# Modelo Basado En Redes Neuronales Para La Clasificación De Imágenes De Tumores Cerebrales

William Peña[1] and Elkin Prada[2]

*Abstract*— **En el siguiente artículo, se construirán dos modelos basados en redes neuronales para el procesamiento de imágenes y la clasificación de tumores cerebrales utilizando el dataset de Kaggle (Brain Tumor Classification (MRI)), con el fin de poner en práctica los conceptos de deep learning y de manera experimental comprobar la relación que existe entre definir una buena arquitectura de red neuronal, la normalización de los datos y el ajuste de hiperparámetros, así como definir el número de iteraciones (Épocas) y una función de pérdida y ajuste de pesos adecuada para variar el modelo y tender a un buen desempeño. Para ello se inicia con la construcción de una red neuronal feed fordward de pocas capas y se realiza el entrenamiento modificando la cantidad de épocas e hiperparámetros, posterior a ello se construye un modelo convolucional con varias capas ocultas y filtros sobre la imagen, se aplicarán capas de Max pooling y flattening para manejar el filtrado y aplanamiento de las imágenes y de esta manera extraer más características en un vector resultante. Finalmente, se comparan los resultados de los modelos en la variación de épocas y ajuste de parámetros y se grafica el desempeño y la función de pérdida a lo largo de cada iteración. Determinando que este problema de depp learning se puede resolver con una red neuronal con pocas capas ocultas, pero con una mejor distribución del dataset y un manejo categórico de las etiquetas haciendo uso de one-hot encoding. De igual manera, se determina que es necesario agregar varias capas dropout para la regularización y evitar el sobre ajuste (overfitting) y aumentar la cantidad de epocas para tener un mejor desempeño y disminuir el error.**

## I. INTRODUCCIÓN

En los últimos años, la aplicación de técnicas de Deep Learning ha revolucionado la forma en que se aborda el procesamiento de imágenes. En particular, las Redes Neuronales Convolucionales (Convolutional Neural Networks o CNNs) se han convertido en una herramienta esencial para la clasificación y detección de objetos en imágenes. Sin embargo, para tareas muy complejas, una sola red neuronal puede no ser suficiente para alcanzar un alto rendimiento en términos de precisión y eficiencia. En este contexto, las Redes Neuronales Convolucionales Apiladas (Stacked CNNs) han surgido como una alternativa prometedora para mejorar la capacidad de clasificación de las CNNs tradicionales. Estas arquitecturas consisten en apilar varía modelo de Red Neuronal Convolucionals capas de redes neuronales convolucionales para formar una red profunda que pueda extraer características complejas de los datos de entrada. En este artículo, se propone una implementación de redes neuronales apiladas para clasificar imágenes de objetos en una base de datos ampliamente utilizada, obteniendo resultados prometedores en términos de precisión y tiempo de procesamiento.

## II. OBJETIVO

El objetivo principal de este proyecto es el de analizar y construir diferentes modelos de redes neuronales para la clasificación de imágenes de tumores celébrales utilizando el dataset de Kaggle (Brain Tumor Classification (MRI)), con el fin de explorar la relación entre parámetros, capas y algoritmos que componen una arquitectura de red neuronal, comprendiendo los aspectos más relevantes a la hora de buscar la solución más factible para un problema de Deep Learning. Para ello se tendrán en cuenta las siguientes etapas de análisis y experimentación.

- Analizar la dimensionalidad de los datos
- Construir modelo de Red Neuronal feed fordward
- Construir modelo de Red Neuronal Convolucional

[1]Maestría en Inteligencia Artificial
[2]Maestría en Ingenieria de Sistemas y Computación

- Análisis de resultados y métricas de desempeño.

## III. Análisis del problema

### *III-A. Descripción del problema*

Un tumor cerebral se erige como una de las enfermedades más implacables tanto en la infancia como en la adultez. Los tumores cerebrales, abarcando del 85 al 90 por ciento de todos los tumores primarios del sistema nervioso central (SNC), poseen un impacto significativo en la salud humana. Anualmente, aproximadamente 11,700 individuos son diagnosticados con esta afección. Para los tumores cerebrales malignos o del SNC, las tasas de supervivencia a cinco años rondan el 34 por ciento en hombres y el 36 por ciento en mujeres. Estos tumores se subdividen en categorías tales como benignos, malignos, pituitarios, entre otros. Para elevar la esperanza de vida de los pacientes, resulta imprescindible implementar tratamientos adecuados, planificaciones minuciosas y diagnósticos precisos.

En el ámbito de la detección de tumores cerebrales, la resonancia magnética (RM) emerge como la técnica más precisa y confiable. A través de esta modalidad de escaneo, se genera una ingente cantidad de datos de imágenes que, posteriormente, son sometidos al escrutinio minucioso de los radiólogos. No obstante, el examen manual adolece de errores inherentes debido a la complejidad que rodea a los tumores cerebrales y sus características distintivas.

Es en este contexto que las técnicas de clasificación automatizada, respaldadas por el aprendizaje automático (ML) y la inteligencia artificial (AI), han demostrado una precisión superior a la clasificación manual. En este sentido, proponer un sistema que despliegue capacidades de detección y clasificación, valiéndose de algoritmos de aprendizaje profundo como las Redes Neuronales Convolucionales (CNN), las Redes Neuronales Artificiales (ANN) y la Transferencia de Aprendizaje (TL), se erige como una herramienta de inmenso valor para los profesionales médicos en todas partes del mundo. Mediante este enfoque, se espera otorgarles un respaldo adicional en sus labores diarias y contribuir a la mejora de la precisión y eficiencia en el diagnóstico de esta enfermedad.

### *III-B. Dimensionalidad de los datos*

Se emplea el dataset Brain Tumor Classification (MRI) proporcionado en Kaggle para la construcción de los modelos. El dataset tiene un peso de 91MB y contiene 3264 imágenes divididas inicialmente en dos paquetes, un paquete para las imágenes de entrenamiento y un paquete para las imágenes de prueba. Cada uno de estos paquetes contiene las imágenes divididas en cuatro clases, que representan tres tipos de tumor y un paquete para aquellas imágenes que no tienen dicha anomalía. En la siguiente tabla se puede observar la distribución de imágenes por cada una de las clases para los datos de entrenamiento y validación.

| Datos de entrenamiento | | | |
|---|---|---|---|
| glioma_tumor | meningioma_tumor | no_tumor | pitutary_tumor |
| 826 | 822 | 395 | 827 |
| Datos de Prueba | | | |
| glioma_tumor | meningioma_tumor | no_tumor | pitutary_tumor |
| 100 | 115 | 105 | 74 |

TABLE I: Cantidad de imágenes

De igual manera, en el siguiente gráfico podemos observar la distribución de imágenes por clase y su porcentaje con respecto a la cantidad total, tanto para las imágenes de prueba como de entrenamiento.

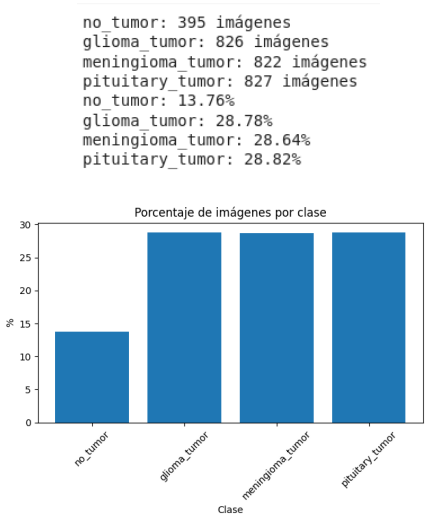



Fig. 1: Imágenes de entrenamiento

Cada paquete correspondiente a una clase, contiene imágenes en diferentes dimensiones a nivel de píxeles, formato de la imagen JPEG en escala

```
no_tumor: 105 imágenes
glioma_tumor: 100 imágenes
meningioma_tumor: 115 imágenes
pituitary_tumor: 74 imágenes
no_tumor: 26.65%
glioma_tumor: 25.38%
meningioma_tumor: 29.19%
pituitary_tumor: 18.78%
```

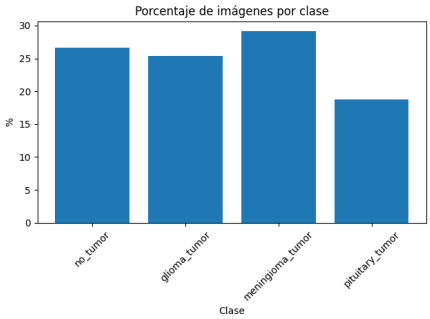
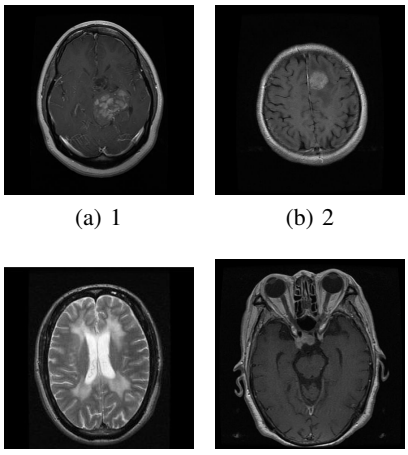



Fig. 2: Imágenes de prueba

de grises. A continuación se aprecia un ejemplo de imagen de cada clase del dataset que se empleará en la construcción de los modelos, nótese la dimensión de la imagen:



(a) 1 (b) 2

(c) 3 (d) 4

Fig. 3: Ejemplo de imágenes del dataset de acuerdo a su clase: 1. gliona, 2. meningioma, 3. no tumor, 4. pitutary.

Explorando un poco las características de una de las imágenes, en este caso la imagen “gg (8).jpg” del paquete glioma_tumor del set de datos de entrenamiento, se puede apreciar que las dimensiones de la imagen son de 512 x 512 de 3 canales, de tipo uint8 con un número total de píxeles en toda la imagen de 786432 píxeles. Estas mismas características se pueden apreciar en las demás imágenes con rangos de 0 a 255 píxeles, dando de entrada información importante para determinar que filtros y algoritmos emplear en el modelo a construir.

### *III-C. Metodología*

La metodología para la construcción de los modelos se basa en los siguientes puntos.

- Realizar nuevamente el dimensionamiento del dataset, es decir, distribuir mejor el dataset de manera que se definan nuevos conjuntos de entrenamiento y pruebas manteniendo la división de los mismos. En este caso utilizaremos train_test_split de Scikit-learn para dicho proceso.
- Leer imágenes y cargarlas en una matriz, se empleará OpenCV (cv2) para generar matrices NumPy basadas en las imágenes reales.
- Ajustar tamaño de las imágenes, redimensionando mediante la aplicación de un factor de escala, esto con el fin de dejar todas las imágenes en un tamaño más sencillo de trabajar.
- Aplicar Algoritmo one-hot encoding, para ello se utilizará ‘to_categorical’ de Keras, de esta manera, convertir el problema en un problema de clasificación, donde las etiquetas (Clases) serán representadas por una matriz binaria.
- Construcción del modelo 1 feed fordward con pocas capas.
- Definición de hiperparámetros, épocas y entrenamiento del modelo 1.
- Ajuste de arquitectura y parámetros.
- Construcción del modelo 2, Red neuronal convolucional.
- Entrenamiento del modelo 2 y ajuste de parámetros.
- Análisis de resultados y comparación de los modelos.

Basados en la serie de pasos mencionados anteriormente, se construirán 2 modelos de red neuronal para el procesamiento de imágenes, con el fin de clasificar encontrar tumores cerebrales. Para ello se compararán los 2 modelos empleados y se realizará un análisis de resultados con valores graficados de efectividad, rendimiento y pérdida de error.

## IV. Modelo Feed fordward

### *IV-A. Construcción*

El modelo 1 se desarrolla en Google Colaboratoy (Colab), ya que nos permite acceso a GPU sin costos adicionales y sin una configuración previa para este laboratorio experimental. A continuación descripción de tecnologías y librerías a utilizar:

- Python v 3.10.11
- Google Colab
- NumPy=1.22.4
- pandas=1.5.3
- TensorFlow=2.12.0
- Keras=2.12.0
- os
- Matplotlib=3.7.1
- cv2
- glob
- zipfile

### *IV-B. Construcción y cargue del dataset*

Se realiza la descarga del dataset Brain Tumor Classification (MRI) proporcionado en Kaggle y se carga a una unidad de drive, posteriormente se monta la unidad a Colab con drive.mount y se descomprimen los archivos haciendo uso de la librería zipfile para tener las imágenes en sus paquetes respectivos e iniciar con la normalización de los datos y la construcción del modelo.

### *IV-C. Modelo*

Se crea la siguiente arquitectura de red neuronal con las siguientes capas representadas en las figuras 4, 5, 6 y 7.
Para la definición y ejecución del modelo de red neuronal, se hace uso de la librería 'Sequential' de Keras, definiendo las capas con los siguientes parámetros.
Una vez agregadas las capas al modelo, veamos ahora la definición del mismo, y el número total de parámetros de cada capa que pasará por la etapa de entrenamiento.

### *IV-D. Entrenamiento*

En esta etapa se configuran los parámetros de compilación del modelo y se define la cantidad de épocas de iteración para el proceso de entrenamiento. Se trabaja con la función de pérdida categorical_crossentropy basada en entropía cruzada (cross-entropy), diseñada para trabajar con etiquetas categóricas

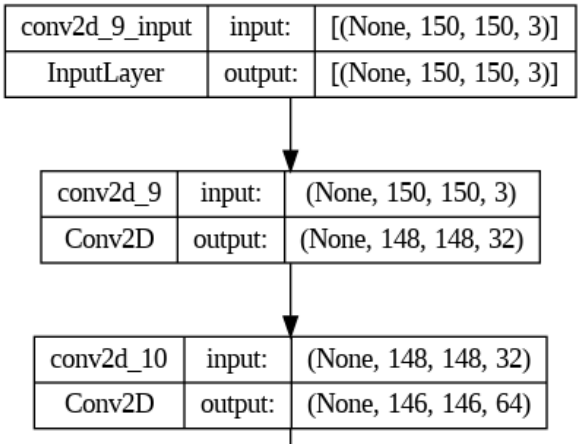



Fig. 4: Arquitectura Modelo 1 - Entrada y primeras 2 capas

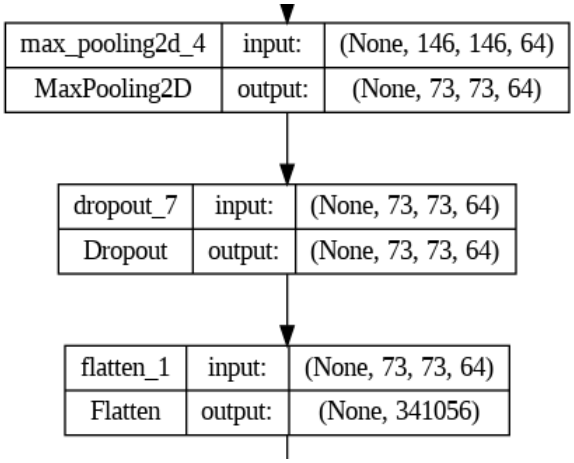



Fig. 5: Capa de MaxPooling, Dropout y Flattening

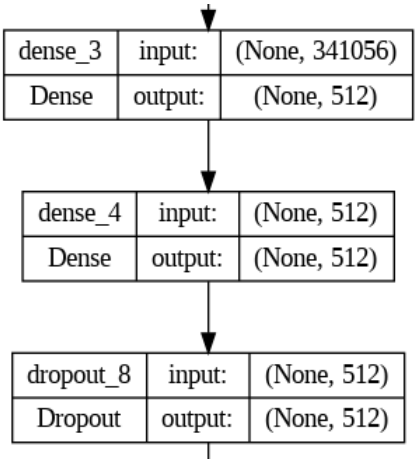



Fig. 6: 2 capas ocultas y dropout

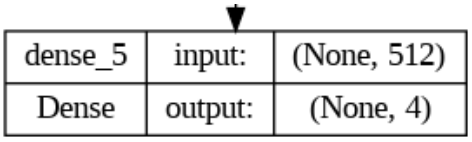



Fig. 7: Capa final de salida

```
[ ]  model1 = Sequential()

[ ]  model1_capa1 = Conv2D(
         filters = 32,
         kernel_size = (3,3),
         activation = 'relu',
         input_shape=(150,150,3)
     )

[ ]  model1_capa2 = Conv2D(
         filters = 64,
         kernel_size = (3,3),
         activation = 'relu'
     )
```

Fig. 8: Inicialización del modelo y primeras 2 capas

```
[ ] #Capa 1: Max Pooling
    model_1_max_pooling_1 = MaxPooling2D(pool_size = (2,2))

[ ] #Capa Dropout 2
    model_1_dropout_1 = Dropout(0.3)

[ ] model_1_flattening = Flatten()
```

Fig. 9: Capa de MaxPooling, Dropout y Flattening

```
    model_1_dense_2 = Dense(
        512,
        activation = "relu"
    )

[ ] model_1_dropout_2 = Dropout(0.3)

[ ] #Capa de salida
    model_1_dense_3 = Dense(
        4,
        activation = "softmax"
    )
```

Fig. 10: 2 capas ocultas, dropout y salida

generadas por el algoritmo one-hot encoding y representa diferencias entre dos distribuciones de probabilidad. Como algoritmo de ajuste de pesos, se hace uso del optimizador 'Adam' (Adaptive Moment Estimation) y finalmente se configura accuracy como métricas de desempeño, que nos

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 148, 148, 32)      896

 conv2d_10 (Conv2D)          (None, 146, 146, 64)      18496

 max_pooling2d_4 (MaxPooling  (None, 73, 73, 64)       0
 2D)

 dropout_7 (Dropout)         (None, 73, 73, 64)        0

 flatten_1 (Flatten)         (None, 341056)            0

 dense_3 (Dense)             (None, 512)               174621184

 dense_4 (Dense)             (None, 512)               262656

 dropout_8 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 4)                 2052

=================================================================
Total params: 174,905,284
Trainable params: 174,905,284
Non-trainable params: 0
```

Fig. 11: Parámetros por capa

permite evaluar el modelo basado en la cantidad de preediciones correctas con respecto al número total de ejemplos.

A continuación se resalta el entrenamiento de este primer modelo con los parámetros ya mencionados para un total de 5 épocas.

```
20s 98ms/step - loss: 76.0294 - accuracy: 0.5376 - val_loss: 0.7508 - val_accuracy: 0.7347
7s 84ms/step - loss: 0.4423 - accuracy: 0.8475 - val_loss: 0.5417 - val_accuracy: 0.8367
7s 84ms/step - loss: 0.1558 - accuracy: 0.9478 - val_loss: 0.4886 - val_accuracy: 0.8741
7s 85ms/step - loss: 0.1000 - accuracy: 0.9773 - val_loss: 0.4405 - val_accuracy: 0.8673
7s 84ms/step - loss: 0.0711 - accuracy: 0.9818 - val_loss: 0.5636 - val_accuracy: 0.8776
```

Fig. 12: Entrenamiento para 5 epocas.

```
11/11 [==============================] - 0s 22ms/step - loss: 0.5599 - accuracy: 0.8685
11/11 [==============================] - 0s 17ms/step - loss: 0.5599 - accuracy: 0.8685

accuracy: 86.85%
```

Fig. 13: Resultado 5 epocas

Como se puede apreciar en la gráfica de función de pérdida, para la parte de entrenamiento, itero muy rápido, inicio con un margen de error muy alto y luego llego a un valor muy bajo de una

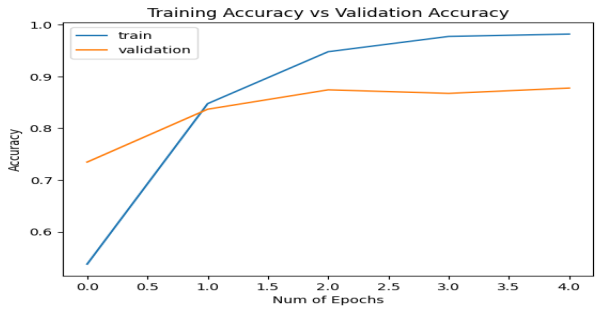



Fig. 14: Gráfica de desempeño accuracy

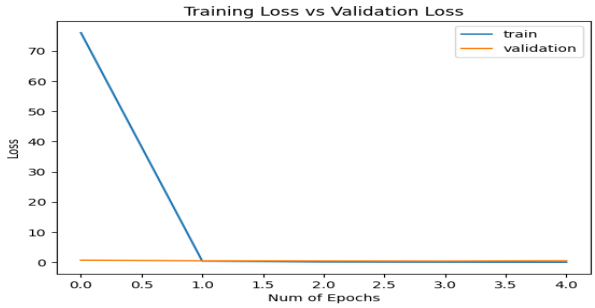



Fig. 15: Función de pérdida Loss

manera muy eficiente. Sin embargo, en los datos de validación esto no sucedió de forma eficiente, se estancó en un punto, lo cual indica un claro ejemplo de overfitting. De esta manera, se procede a iterar con una cantidad más alta de épocas, en este caso 20 épocas, pero el resultado no es muy favorable.

Teniendo en cuenta lo anterior, se realiza un ajuste sobre la arquitectura del modelo, agregando más capas y ajustando los hiperparámetros, iterando sobre 20 épocas con la arquitectura expuesta en el modelo hibrido.

### *IV-E. Implementación*

## V. Modelo Convolucional

### *V-A. Construcción*

En la construcción del modelo de Red Neuronal Convolucional (CNN) para la clasificación de imágenes de tumores, se utilizó principalmente la biblioteca especializada TensorFlow y Keras. Además, se emplearon algunas otras bibliotecas auxiliares como NumPy y seaborn para la manipulación de datos y la generación de gráficas.

El proceso de construcción del modelo CNN implica la creación de capas convolucionales, capas de agrupación (pooling) y capas totalmente conectadas. Las capas convolucionales desempeñan un papel fundamental al extraer características relevantes de las imágenes, mientras que las capas de agrupación reducen la dimensionalidad y preservan las características más importantes. Por su parte, las capas totalmente conectadas son responsables de llevar a cabo la clasificación final de las características extraídas. Asimismo, se incluyeron capas de dropout con el fin de mitigar el sobre ajuste del modelo.

En cuanto al algoritmo de optimización, se optó por utilizar Adam, el cual es ampliamente reconocido por su eficiencia y capacidad para adaptar la tasa de aprendizaje de manera dinámica durante el proceso de entrenamiento.

Se exploraron varias configuraciones, a continuación se muestran.

**Modelo 1**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 conv2d_4 (Conv2D)           (None, 5, 5, 256)         295168

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 64)                409664

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 835,268
Trainable params: 835,268
Non-trainable params: 0
_________________________________________________________________
```

Fig. 16: CNN con 4.4447.044 parámetros

Esta arquitectura de CNN consta de varias capas convolucionales y de agrupación (pooling), junto con capas de dropout para evitar el sobre

ajuste. Las capas convolucionales tienen diferentes tamaños de filtro y generan mapas de características de salida. El modelo también incluye capas totalmente conectadas que realizan la clasificación final. En total, la red tiene 835,268 parámetros entrenables. El número de salida final es 4, lo que indica que se utiliza para la clasificación de 4 clases diferentes de imágenes de tumores.

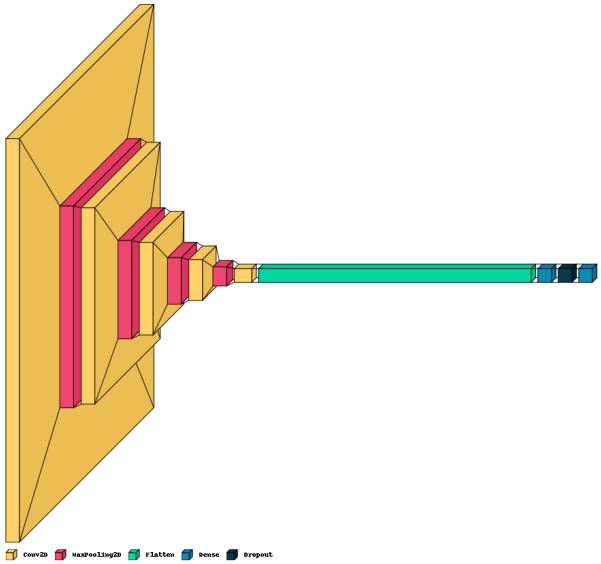



Fig. 17: Esquema de la CNN con 835.268 parámetros

**Modelo 2**

Esta red neuronal utiliza un número menor de capas convolucionales y de pooling para extraer características relevantes de las imágenes.

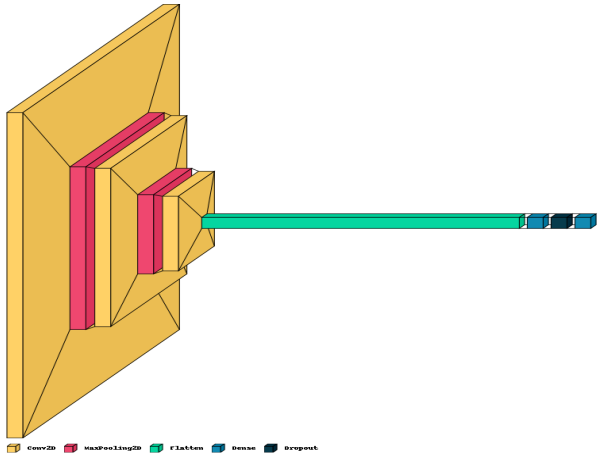



Fig. 18: Esquema de la CNN con 835,268 parámetros

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 148, 148, 32)      896

 max_pooling2d_12 (MaxPoolin  (None, 74, 74, 32)       0
 g2D)

 conv2d_19 (Conv2D)          (None, 72, 72, 64)        18496

 max_pooling2d_13 (MaxPoolin  (None, 36, 36, 64)       0
 g2D)

 conv2d_20 (Conv2D)          (None, 34, 34, 64)        36928

 flatten_6 (Flatten)         (None, 73984)             0

 dense_12 (Dense)            (None, 64)                4735040

 dropout_6 (Dropout)         (None, 64)                0

 dense_13 (Dense)            (None, 4)                 260

=================================================================
Total params: 4,791,620
Trainable params: 4,791,620
Non-trainable params: 0
_________________________________________________________________
```

Fig. 19: CNN con 4.4791.620 parámetros

### *V-B. Entrenamiento*

El entrenamiento de la Red Neuronal Convolucional (CNN) se realizó mediante un proceso iterativo que constó de 10 épocas. Con el objetivo de mejorar la eficacia y la generalización del modelo, se optó por unificar las imágenes de entrenamiento y prueba en un único conjunto de datos. A partir de este conjunto consolidado, se llevó a cabo una cuidadosa partición que permitió establecer un conjunto de entrenamiento y un conjunto de validación, con una proporción del 80 % y 20 %, respectivamente.

Esta estrategia de unificación y partición de datos garantiza una mejor evaluación del rendimiento del modelo, ya que se utiliza un conjunto de validación independiente para ajustar los hiperparámetros y controlar el proceso de entrenamiento. Además, evita la posibilidad de que el modelo memorice o sobre ajuste los datos de entrenamiento específicos, lo cual podría comprometer su capacidad para generalizar y clasificar correctamente nuevas imágenes. Al unificar las imágenes de entrenamiento y prueba en un único conjunto de datos, se logra una mayor variabilidad y diversidad

en las muestras presentadas a la CNN durante el entrenamiento. Esto contribuye a mejorar la robustez y la capacidad de generalización del modelo, ya que se expone a una amplia gama de características y patrones presentes en las imágenes de tumores.

Posteriormente, la partición del conjunto de datos en un 80 % para entrenamiento y un 20 % para validación permite establecer un equilibrio entre la cantidad de datos utilizados para el ajuste de los pesos de la red y la evaluación del desempeño del modelo. El conjunto de entrenamiento, con el 80 % de las imágenes, se emplea para actualizar los pesos de la CNN a través de la retropropagación del error y minimizar la pérdida. Mientras tanto, el conjunto de validación, con el 20 % restante de las imágenes, se utiliza para monitorear el rendimiento del modelo y ajustar los hiperparámetros con el fin de evitar el sobre ajuste y garantizar una clasificación precisa.

**Modelo 1**

El entrenamiento de la red neuronal se realizó durante 30 épocas. En cada época, se procesaron 82 lotes de datos, con una duración promedio de 2 segundos por lote. Los resultados muestran que el modelo fue capaz de aprender y mejorar su rendimiento a lo largo del entrenamiento, logrando una alta precisión en la clasificación de los datos de prueba con una cantidad relativamente pequeña de parámetros frente al modelo 2.

**Modelo 2**

Durante el entrenamiento de la red neuronal, se realizaron 10 épocas de entrenamiento. En cada época, se procesaron 82 lotes de datos, con una duración promedio de 2 segundos por lote. Durante la primera época, se obtuvo una pérdida (loss) de 1.1328 y una precisión (accuracy) de 0.4964 en el conjunto de entrenamiento, mientras que en el conjunto de validación se obtuvo una pérdida de 0.9059 y una precisión de 0.6646. A medida que avanzaban las épocas, tanto la pérdida como la precisión mejoraron significativamente. Al finalizar la décima época, se obtuvo una pérdida de 0.1302 y una precisión de 0.9494 en el conjunto de entrenamiento, y una pérdida de 0.4115 y una precisión de 0.8943 en el conjunto de validación. Estos resultados muestran que el modelo fue capaz de aprender y mejorar su rendimiento a lo largo del entrenamiento, logrando una alta precisión en la clasificación de los datos de prueba.

## VI. MODELO HÍBRIDO

Los modelos híbridos que combinan una red neuronal convolucional (CNN) y un modelo feed forward han demostrado ser eficaces en la clasificación de imágenes. Estos modelos aprovechan las capacidades de extracción de características de la CNN y la capacidad de aprendizaje de representaciones de alto nivel del modelo feed forward.

En estos modelos, la CNN se encarga de procesar las imágenes de entrada y extraer características relevantes a través de capas convolucionales y capas de agrupación. Estas capas convolucionales aplican filtros para detectar patrones locales en las imágenes y reducir la dimensionalidad, mientras que las capas de agrupación realizan un muestreo máximo para conservar las características más importantes.

Después de la etapa de extracción de características de la CNN, se utiliza un modelo feed forward para realizar la clasificación de alto nivel. El modelo feed forward consiste en capas densas completamente conectadas que aprenden a asociar las características extraídas por la CNN con las clases de clasificación. Estas capas densas son capaces de capturar relaciones no lineales y realizar una clasificación más precisa.

La combinación de la CNN y el modelo feed forward se realiza mediante la conexión de la salida de la CNN a la entrada del modelo feed forward. Esto permite que las características extraídas por la CNN se utilicen como entrada para el modelo feed forward, lo que mejora la capacidad del modelo para aprender representaciones más ricas y discriminativas.

El entrenamiento de estos modelos híbridos se realiza en etapas. Primero, se entrenan las capas de la CNN utilizando técnicas de aprendizaje profundo, como el descenso de gradiente estocástico. Luego, se congelan los pesos de la CNN y se entrenan las capas del modelo feed forward utilizando el gradiente descendente.

La combinación de la CNN y el modelo feed forward en un modelo híbrido permite aprovechar las fortalezas de ambos enfoques. La CNN es

capaz de capturar características locales y aprender representaciones jerárquicas de las imágenes, mientras que el modelo feed forward realiza la clasificación de alto nivel basada en estas características aprendidas. Esto resulta en modelos más potentes y precisos para la clasificación de imágenes en una variedad de tareas, como reconocimiento facial, detección de objetos y clasificación de enfermedades en imágenes médicas.

### *VI-A. Construcción*

La construcción del modelo híbrido se hace con una arquitectura CNN utilizando capas convolucionales y capas de agrupación (pooling), seguido de la definición del modelo feed forward. Luego, se combinan ambas arquitecturas en un solo modelo utilizando la clase Model de Keras. Finalmente, se compila, entrena y evalúa el modelo combinado utilizando los conjuntos de datos de entrenamiento, validación y prueba.

### *VI-B. Entrenamiento*

## VII. Resultados

**Modelo Feed forward**

**Modelo 1 CNN** El modelo se entrenó durante 30 épocas utilizando un conjunto de datos de entrenamiento y validación. En cada época, se realizaron 82 iteraciones (pasos) de entrenamiento.

La función de pérdida (loss) y la precisión (accuracy) se utilizaron como métricas para evaluar el rendimiento del modelo durante el entrenamiento.

Al comienzo del entrenamiento, en la primera época, se obtuvo una pérdida de 1.2777 y una precisión de 0.3945 en el conjunto de entrenamiento. En el conjunto de validación, la pérdida fue de 1.0564 y la precisión de 0.5498.

A medida que avanzaba el entrenamiento, tanto la pérdida como la precisión mejoraron progresivamente. Después de 30 épocas, la pérdida en el conjunto de entrenamiento se redujo a 0.0438, mientras que la precisión aumentó a 0.9828. En el conjunto de validación, la pérdida final fue de 0.4997 y la precisión de 0.9280.

**Modelo 2 CNN**

Se puede ver que el modelo tuvo un rendimiento sobresaliente al clasificar correctamente 176 instancias de la clase 3. Solo cometió 1 error al clasificar una instancia de la clase 0 y 3 errores al clasificar instancias de la clase 1.

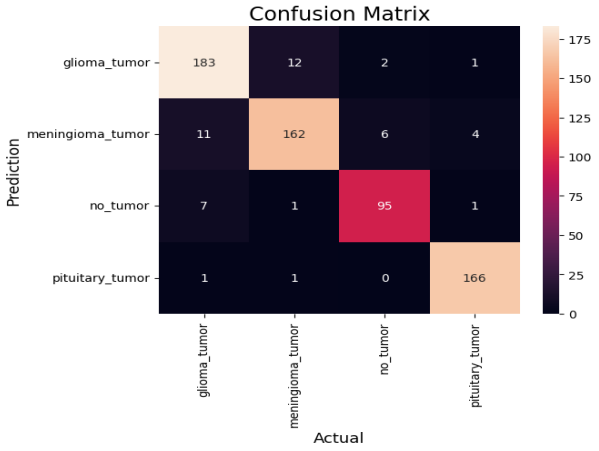



Fig. 20: Matriz de confusión modelo 1

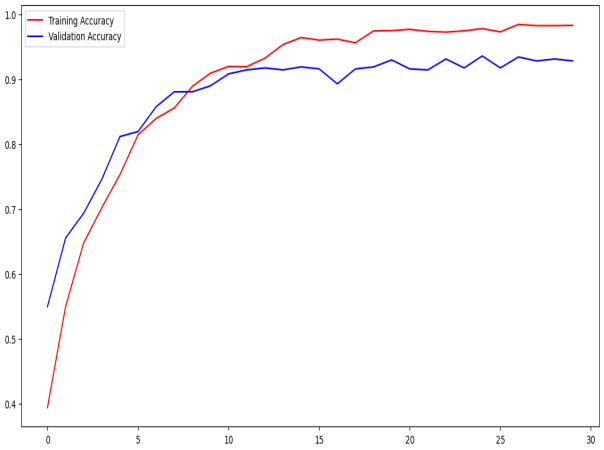



Fig. 21: Desempeños de la exactitud del entrenamiento y la validación

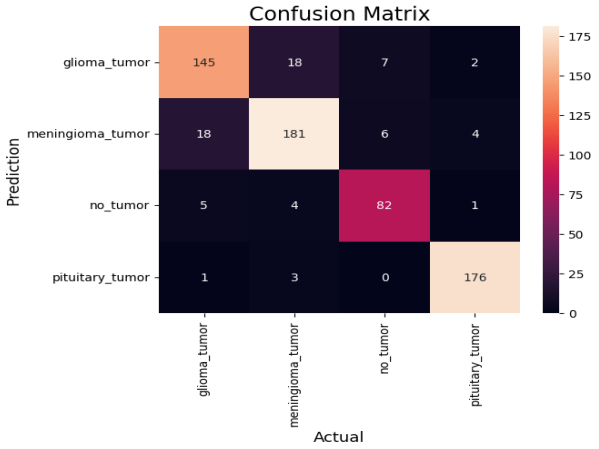



Fig. 22: Matriz de confusión modelo 2

## VIII. Modelo hibridó

### *VIII-A. Modelo*

Teniendo en cuenta los resultados del primer modelo, cuya tendencia era hacia el sobre ajus-

| Resultados | | |
|---|---|---|
| **Métrica** | **modelo 1** | **modelo 2** |
| Accuracy | 0.9280245022970903 | 0.8943338437978561 |
| Precision | 0.9278427020256547 | 0.8939282806145309 |
| Recall | 0.9280245022970903 | 0.8943338437978561 |
| F1-score | 0.9277714707425055 | 0.8940490659771071 |

TABLE II: Métricas de rendimiento

te (overfitting). Se construye un nuevo modelo, el cual costa de más capas ocultas, más capas MaxPooling y más capas dropout con el fin de extraer más características y regularizar el modelo para prevenir el sobre ajuste. De esta manera, con las capas dropout, se desactivan de forma aleatoria un conjunto de neuronas en el proceso de entrenamiento, forzando así a la red neuronal un aprendizaje orientado a un subconjunto aleatorio de neuronas. Nótese, en la siguiente figura (22), la arquitectura empleada para este nuevo modelo con nuevas capas y un número de parámetros superior.

### *VIII-B. Entrenamiento*

Se realiza el entrenamiento con los mismos parámetros del modelo anterior, función de pérdida categorical_crossentropy, optimizador "Adam" y métricas de desempeño accuracy. Inicialmente se itera con 10 épocas, pero para obtener mejores resultados se aumenta el número de épocas a un total de 20 épocas. En la figura 23, se puede apreciar el paso a paso de cada iteracón el comportamiento de la función de perdida y el desempeño de los casos acertados en la métrica accuracy.

### *VIII-C. Resultados*

Para este modelo se obtienen resultados más favorables, se pueden apreciar en la fig. 24 y 25 donde la gráfica de los datos de entrenamiento está relacionada con los datos de prueba, con un accuracy de 86.54 % A medida que pasaba por cada iteración el error disminuía considerablemente y el desempeño de casos de predicción acertados con respecto al número de ejemplos aumentaba para llegar a un total de 86 %.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896

conv2d_1 (Conv2D)            (None, 146, 146, 64)      18496

max_pooling2d (MaxPooling2D  (None, 73, 73, 64)        0
)

dropout (Dropout)            (None, 73, 73, 64)        0

conv2d_2 (Conv2D)            (None, 71, 71, 64)        36928

conv2d_3 (Conv2D)            (None, 69, 69, 64)        36928

dropout_1 (Dropout)          (None, 69, 69, 64)        0

max_pooling2d_1 (MaxPooling  (None, 34, 34, 64)        0
2D)

dropout_2 (Dropout)          (None, 34, 34, 64)        0

conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856

conv2d_5 (Conv2D)            (None, 30, 30, 128)       147584

conv2d_6 (Conv2D)            (None, 28, 28, 128)       147584

max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)       0
2D)

dropout_3 (Dropout)          (None, 14, 14, 128)       0

conv2d_7 (Conv2D)            (None, 12, 12, 128)       147584

conv2d_8 (Conv2D)            (None, 10, 10, 256)       295168

max_pooling2d_3 (MaxPooling  (None, 5, 5, 256)         0
2D)

dropout_4 (Dropout)          (None, 5, 5, 256)         0

flatten (Flatten)            (None, 6400)              0

dense (Dense)                (None, 512)               3277312

dense_1 (Dense)              (None, 512)               262656

dropout_5 (Dropout)          (None, 512)               0

dense_2 (Dense)              (None, 4)                 2052

=================================================================
Total params: 4,447,044
Trainable params: 4,447,044
Non-trainable params: 0
```

Fig. 23: Arquitectura modelo hibridó, se agregan más capas MaxPooling, Dropout y Flattening

## IX. Conclusiones

Podemos concluir los siguientes puntos de acuerdo a la creación y ajuste de los modelos.

- Es un problema complejo, para la solución del mismo se debe tener un dataset más amplio para un conjunto de datos de validación y aplicar diferentes algoritmos de normalización y filtrado, en este caso fue necesario mezclar nuevamente los ejemplos y dividir de nuevo los datos de entrenamiento y prueba, funciona mejor con la interpretación categórica de las etiquetas usando one-hot encoding.

```
28s 124ms/step - loss: 1.7157 - accuracy: 0.2868 - val_loss: 1.3482 - val_accuracy: 0.2891
8s 96ms/step - loss: 1.1492 - accuracy: 0.4862 - val_loss: 1.0429 - val_accuracy: 0.5238
8s 96ms/step - loss: 0.9324 - accuracy: 0.5846 - val_loss: 0.9160 - val_accuracy: 0.6122
8s 96ms/step - loss: 0.8164 - accuracy: 0.6413 - val_loss: 0.7495 - val_accuracy: 0.6803
8s 98ms/step - loss: 0.7019 - accuracy: 0.7030 - val_loss: 0.8179 - val_accuracy: 0.6361
8s 96ms/step - loss: 0.6661 - accuracy: 0.7212 - val_loss: 0.6690 - val_accuracy: 0.7347
8s 98ms/step - loss: 0.6070 - accuracy: 0.7442 - val_loss: 0.6002 - val_accuracy: 0.7517
8s 97ms/step - loss: 0.5418 - accuracy: 0.7719 - val_loss: 0.5616 - val_accuracy: 0.7381
8s 98ms/step - loss: 0.4828 - accuracy: 0.8036 - val_loss: 0.5255 - val_accuracy: 0.8095
8s 99ms/step - loss: 0.4124 - accuracy: 0.8343 - val_loss: 0.5305 - val_accuracy: 0.8299
8s 99ms/step - loss: 0.3734 - accuracy: 0.8513 - val_loss: 0.4612 - val_accuracy: 0.8367
8s 100ms/step - loss: 0.3339 - accuracy: 0.8763 - val_loss: 0.4164 - val_accuracy: 0.8673
8s 100ms/step - loss: 0.2718 - accuracy: 0.8956 - val_loss: 0.4313 - val_accuracy: 0.8571
8s 99ms/step - loss: 0.2483 - accuracy: 0.9024 - val_loss: 0.4584 - val_accuracy: 0.8639
8s 98ms/step - loss: 0.2141 - accuracy: 0.9255 - val_loss: 0.3963 - val_accuracy: 0.8707
8s 98ms/step - loss: 0.1956 - accuracy: 0.9319 - val_loss: 0.4038 - val_accuracy: 0.8741
8s 98ms/step - loss: 0.1802 - accuracy: 0.9440 - val_loss: 0.3948 - val_accuracy: 0.8605
8s 98ms/step - loss: 0.1265 - accuracy: 0.9550 - val_loss: 0.4207 - val_accuracy: 0.8605
8s 96ms/step - loss: 0.1605 - accuracy: 0.9474 - val_loss: 0.3572 - val_accuracy: 0.8844
8s 98ms/step - loss: 0.1588 - accuracy: 0.9463 - val_loss: 0.4099 - val_accuracy: 0.8741
```

Fig. 24: Entrenamiento modelo hibrido, 20 epocas

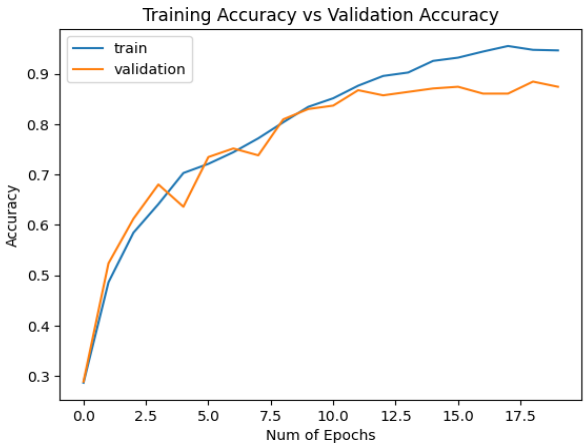



Fig. 25: Accuracy

- Es necesario agregar más capas de Max pooling y dropout para descomponer en más características y evitar overfitting.
- En este problema, fue importante el número de épocas, fue necesario aumentar el número de épocas para aumentar él desempeñó y bajar la pérdida.
- En el ámbito de los modelos convolucionales, se han obtenido resultados notablemente superiores al emplear una red con un 25 % de los parámetros de la red más grande. Este enfoque se basa en la idea de que la efectividad de una red neuronal no solo depende de su tamaño o complejidad, sino de la capacidad de aprender y extraer características relevantes de los datos. Al reducir la cantidad de parámetros de la red, se promueve una mayor eficiencia computacional y una menor propensión al sobre ajuste.
- Al reducir la cantidad de parámetros, se logra una mejor generalización del modelo, lo que significa que este será capaz de desempeñarse de manera más efectiva en datos no vistos previamente. Esta propiedad es esencial en aplicaciones del mundo real, donde la capacidad de adaptación a diferentes escenarios y conjuntos de datos es fundamental.

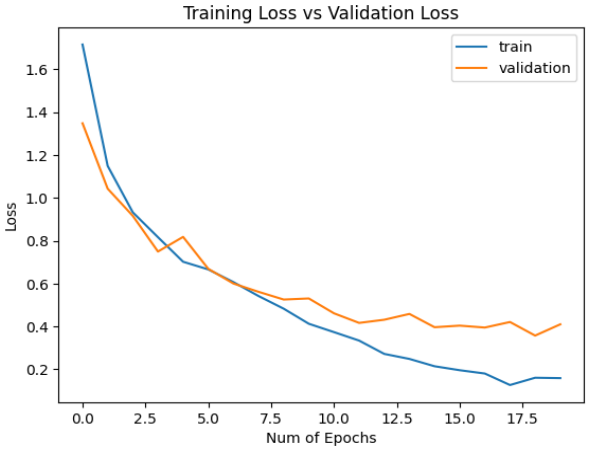



Fig. 26: Loss

## REFERENCES


[1] 1. Yan, M.; Ruan, Z.; Qiu, M. (2007). Çylindrical Invisibility Cloak with Simplified Material Parameters is Inherently Visible".

[2] Ruan, Z.; Yan, M.; Neff, C. W.; Qiu, M. (2007). Ïdeal Cylindrical Cloak: Perfect but Sensitive to Tiny Perturbations".

[3] Greenleaf, A.; Kurylev, Y.; Lassas, M.; Uhlmann, G. (2007). Ïmprovement of cylindrical cloaking with the SHS lining".

[4] Yu Luo, Jingjing Zhang, Hongsheng Chen, Bae-Ian Wu, and Jin Au Kong(2007). .[A] new strategy to conceal an object from electromagnetic wave"

[5] Chattopadhyay, A., Maitra, M. (2022). MRI-based brain tumour image detection using CNN based deep learning method. Neuroscience Informatics, 2(4), 100060. https://doi.org/10.1016/j.neuri.2022.100060

[6] T. Hossain, F. S. Shishir, M. Ashraf, M. A. Al Nasim and F. Muhammad Shah, "Brain Tumor Detection Using Convolutional Neural Network,"2019 1st International Conference on Advances in Science, Engineering and Robotics

Technology (ICASERT), Dhaka, Bangladesh, 2019, pp. 1-6,
doi: 10.1109/ICASERT.2019.8934561.