Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Uso de inteligencia artificial para reforzamiento del diagnóstico de TDAH

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física Aplicada

por

Fernando Sánchez Ortega

Asesorado por

Dr. Javier Miguel Hernández López
Dr.José Gerardo Suárez García

Puebla Pue.
Noviembre de 2024

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Uso de inteligencia artificial para reforzamiento del diagnóstico de TDAH

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física Aplicada

por

Fernando Sánchez Ortega

Asesorado por

Dr. Javier Miguel Hernández López
Dr.José Gerardo Suárez García

Puebla Pue.
Noviembre de 2024

Título: Uso de inteligencia artificial para reforzamiento del diagnóstico de TDAH
Estudiante: Fernando Sánchez Ortega

COMITÉ

Dra. Ma. del Rosario Pastrana Sánchez
Presidente

Dr Héctor Novales Sánchez
Secretario

M.C. Margarita Amaro Aranda
Vocal

Vocal

Dr. Javier Miguel Hernández López
Dr.José Gerardo Suárez García
Asesor

Índice general

Capítulo 1

Agradecimientos

Agradezco a mi padre y madre por darme lo necesario para poder llegar hasta donde estoy.

Capítulo 2

Introducción

2.1. ¿Qué problema se quiere resolver en esta tesis?

En esta tesis se quiere resolver el problema de que la enfermedad de TDAH no puede ser diagnosticada ante una resonancia magnética en principio, sino que su diagnóstico depende de cuestionarios de caracteres psicológicos y lo experto que sea el psicólogo/neurólogo en su diagnóstico. ¿Pero que tal si la respuesta está al alcance de otros métodos complementarios, basados en mediciones de variables físicas de la persona en cuestión? En las resonancias magnéticas pero que no podemos verlas con nuestro ojo desnudo, i.e. de las imágenes directamente. El auge de la inteligencia artificial nos da herramientas que permiten que la información médica se pueda clasificar y mejorar e incluso en un futuro nos permita mejorar las posibilidades de detección de esta y otras condiciones médicas, basados en herramientas físicas y computacionales.

2.2. ¿Qué es la inteligencia artificial?

Desde la década de los 50 se intentó hacer de la inteligencia artificial (IA) un área fuerte, pero era algo muy adelantado a su época, ahora en su segundo (tercer, cuarto) "round" viene a demostrar sus capacidades con la llegada de inteligencias artificiales que tratan el lenguaje natural con habilidades y posibilidades no pensadas antes. Herramientas tales como "ChatGPT", Llama3, el reconocimiento de imágenes, la identificación de sentimientos en redes sociales, reconocimiento de audio, regresiones, algoritmos para asistir al dermatopatólogo en el diagnóstico, asistencia virtual y un sin fin de aplicaciones a las cuales la IA abre paso. [1][2]

La IA es una nueva tecnología la cual pretende resolver problemas que son intelectualmente difíciles para los humanos pero relativamente sencillos para las computadoras. Los retos al usar este tipo de tecnología son los problemas en los cuales no se pueden definir un conjunto de reglas fijas, pongamos un ejemplo de los casos fáciles para una IA como lo sería el clásico juego de gato o convertir grados Celsius a Fahrenheit, y luego unos casos que serían un reto para esta tecnología como sería identificar objetos dentro de una imagen o generar una respuesta en un lenguaje el cual constantemente cambian las reglas gramaticales.[3][4][5]

2.3. ¿Qué es el *machine learning* (aprendizaje automático)?

"Es un subdominio de la IA que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente a partir de la experiencia, sin ser explícitamente programados para ello. Se basa en la hipótesis de crear un modelo, y tratar de mejorarlo ajustando más datos en el modelo a lo largo del tiempo."[1]

Concretamente, los elementos que forman parte de cualquier algoritmo de *machine learning* son: experiencia pasada (datos de entrenamiento), la función que irá ajustando parámetros e irá aprendiendo de la experiencia pasada (función hipótesis) y el modelo finalmente el cual se le pasarán los datos para poder hacer una predicción. Esto se debería listar de la siguiente forma al momento de entrenar un modelo:

1. Conjunto de datos
2. Algoritmo de aprendizaje
3. h (función hipótesis)
4. Modelo

Para poder actuar con datos nuevos de forma que:

1. x (dato nuevo)
2. h (función hipótesis)
3. y (predicción)

2.4. *End-to-End learning*

El rendimiento de los algoritmos de *machine learning* depende en gran medida de la representación de los datos que se les proporciona; muchas tareas pueden resolverse diseñando un conjunto adecuado de características a extraer para esa tarea y proporcionando un algoritmo de *machine learning*, pero para los casos en los que no, surge una técnica llamada *Representation learning* (aprendizaje de representación).

2.5. ¿Qué es el *deep learning*?

El *deep learning* (aprendizaje profundo) va a consistir en una parte de las técnicas de *machine learning* que resuelve el problema de las técnicas de *Representation learning*, introduciendo representaciones que se expresan en términos de otras representaciones más simples, se fundamenta en la construcción de conceptos complejos a partir de conceptos más simples. Es fundamental, para que los algoritmos de *deep learning* funcionen correctamente, que estos tengan una gran cantidad de datos. Esto es una limitante para algunos casos de estudio.

2.6. Aprendizaje supervisado y no supervisado

Para el aprendizaje supervisado, el *learner* (la parte que aprende) recibe un conjunto de ejemplos etiquetados como datos de entrenamiento y hace predicciones para todos los puntos por conocer. Este es el escenario más común asociado con problemas de clasificación, regresión y *ranking* (ordenar por importancia) [6]. El problema general de aprendizaje supervisado de clasificación consiste en utilizar información etiquetada para definir una función de predicción de clasificación precisa para todos los puntos [7]. La función resultante es utilizada posteriormente para predecir valores a partir de ejemplos de datos no etiquetados.

Para el aprendizaje no supervisado el *learner* recibe exclusivamente datos de entrenamiento sin etiquetar y hace predicciones para todos los puntos por conocer. Dado que, en general, no hay tantos ejemplos etiquetados disponibles en ese entorno, puede resultar difícil evaluar cuantitativamente el desempeño de un *learner*. La agrupación y la reducción de dimensionalidad son ejemplos de problemas de aprendizaje no supervisados[6]. Este tipo de problemas aparece comúnmente en el análisis de grandes volúmenes de datos en diversas áreas de la ciencia y, en particular, de física.

Capítulo 3

Redes neuronales convolucionales

3.1. ¿Qué es una red neuronal?

Es un tipo de algoritmo de *machine learning* inspirado en un modelo básico de las neuronas biológicas. Este tipo de algoritmo se introduce por primera vez en 1943 por el neurosicólogo Warren McCullop y el matemático Walter Pitts [8]. Al momento de escribir esta tesis han pasado 80 años, y algunos se preguntarán por qué ha pasado tanto tiempo de la creación del modelo pero apenas unos años atrás empezamos a escuchar sobre redes neuronales. Esta disciplina pasó por dos periodos de inactividad en los cuales quedó un poco olvidada. El primero fue por el "Informe lighthill" el cual dio un pronóstico muy pesimista para muchos aspectos centrales de la investigación en este campo, afirmando que "en ninguna parte del campo los descubrimientos realizados hasta ese momento produjeron el impacto principal que se prometió entonces". Esto hizo que de 1974-1980 esta tecnología fuera olvidada. El segundo impasse fue tras la caída de los sistemas expertos y el colapso del mercado de las máquinas "Lisp", muy populares en los 80. A medida que las expectativas sobre los sistemas expertos no se cumplieron, la inversión en IA disminuyó drásticamente, y la comunidad de investigación se enfrentó nuevamente a un recorte masivo en la financiación y no fue hasta que en 2012, Geoffrey Hinton ganó el reto "ImageNet" con una red convolucional para clasificar imágenes que el área revivió [9] [10].

La razón por la cual hoy en día está en un auge la IA es porque las empresas ahora tienen muchos datos, los cuales ya cuentan como un activo que se puede vender, por lo que la IA, la ciencia de datos y el *Big data* ahora son indispensables, dado el florecimiento del área de servicios en línea por internet. Además, el poder de computo comercial del que ahora se dispone al alcance de la población es de mayor rendimiento del que tenían acceso las personas hace 80 años. Un claro ejemplo sería los GPU (hardware que permiten procesar grandes cantidades de datos de manera más rápida y eficiente que los CPUs tradicionales utilizados en la mayoría de las computadoras) y más específicamente para inteligencia artificial las TPU (hardware específicamente para realizar operaciones con tensores). Los algoritmos de redes neuronales también han mejorado, en particular las nuevas metodologías que implementan el *back propagation* (retropropagación). Esta técnica permite el acceso a la resolución a problemas de forma no lineal, a diferencia de las técnicas de años atrás en donde todo el procesamiento era lineal y por ende sólo podía resolver problemas lineales.

3.2. La neurona MP

La neurona de McCullop y Pitts es la primera neurona artificial de la historia. Se caracteriza porque recibe uno o más valores binarios y retorna otro valor binario, activa su salida cuando más de un número determinado de valores de entrada se encuentran activos. Debe establecerse

manualmente el número de valores de entrada que deben estar activos a este valor se le denomina *threshold* (θ). En la Figura 3.1, se muestra una neurona MP donde **Z** es la función de agregación,

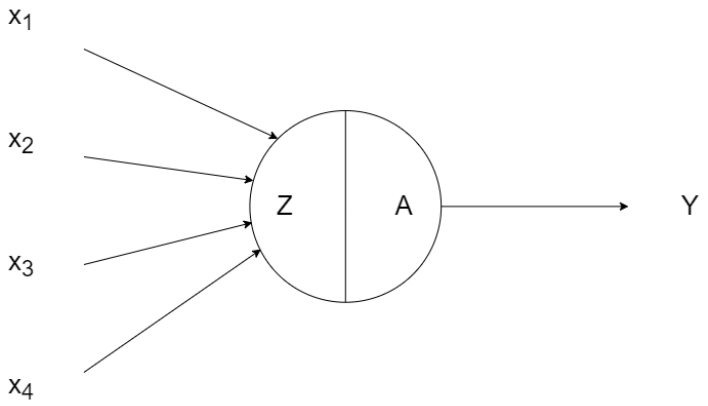


Figura 3.1: Neurona MP.

A es la función de activación y **Y** es la salida binaria de la neurona, $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$,$\mathbf{x}_4$, son entradas binarias, por lo que podemos escribir la neurona MP como

$$Z(x) = \sum_{\mathrm{i}=1}^{\mathrm{n}} x_{\mathrm{i}} \tag{3.1}$$

con lo que la función de activación definida es:

$$A(Z(x)) = \begin{cases} 1 & \text{si Z(x)} \ \geq \theta, \\ 0 & \text{si Z(x)} < \theta \end{cases}$$

Tiene dos tipos de entradas: inhibidoras y excitadoras. Los inhibidores tendrían más valor para la suma por lo que superaría el *threshold* (corte) cuando esa entrada inhibidora esté en 1. Si está en 0 dependerá únicamente de las entradas excitadoras. Después de esto la neurona ya es útil para representar algunas funciones lógicas como lo son "AND" y "OR" asignando un *threshold* en 2 o en 1 respectivamente. Sin embargo, tiene sus limitaciones tales como que solo acepta valores binarios y la salida será también un valor binario. El *threshold* debe ser seleccionado de manera manual. Todas las entradas son iguales, no se puede asignar un mayor peso a alguna de ellas, no son capaces de resolver problemas que no sean linealmente separables, por ejemplo la operación "XOR".

3.3. El perceptrón

Basado en el modelo de neuronas biológicas propuesto por Frank Rosenblatt en 1958, en 1969 fue refinado y analizado a detalle por Marvin Minsky y Seymour Papert [11]. Los autores mejoran el planteamiento de la neurona MP con el concepto de peso (un valor numérico en las entradas y planteando un mecanismo para ajustarlos, lo que hoy se conoce como "aprendizaje"). Se caracteriza por ser un algoritmo que tiene varias neuronas TLU *Threshold Logic Unit* la cual computa una suma parametrizada de las entradas:

$$Z(x) = \sum_{\mathrm{i}=0}^{\mathrm{n}} x_{\mathrm{i}} * \omega_{\mathrm{i}} \tag{3.2}$$

donde los términos ω son los pesos de las respectivas entradas y ahora se elimina el *threshold* y va a existir una entrada $\mathbf{x}_0 = \mathbf{1}$ con un peso θ. La función de activación ahora es:

$$A(Z(x)) = \begin{cases} 1 & \text{si Z(x)} \ \geq 0, \\ 0 & \text{si Z(x)} < 0 \end{cases}$$

este peso adicional que se añadió comúnmente se llama término "bias" también llamado término de parcialidad donde el término **b** sería el término "bias".

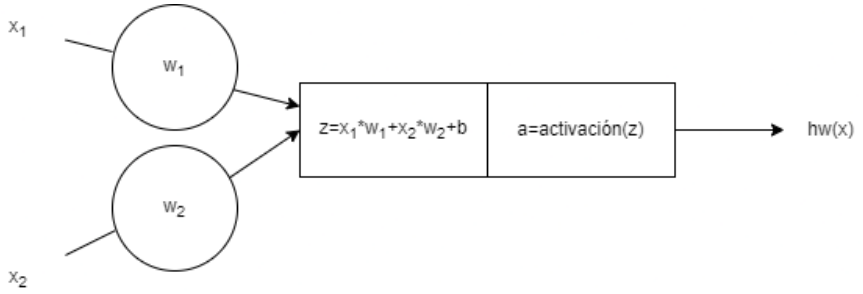


Figura 3.2: Neurona TLU.

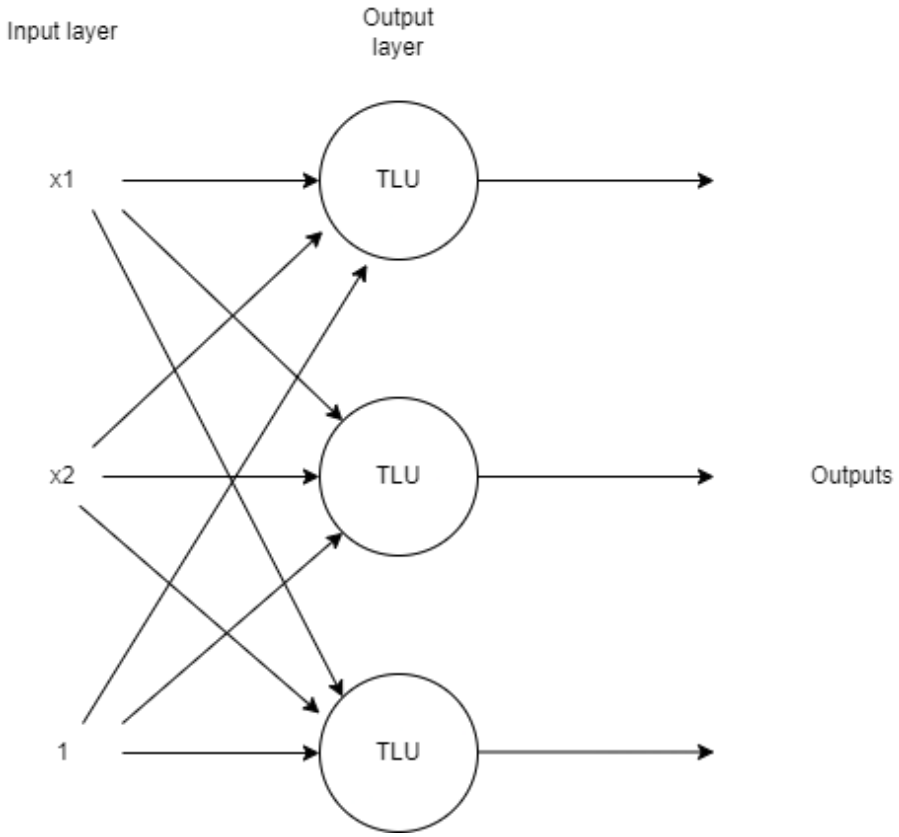


Figura 3.3: Perceptrón de una única capa.

El perceptrón corresponde a una arquitectura por una única capa de neuronas TLUs donde a los pesos los denominamos parámetros del modelo, la capa de las entradas se denomina *input layer*

y la capa por donde estarán las salidas se denominará *output layer*, este tipo de algoritmo permite la clasificación de instancias en clases binarias de manera simultánea.

3.4. ¿Qué son las funciones de activación?

Tenemos varias funciones de activación algunas de las más comunes son: "heaviside step function":

$$heaviside(z) = \begin{cases} 1 & \text{si z } \geq 0, \\ 0 & \text{si z} < 0 \end{cases}$$

"Sign function":

$$Sign(z) = \begin{cases} 1 & \text{si z } > 0, \\ 0 & \text{si z } = 0, \\ -1 & \text{si z} < 0 \end{cases}$$

El perceptrón acepta entradas no binarias y sus salidas pueden variar de ser binarias a ser multiclase. Para que los resultados de una TLU sean binarios, o sea 0 o 1, se asigna la función *heaviside*. Pero en el caso de la función *sign* puede asignar valores negativos. Las funciones de activación pueden variar en las diferentes TLU de un perceptrón.

3.4.1. Función sigmoide

La función sigmoide retornará un valor continuo en vez de un valor binario, la forma de interpretar este valor binario resultado de la función sigmoide es como si fuera una probabilidad de pertenecer a la clase positiva. Si el resultado de la función sigmoide es 0.6 significará que tiene un 60 % de probabilidad de pertenecer a la clase positiva. [12]

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

3.4.2. Función ReLu

La función ReLu (*Rectified Linear Unit*) es una función de activación comúnmente utilizada en redes neuronales, incluidas las redes convolucionales (CNNs). La función ReLU se define como $f(x) = max(0, x)$, lo que significa que devuelve cero si el valor de entrada es negativo y devuelve el mismo valor si es positivo.

En el contexto de una capa Conv3D en "Keras", la función ReLU se aplica elemento por elemento a los datos de salida de la capa convolucional. Esto ayuda a introducir no linealidad en la red y a superar el problema de la desaparición del gradiente, que puede ocurrir con funciones de activación saturadas como la función sigmoide. Se proporcionarán más detalles posteriormente.

3.5. Perceptrón multicapa

El perceptrón multicapa(MLP) o *Deep Forward network*(DFN) o *Deep Neural Network*(DNN) es el modelo más popular dentro del "deep learning". Se denomina *feedforward* porque la información fluye desde las entradas hasta la salida sin conexiones de *feedback* (es decir, la información siempre fluye en la misma dirección). Se caracterizan por la composición de numerosas funciones, teniendo en cuenta el perceptrón ahora podemos considerar una capa más llamada *hidden layer* (capa oculta) de la cual sus salidas serán las entradas de la siguiente capa, pueden ser la *output layer* (capa de salida) o bien otra *hidden layer*.

3.6. Componentes de una red neuronal artificial

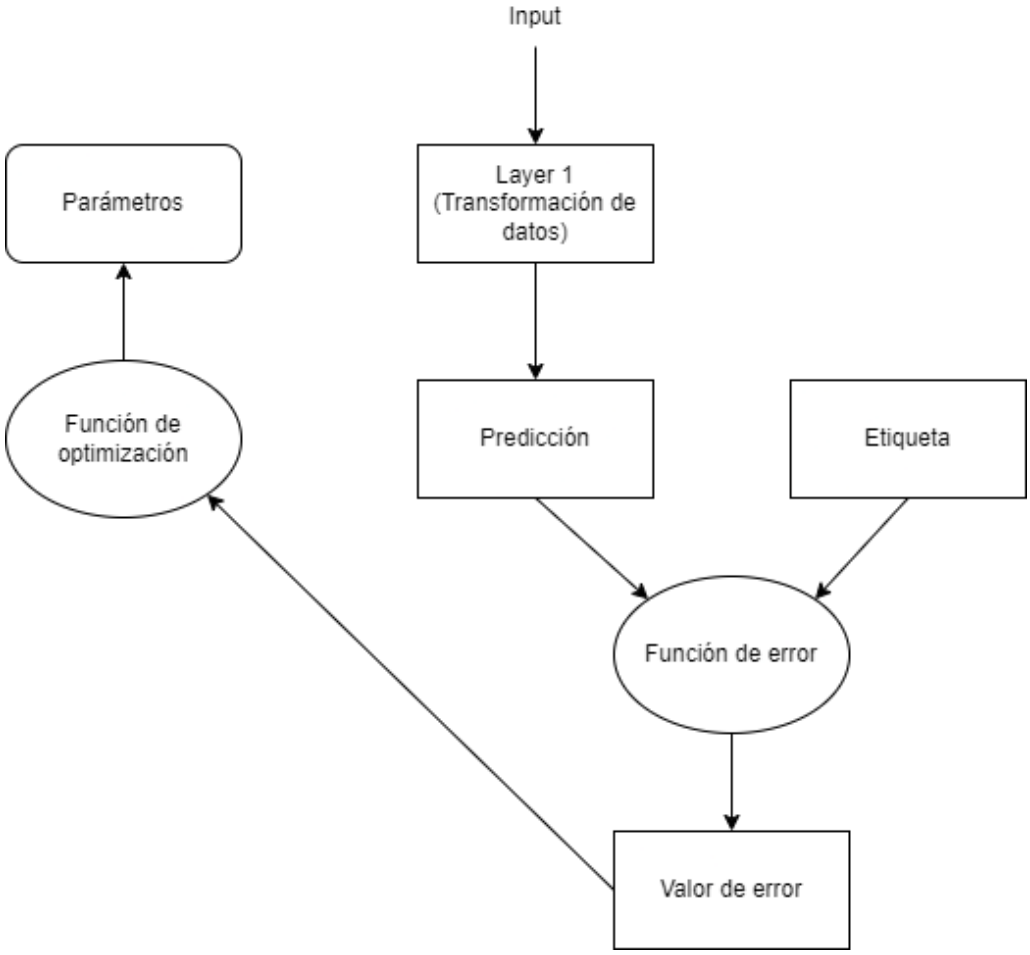


Figura 3.4: Componentes de una red neuronal artificial.

Para el entrenamiento de una red neuronal artificial el flujo será como se muestra en la figura 3.4: *input* es la capa de datos mientras que *layer* 1 es una capa de TLUs la cual tendrá una predicción que sería la salida de ese respectivo perceptrón la cual va a ser comparada con la etiqueta mediante la función de error o también llamada función de pérdida. La función de error tendrá como resultado el valor del error el cual pasará a la función de optimización, esto proporcionará como resultado los nuevos parámetros, cuando son modelos con más capas se actualizan todos los parámetros. Todas las capas excepto la *output layer* incluyen la neurona "bias". Todas las capas se encuentran conectadas con las siguientes capas, cuando una red neuronal artificial tiene mas de dos capas se denomina red neuronal profunda. Hasta 1986 D. E. Rumelhart presenta un algoritmo que revoluciona la manera de entrenar el perceptrón multicapa, el algoritmo "backpropagation" [13] lo cual no fue nada más y nada menos que cambiar la función de activación *heaviside* por la función sigmoide la cual se sigue utilizando al día de hoy, existen otras como funciones de activación como lo son tanh(z) y ReLu(z).

3.6.1. *Forward propagation*

El proceso conocido como *forward propagation* del cual depende el tipo de arquitecturas *feed-forward networks* se fundamenta en que la información fluye desde las entradas hasta las salidas sin ninguna conexión de *feedback*, es decir, el *output* de una neurona siempre se va a corresponder con el input de una neurona posterior y nunca se va a corresponder con el input de una neurona anterior. El *forward propagation* se utiliza al hacer el entrenamiento y al hacer una predicción "ya con los parámetros finales".

3.6.2. Función de coste

La función de error toma los pesos aleatorios con los que se inicializa la red neuronal y toma el valor del dato etiquetado y lo compara contra el valor de la predicción hecha para poder ajustar los pesos.

Al estar en la función sigmoide su función de coste sería

$$\mathcal{L}(y_{\text{predicción}}, y_{\text{etiqueta}}) = -\left(y_{\text{etiqueta}} \log(y_{\text{predicción}}) + (1 - y_{\text{etiqueta}}) \log(1 - y_{\text{predicción}})\right) \tag{3.4}$$

por lo que cuando la etiqueta es 0

$$\mathcal{L}(y_{\text{predicción}}, 0) = log(1 - y_{\text{predicción}})) \tag{3.5}$$

y cuando es 1

$$\mathcal{L}(y_{\text{predicción}}, 1) = -(log(y_{\text{predicción}})) \tag{3.6}$$

la función global de error sería

$$\mathcal{J}(w, B) \frac{1}{m} \sum_{\text{i}=1}^{\text{m}} \mathcal{L}(y_{\text{predicción}^{\text{i}}}, y_{\text{etiqueta}^{\text{i}}}) \tag{3.7}$$

por lo que

$$\mathcal{J}(w, B) \frac{1}{m} \sum_{\text{i}=1}^{\text{m}} -(y_{\text{etiqueta}^{\text{i}}} log(y_{\text{predicción}^{\text{i}}}) + (1 - y_{\text{etiqueta}^{\text{i}}}) log(1 - y_{\text{predicción}^{\text{i}}})) \tag{3.8}$$

donde m es el número total de datos y $\mathcal{J}(\mathbf{w}, \mathbf{B})$ es denominado *Cross Entropy Loss*.

3.6.3. *Gradient descent* (Gradiente descendente)

La función de optimización tomará la función de coste *Cross Entropy Loss* inicializa los parámetros ω aleatoriamente de tal modo que los parámetros se actualizarán de la siguiente forma:

$$\omega_{\text{n}} = \omega_{\text{n}-1} - \eta \frac{d\mathcal{J}}{d\omega_{\text{n}-1}} \tag{3.9}$$

donde η es el llamado *learning rate* (grado o razón de aprendizaje) con lo que se puede notar que la derivada de $\mathcal{J}$ respecto de ω va a ser 0 cuando ω ya no varíe con lo cual ω sería un mínimo global

$$\omega_{\text{n}} = \omega_{\text{n}-1} - \eta(x(y_{\text{predicción}} - y_{\text{etiqueta}})) \tag{3.10}$$

Suponiendo una red neuronal con una única TLU con dos entradas tendríamos

$$\mathcal{J} = 0, d\omega_1 = 0, d\omega_2 = 0, db = 0 \tag{3.11}$$

inicializando las sumas en 0

$$\mathcal{J}_{\text{nuevo}} = \mathcal{J}_{\text{anterior}} + \mathcal{L}(y_{\text{predicción}^{\text{i}}}, y_{\text{etiqueta}^{\text{i}}}) \tag{3.12}$$

$$d\omega_1 = d\omega_1 + \frac{\mathcal{J}}{d\omega_1} = d\omega_1 + x_{\mathrm{i}}(y_{\text{predicción}^{\mathrm{i}}} - y_{\text{etiqueta}^{\mathrm{i}}}) \tag{3.13}$$

$$d\omega_2 = d\omega_2 + \frac{\mathcal{J}}{d\omega_2} = d\omega_2 + x_{\mathrm{i}}(y_{\text{predicción}^{\mathrm{i}}} - y_{\text{etiqueta}^{\mathrm{i}}}) \tag{3.14}$$

$$db = db + \frac{\mathcal{J}}{db} = db + (y_{\text{predicción}^{\mathrm{i}}} - y_{\text{etiqueta}^{\mathrm{i}}}) \tag{3.15}$$

iterando en el **i**-ésimo dato ajustando los parámetros del modelo tanto el término bias. Por lo que al final tendríamos

$$\mathcal{J} = \frac{\mathcal{J}}{m}, d\omega_1 = \frac{d\omega_1}{m}, d\omega_2 = \frac{d\omega_2}{m}, db = \frac{db}{m} \tag{3.16}$$

entonces

$$\omega_{1_{\mathrm{n}}} = \omega_{1_{\mathrm{n}-1}} - \eta d\omega_{1_{\mathrm{n}-1}}, \omega_{2_{\mathrm{n}}} = \omega_{2_{\mathrm{n}-1}} - \eta d\omega_{2_{\mathrm{n}-1}}, b = b_{\mathrm{n}-1} - \eta db_{\mathrm{n}-1} \tag{3.17}$$

de este modo tendríamos la primera actualización de los parámetros.

3.6.4. Grafo computacional

Un grafo computacional es un grafo dirigido cuyos nodos se van a corresponder con una operación matemática y los arcos con valores de entrada/salida. Tomando en cuenta una red neuronal con una sola TLU supongamos dos entradas, x_1 y x_2, tendríamos que los pesos y el término bias serían los siguientes por "gradient decent"

$$\omega_{1_{\mathrm{n}}} = \omega_{1_{\mathrm{n}-1}} - \eta\frac{\partial\mathcal{L}}{\partial\omega_{1_{\mathrm{n}-1}}}; \omega_{2_{\mathrm{n}}} = \omega_{2_{\mathrm{n}-1}} - \eta\frac{\partial\mathcal{L}}{\partial\omega_{2_{\mathrm{n}-1}}}; b_{\mathrm{n}} = b_{\mathrm{n}-1} - \eta\frac{\partial\mathcal{L}}{\partial b} \tag{3.18}$$

siguiendo la regla de la cadena tenemos

$$\frac{\partial\mathcal{L}}{\partial\omega_1} = \frac{\partial\mathcal{L}}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial\omega_1} \tag{3.19}$$

esto es

$$\frac{\partial\mathcal{L}}{\partial a} = \frac{\partial}{\partial a}\left[-\left(y_{\text{etiqueta}}\log(y_{\text{predicción}}) + (1 - y_{\text{etiqueta}})\log(1 - y_{\text{predicción}})\right)\right] \tag{3.20}$$

donde $\mathcal{L}$ es nuestra función de error

$$\frac{\partial\mathcal{L}}{\partial a} = \frac{1 - y_{\text{etiqueta}}}{1 - y_{\text{predicción}}} - \frac{y_{\text{etiqueta}}}{y_{\text{predicción}}} \tag{3.21}$$

después tenemos

$$\frac{\partial y_{\text{predicción}}}{\partial z} = \frac{\partial}{\partial z}\left[\frac{1}{1 + e^{-z}}\right] = y_{\text{predicción}}\left(1 - y_{\text{predicción}}\right) \tag{3.22}$$

donde $y_{\text{predicción}}$ es la función de activación sigmoide después tenemos que:

$$\frac{\partial z}{\partial\omega_1} = \frac{\partial}{\partial\omega_1}\left[\omega_1 x_1 + \omega_2 x_2 + b\right] = x_1 \tag{3.23}$$

para finalmente poner la ecuación 3.19 de la siguiente forma:

$$\frac{\partial\mathcal{L}}{\partial\omega_1} = \left(\frac{1 - y_{\text{etiqueta}}}{1 - y_{\text{predicción}}} - \frac{y_{\text{etiqueta}}}{y_{\text{predicción}}}\right)\left(y_{\text{predicción}}\left(1 - y_{\text{predicción}}\right)\right)\left(x_1\right) \tag{3.24}$$

esto es

$$\frac{\partial\mathcal{L}}{\partial\omega_1} = \left(y_{\text{predicción}} - y_{\text{etiqueta}}\right)x_1 \tag{3.25}$$

por lo que la actualización del parámetro ω_1 quedaría como

$$\omega_{1_{\mathrm{n}}} = \omega_{1_{\mathrm{n}-1}} - \eta\left(y_{\text{predicción}} - y_{\text{etiqueta}}\right)x_1 \tag{3.26}$$

donde el miembro izquierdo de las ecuaciones 3.18 3.26 son los parámetros actualizados.

3.6.5. *Backward propagation*

Al proceso de calcular las derivadas de la función de error respecto de los pesos para poder actualizar los valores de los pesos se le llama *Backward propagation.* [14]

Primero echemos un vistazo a la estructura básica de Redes Neuronales Convolucionales(CNN)

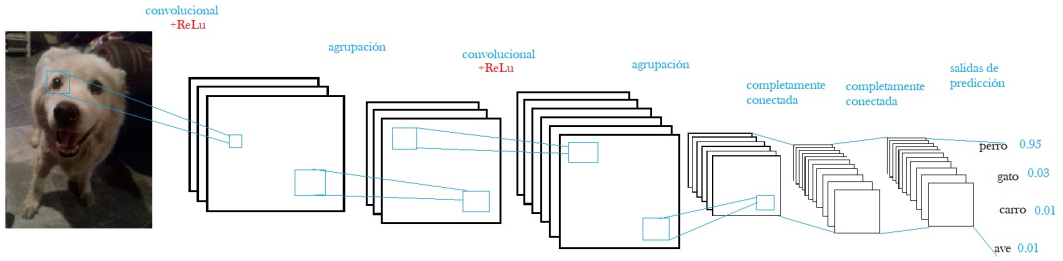


Figura 3.5: Estructura de una CNN.

La imagen anterior 3.5 es un modelo CNN para el reconocimiento de patrones. Se puede ver que la imagen es nuestra capa de entrada. La computadora entiende que ingresa varias matrices, que es básicamente lo mismo que DNN.

La siguiente capa de convolución, que es exclusiva de CNN, la función de activación para la capa de convolución, utiliza ReLu función de la cual hablamos en el capítulo anterior: $\mathbf{ReLu}(\mathbf{x}) = \mathbf{max}(\mathbf{0}, \mathbf{x})$, después de la capa convolucional esta la capa de agrupación también llamada capa de "pool" o agrupamiento, igual exclusiva de CNN esta capa de "pooling" no cuenta con función de activación.

La combinación de $\mathbf{capa\_convolucional} + \mathbf{capa\_de\_agrupación}$ puede aparecer muchas veces en la capa oculta, dos veces en la figura anterior, de hecho este número se basa en las necesidades del modelo. Por supuesto, también podemos usar de manera flexible $\mathbf{capa\_convolucional} + \mathbf{capa\_convolucional}$, o $\mathbf{capa\_convolucional} + \mathbf{capa\_convolucional} + \mathbf{capa\_de\_agrupación}$, que no tiene límites al construir un modelo, pero la CNN más común es una combinación de varias $\mathbf{capas\_convolucionales} + \mathbf{capas\_de\_agrupación}$, como en la estructura CNN en la imagen anterior.

Después de varias $\mathbf{capas\_convolucionales} + \mathbf{capas\_de\_agrupación}$ hay una capa completamente conectada, esta capa completamente conectada es en realidad la estructura del capítulo anterior DNN excepto que en el caso de la imagen nuestra DNN usa la función de activación softmax la cual puede clasificar a varias clases para la clasificación de reconocimiento de imágenes. [15]

3.7. ¿Qué es la convolución?

La expresión de la convolución es la siguiente:

$$S(t) = \int x(t-a)w(a)da \tag{3.27}$$

lo cual en su forma discreta es

$$s(t) = \sum_a x(t-a)w(a) \tag{3.28}$$

si esta formula se expresa como una matriz, puede ser:

$$s(t) = (X * W)(t) \tag{3.29}$$

donde el * representa convolución. Si es una convolución bidimensional, la expresión es:

$$s(i,j) = (X * W)(i,j) = \sum_{\mathrm{m}} \sum_{\mathrm{n}} x(i-m, j-n)w(m,n) \tag{3.30}$$

En CNN, aunque también hablamos de convolución, nuestra fórmula de convolución es ligeramente diferente de la definición en matemáticas estrictas. Por ejemplo, para convolución bidimensional, se define como:

$$s(i,j) = (X * W)(i,j) = \sum_{\mathrm{m}} \sum_{\mathrm{n}} x(i+m, j+n)w(m,n) \tag{3.31}$$

Aunque esta fórmula no es una convolución en sentido estricto desde un punto de vista matemático, todos los expertos la llaman así, por lo que nosotros también hacemos lo mismo. La convolución de CNN mencionada más adelante se refiere a la ecuación 3.31.

Entre ellos, llamamos a W nuestro núcleo de convolución y X es nuestra entrada. Si X es una matriz de entrada bidimensional y W también es una matriz bidimensional. Pero si X es un tensor multidimensional, entonces W también es un tensor multidimensional.

3.8. Capas convolucionales

Suponiendo que tenemos una imagen de 3 píxeles de alto y 4 píxeles de ancho y nuestro núcleo de convolución es de 2×2 píxeles, por lo que un núcleo de convolución (kernel) aplicado a nuestra matriz imagen tendremos una matriz de 3×2, es decir, nuestra nueva imagen convolucionada teniendo 3 píxeles de ancho y 2 píxeles de alto.

Sea A Nuestra imagen de 3×4:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

Y sea B nuestro núcleo de convolución 2×2:

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$C = A * B$, se calcula como:

$$C = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{12} + a_{21}b_{21} + a_{22}b_{22}) & (a_{12}b_{11} + a_{13}b_{12} + a_{22}b_{21} + a_{23}b_{22}) & (a_{13}b_{11} + a_{14}b_{12} + a_{23}b_{21} + a_{24}b_{22}) \\ (a_{21}b_{11} + a_{22}b_{12} + a_{31}b_{21} + a_{32}b_{22}) & (a_{22}b_{11} + a_{23}b_{12} + a_{32}b_{21} + a_{33}b_{22}) & (a_{23}b_{11} + a_{24}b_{12} + a_{33}b_{21} + a_{34}b_{22}) \end{bmatrix}$$

Note que el * denota convolución y no producto punto, a este proceso de pasar un núcleo a través de una imagen, multiplicar los valores, sumar el resultado y obtener una nueva imagen es llamado convolución. En nuestro caso este proceso se repetirá 3 veces si nuestra imagen es de color separándose en colores RGB y una sola vez si nuestra imagen en blanco y negro (1 solo canal). Una imagen en color como tensor se podría escribir como **ancho** $\times$ **alto** $\times$ **canales** por lo que una imagen puede llegar a escribirse como:

$$s(i,j) = (X * W)(i,j) + b = \sum_{\mathrm{k}=1}^{\mathrm{n\_in}} (X_{\mathrm{k}} * W_{\mathrm{k}})(i,j) + b \tag{3.32}$$

donde **n_in** es el número de matrices de entrada o la dimensión de la última dimensión del tensor al final de la capa de convolución se suele agregar la función de activación relu para cambiar todos los números negativos a 0.

3.9. Capas de agrupación

Existen dos tipos de capas de agrupación: agrupación máxima y agrupación promedio.

3.9.1. Agrupación máxima

Suponga una matriz de 4×4, si el paso es de 2 simplemente tomará el máximo dentro de la matriz de 4×4 una submatriz de 2×2 de la cual obtendrá los 4 números dentro de esta y tomará el máximo para generar otra matriz de una dimensionalidad más pequeña.

En este ejemplo suponga

$$K = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & q & o \end{bmatrix}$$

suponga que a la matriz K se le aplica la capa de agrupación "max" de 2×2, si el máximo de a, b, e y f es a; c, d, g y h es d; i, j, m y n es i;k, l, q y o es k, tendremos la siguiente matriz resultante

$$k_{\max} = \begin{bmatrix} a & d \\ i & k \end{bmatrix}$$

3.9.2. Agrupación promedio

Suponga una matriz de 4×4, si el paso es de 2 simplemente tomará el promedio dentro de la matriz de 4×4 una submatriz de 2×2 de la cual obtendrá los 4 números dentro de esta y tomará el promedio para generar otra matriz de una dimensionalidad mas pequeña.

En este ejemplo suponga

$$K = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & q & o \end{bmatrix}$$

suponga que a la matriz K se le aplica la capa de agrupación "avg" de 2×2, si el promedio de a,b,e y f es q; c,d,g y h es ω; i,j,m y n es p;k,l,q y o es η, tendremos la siguiente matriz resultante

$$k_{\text{avg}} = \begin{bmatrix} q & \omega \\ p & \eta \end{bmatrix}$$

a todo este proceso de convolucionar y agrupar se le llama *Forward propagation* dentro de las redes CNN.

3.10. *Backward propagation* en CNN

3.10.1. Separación de *Backward propagation* en CNN

Para aplicar el algoritmo de *Backward propagation* de DNN a CNN, hay varios problemas que deben resolverse:

1. La capa de agrupación no tiene función de activación. Este problema es más fácil de resolver. Podemos hacer que la función de activación de la capa de agrupación sea la siguiente: $\sigma(z) = z$, es decir, después de la activación, se convierte en sí mismo. De esta forma, la derivada de la función de activación de la capa de agrupación es 1.

2. La capa de agrupación comprime la entrada durante el *Forward propagation*, por lo que ahora necesitamos derivarla hacia adelante y hacia atrás, δ^{l-1}, este método de derivación es completamente diferente de DNN.

3. La capa convolucional es la salida de la capa actual obtenida mediante convolución tensorial o la suma de varias convoluciones matriciales. Esto es muy diferente de DNN. La capa completamente conectada de DNN realiza directamente la multiplicación de matrices para obtener la salida de la capa actual. De esta manera, cuando la capa convolucional se propaga hacia atrás, la capa anterior δ^{l-1} en el método de cálculo recursivo es definitivamente diferente.

4. Para la capa convolucional, ya que en W la operación utilizada es la convolución, luego de que δ^l derive los valores de todos los núcleos de convolución de esta capa. W, b el camino también es diferente.

   Como se puede ver en lo anterior, el problema 1 es relativamente fácil de resolver, pero los problemas 2, 3 y 4 requieren mucho pensamiento, y los problemas 2, 3 y 4 también son la clave para resolver el algoritmo de *Backward propagation* de CNN. Además, todos deberían prestar atención al hecho de que en DNN a_l, z_l, son solo un vector, y en nuestra CNN a_l, z_l, todos son un tensor. Este tensor es tridimensional, es decir, consta de varias submatrices de entrada.

   A continuación, estudiaremos el algoritmo de *Backward propagation* de CNN paso a paso para las preguntas 2, 3 y 4.

   Durante el proceso de investigación, cabe señalar que dado que la capa de convolución puede tener múltiples núcleos de convolución, los métodos de procesamiento de cada núcleo de convolución son exactamente iguales e independientes. Para simplificar la complejidad de la fórmula del algoritmo, mencionamos la convolución a continuación. Kernel es uno de varios núcleos de convolución en la capa convolucional.

3.10.2. Capa de agrupación conocida, restaura capa oculta anterior

Primero resolvemos el problema 2 anterior. Si se conoce la capa de agrupación δ^l, deriva la capa oculta anterior δ^{l-1}, en el algoritmo de *Forward propagation*, en la capa de agrupación, generalmente usamos "MAX" o "AVG" para agrupar la entrada, y se conoce el tamaño del área de agrupación. Ahora le damos la vuelta y partimos del error reducido.

δ^l restaura el error correspondiente al área más grande anterior.

Durante el *Backward propagation*, primero ponemos δ^l el tamaño de la matriz de todas las submatrices, se restaura al tamaño antes de la agrupación y luego, si es "MAX", entonces δ^l. Los valores de cada área local de agrupación de todas las submatrices se colocan en la posición donde el algoritmo de "fordward propagation" obtuvo el valor máximo. Si es Promedio, ponga δ^l, los valores de cada área local agrupada de todas las submatrices se promedian y se colocan en la posición de la submatriz restaurada. Este proceso generalmente se denomina muestreo ascendente.

Se puede expresar fácilmente con un ejemplo: supongamos que el tamaño del área de nuestra capa de agrupamiento es 2×2 donde δ^l la k-ésima submatriz es:

$$\delta_k^l = \begin{pmatrix} 2 & 8 \\ 4 & 6 \end{pmatrix} \tag{3.33}$$

Dado que el área de la agrupación es de 2×2, primero hablamos de $\delta_{\mathrm{k}}^{\mathrm{l}}$Para restaurar, se convierte en:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 8 & 0 \\ 0 & 4 & 6 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.34}$$

Si es "MAX", suponiendo que las posiciones de valor máximo que registramos previamente durante en el *Forward propagation* son superior izquierda, inferior derecha, superior derecha e inferior izquierda, entonces la matriz convertida es:

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 6 & 0 \end{pmatrix} \tag{3.35}$$

Si es Promedio, la matriz convertida es:

$$\begin{pmatrix} 2 & 2 & 8 & 8 \\ 2 & 2 & 8 & 8 \\ 4 & 4 & 6 & 6 \\ 4 & 4 & 6 & 6 \end{pmatrix} \tag{3.36}$$

De esta forma conseguimos la capa superior. $\frac{\partial J(W,b)}{\partial a_k^{l-1}}$ valor, para obtener δ_k^{l-1}:

$$\delta_k^{l-1} = (\frac{\partial a_k^{l-1}}{\partial z_k^{l-1}})^T \frac{\partial J(W,b)}{\partial a_k^{l-1}} = upsample(\delta_k^l) \odot \sigma^{'}(z_k^{l-1}) \tag{3.37}$$

Entre ellos, la función "upsample" completa la lógica de agrupar la amplificación de la matriz de errores y la redistribución de errores.

3.10.3. Capa de convolución conocida, restaura capa oculta anterior

Resumamos δ^{l-1} para tensores, tenemos:

$$\delta^{l-1} = upsample(\delta^l) \odot \sigma^{'}(z^{l-1}) \tag{3.38}$$

Para *Backward propagation* de la capa convolucional, primero recordamos la fórmula de *Forward propagation* de la capa convolucional:

$$a^l = \sigma(z^l) = \sigma(a^{l-1} * W^l + b^l) \tag{3.39}$$

en n_{in} es el número de submatrices de entrada de la capa oculta anterior. En DNN sabemos que entre δ^{l-1} y δ^l la relación de recurrencia es:

$$\delta^l = \frac{\partial J(W,b)}{\partial z^l} = (\frac{\partial z^{l+1}}{\partial z^l})^T \frac{\partial J(W,b)}{\partial z^{l+1}} = (\frac{\partial z^{l+1}}{\partial z^l})^T \delta^{l+1} \tag{3.40}$$

Por tanto es necesario deducir δ^{l-1} y δ^l se debe calcular la relación de recurrencia de $\frac{\partial z^l}{\partial z^{l-1}}$ expresión de gradiente.

note que entre z^l y z^{l-1} la relación es:

$$z^l = a^{l-1} * W^l + b^l = \sigma(z^{l-1}) * W^l + b^l \tag{3.41}$$

entonces tenemos:

$$\delta^{l-1} = (\frac{\partial z^l}{\partial z^{l-1}})^T \delta^l = \delta^l * rot180(W^l) \odot \sigma^{'}(z^{l-1}) \tag{3.42}$$

La fórmula aquí es en realidad similar a la de DNN. La diferencia es que al derivar la fórmula que contiene convolución, el núcleo de convolución se gira 180 grados. Es decir, en la fórmula $rot180()$, girar 180 grados significa girar hacia arriba y hacia abajo una vez y luego girar hacia la izquierda y hacia la derecha. En DNN esto es sólo la transpuesta de la matriz. ¿Entonces por qué dado que todos estos son tensores hay demasiados parámetros para deducirlos directamente? Aquí usamos un ejemplo simple para ilustrar por qué es necesario invertir el núcleo de convolución después de la derivación.

Supongamos que nuestra $l-1$ es la salida de la capa. a^{l-1} es una matriz de 3x3, l es el núcleo de convolución de la capa. W^l es una matriz de 2×2, con un paso de 1 píxel, la salida z^l es una matriz de 2×2. Simplificamos b^l a 0, entonces hay

$$a^{l-1} * W^l = z^l \tag{3.43}$$

pongamos subíndices en "a" y W la expresión matricial de es la siguiente:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} * \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix} \tag{3.44}$$

Usando la definición de convolución, es fácil concluir:

$$z_{11} = a_{11}w_{11} + a_{12}w_{12} + a_{21}w_{21} + a_{22}w_{22} \tag{3.45}$$
$$z_{12} = a_{12}w_{11} + a_{13}w_{12} + a_{22}w_{21} + a_{23}w_{22} \tag{3.46}$$
$$z_{21} = a_{21}w_{11} + a_{22}w_{12} + a_{31}w_{21} + a_{32}w_{22} \tag{3.47}$$
$$z_{22} = a_{22}w_{11} + a_{23}w_{12} + a_{32}w_{21} + a_{33}w_{22} \tag{3.48}$$
$$\tag{3.49}$$

Luego simulamos la derivación inversa:

$$\nabla a^{l-1} = \frac{\partial J(W,b)}{\partial a^{l-1}} = (\frac{\partial z^l}{\partial a^{l-1}})^T \frac{\partial J(W,b)}{\partial z^l} = (\frac{\partial z^l}{\partial a^{l-1}})^T \delta^l \tag{3.50}$$

Se puede ver en la fórmula anterior que para a^{l-1} el error de gradiente ∇a^{l-1}, igual a la l el error de gradiente de la capa se multiplica por $\frac{\partial z^l}{\partial a^{l-1}}$, $\frac{\partial z^l}{\partial a^{l-1}}$ corresponde al ejemplo anterior relacionado al valor W. Asumir z como el error de *Backward propagation* correspondiente a la matriz de 2×2 compuesta por $d_{11}, d_{12}, d_{21}, d_{22}$, usando la fórmula de gradiente anterior y cuatro ecuaciones, podemos escribir respectivamente ∇a^{l-1} el gradiente de 9 escalares.

Por ejemplo, para a_{11} el gradiente es

$$\nabla a_{11} = \delta_{11}w_{11} \tag{3.51}$$

ya que en las 4 ecuaciones a_{11} y z_{11} hay una relación de producto, por lo que tenemos:

Para el gradiente de a_{12}

$$\nabla a_{12} = \delta_{11}w_{12} + \delta_{12}w_{11} \tag{3.52}$$

ya que en las 4 ecuaciones a_{12} y z_{12}, z_{11} hay una relación de producto.

De la misma manera obtenemos:

$$\nabla a_{13} = \delta_{12}w_{12} \tag{3.53}$$
$$\nabla a_{21} = \delta_{11}w_{21} + \delta_{21}w_{11} \tag{3.54}$$
$$\nabla a_{22} = \delta_{11}w_{22} + \delta_{12}w_{21} + \delta_{21}w_{12} + \delta_{22}w_{11} \tag{3.55}$$
$$\nabla a_{23} = \delta_{12}w_{22} + \delta_{22}w_{12} \tag{3.56}$$
$$\nabla a_{31} = \delta_{21}w_{21} \tag{3.57}$$
$$\nabla a_{32} = \delta_{21}w_{22} + \delta_{22}w_{21} \tag{3.58}$$
$$\nabla a_{33} = \delta_{22}w_{22} \tag{3.59}$$
$$\tag{3.60}$$

Las 9 fórmulas anteriores en realidad se pueden expresar en forma de una matriz convolucional, es decir:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_{11} & \delta_{12} & 0 \\ 0 & \delta_{21} & \delta_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{pmatrix} = \begin{pmatrix} \nabla a_{11} & \nabla a_{12} & \nabla a_{13} \\ \nabla a_{21} & \nabla a_{22} & \nabla a_{23} \\ \nabla a_{31} & \nabla a_{32} & \nabla a_{33} \end{pmatrix} \tag{3.61}$$

Para cumplir con el cálculo del gradiente, llenamos un círculo de ceros alrededor de la matriz de error. En este momento, convolucionamos el núcleo de convolución con el error de gradiente del *Backward propagation* para obtener el error de gradiente anterior. Este ejemplo presenta intuitivamente por qué el núcleo de convolución debe invertirse 180 grados al propagar hacia atrás una fórmula que contiene convolución.

Lo anterior es el proceso de *Backward propagation* del error de la capa convolucional.

3.10.4. Capa convolucional conocida deduce error de gradiente

Bien, ahora podemos derivar el error de gradiente de cada capa. Para la capa δ^l completamente conectada, la capa se puede calcular de acuerdo con el algoritmo de *Backward propagation* de DNN. Gradiente de W, b , mientras que la capa de agrupación W, b no, no es necesario preguntar W, b degradado. Solo capa convolucional W, b.

Tenga en cuenta que la capa convolucional z y W, b la relación es:

$$z^l = a^{l-1} * W^l + b \tag{3.62}$$

Entonces tenemos:

$$\frac{\partial J(W,b)}{\partial W^l} = a^{l-1} * \delta^l \tag{3.63}$$

Tenga en cuenta que el núcleo de convolución no se invierte en este momento, principalmente porque se trata de una derivación dentro de la capa, en lugar de un *Backward propagation* a la derivación de la capa anterior. Podemos analizar el proceso específico [16].

Un ejemplo simplificado es el mismo que el de la subsección 3.10.3. La entrada aquí es una matriz, no un tensor. Luego, para la l-ésima capa, la derivada de una determinada matriz de núcleo de convolución W se puede expresar de la siguiente manera:

$$\frac{\partial J(W,b)}{\partial W^l_{pq}} = \sum_i \sum_j (\delta^l_{ij} a^{l-1}_{i+p-1,j+q-1}) \tag{3.64}$$

Supongamos que las entradas son de forma a siendo una matriz 4×4, el núcleo de convolución W es una matriz de 3×3, la salida z es una matriz de 2×2, luego de *Backward propagation* z es el error de gradiente d también es una matriz de 2×2.

Entonces según la fórmula anterior tenemos:

$$\frac{\partial J(W,b)}{\partial W^l_{11}} = a_{11}\delta_{11} + a_{12}\delta_{12} + a_{21}\delta_{21} + a_{22}\delta_{22} \tag{3.65}$$

$$\frac{\partial J(W,b)}{\partial W^l_{12}} = a_{12}\delta_{11} + a_{13}\delta_{12} + a_{22}\delta_{21} + a_{23}\delta_{22} \tag{3.66}$$

$$\frac{\partial J(W,b)}{\partial W^l_{13}} = a_{13}\delta_{11} + a_{14}\delta_{12} + a_{23}\delta_{21} + a_{24}\delta_{22} \tag{3.67}$$

$$\frac{\partial J(W,b)}{\partial W^l_{21}} = a_{21}\delta_{11} + a_{22}\delta_{12} + a_{31}\delta_{21} + a_{32}\delta_{22} \tag{3.68}$$

$$\tag{3.69}$$

Al final podemos obtener un total de 9 fórmulas. Después de organizar en forma matricial, podemos obtener:

$$\frac{\partial J(W,b)}{\partial W^1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} * \begin{pmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{pmatrix} \tag{3.70}$$

De esto podemos ver claramente la razón por la que no retrocedimos esta vez.

Para b, es ligeramente especial, porque δ^l es un tensor de alta dimensión, y b es solo un vector y no se puede sumar directamente como δ^l igual en DNN . El enfoque habitual es δ^l los elementos de cada submatriz se suman por separado para obtener un vector de error, que es el gradiente de b:

$$\frac{\partial J(W,b)}{\partial b^l} = \sum_{\mathrm{u,v}} (\delta^l)_{\mathrm{u,v}} \tag{3.71}$$

3.10.5. Redes convolucionales 3D

La principal diferencia entre la convolución 3D y la convolución 2D radica en la dimensionalidad de los datos de entrada y los filtros. En la convolución 2D los datos de entrada y los filtros son matrices bidimensionales, mientras que en la convolución 3D son volúmenes tridimensionales. Esto implica que en la convolución 3D los filtros se deslizan a lo largo de tres ejes en lugar de dos, lo que aumenta significativamente la complejidad computacional. Además, en el *Backward propagation*, se deben calcular gradientes tridimensionales en lugar de bidimensionales, lo que requiere operaciones adicionales.

Capítulo 4

Resonancia magnética

4.1. Un poco de contexto

4.1.1. Orígenes en la física

La resonancia magnética surge de la combinación de principios de la mecánica cuántica y del electromagnetismo clásico. Desde principios del siglo XX, los científicos descubrieron que ciertos núcleos atómicos poseen un momento magnético intrínseco debido a su espín nuclear. El espín es una propiedad cuántica fundamental que, a diferencia de la rotación clásica, no tiene un análogo directo en el mundo macroscópico, pero genera un momento magnético detectable. En particular, el protón, presente en abundancia en el hidrógeno, posee un momento magnético que puede interactuar con campos magnéticos externos, absorbiendo y emitiendo energía en la región de radiofrecuencia.

Durante la década de 1940, Felix Bloch y Edward Purcell realizaron experimentos independientes que confirmaron que estas transiciones energéticas podían ser detectadas. Su trabajo demostró la posibilidad de medir la magnetización de núcleos en distintos materiales y abrió la puerta al desarrollo de técnicas de espectroscopía nuclear. Por este logro, ambos recibieron el Premio Nobel de Física en 1952, consolidando así la base teórica y experimental de lo que más tarde se conocería como resonancia magnética nuclear (RMN).

4.1.2. De la química a la medicina

Inicialmente, la RMN se utilizaba en el análisis químico y estructural de moléculas, permitiendo determinar la composición y disposición de átomos en compuestos complejos. Sin embargo, a medida que se comprendió la abundancia de protones de hidrógeno en los tejidos biológicos, los investigadores comenzaron a explorar su potencial para generar imágenes médicas. A diferencia de los rayos X, que implican radiación ionizante, la resonancia magnética ofrecía una alternativa segura y no invasiva para estudiar el cuerpo humano, con un alto contraste entre distintos tejidos blandos.

El avance hacia la imagenología médica fue posible gracias al desarrollo de gradientes de campo magnético. Al aplicar campos magnéticos cuya intensidad varía en el espacio, la frecuencia de precesión de los protones se vuelve dependiente de su ubicación. Esta codificación espacial permite reconstruir imágenes tridimensionales que reflejan la distribución de protones en el cuerpo. Paul Lauterbur y Peter Mansfield fueron los pioneros en la década de 1970 en aplicar estos conceptos, lo que les valió el Premio Nobel de Medicina en 2003. Este hito marcó la transición de la RMN de una herramienta química a una técnica de imagenología revolucionaria.

4.1.3. Principios físicos fundamentales

El funcionamiento de la resonancia magnética se basa en la interacción entre los protones de hidrógeno y un campo magnético estático fuerte, generalmente denotado como B_0. Los protones tienden a alinearse con el campo en dirección paralela (bajo energía) o antiparalela (alta energía), generando una magnetización neta en la dirección del campo aplicado. La precesión de estos momentos magnéticos alrededor de B_0 ocurre a la frecuencia de Larmor, determinada por la relación:

$$\omega_0 = \gamma B_0, \tag{4.1}$$

donde γ es la constante giromagnética del protón (42,58 MHz/T). Por ejemplo, en un resonador clínico de 1.5 T, la frecuencia de Larmor es aproximadamente 63,9 MHz, dentro del rango de radiofrecuencia.

4.1.4. Relajación y contraste de tejidos

Cuando los protones son excitados por un pulso de radiofrecuencia, se alejan de su estado de equilibrio y posteriormente retornan, emitiendo energía. Existen dos mecanismos principales de relajación:

- **Relajación longitudinal (T_1)**: recuperación de la magnetización en la dirección del campo B_0.
- **Relajación transversal (T_2)**: pérdida de coherencia de los protones en el plano perpendicular a B_0.

Cada tejido presenta valores característicos de T_1 y T_2, generando contraste natural en las imágenes. Por ejemplo, el agua libre suele tener T_2 prolongado, mientras que la grasa presenta tiempos más cortos. Esta diferenciación es la base del alto poder de discriminación tisular de la resonancia magnética, permitiendo distinguir tejidos sanos de patológicos.

4.1.5. Impacto tecnológico y aplicaciones modernas

La integración de principios físicos y avances tecnológicos hizo posible el desarrollo de algoritmos de reconstrucción de imágenes, acelerando la obtención de mapas tridimensionales precisos. Hoy en día, la resonancia magnética no solo permite visualizar anatomía y estructuras internas, sino que también se utiliza en estudios funcionales, espectroscopía metabólica y en la investigación de nuevas terapias. Además, la disponibilidad de grandes volúmenes de datos de imágenes ha facilitado el uso de inteligencia artificial para clasificación, segmentación y análisis avanzado, consolidando la resonancia magnética como una herramienta central en la medicina moderna [19].

4.1.6. Resumen histórico y científico

En síntesis, la resonancia magnética es el resultado de la unión entre descubrimientos teóricos de la física cuántica, experimentos pioneros en espectroscopía nuclear y la innovación tecnológica en imanes, gradientes espaciales y procesamiento de señales. Su evolución desde la RMN química hasta la imagenología médica ilustra cómo un concepto científico fundamental puede transformarse en una herramienta con impacto profundo en la salud humana.

4.2. ¿Qué es la resonancia magnética?

La resonancia magnética (RM) tiene su origen en la técnica de resonancia magnética nuclear (RMN) aplicada a la obtención de imágenes médicas. El término “magnética” hace referencia al uso de distintos campos magnéticos, mientras que “resonancia” alude a la necesidad de ajustar

la frecuencia de un campo magnético oscilante para que coincida con la frecuencia de precesión del espín de ciertos núcleos (lo que da lugar al término "nuclear") presentes en las moléculas del tejido. Aunque el nombre más preciso sería RMN, se ha evitado el uso del término "nuclear" por la connotación negativa que puede tener. A pesar de que este término solo describe un aspecto inofensivo relacionado con el espín del núcleo atómico, tanto el público como los profesionales de la salud han optado por usar simplemente RM. Es un método de adquisición de imágenes que se basa en la utilización de ondas de radio frecuencia aplicadas por un campo magnético potente generado por un imán. No utiliza radiación ionizante.

4.3. Partes de un resonador

Figura 4.1: Resonador Magnético prodiva de 1.5 T (ejemplo) [17]

4.3.1. Imán

La potencia de estos imanes se mide en teslas, (10,000 Gauss = 1 tesla), el imán de un resonador suele ir entre 0.38 teslas - 3 teslas, para sistemas comerciales. Existen resonadores de hasta 14 teslas para investigación.

4.3.2. Antenas emisoras-receptoras de radiofrecuencia

Estas emiten y receptan las ondas de radiofrecuencia (RF)

4.3.3. Aislamiento magnético y de radiofrecuencia

Efecto de la jaula de Faraday

La cobertura de estos cuartos suele tener placas y/o mallas de cobre de la totalidad de las paredes, puertas y ventanas de la sala donde se ubica el resonador. Lo cual tiene como finalidad que las ondas de RF provenientes del exterior no modifiquen la obtención de imágenes. Lo anterior se logra utilizando materiales que sean altamente permeables al campo magnético y que desvíen o canalicen el flujo magnético alrededor del objeto que se desea proteger. Las paredes aislantes suelen estar hechas de materiales metálicos altamente conductores.

4.4. Física de la resonancia magnética

Recordemos que la materia está formada por átomos que poseen

- Núcleo: Protones y neutrones
- Corteza: Electrones

4.4.1. El espín

El espín en un átomo es una propiedad cuántica fundamental asociada a las partículas subatómicas, como electrones, protones y neutrones. Se refiere al momento angular intrínseco de la partícula, que no tiene una analogía clásica directa. El espín de un protón o electrón genera un momento magnético intrínseco, lo que significa que el protón o electrón tiene una propiedad magnética asociada con su espín. El espín de un núcleo está determinado por el número cuántico del espín del conjunto total de sus elementos.

La magnetización neta es el resultado de alinear los momentos magnéticos individuales de los átomos o electrones dentro de un material en una dirección preferencial, ya sea natural o inducida. [18]

La magnetización neta nula significa que no hay un campo magnético neto dentro del material y que los momentos magnéticos individuales dentro del material están orientados de manera aleatoria y se cancelan mutuamente.[18]

Siendo el agua un constituyente fundamental en casi todas las partes del cuerpo humano, es natural usarlo como método de estudio del mismo. En particular, estudiar al átomo de hidrógeno como el elemento a distinguir en la molécula de agua. El átomo de hidrógeno tiene un momento magnético asociado con el espín de su electrón. El electrón en un átomo de hidrógeno, al estar en movimiento, genera un momento magnético intrínseco debido a su espín.[18]

Precesión incoherente

El movimiento de precesión es una rotación provocada por un campo magnético externo caracterizado por un movimiento cónico; los átomos precesan a una frecuencia determinada por la intensidad del campo magnético externo, pero en distinta fase (Ecuación de Larmor)

$$\omega = \gamma \vec{B_0} \tag{4.2}$$

donde γ es la constante giromagnética característica de cada elemento y $\tilde{\mathbf{B_0}}$ es la intensidad del campo magnético (en teslas). Esto es, a mayor intensidad de campo magnético, mayor intensidad de frecuencia de precesión[18]. Si todos los átomos de hidrógeno precesan en la misma fase, hablamos de precesión coherente. [18].

4.4.2. ¿Qué pasa dentro de un resonador magnético?

Dentro de un resonador magnético, al aplicar un campo magnético externo al paciente, los átomos de hidrógeno dentro de las moléculas de agua y moléculas de grasa, sucede un fenómeno llamado polarización de la magnetización, el cual consiste en que sus átomos de hidrógeno del paciente alinean su campo magnético con el campo magnético externo. La alineación puede ser en sentido paralelo (menor nivel de energía) o antiparalelo (mayor energía) al campo magnético. La alineación predominante es la que implica menor uso de energía, es decir, en sentido paralelo. Una vez alineados los campos magnéticos de los átomos de hidrógeno, estos empiezan a precesar incoherentemente; luego se aplican ondas de RF con las cuales se da otro fenómeno llamado rotación de la magnetización o magnetización longitudinal, en el cual los átomos de hidrógeno cambian el sentido de orientación de su campo magnético, y estos empezarán a precesar coherentemente. Las ondas de RF poseen frecuencia similar a la frecuencia de precesión de los átomos. Esto permite un intercambio de energía, fenómeno que se conoce como resonancia magnética nuclear.

Luego de esto se interrumpen las ondas de RF, la magnetización vuelve a su estado de reposo, con lo cual los átomos de hidrógeno vuelven a estar alineados con el campo magnético externo y estos al volver a alinearse despiden energía en forma de señales de RF. Este fenómeno es conocido como T1 (Relajación longitudinal). Esta señal se conoce como señal de recuperación de espín o señal de longitud de eco y tiene una amplitud que depende de la constante de tiempo T1 del tejido y del tiempo transcurrido desde la excitación. Los átomos de hidrógeno vuelven a precesar de forma incoherente, es decir, todos en diferentes fases, a este otro fenómeno se le llama T2 (relajación transversal) con el cual también despiden energía en forma de señales de RF, que disminuye exponencialmente con el tiempo. Esta señal se conoce como señal de eco de espín o señal de decaimiento y su amplitud disminuye con una constante de tiempo T2.

4.4.3. Tipos de materiales

Diamagnéticas

Las sustancias diamagnéticas no tienen electrones orbitales des-apareados. Cuando una sustancia de este tipo se coloca en un campo magnético externo $\vec{B_0}$, se induce un campo magnético débil (M) en la dirección opuesta a $\vec{B_0}$. Como resultado, el campo magnético efectivo se reduce. Por lo tanto, las sustancias diamagnéticas tienen una susceptibilidad magnética negativa pequeña χ (es decir, χ <0 y μ <1). Básicamente, son no magnéticas. La gran mayoría de los tejidos del cuerpo tienen esta propiedad. Un ejemplo de efecto diamagnético es la distorsión que se produce en una interfaz aire-tejido (como alrededor de los senos paranasales).

Paramagnéticas

Las sustancias paramagnéticas tienen electrones orbitales des-apareados. Se magnetizan mientras el campo magnético externo $\vec{B_0}$ está activo y se desmagnetizan una vez que el campo se ha apagado. Su campo magnético inducido (M) está en la misma dirección que el campo magnético externo. En consecuencia, su presencia provoca un aumento en el campo magnético efectivo. Por lo tanto, tienen un χ positivo pequeño (es decir, χ >0 y μ >1) y son débilmente atraídos por el campo magnético externo. En tales sustancias, las interacciones dipolo-dipolo (es decir, protón-protón y protón-electrón) causan acortamiento de T1 (señal brillante en imágenes ponderadas en T1). El elemento de la tabla periódica con el mayor número de electrones desapareados es el elemento de tierras raras gadolinio (Gd) con siete electrones desapareados, que es una sustancia paramagnética fuerte. Gd es un miembro del grupo de los lantánidos en la tabla periódica. El elemento de tierras raras disprosio (Dy) es otra sustancia paramagnética fuerte que pertenece a este grupo. Ciertos productos de degradación de la hemoglobina son paramagnéticos: la desoxihemoglobina tiene cuatro electrones desapareados y la metahemoglobina tiene cinco. La hemosiderina, la etapa final de la hemorragia, contiene, en comparación, más de 10.000 electrones des-apareados. La hemosiderina

pertenece a un grupo de sustancias llamadas superparamagnéticas, que tienen susceptibilidades magnéticas de 100 a 1000 veces más fuertes que las sustancias paramagnéticas.

Ferromagnéticas

Las sustancias ferromagnéticas son fuertemente atraídas por un campo magnético. Quedan permanentemente magnetizadas incluso después de que el campo magnético se haya apagado. Tienen un χ positivo grande, incluso mayor que el de las sustancias superparamagnéticas. Se conocen tres tipos de ferroimanes: hierro (Fe), cobalto (Co) y níquel (Ni). Algunos ejemplos son los clips de aneurisma y la metralla.

4.4.4. Pulso RF 90°

En respuesta a un campo magnético intenso en la dirección z, los espines se alinean. Esto da como resultado una magnetización neta, M_0. A continuación, aplicamos un pulso de RF externo que invierte el vector de magnetización 90° en el plano x-y. Cuando el vector de magnetización está en el plano x-y, lo llamamos M_{xy}.

$$M_{xy} = \text{componente de } M_0 \text{ en el plano x-y} \tag{4.3}$$

Si todo el vector se invierte en el plano x–y, entonces la magnitud de M_{xy} es igual a la magnitud del vector M_0. Esto se llama inversión de 90°. El pulso que causa la inversión de 90° se llama pulso de RF de 90°.

4.4.5. Tiempo de relajación T1

El término relajación significa que los espines se relajan y vuelven a su estado de energía más bajo o al estado de equilibrio, (el equilibrio, por definición, es el estado de energía más bajo posible). Una vez que se apaga el pulso de RF, los protones tendrán que realinear con el eje del campo magnético $\vec{B_0}$ y ceder todo su exceso de energía.

T1 se denomina tiempo de relajación longitudinal porque se refiere al tiempo que tardan los espines en realinear a lo largo del eje longitudinal (z). T1 también se denomina tiempo de relajación de espín-red porque se refiere al tiempo que tardan los espines en devolver la energía que obtuvieron del pulso de RF a la red circundante para volver a su estado de equilibrio.

Inmediatamente después del pulso de 90°, la magnetización M_{xy} precesa dentro del plano x–y, oscilando alrededor del eje z con todos los protones rotando en fase. Después de que la magnetización se haya invertido 90° en el plano x–y, el pulso de RF se apaga. Un principio general de la termodinámica es que cada sistema busca su nivel de energía más bajo. Por lo tanto, después de que se apaga el pulso de RF, ocurrirán los dos eventos siguientes:

- Los espines volverán al estado de energía más bajo.
- Los espines se desfasarán entre sí.

Estos eventos son el resultado de dos procesos simultáneos pero separados que ocurren después de que se apaga el pulso de RF:

1. El componente M_{xy} del vector de magnetización disminuye rápidamente.
2. El componente M_z se recupera lentamente a lo largo del eje z. Siguiendo la siguiente ecuación:

$$M_z(t) = M_0(1 - e^{-\frac{t}{T1}}) \tag{4.4}$$

Para resumir T1 es la constante de tiempo que tarda en volver a ser la Magnetización M_z nuevamente M_0 cuando se retiró el pulso de RF.[22]

4.4.6. Tiempo de relajación T2

T2 es la constante de tiempo que tarda la magnetizaciónM_{xy} en decaer siguiendo la siguiente ecuación:

$$M_{xy}(t) = M_0 e^{-\frac{t}{T2}} \tag{4.5}$$

Observe que la recuperación de la magnetización a lo largo del eje z y la desintegración de la magnetización dentro del plano x-y son dos procesos independientes que ocurren a dos velocidades diferentes. Tomemos un proceso exponencial simple. Esperaríamos que la velocidad a la que este proceso decae en el plano x-y sea la misma que la de su crecimiento a lo largo del eje z. Esto no es lo mismo en el sistema de imágenes por resonancia magnética que estamos analizando porque este sistema implica un proceso mucho más complicado. La desintegración de T2 ocurre de 5 a 10 veces más rápidamente que la recuperación de T1. Para entender esto, necesitamos entender el concepto de desfase el cual consiste en: Después de apagar el pulso de RF de 90°, todos los espines están en fase; están todos alineados en la misma dirección y giran a la misma frecuencia ω_0. Hay dos fenómenos que harán que los espines se desfasen: interacciones entre espines e inhomogeneidades del campo externo.

- Interacciones entre espines individuales

  Cuando dos espines están uno al lado del otro, el campo magnético de un protón afecta al protón que está a su lado. Supongamos que un protón está alineado con el campo y el otro está en contra de él. El protón alineado con $\vec{B_0}$ crea un campo magnético ligeramente más alto para su vecino, de modo que el protón n.° 1 está expuesto al campo magnético $\vec{B_0}$ más un pequeño campo magnético creado por el otro protón (ΔB). La frecuencia de precesión del protón aumentará entonces ligeramente como

  $$\omega(\text{protón n.}^\text{o}\ 1) = (\vec{B_0} + \Delta\vec{B}) \tag{4.6}$$

  Por otro lado, el otro protón está expuesto a un campo magnético ligeramente más débil porque el otro protón apunta en contra de $\vec{B_0}$. Por lo tanto, su exposición total a la intensidad del campo magnético será ligeramente menor. La frecuencia de precesión del protón n.° 2 disminuirá ligeramente como

  $$\omega(\text{protón n.}^\text{o}\ 2) = (\vec{B_0} - \Delta\vec{B}) \tag{4.7}$$

  La diferencia en el entorno magnético creado por estas interacciones protón-protón puede ser muy pequeña, pero marca una diferencia en la homogeneidad general del campo magnético al que están expuestos los espines. Por lo tanto, la primera causa del desfase es inherente al tejido. Se denomina interacción espín-espín. Esta interacción es una propiedad inherente a cada tejido y se mide mediante T2.

- Inhomogeneidad del campo magnético externo

  Este es el segundo fenómeno que hace que los espines se desfasen. No importa lo bueno que sea el sistema que tengamos, no importa lo estable que sea el campo magnético externo, todavía existe alguna variación en la homogeneidad del campo magnético (generalmente medida en partes por millón).

  La inhomogeneidad del campo magnético externo hace que los protones en diferentes ubicaciones precesen a diferentes frecuencias porque cada espín está expuesto a una intensidad de campo magnético ligeramente diferente. Estas frecuencias variables son muy cercanas entre sí y muy cercanas a la verdadera frecuencia de Larmor; sin embargo, estas pequeñas diferencias en la frecuencia dan como resultado un desfase del espín.

Diferencias entre T2 y T2*

La desintegración de T2* depende de ambos:

- Campo magnético externo
- Interacciones espín-espín

La desintegración de T2 depende principalmente de

- Interacciones espín-espín

La T2 de un tejido, dado que depende únicamente de las interacciones espín-espín, es fija: no tenemos control sobre lo que los espines se hacen entre sí. La T2* depende de la homogeneidad del campo magnético externo, por lo que no es fija. Varía según la uniformidad del imán principal. La T2* siempre es menor que la T2. La desintegración de T2* es siempre más rápida que la de T2. La siguiente ecuación relaciona ambas:

$$\frac{1}{T2*} = \frac{1}{T2} + \gamma \Delta B \tag{4.8}$$

4.4.7. Tipos de cortes

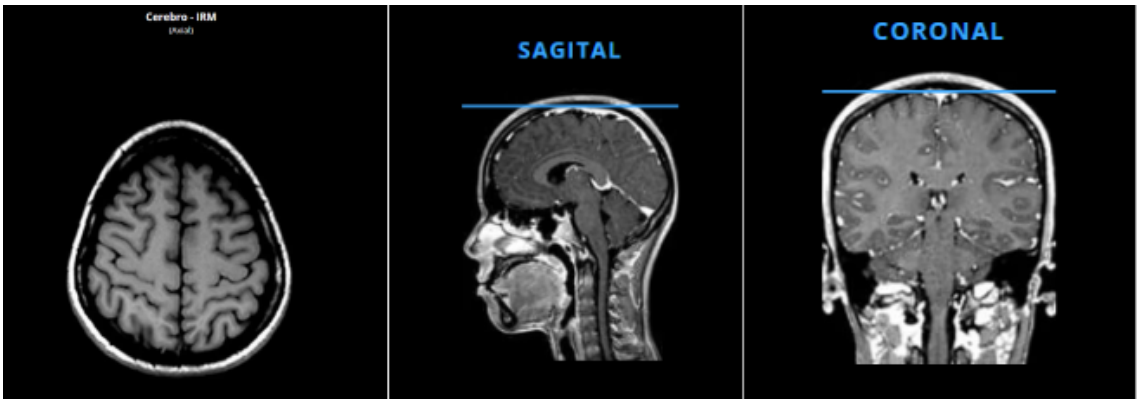


Figura 4.2: corte axial a), sagital b) y coronal c) [20].

existen los cortes axiales, sagitales y coronales, Para nuestro caso nos enfocaremos en los cortes axiales, ver figura 4.2

4.4.8. Características de señal

Cuando en una imagen obtenida por resonancia magnética si algo es muy blanco se le suele llamar hiperintenso, cuando algo es muy oscuro se le suele llamar hipointenso, en la tabla 4.1 se puede ver como deberían ser los colores en una imagen de resonancia magnética para una imagen en T1 o T2 [21]

4.5. Precesión en un campo magnético

La aguja de una brújula oscila hacia el polo norte magnético de la Tierra porque un dipolo magnético, como la aguja de una brújula, experimenta un torque cuando se coloca en un campo magnético uniforme. Similarmente, un protón que está en un campo magnético experimenta un

	T1	T2	señal T1	señal T2
Agua	Largo	Largo	Hipo	Hiper
Grasa	Corto	Corto	Hiper	Hipo
Tejido patológico	Corto	Largo	Hipo	Hiper

Tabla 4.1: intensidad de señales T1 y T2 en diferentes medios

torque ya que es un dipolo magnético, sin embargo, su respuesta a este campo difiere de la de la aguja ya que este es un cuerpo giratorio; en lugar de alinearse con el campo magnético este precesa al rededor del campo magnético.

Sabemos que en un campo magnético $\tilde{\mathbf{B}}$ los estados estacionarios de un espín medio para un dipolo magnético μ tiene energías de

$$E_{\pm} = \mp \mu B \tag{4.9}$$

la evolución del tiempo de cualquier estado de espín es

$$|\psi, t\rangle = a_{-} e^{-iE_{-}t/\hbar} |-\rangle + a_{+} e^{-iE_{+}t/\hbar} |+\rangle \tag{4.10}$$

donde $|\pm\rangle$ son los eigenkets de $\vec{B} \cdot \vec{S}$ y las amplitudes constantes $a_{\pm}$ especifican la condición inicial $|\psi, 0\rangle = a_{-} |-\rangle + a_{+} |+\rangle$. Suponga que inicialmente una medida del espín paralelo a $n \equiv (sin(\theta), 0, cos(\theta))$ es ciertamente $\frac{1}{2}$; sabemos que si $\vec{n}$ es un vector unitario con coordenadas polares (θ, ϕ).El estado de una partícula de espín medio en el que una medición del componente del espín a lo largo de $\vec{n}$ seguramente producirá $\frac{1}{2}\hbar$ es

$$|+, n\rangle = sin(\frac{\theta}{2}) e^{i\frac{\phi}{2}} |-\rangle + cos(\frac{\theta}{2}) e^{-i\frac{\phi}{2}} |+\rangle \tag{4.11}$$

entonces tenemos $a_{-} = sin(\frac{\theta}{2})$ y $a_{+} = cos(\frac{\theta}{2})$ por lo que al tiempo t el estado del protón es

$$|\psi, t\rangle = sin(\frac{\theta}{2}) e^{-i\frac{E_{-}t}{\hbar}} |-\rangle + cos(\frac{\theta}{2}) e^{-i\frac{E_{+}t}{\hbar}} |+\rangle \tag{4.12}$$

$$= sin(\frac{\theta}{2}) e^{-i\frac{\phi}{2}} |-\rangle + cos(\frac{\theta}{2}) e^{-i\frac{\phi}{2}} |+\rangle \tag{4.13}$$

donde

$$\phi = \frac{2E_{+}t}{\hbar} = \omega t; \omega = \frac{-2\mu_p B}{\hbar} \tag{4.14}$$

donde μ_p es el momento magnético de un protón. Esto es justo $|+, n'\rangle$ en donde una medida de espín paralelo a $n' = (sin(\theta)cos(\phi), sin(\theta)sin(\phi), cos(\theta))$ es justamente $\frac{1}{2}$. Consecuentemente la dirección en la cual una medida de espín es justamente $\frac{1}{2}$ rotando al rededor de la dirección de B a una frecuencia ω. Esto refleja el comportamiento esperado en física clásica de un dipolo magnético de magnitud μ que tiene un momento espín angular de $\frac{1}{2}\hbar$; su espín axial precesariá alrededor de B a una frecuencia angular ω. Cuando el material que contiene átomos de hidrógeno unido químicamente está inmerso en un poderoso campo magnético, la mayoría de los protones alinean su espín con el campo B en orden para minimizar su energía. Radiación de frecuencia ω tiene justo la energía requerida para golpear al protón en su nivel más alto de energía, estado en el cual no esta alineado con B (perpendicular). Consecuentemente, cada radiación es fácilmente absorbida por una muestra, considerando que no hay radiación de frecuencias vecinas. Como el análisis muestra, la interferencia cuántica entre los estados alineados y no alineados causa el valor esperado del momento magnético para precesar a una frecuencia angular ω y el momento magnético de precesión se acopla resonantemente al campo de radiación impuesto [23]

Capítulo 5

Tratamiento de imágenes

5.1. ¿Por que hacer un programa?

El captar pequeños detalles para la vista humana en un estudio de resonancia magnética es algo que solo médicos especialistas pueden hacer, como sucede al diagnosticar una enfermedad como el TDAH, por lo que no vendría mal la ayuda de un ojo que extraiga características complejas de imágenes o detalles muy pequeños imperceptibles para el ojo humano, tarea que puede ser realizada por una red neuronal convolucional.

En este programa lo definimos como un algoritmo de aprendizaje supervisado ya que tenemos etiquetadas las imágenes como positivo y negativo a TDAH de cerebros de infantes, imágenes obtenidas de estudios de resonancia magnética obtenidos en T1(Relajación longitudinal). por lo cual este modelo es útil cuando se trata de clasificar únicamente imágenes en T1, para cerebros de infantes en cortes axiales.

Este programa fue diseñado con intenciones de ayudar a un médico profesional o especialista en el área a poder diagnosticar TDAH mediante un estudio de resonancia magnética debido a que no existe una prueba única para diagnosticar el TDAH. [24]

La imagen se trató de la siguiente forma: Para leer el volumen de datos de resonancia magnética se utilizó la librería nibabel una herramienta muy útil en el campo de la neuroimagen y la ciencia de datos médicos. Su función principal es leer y escribir archivos de imágenes médicas en formatos como NIfTI, en este caso se uso para obtener los volúmenes de datos de resonancias magnéticas en archivos .nii ver función "read_nifti_file" apéndice A.2

5.2. Normalización

Se hizo una normalización de los datos al ser imágenes en blanco y negro la normalización va de 0-255 para evitar datos extraños, cualquier valor en el volumen que sea menor que el valor mínimo se establece en el valor mínimo, y cualquier valor que sea mayor que el valor máximo se establece en el valor máximo. Esto asegura que los valores estén dentro del rango definido. Ver función "normalize" en apéndice A.3

5.3. Redimensionamiento

Se redimensiona el volumen a un número especifico de por ancho,alto y largo de los datos, dejando la profundidad en 64, alto y ancho en 128 haciendo uso de la función zoom de la clase ndimage de la librería scipy. Ver función "resize_volume" en apéndice A.4

Finalmente mezclamos todo el tratamiento de imágenes en una sola función. Ver función "process_scan" en apéndice A.5

Capítulo 6

Procesamiento de datos de entrenamiento

Primero necesitamos saber con cuantos datos contamos así que todos nuestros datos positivos y negativos fueron guardados en respectivas carpetas con POS o NEG a ADHD respectivamente. (ver apéndice) A.6

Por lo que la carpeta POS contiene: 123 archivos y la carpeta NEG contiene: 99 archivos, guardamos los archivos de ambas carpetas en numpy arrays para poder utilizarlos con la función para normalizar nuestros datos y separamos los datos en muestras de entrenamiento y validación en un 70 % y 30 % respectivamente (ver apéndice) A.7

De modo que tenemos un número de muestras en entrenamiento y validación son 148 y 74 respectivamente. Luego usamos una técnica para aumentar los datos llamada "data augmentation" agregando ángulos de -20,-10,-5,5,10,20; el decorador "@tf.function" permite que la función sea ejecutada de manera eficiente por el motor de "TensorFlow". "tf.numpy_function" para envolver la función de Python "scipy_rotate" dentro del gráfico computacional de "TensorFlow". Esto permite que la función de rotación de "SciPy" sea ejecutada como parte del gráfico de "TensorFlow". El volumen original (volume) se pasa como argumento a la función "scipy_rotate" varias de las funciones usadas en este código son modificaciones a un programa de tomografías computarizada [25] (ver apéndice) A.8

"train_dataset" es un objeto que contiene los datos de entrenamiento después de ser barajados, preprocesados, agrupados en lotes y precargados en memoria para el entrenamiento del modelo del mismo modo "validation_dataset" (ver apéndice) A.9

Luego utilizamos el primer dato que hay en nuestros datos de entrenamiento para visualizar el corte y saber que podrá clasificar nuestro modelo (ver apéndice) A.10

Podemos visualizar en la figura 6.1 un corte axial de un cerebro por lo que sabemos que contamos con 64 imágenes de este tipo de corte que mapean un solo cerebro por archivo de dimensiones de 128x128.

Visualizando en la figura 6.2 40 de esas 64 capas podemos ver que no todas traen la misma información por lo que nuestra normalización y redimensionamiento fueron útiles para que en todo nuestro volumen se encuentre información util que tenga la forma del tamaño de la entrada de nuestra red neuronal (ver apéndice) A.11 Luego llega el momento de hacer nuestro modelo, el cual consiste en tener una capa de entrada del tamaño de la imagen que es de 128x128x64, luego pasa a una capa de convolución, luego una capa de agrupamiento máximo y una de normalización de batch. Estas tres capas repiten el proceso 4 veces con diferentes filtros. Luego, después de pasar por el proceso de convolución entra a una capa de agrupamiento promedio y finalmente a una capa completamente conectada de 512 entradas para finalizar en una capa de salida de una sola neurona, la cual tiene como función de activación la función sigmoide la cual ayuda a clasificar a

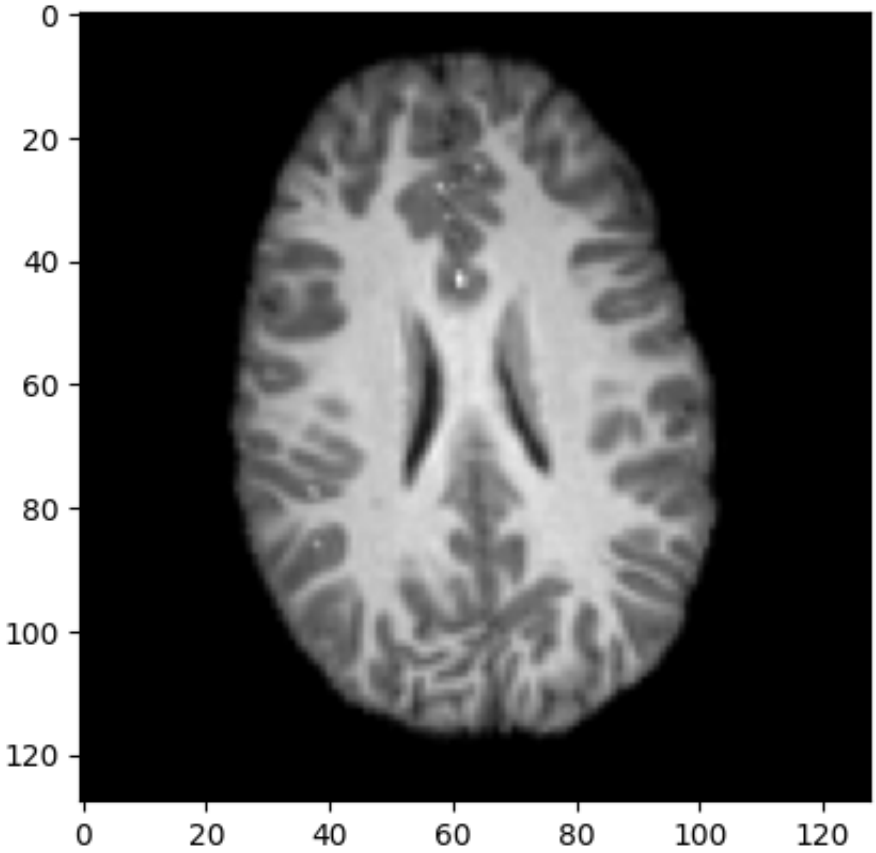


Figura 6.1: Capas axial.

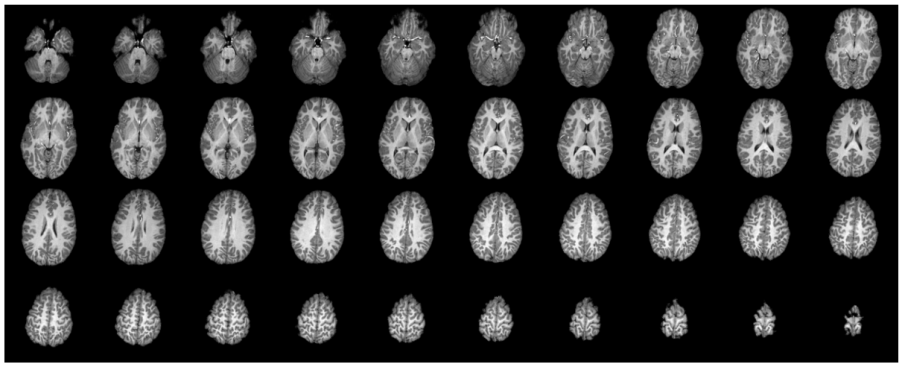


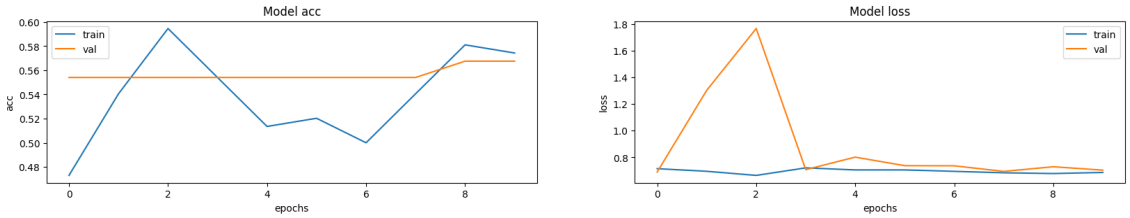
Figura 6.2: 40 capas axiales.

una salida binaria es decir para nuestro caso, positivo o negativo a TDAH. Al utilizar ReLU en una capa Conv3D, se espera que la red aprenda representaciones más ricas y complejas de los datos, lo que potencialmente mejora el rendimiento en tareas de reconocimiento de patrones en volúmenes de datos tridimensionales. A.12

dando como resultado “Total params”: 1,352,897 “Trainable params”: 1,351,873 “Non-trainable params”: 1,024

listo el modelo se procedió a entrenar (ver apéndice) A.13 dando como resultado.
'textitepoch 1/10
$74/74 - 2300s - loss : 0{,}7152 - acc : 0{,}4730 - val\_loss : 0{,}6898 - val\_acc : 0{,}5541 - 2300s/epoch - 31s/step$
'textitepoch 2/10
$74/74 - 2296s - loss : 0{,}6954 - acc : 0{,}5405 - val\_loss : 1{,}3028 - val\_acc : 0{,}5541 - 2296s/epoch - 31s/step$
'textitepoch 3/10
$74/74 - 2283s - loss : 0{,}6646 - acc : 0{,}5946 - val\_loss : 1{,}7666 - val\_acc : 0{,}5541 - 2283s/epoch - 31s/step$
'textitepoch 4/10
$74/74 - 2231s - loss : 0{,}7211 - acc : 0{,}5541 - val\_loss : 0{,}7077 - val\_acc : 0{,}5541 - 2231s/epoch - 30s/step$
'textitepoch 5/10
$74/74 - 2277s - loss : 0{,}7057 - acc : 0{,}5135 - val\_loss : 0{,}8016 - val\_acc : 0{,}5541 - 2277s/epoch - 31s/step$
'textitepoch 6/10
$74/74 - 2224s - loss : 0{,}7056 - acc : 0{,}5203 - val\_loss : 0{,}7375 - val\_acc : 0{,}5541 - 2224s/epoch - 30s/step$
'textitepoch 7/10
$74/74 - 2263s - loss : 0{,}6951 - acc : 0{,}5000 - val\_loss : 0{,}7366 - val\_acc : 0{,}5541 - 2263s/epoch - 31s/step$
'textitepoch 8/10
$74/74 - 2264s - loss : 0{,}6841 - acc : 0{,}5405 - val\_loss : 0{,}6947 - val\_acc : 0{,}5541 - 2264s/epoch - 31s/step$
'textitepoch 9/10
$74/74 - 2260s - loss : 0{,}6785 - acc : 0{,}5811 - val\_loss : 0{,}7295 - val\_acc : 0{,}5676 - 2260s/epoch - 31s/step$
'textitepoch 10/10
$74/74 - 2254s - loss : 0{,}6868 - acc : 0{,}5743 - val\_loss : 0{,}7041 - val\_acc : 0{,}5676 - 2254s/epoch - 30s/step$

Al ver las gráficas podemos ver que la pérdida no cambia mucho después de la 'textitepoch 7 y la exactitud del como predice podemos ver que varía bastante en la 'textitepoch 8. (ver apéndice) A.14



$1/1[==============================] - 4s4s/step$
Este modelo es 58.52 por ciento seguro de que es normal
Este modelo es 41.48 por ciento seguro de que es abnormal . Ver A.15

Capítulo 7

Resultados y Conclusiones

Dados los resultados arrojados por el modelo podemos concluir que la estructura de la red convolucional 3D es una muy buena forma de poder clasificar imágenes de volúmenes médicos, sin embargo a pesar de la cantidad de datos con las que se entrenó no fue suficiente para considerarlo un buen modelo. En resumen, este modelo sirve para ayudar a un médico o especialista a diagnosticar TDAH, sin embargo, no es muy exacto ya que se entrenó con muy pocos datos a pesar de haber hecho *data augmentation*. Si se cuenta con imágenes de cortes axiales, sagitales o coronales se pueden usar para entrenar esta misma red y obtendrá un resultado en binario. Es decir, un **SI** con su respectivo porcentaje y un **NO** con el porcentaje restante, si un médico especialista cuenta con un volumen de datos obtenido de alguna resonancia magnética supongamos:

- Trastornos cerebrales.
- Lesiones deportivas.
- Problemas musculo-esqueléticos.
- Anormalidades vasculares.
- Problemas pélvicos femeninos.
- Problemas de próstata.

y los datos son de un tamaño similar, es decir, $128 \times 128 \times 64$ puede usar esta red neuronal para ayudarse a diagnosticar alguna enfermedad que pueda ser visible para la resonancia magnética y tal vez no para el ojo humano.

En esencia funciona de la siguiente forma: mientras más datos de casos positivos y negativos en el entrenamiento mejor será la predicción.

Este trabajo se puede retomar para mejorar el modelo si se aumenta el número de datos de entrenamiento o si se varían los hiperparpámetros como lo son capas, neuronas y orden de capas de convolución. Como se pudo ver en resultado de este programa, el problema se estudió desde el lado de la red neuronal, por lo que se requeriría perfeccionar el modelo. No obstante, el presente estudio orienta este enfoque de clasificación hacia una solución prometedora.

Apéndice A

Código fuente del programa

A.1. Librerías

Se usaron las siguientes librerías para este programa.

```
import datetime
import nibabel as nib
import numpy as np
import matplotlib.pyplot as plt
import nibabel as nib
import os
import pandas as pd
from scipy import ndimage
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import random
```

A.2. Nifti file

```
def read_nifti_file(filepath):
    """Lee y carga el volumen"""
    # Lee el archivo
    scan = nib.load(filepath)
    # Se obtienen los datos
    scan = scan.get_fdata()
    return scan
```

A.3. Normalizando

```
def normalize(volume):
    """Normaliza el volumen"""
    min = 0
    max = 255
    volume[volume < min] = min
    volume[volume > max] = max
    volume = (volume - min) / (max - min)
    volume = volume.astype("float32")
    return volume
```

A.4. Redimensionando

```
def resize_volume(img):
    """redimensiona la imagen atraves del ee z"""
    # ajusta las dimensiones a 64x128x128 ya que no
    # alcanzan a llenar los datos para ser de 256
    desired_depth = 64
    desired_width = 128
    desired_height = 128
    # obtiene la profundidad actual
    current_depth = img.shape[-1]
    current_width = img.shape[0]
    current_height = img.shape[1]
    # calcula el factor de profundidad
    depth = current_depth / desired_depth
    width = current_width / desired_width
    height = current_height / desired_height
    depth_factor = 1 / depth
    width_factor = 1 / width
    height_factor = 1 / height
    # Se rota
    img = ndimage.rotate(img, 90, reshape=False)
    # Redimensiona a través del eje z
    img = ndimage.zoom(img,(width_factor, height_factor, depth_factor), order=1)
    return img
```

A.5. Flujo de de tratamiento de imágenes

```
def process_scan(path):
    """lee y redimensiona el volumen"""
    # lee la imagen
    volume = read_nifti_file(path)
    # Normaliza
    volume = normalize(volume)
    # # Redimensiona alto, ancho y profundidad
    volume = resize_volume(volume)
    return volume
```

A.6. Leyendo datos

```
#Obtiene los datos de la carpeta de positivos que son los datos normales
normal_scan_paths = [
    os.path.join(os.getcwd(), "./ADHD_POS", x)
    for x in os.listdir("./ADHD_POS")
    if not x.startswith(".ipynb_checkpoints")
    # Filtrar el archivo .ipynb_checkpoints

]
#Obtiene los datos de la carpeta de negativos que son los datos no normales
abnormal_scan_paths = [
    os.path.join(os.getcwd(), "./ADHD_NEG", x)
    for x in os.listdir("./ADHD_NEG")
    if not x.startswith(".ipynb_checkpoints")
    # Filtrar el archivo .ipynb_checkpoints
]

print(f'La carpeta POS contiene : {str(len(normal_scan_paths))} archivos')
# print(normal_scan_paths)
print(f'La carpeta NEG contiene : {str(len(abnormal_scan_paths))} archivos')
# print(abnormal_scan_paths)
```

A.7. Etiquetado de datos

```
abnormal_scans = np.array([process_scan(path) for path in abnormal_scan_paths])
normal_scans = np.array([process_scan(path) for path in normal_scan_paths])

# asignamos 1 para los casos anormales y 0 para los casos normales
abnormal_labels = np.array([1 for _ in range(len(abnormal_scans))])
normal_labels = np.array([0 for _ in range(len(normal_scans))])

# separamos los datos en proporción 1:3 para entrenamiento.
x_train = np.concatenate((abnormal_scans[:66], normal_scans[:82]), axis=0)
y_train = np.concatenate((abnormal_labels[:66], normal_labels[:82]), axis=0)
x_val = np.concatenate((abnormal_scans[66:], normal_scans[82:]), axis=0)
y_val = np.concatenate((abnormal_labels[66:], normal_labels[82:]), axis=0)
print(
    "Número de muestras en entrenamiento y validación son %d y %d respectivamente."
    % (x_train.shape[0], x_val.shape[0])
)
```

A.8. Rotación

```
@tf.function
def rotate(volume):
    """Rotate the volume by a few degrees"""
    def scipy_rotate(volume):
        # define some rotation angles
        angles = [-20, -10, -5, 5, 10, 20]
        # pick angles at random
        angle = random.choice(angles)
        # rotate volume
        volume = ndimage.rotate(volume, angle, reshape=False)
        volume[volume < 0] = 0
        volume[volume > 1] = 1
        return volume
    augmented_volume = tf.numpy_function(scipy_rotate, [volume], tf.float32)
    return augmented_volume
def train_preprocessing(volume, label):
    """Process training data by rotating and adding a channel."""
    # Rotate volume
    volume = rotate(volume)
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
```

A.9. Datos de entrenamiento y validación

```
# Define data loaders.
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))

batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

A.10. Capa de dato 1

```
data = train_dataset.take(1)
print(data)
images, labels = list(data)[0]
images = images.numpy()
image = images[0]
print("Dimension of the RMI scan is:", image.shape)
plt.imshow(np.squeeze(image[:, :, image.shape[2]//2]), cmap="gray")
```

A.11. Graficando capas

```
def plot_slices(num_rows, num_columns, width, height, data):
    """Graficando 40 capas de las imagenes de RMI"""
    data = np.rot90(np.array(data))
    data = np.transpose(data)
    data = np.reshape(data, (num_rows, num_columns, width, height))
    rows_data, columns_data = data.shape[0], data.shape[1]
    heights = [slc[0].shape[0] for slc in data]
    widths = [slc.shape[1] for slc in data[0]]
    fig_width = 12.0
    fig_height = fig_width * sum(heights) / sum(widths)
    f, axarr = plt.subplots(
        rows_data,
        columns_data,
        figsize=(fig_width, fig_height),
        gridspec_kw={"height_ratios": heights},
    )
    for i in range(rows_data):
        for j in range(columns_data):
            axarr[i, j].imshow(data[i][j], cmap="gray")
            axarr[i, j].axis("off")
    plt.subplots_adjust(wspace=0, hspace=0, left=0, right=1, bottom=0, top=1)
    plt.show()
plot_slices(4, 10, 128, 128, image[:, :, 12:52])
```

A.12. Red Convolucional

```
def get_model(width=128, height=128, depth=64):
    """Build a 3D convolutional neural network model."""

    inputs = keras.Input((width, height, depth, 1))

    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=128, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=256, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=512, activation="relu")(x)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Dense(units=1, activation="sigmoid")(x)

    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model


# Build model.
model = get_model(width=128, height=128, depth=64)
model.summary()
```

A.13. Entrenamiento

```
# Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)
model.compile(
    loss="binary_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    metrics=["acc"],
)

# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    "3d_image_classification.h5", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)

# Train the model, doing validation at the end of each epoch
epochs = 10
model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=epochs,
    shuffle=True,
    verbose=2,
    callbacks=[checkpoint_cb, early_stopping_cb],
)
```

A.14. Graficando entrenamiento

```
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()

for i, metric in enumerate(["acc", "loss"]):
    ax[i].plot(model.history.history[metric])
    ax[i].plot(model.history.history["val_" + metric])
    ax[i].set_title("Model {}".format(metric))
    ax[i].set_xlabel("epochs")
    ax[i].set_ylabel(metric)
    ax[i].legend(["train", "val"])
```

A.15. Probabilidad de certeza del modelo

```
# Load best weights.
model.load_weights("3d_image_classification.h5")
prediction = model.predict(np.expand_dims(x_val[0], axis=0))[0]
scores = [1 - prediction[0], prediction[0]]

class_names = ["normal", "abnormal"]
for score, name in zip(scores, class_names):
    print(
        "Este modelo es %.2f porciento seguro de que es %s"
        % ((100 * score), name)
    )
```

Bibliografía

[1] Stuart J. Russell and Peter Norvig, Artifitial Intelligence A Modern Approach, Fourth Edition

[2] Mar Llamas-Velasco, Enrique Ovejero-Merino, Inteligencia artificial en el diagnóstico dermatopatológico, Piel, 2024, ISSN 0213-9251, https://doi.bibliotecabuap.elogim.com/10.1016/j.piel.2024.01.002.

[3] Aprende Inteligencia Artificial y Deep Learning con Python, Tensorflow y Keras, conviértete en experto en Deep Learning. Udemy, Santiago Hernandez

[4] Aprende Machine Learning y Data Science con Python, ¡conviértete en un experto en Machine Learning con Python! Udemy, Santiago Hernandez

[5] Dominando el Aprendizaje Profundo con Keras en Python: Desarrollo y Evaluación de Modelos Neuronales de 0 a experto. Udemy, PhD. Manuel Castillo-Cara

[6] Mehryar Mohri, Afshin Rostamizadeh, Ameed Talwalkar(2012) Foundations of machine Learning, The MIT Press ,7

[7] Mehryar Mohri, Afshin Rostamizadeh, Ameed Talwalkar(2012) Foundations of machine Learning, The MIT Press ,210

[8] Warren S. McCulloch, Walter Pitts, University of Chicago, Chicago, U.S.A. A logical calculus of the ideas immanent in nervous activity, University of Illinois, College of Medicine, Department of Psychiatry at the Illinois Neuropsychiatric Institute, 1943

[9] Lighthill, James (1973). .ªrtificial Intelligence: A General Survey". Artificial Intelligence: A paper symposium. UK: Science Research Council.

[10] Alex K. ,Ilya S, Walter Pitts, Geoffrey E. H., University of Toronto, ImageNet Classification with Deep Convolutional Neural Networks, University of Illinois

[11] Marvin Minsky, Seymour Papert, Massachusetts Institute of Technology, Perceptrons An introduction to Comptational Geometry, the MIT Press

[12] Banda, Hugo. (2014). Inteligencia Artificial: Principios y Aplicaciones.

[13] David E. Rumelhart, Geoffrey E. Hilton, Ronald J Williams, Institute for cognitive science university of California, San Diego, Learning internal representations by error propagation, September 1985

[14] https://www.cnblogs.com/pinard/p/6422831.html

[15] https://www.cnblogs.com/pinard/p/6489633.html

[16] https://www.cnblogs.com/pinard/p/6494810.html

[17] https://www.philips.com.mx/healthcare/solutions/magnetic-resonance

[18] Minhas, A.S., Oliver, R. (2022). Magnetic Resonance Imaging Basics. In: Sadleir, R., Minhas, A.S. (eds) Electrical Properties of Tissues. Advances in Experimental Medicine and Biology, vol 1380. Springer, Cham. https://doi.org/10.1007/978-3-031-03873-0_3

[19] Brown, R. W., Cheng, Y.-C. N., Haacke, E. M., Thompson, M. R., & Venkatesan, R. (2014).

[20] Cerebro IRM de cortes axiales : anatomía normal | e-Anatomy https://www.imaios.com/es/e-anatomy/cerebro/irm-axial-cerebral

[21] Canal oficial de YouTube de la II Cátedra de Diagnóstico por Imágenes y Terapia Radiante de la Facultad de Ciencias Médicas de la Universidad Nacional de Córdoba en Argentina.

[22] MLA, 9.ª edición (Modern Language Assoc.) Ray H. Hashemi, et al. MRI: The Basics. Wolters Kluwer Health, 2018. APA, 7.ª edición (American Psychological Assoc.)Ray H. Hashemi, Christopher J. Lisanti, & William Bradley. (2018). MRI: The Basics: Vol. Fourth edition. Wolters Kluwer Health.

[23] James Binney and David skinner, The physics of Quantum Mechanics,Oxford University press 2014 180-188

[24] https://medlineplus.gov/spanish/attentiondeficithyperactivitydisorder.html

[25] Hasib Zunair. (2020). 3D image classification from CT scans. https://keras.io/examples/vision/3D_image_classification/