes neuronales, y este es el principio que utilizan la mayoría de las arquitecturas actuales, en la cual lo que cambia es la función de activación, entonces tenemos en una red neuronal tenemos entradas y luego salidas de la primera capa, que a su vez se convierten entradas de la segunda capa neuronal, y la salida de esta capa, serían los valores esperados, y así para N capas de la red.

Cabe resaltar, que el método de cálculo que se utilizan para entrenar las redes neuronales lleva el nombre backpropagation o propagación hacia atrás de los errores.

C. Red Neuronal Convolucional

Las Redes Neuronales Densa son comúnmente para variable numéricas, que se puede extrapolar para la solución de diferentes situaciones, siempre y cuando utilicen variables numéricas y estructura de datos tabulares.

Sin embargo, al trabajar con información espacial como imágenes o videos, ocupamos una diferente estructura de red neuronal que mejore el tiempo de entrenamiento y que aprenda lo suficiente para clasificar las imágenes.



Img 4. Diagrama de funcionamiento de una CNN.

Las Redes Neuronales Convolucionales o inglés Convolutional Neural Network (CNN), este de tipo de arquitectura consiste en un preprocesamiento para extraer la información importante de las imágenes con el objetivo que cada neurona aprenda eficientemente la información de las imágenes.

Cabe recalcar, una foto es un representación de en dos dimensiones de tres canales (R,G,B), no obstante, en la imagen anterior está en escala de grises, por lo que es de un canal (28,28,1), esto se representa que tenemos una imagen de 28px * 28px en un canal de escala de grises, el fin de esta anotación es poder pasar los parámetros adecuados a nuestro modelo para que aprenda, ejemplo: si estaríamos trabajando con redes densas, tendríamos una red de (28px*28px)784 neuronas de entrada y 10 neuronas de salida, representando cada neurona de salida una clase (0..9).

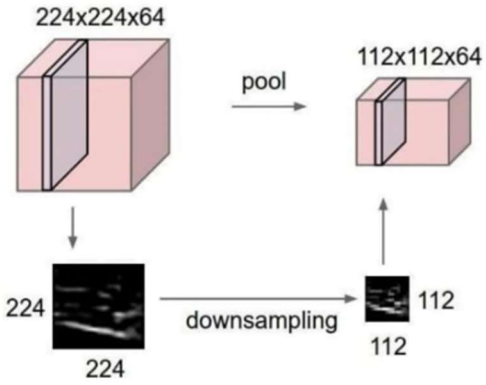
Por otra parte, queremos agregar mayor complejidad y poder descriptivo para nuestro modelo, por ende, primeramente, se le aplicará filtros para extraer la información esencial de la foto, así también reducir el consumo de recursos al momento de entrenar nuestro modelo.

Un filtro o kernel, en cuestiones matemática es un producto punto entre un filtro y una porción de la imagen de entrada,

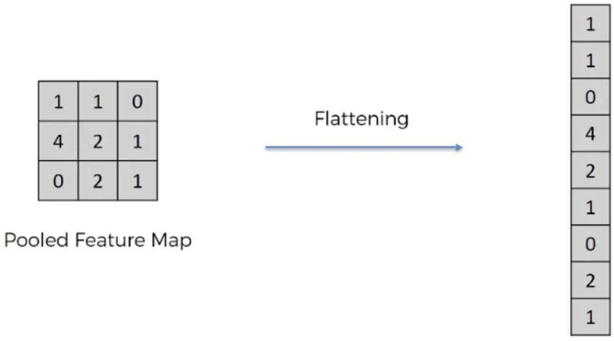
prácticamente es una pequeña matriz de pesos que se desliza sobre la imagen de entrada.

Luego de ello, tenemos la función max-pooling que se utiliza para explorar las características de la imagen.
La ventana se desplaza a través del mapa de características en pasos definidos, que se conocen como stride. La operación consiste en evaluar cada posición de la ventana, se toma el valor máximo dentro de la región cubierta por la ventana.

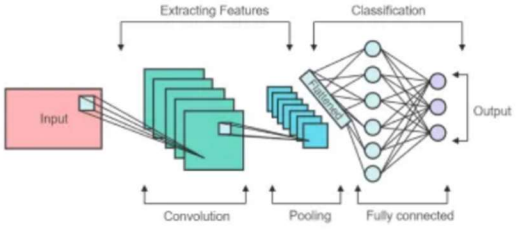
Este valor máximo se convierte en el nuevo valor en la posición correspondiente del mapa de salida, este elemento se utiliza para reducir tamaño de la imagen.



Luego tenemos el flatten es una operación clave para las CNN, ya que convierte un tensor multidimensional en un vector unidimensional, con el objetivo de conectar capas de las características extraídas (convolucionales y de pooling) a las capas completamente conectadas.



Al entender que la esencia de una CNN es combinar pre-procesos de transformación para agregar información esencial para entrenar nuestro modelo.



D. *Hiperparámetros Redes Neuronales*

Los hiperparámetros son variables que no se aprenden directamente del proceso de entrenamiento del modelo. Se establece antes del proceso del entrenamiento de la red y afectan en la forma en que se entrena el modelo.

Los parámetros de una red neuronal son los valores (pesos y sesgos) que aprende el algoritmo durante el proceso de entrenamiento para minimizar la función de coste y mejorar las predicciones, que en otras palabras son las conexiones ponderadas entre las neuronas en la red.

En los hiperparámetros lo podemos clasificar en dos grupos:

1. Nivel estructural; número de neuronas por capas, número de capas, inicialización de los pesos, funciones de activación, etc…
2. Nivel del algoritmo de aprendizaje; epochs, momentum, learning rate, batch size, etc…

Por otro lado, los hiperparámetros que tienen el mayor impacto en el rendimiento y capacidad de generalización de la red neuronal son (Javier, 2024):

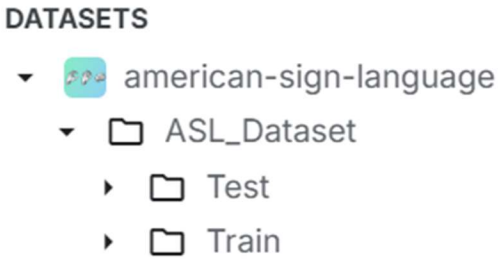
- Numero de capas y la capacidad de neuronas
- Tasa de aprendizaje
- Regularización
- Tamaño del lote
- Función de activación

IV. DATASET

En este proyecto abordaremos el lenguaje de señas, por lo que se ocupamos un dataset de Kaggle “American Sign Langage”, el cual cuenta con 28 clases, que son las 26 letras del vocabulario americano, espacio y vacío, la información total son 4.98Gb entre la sección de entrenamiento y prueba, cada imagen es de 400x400 en RGB. Dataset

V. ANÁLISIS DEL DATASET

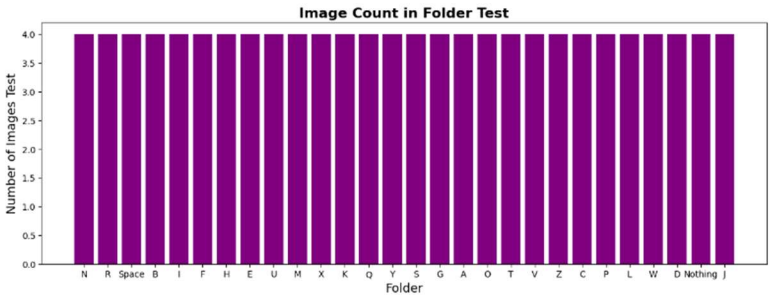
La estructura del dataset consiste en folder de entrenamiento y prueba, cada folder tiene sub-folder con las 28 clases, y cada sub-folder tiene guardado las fotos correspondientes a su clase.



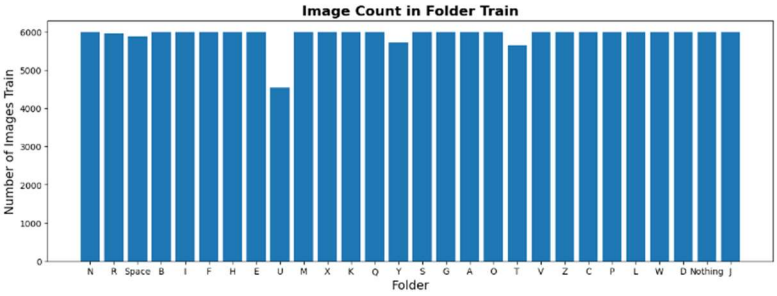
En el set de entrenamiento y prueba tienen las siguientes clases guardas: 'N', 'R', 'Space', 'B', 'I', 'F', 'H', 'E', 'U', 'M', 'X', 'K', 'Q', 'Y',

'S', 'G', 'A', 'O', 'T', 'V', 'Z', 'C', 'P', 'L', 'W', 'D', 'Nothing', 'J'.

La distribución del folder test y train:



En el folder Test está conformado por las 28 clases mencionadas anteriormente, y cada clase cuenta con 4 imágenes para hacer predicciones.



En el folder Train está conformado por las 28 clases mencionadas anteriormente, y cada clase cuenta con más 4000 imágenes, no obstante, la distribución no es homogénea.

A continuación, haremos la división de nuestro dataset para definir set de entrenamiento, validación y prueba, para ello utilizaremos la extensión ImageDataGenerator de Keras para hacer directo la división de los datos y transformaciones pertinentes a las imágenes.

El folder Train cuenta 165,670 imágenes totales, por lo que, se ocuparan 80% para entrenamiento y 20% validación, y el folder Test son 112 imágenes totales, estas imágenes las ocuparemos para predicciones de nuestro modelo. Además, para los tres sets, se distribuirán en batches de 16 imágenes, y un reajuste de tamaño de las imágenes de 224px * 224px.

Debo de resaltar, que, al set de entrenamiento y validación, aplicaremos normalización (0 a 1), cortes aleatorios de hasta 20% de la imagen, zoom de hasta 20% de la imagen, volteamos aleatoriamente las imágenes horizontales, giramos aleatoriamente las imágenes hasta 30 grados, cambio de altura de las imágenes hasta un 20%, y reserva el 20% de los datos para su validación.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,           # Normal
    shear_range=0.2,          # Apply
    zoom_range=0.2,           # Random
    horizontal_flip=True,     # Flip t
    rotation_range=30,        # Rotate
    width_shift_range=0.2,    # Shift
    height_shift_range=0.2,   # Shift
    validation_split=0.2      # Reserv
)
```

Las transformaciones anteriores tienen el propósito de aumentar la cantidad de los datos y variedad de las imágenes con el fin de evitar el overfitting, que es, que el modelo sobre aprender los patrones de las imágenes.

De otra manera, al set de prueba solo vamos a normalizar (0 a 1) las imágenes.

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

VI. MODEL RED NEURONAL CONVOLUCIONAL

Al tener los datos que se utilizarán para entrenar, validar y prueba, podemos empezar con la construcción de nuestra red neuronal convolucional, anteriormente mencionamos que consiste un preprocesado en donde se extrae características esenciales de las imágenes.

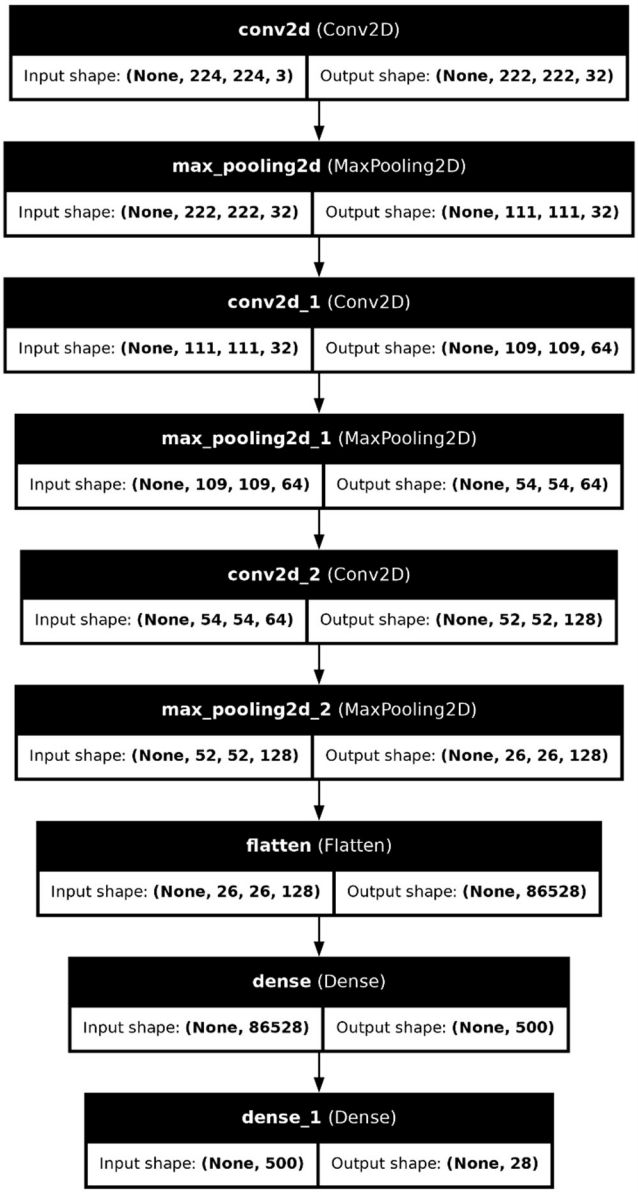
Para la construcción de nuestro modelo estaremos trabajando con Keras. Importamos las dependencias del framework que ocuparemos

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D , Dropout , Dense, Conv2D, MaxPooling2D, Flatten, Input
```

Diseñamos una red neuronal convolucional, que consta de una entrada de 224px*224px, una capa convolucional.

```
cnn=Sequential()
cnn.add(Input(shape=(224, 224, 3)))
cnn.add(Conv2D(32,(3,3), activation='relu'))
cnn.add(MaxPooling2D(2,2))
cnn.add(Conv2D(64, (3,3), activation='relu'))
cnn.add(MaxPooling2D(2,2))
cnn.add(Conv2D(128,(3,3), activation='relu'))
cnn.add(MaxPooling2D(2,2))
cnn.add(Flatten())
cnn.add(Dense(500, activation='relu'))
cnn.add(Dense(units=train_generator.num_classes,activation='softmax'))
```

```
cnn.compile(
    optimizer=Adam(),
    loss='categorical_crossentropy',
    metrics=[
        Recall,
        Precision,
        'accuracy'
    ]
)
```




```
history2=cnn.fit(train_generator,
                 epochs=5,
                 validation_data=validation_generator,
                 steps_per_epoch=train_generator.samples // train_generator.batch_size,
                 validation_steps=validation_generator.samples // validation_generator.batch_size)
```

```
Epoch 1/5
8283/8283 ━━━━━━━━━━━━━━━━━━━━ 2556s 308ms/step - accuracy: 0.6107 - loss: 1.2188 - precision_1: 0.8070 - recall_1: 0.5201 - val_accuracy: 0.9122 - val_loss: 0.2577 - val_precision_1: 0.9282 - val_recall_1: 0.8993
Epoch 2/5
8283/8283 ━━━━━━━━━━━━━━━━━━━━ 1s 107us/step - accuracy: 1.0000 - loss: 0.0808 - precision_1: 1.0000 - recall_1: 1.0000 - val_accuracy: 1.0000 - val_loss: 0.0153 - val_precision_1: 1.0000 - val_recall_1: 1.0000
Epoch 3/5
8283/8283 ━━━━━━━━━━━━━━━━━━━━ 2542s 307ms/step - accuracy: 0.9410 - loss: 0.1729 - precision_1: 0.9485 - recall_1: 0.9353 - val_accuracy: 0.9109 - val_loss: 0.2912 - val_precision_1: 0.9232 - val_recall_1: 0.9022
Epoch 4/5
8283/8283 ━━━━━━━━━━━━━━━━━━━━ 0s 13us/step - accuracy: 0.8125 - loss: 0.5723 - precision_1: 0.8125 - recall_1: 0.8125 - val_accuracy: 1.0000 - val_loss: 0.0247 - val_precision_1: 1.0000 - val_recall_1: 1.0000
Epoch 5/5
8283/8283 ━━━━━━━━━━━━━━━━━━━━ 0s 251ms/step - accuracy: 0.9620 - loss: 0.1125 - precision_1: 0.9661 - recall_1: 0.9598
```

```
results2 = cnn.evaluate(test_generator)
print(results2)
```

```
val_accuracy: 0.9501 - val_loss: 0.1668 - val_precision_1: 0.9541 - val_recall_1: 0.9472
```

VII. MODEL RED NEURONAL CONVOLUCIONAL CON MODELO PRE-ENTRENADO

VIII. RESULTADOS

IX. IMPLEMENTACIÓN

X. CONCLUSIONES

El título debe estar en fuente tamaño 24 puntos. Los nombres de los autores en tamaño de 11 puntos. El nombre de la universidad y departamentos en letra tamaño 10 puntos y cursiva y finalmente los correos electrónicos en tamaño 9 puntos con una fuente tipo Courier.

TABLA I
TAMAÑOS DE FUENTE PARA ARTÍCULOS

Tamaño	Apariencia (en Time New Roman ó Times)		
	Regular	**Negrita**	**Cursiva**
8	Contenidos de tablas Título de figures Referencias a objetos	**Negrita**	*Cursiva*
9	Direcciones de correo electrónico (usar fuente Courier) Cuerpo del articulo	**Negrita Cuerpo del abstract**	*Cursiva*
10	Subtítulos	**Negrita**	*Cursiva*
11	Nombre del autor	**Negrita**	*Cursiva*
24	Título del artículo		

REFERENCIAS

Artificial Intelligence: A Modern Approach, 4th Global ed. (2022). Berkeley.edu. https://aima.cs.berkeley.edu/global-index.html

COCO Dataset Structure | Understanding Bounding Box Annotations for Object Detection
By Neuralception Container: YouTube Year: 2022 URL: https://www.youtube.com/watch?v=TLvdlDgZ3G0

GeeksforGeeks. (2017, August 21). *Introduction to Convolution Neural Network*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Javier. (2024, May 7). *Hiperparametros de una red neuronal*. Javierheras.website; Inteligencia Artificial. https://blog.javierheras.website/hiperparametros-de-una-red-neuronal/

https://www.youtube.com/watch?v=Y_WAXe4LprY&list=PLWzLQn_hxe6YL-9JiKwIZqIuB8jliA-Fs