



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM VALLE DE MÉXICO

**Aprendizaje incremental para la tarea de reconocimiento
de dígitos con Redes Neuronales Artificiales**

TRABAJO DE TESIS

Que para obtener el Título de

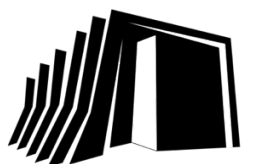
INGENIERO EN SISTEMAS Y COMUNICACIONES

P r e s e n t a

C. González Hernández Luis Ángel

Asesor: Dr. Víctor Manuel Landassuri Moreno

Atizapán de Zaragoza, Edo. de Méx. Septiembre 2023



Centro Universitario
UAEM Valle de México

Resumen

El aprendizaje incremental es un área de la Inteligencia Artificial la cual permite agregar nuevo conocimiento a un modelo (e.g. Redes Neuronales Artificiales) sin la necesidad de entrenar el modelo con toda la información histórica de la tarea en cuestión [1]. En el presente trabajo de investigación se ocupará el modelo de Redes Neuronales Artificiales enfocada en la clasificación de dígitos escritos a mano usando el algoritmo de entrenamiento de backpropagation, con redes Multi capa Perceptron y duplicación de pesos múltiples simulando memoria a corto y largo plazo para mejorar los resultados presentados en [1].

Índice general

| | |
|---|-----------|
| 1. Introducción | 2 |
| 2. Planeamiento del problema | 4 |
| 3. Objetivos | 7 |
| 3.1. Objetivos específicos | 7 |
| 4. Preguntas de investigación | 8 |
| 5. Justificación | 9 |
| 6. Delimitación | 12 |
| 7. Revisión de la literatura | 13 |
| 7.1. Redes neuronales artificiales | 13 |
| 7.1.1. Función de activación | 14 |
| 7.1.2. Redes neuronales de perceptrón multicapa | 15 |
| 7.1.3. Algoritmo backpropagation | 16 |
| 7.2. Aprendizaje incremental | 17 |
| 7.2.1. Algoritmos de aprendizaje incremental | 18 |
| 8. Metodología | 21 |
| 9. Resultados | 22 |
| 10. Organización del capitulado | 28 |

Índice de figuras

| | |
|--|----|
| 7.1. Red Neuronal Artificial Básica | 13 |
| 7.2. Función Escalonada | 15 |
| 7.3. Perceptron Multicapa | 16 |
| 7.4. Dos enfoques tradicionales del aprendizaje incremental. | 18 |
| 9.1. Primeros Resultados | 22 |
| 9.2. Precisión por época | 23 |
| 9.3. Perdida por época | 24 |
| 9.4. Evaluación en la precisión | 24 |
| 9.5. Segundos Resultados | 25 |
| 9.6. Precisión de época con pesos duplicados | 26 |
| 9.7. Perdida de época | 26 |
| 9.8. Evaluación en la precisión | 27 |

Capítulo 1

Introducción

La inteligencia artificial (IA) es una área del conocimiento que se enfoca en poder hacer máquinas que se enfocan en comportamiento y razonamiento humano, para que en momento dado, se pueda interactuar con una máquina. Así mismo, también es posible pensar que mucho del desarrollo en el área de inteligencia artificial, es el poder tener mejores herramientas que ayuden a las actividades diarias.

En este sentido, un área de la IA es el llamado Aprendizaje Máquina, donde se estudian algoritmos que permiten aprender de forma automática una tarea. Así, una de las técnicas más conocidas en la actualidad, dentro del área de IA son las Redes Neuronales Artificiales (RNAs), siendo técnicas que realizan procesos matemáticos para poder aprenderse tareas a resolver. Algunas áreas en las que son útiles las RNAs son en el aprendizaje de tareas no lineales, como la predicción de la capacidad de la red 5G, basada en el tráfico diario de este [2] o clasificación, por ejemplo la clasificación de metales y rocas por medio de RNAs y lógica difusa [3].

Las RNAs están formadas por neuronas artificiales que simulan a las biológicas. Así los procesos químicos que suceden en el cerebro, se simulan computacionalmente a través de señales que viajen a través de las neuronas artificiales, de aquí en adelante simplemente se referirá a ellas como "neuronas". Las neuronas en una RNA cuentan con una

estructura distribuida en paralelo, presentando una buena habilidad de aprendizaje [4].

Dentro del aprendizaje máquina cuando una técnica, por ejemplo RNA, se enfoca en aprender una tarea, se conoce como *algoritmo lineal sin memoria*, siendo uno de los métodos más empleados desde el inicio de las RNAs [5]. Sin embargo, si es necesario incorporar nueva información del problema, es necesario volver a entrenar todo el modelo, considerando toda la información existente, esto es, la anterior y la nueva que acaba de llegar, es ahí donde nace el concepto de Aprendizaje Incremental, siendo un área enfocada en poder incorporar información del problema en cuestión, sin tener que volver a re-entrenar todo el modelo.

Derivado del aprendizaje incremental se desprende el concepto de memoria dentro de la IA, analizando como un algoritmo de aprendizaje máquina puede olvidar la información que se usó en un entrenamiento previo al entrenar con información más reciente. Si se hace la analogía con los humanos, la memoria es un factor importante para estudiar considerando la pérdida de información aprendida, así, este es un problema biológico, el cual tanto afecta a los humanos como a las máquinas. Por ello, se han elaborado distintos experimentos para poder combatir esta problemática. Uno de estos es el caso de [1], el cual propone el manejo de RNAs con pesos dobles, donde la primer capa de pesos esta enfocada a comportarse como memoria a corto plazo, y la segunda como memoria a largo plazo. Los experimentos mostrados en [1] permiten notar un mejora en tareas de aprendizaje incremental, teniendo menos pérdida de información en comparación de implementaciones anteriores como el algoritmo Learn++ [6, 7].

anteriores como el algoritmo Learn++ [6, 7].

Así, el presente trabajo de investigación esta enfocado en poder explorar nuevas configuraciones. Así, el presente trabajo de investigación esta enfocado en poder explorar nuevas configuraciones de pesos duplicados para poder extender el trabajo previamente presentado en [1]. de pesos duplicados para poder extender el trabajo previamente presentado en [1].

Capítulo 2

Planeamiento del problema

Las Redes Neuronales Artificiales tienen la habilidad de poderse aprender una tarea y poder hacer o resolver tareas de predicción o clasificación básicamente. En este sentido, con algoritmos de aprendizaje como el Backpropagation, permite ajustar los pesos de una RNA para que esta pueda empezar a resolver una tarea particular. No obstante, muchas de las tareas que se resuelven en la vida diaria, van generando mas información con el tiempo, por ejemplo, el comportamiento de una serie financiera, o bien la predicción del clima en una determinada región. Así, se tiene el aprendizaje incremental, siendo un método poco explorado enfocado en poder aprender nueva información del problema, sin tener que volver a entrenar todo el modelo con la información anterior y la nueva que acaba de llegar, esto es, en los modelos actuales de aprendizaje máquina, si se usa un conjunto de datos para entrenar un modelo en específico, dicho modelo es funcional para dicho conjunto de datos y la información que ello representa. Sin embargo, si es necesario incorporar nueva información al modelo, es necesario recolectar dicha información nueva, agregarla a la que se tenía anteriormente, y volver a entrenar todo el modelo para que éste pueda incorporar la nueva información.

De esta forma, una red ya entrenada con un primer conjunto de datos d_1 se desea entrenar con un nuevo conjunto de datos del mismo problema d_2 , al entrenar la red con d_2 se perderá el conocimiento aprendido por d_1 . Si no se desea utilizar un modelo de

aprendizaje incremental, se tendría que juntar el conjunto d_1 y d_2 en un solo conjunto y volver a entrenar la RNA para así poder incorporar el nuevo conocimiento (d_2) a la RNA. Si se desea utilizar el método de aprendizaje incremental, se puede entrenar en un primer momento a la red con el conjunto d_1 , posteriormente con d_2 teniendo poca pérdida de información de d_1 . Si más adelante llega mas información del problema (d_3) que se desee incorporar a la base de conocimientos de la RNA, entonces solo habrá que entrenar la RNA con d_3 usando el modelo incremental para tener una pérdida mínima de información de d_1 y d_2 .

De esta forma, el presente trabajo tomará como base la investigación de [1], en donde utiliza una configuración de pesos dobles (a una RNA se duplican todos sus pesos), donde a una capa de pesos duplicados se enlaza con una tasa de aprendizaje alta, para simular un rápido aprendizaje, y por ende simular lo que sería memoria a corto plazo. En su contraparte, la segunda capa de pesos duplicados, se enlaza con una tasa de aprendizaje baja para aprender lentamente un problema, simulando la memoria a largo plazo. Es decir, al momento de aprender una tarea nueva, una capa de pesos aprenderá muy rápido la nueva tarea (tasa alta de aprendizaje) y por ende olvidará más rápidamente los datos ya aprendidos anteriormente, por otro lado, la segunda capa de pesos duplicados, aprenderá muy lentamente los nuevos datos que llegan, y por ende olvidando poco la información que anteriormente se aprendió. Considerando ello y también que la RNA trabaja en conjunto con ambas capas de pesos duplicados, se pondera la salida de la RNA para que pueda contemplar la información nueva que se acaba de agregar a la RNA en ambas capas, así como la información que se acaba de olvidar de ambas capas de pesos.

El problema principal del aprendizaje incremental mostrado en [1], es que entre más conjuntos de datos nuevos que lleguen, mas se olvidaran los primeros conjuntos que se aprendieron, lo cual no es tan útil si se contempla que en el futuro de una RNA podrán existir 10 o 20 etapas de entrenamiento incremental con nuevos conjuntos de datos que

se vayan recolectando.

Por ello es indispensable poder explorar nuevas configuraciones de RNAs que permitan mejorar los métodos actuales para permitir una menor cantidad de olvido conforme llegue nueva información al modelo. Donde al igual que en trabajos anteriores, el presente trabajo se basará en conceptos de memoria a corto y largo plazo, y en lugar de hacer una copia de los pesos actuales y tener dos tasas de aprendizaje, una rápida para simular la memoria a corto plazo, y otra tasa de aprendizaje para simular la memoria a largo plazo, se explorará por hacer mas copias de los pesos, teniendo más tasas de aprendizaje que operen en cada una de dichas copias.

Capítulo 3

Objetivos

Diseñar una red neuronal artificial para aprendizaje incremental basada en el principio de la memoria a corto y largo plazo, buscando usar más de dos capas de pesos duplicados para el reconocimiento de dígitos, y con una menor pérdida de información de trabajos previos.

3.1. Objetivos específicos

1. Implementar el algoritmo mostrado en [1] para el reconocimiento de dígitos con aprendizaje a corto y largo plazo con los parámetros que ahí se indican.
2. Obtener el conjunto de datos de Optical Digits, limpiar los datos y prepararlos según lo indicado con [1].
3. Separar el conjunto de entrenamiento y de prueba de acuerdo a lo que se explica en el artículo de Bullinaria y probar el primer código implementado en miras de comprobar los resultados previamente mostrados en [1].
4. Tomar como base el algoritmo implementado, y extenderlo para permitir mas de dos pesos duplicados, aplicando el conjunto de datos previamente mostrado.
5. Comparar ambas implementaciones en busca de una reducción significativa de las tasas de aprendizaje con respecto a trabajos previos en la literatura.

Capítulo 4

Preguntas de investigación

1. ¿Existen trabajos más recientes sobre aprendizaje incremental?
2. La memoria a corto y largo plazo, representa dos extremos, pero ¿será posible subdividir más esa clasificación, para que más puntos intermedios puedan mejorar la memoria a corto y largo plazo, al mismo tiempo que el rendimiento en el aprendizaje incremental?
3. ¿El uso del aprendizaje incremental va a hacer que el tiempo de entrenamiento sea más rápido o al contrario?

Capítulo 5

Justificación

Las redes neuronales permiten el aprendizaje automático y la resolución de distintos problemas, pero como se comentó anteriormente, las técnicas de aprendizaje máquina, tienen una deficiencia que es al momento de aumentar los nuevos bloques de datos que llegan para aprender, se obtiene un deterioro en el rendimiento de aprendizaje de información y olvido de la información anterior [1].

Los resultados que se han obtenido no han funcionado a la perfección, la memoria a corto plazo olvida poco pero va olvidando, y lo ideal sería que no olvidara. Biológicamente, los humanos pueden aprender nuevas tareas, o información nueva de un problema, y no olvida de forma significativa lo que anteriormente aprendió, no obstante, eso no pasa actualmente con las RNA y en general con cualquier algoritmo de aprendizaje máquina. En otro sentido, los humanos ya tienen cierta configuración en el cerebro que les permite aprender como se hace actualmente, y se puede afirmar que por el momento no hay ningún procedimiento (ya sea quirúrgico o no) que permita modificar la estructura del cerebro para aprender mas y olvidar menos.

No obstante, computacionalmente nada puede impedir que se experimente con más configuraciones y llegar al punto en donde toda la información que ingrese a un modelo (por ejemplo RNAs) se acumule, y si no hay problema de almacenamiento, que ésta se

siga acumulando y que no olvide, esto podría ser bueno en diversas situaciones.

Desde un punto de vista computacional, si llega nueva información y no se ocupa aprendizaje incremental, esto implicará volver a entrenar todo el sistema con la información anterior y la actual (por ejemplo, d_1 y d_2) y considerando que una de las desventajas que tienen la RNAs es que el entrenamiento es un cuello de botella, siendo este donde se lleva la mayor parte de cómputo y por consiguiente de energía. Lo mencionado anteriormente implica que volver a entrenar con toda la información acumulada, gastará más energía y tiempo, que si solo se entrena con la nueva información que llega al modelo. En su contraparte, existe una gran variedad de herramientas, las cuales permiten codificar una red neuronal artificial con librerías ya preexistentes, para esta investigación solo se expondrán 2 empresas, siendo estas las más importantes: Microsoft y Google. La primera cuenta con la plataforma de Azure que renta una maquina virtual donde se puede programar en Python, y la segunda, cuenta con Google Colab que de igual forma brinda una máquina virtual para realizar experimentos de Maching Learning, la única diferencia contra Azure es que dicha herramienta es gratuita, y una similitud que tienen es que en ambas herramientas se puede programar en el mismo lenguaje.

Como se puede observar, las herramientas mencionadas permiten la programación en Pyhton, y esto se debe a que dicho lenguaje es una herramienta de software libre que no requiere licencia, es relativamente fácil poder depurar un código y permite acelerar más el desarrollo de aplicaciones, a diferencia de otros lenguajes más estructurados como C o Java, además tiene más librerías para el desarrollo de Maching Learning, por ejemplo, TensorFlow, Numpi, entre otras.

TensorFlow es una librería de Python que permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos, en la presente investigación se usará dado a que favorece la creación de una RNA, permite la elaboración de cualquier tipo de algoritmo de Machine Learning, cabe mencionar que también se

puede usar para Deep Learning, facilita la adquisición de datos modelos de capacitación, predicciones y refinamiento de resultados, está disponible para el uso en computadores personales, pero es recomendado usarlo en su propio editor en la nube que es Colab.

Keras es un framework de alto nivel para el aprendizaje, escrito en Python y capaz de correr sobre los frameworks TensorFlow, por esta razón se usará para la presente investigación, pues facilita los procesos de experimentación rápida, y ya que al ejecutarse en TensorFlow, por consiguiente se puede ejecutar en Colab.

Capítulo 6

Delimitación

En la presente investigación solamente se utilizará redes neuronales del tipo perceptrón multicapa, donde cabe mencionar que este no es el único tipo de red que existe, i.e., también se tiene Redes Neuronales Convolucionales (RNC), Redes Neuronales Recurrentes (RNR) o bien Redes de Base Radial (RBR) [8]. Así mismo, no se abordará el uso de técnicas de optimización como lo son los algoritmos genéticos, y únicamente se limitará a explorar la mejora en rendimiento al tener más de dos capas duplicadas de pesos en la red con aprendizaje incremental. Así mismo, no se abordarán modelos como las redes profundas u otro conjunto de datos y se limitará el trabajo a lo antes mencionado.

Capítulo 7

Revisión de la literatura

7.1. Redes neuronales artificiales

Las redes neuronales artificiales (RNA) son modelos computacionales de la Inteligencia Artificial los cuales contienen simples unidades de procesamiento llamadas neuronas. Ellas se inspiran en el cerebro humano, tomando como base la conectividad entre neuronas y el aprendizaje que pueden tener. Un perceptron o neurona (artificial) solamente resuelve problemas lineales y tiene la siguiente forma:

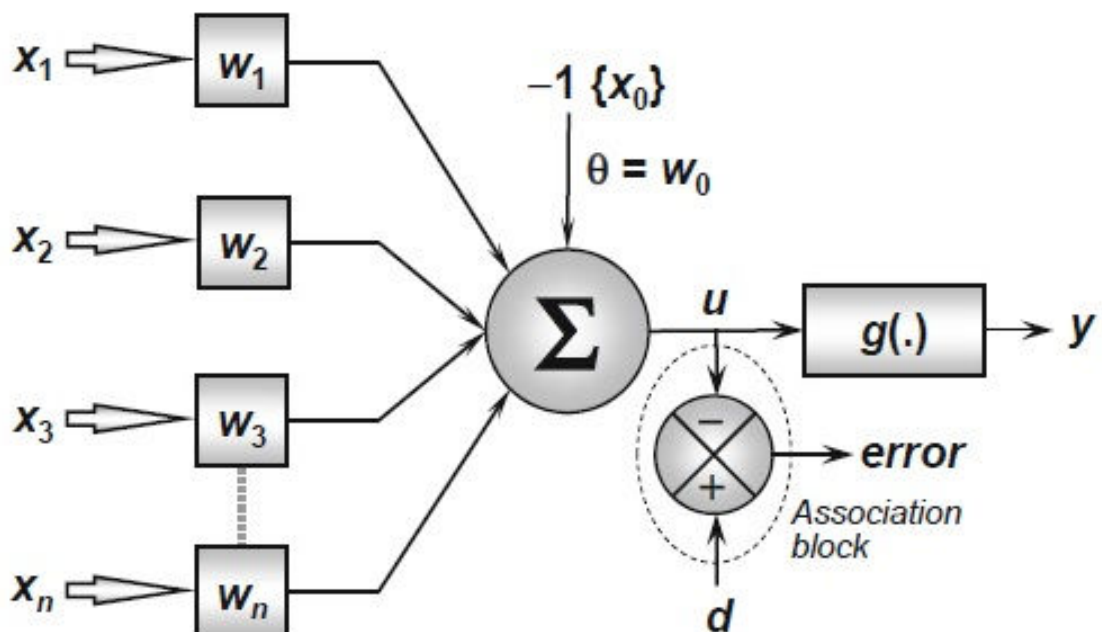


Figura 7.1: Red Neuronal Artificial Básica

Donde Σ es la representación matemática de la neurona., x_1, x_2, \dots, x_n son las variables de entrada a la red. w_1, w_2, \dots, w_n son los pesos con los cuales se van a ponderar las entradas, es decir multiplicar cuando la información entra en la neurona. Posterior a multiplicar el peso por la entrada correspondiente, se suman todos esos valores $w_1x_1 + w_2x_2 + w_3x_3$.

Al revisar esta formula, se puede observar que se parece a la operación de una regresión la cual es: $y = w_0 + w_ix_i$, de esta forma, internamente la neurona realiza una regresión lineal. En su contraparte, el parámetro que permite a la neurona trazar una recta cruzando el eje y en el plano cartesiano (eje de las ordenadas), a ello se conoce como sesgo (del inglés *bias*), este valor se agrega a la conexión, el cual usualmente se le da un valor de 1. Agregando este nuevo valor a la fórmula, queda de la siguiente manera: $y = \Sigma w_ix_i + w_0b$, donde b es el sesgo.

Un inconveniente del uso de una sola neurona para experimentos es que solo va a resolver ejercicios parecidos a la puerta lógica AND u OR.

Existen problemáticas de solo usar una sola neurona e.g problemas de tipo compuerta XOR.

Para solucionarlo se usan dos o más neuronas, además de la función de activación, que es la que permite pasar la información de una neurona a otra, en un rango especificado, y la cual se describirá en la siguiente sección.

7.1.1. Función de activación

Dicho método se utiliza cuando el modelo de RNA contiene dos o más neuronas. Esta función lo que provoca es dar al modelo una salida no lineal, para eso la segunda fórmula presentada es distorsionada para quedar de la siguiente manera: $f(w_1x_1 + w_2x_2 + w_3x_3 + b_0)$ para el caso de 3 entradas.

Al hablar de funciones de activación se deben de comentar las más comunes, como lo es la función escalonada.

Tenemos la función escalonada, la cual se representa con la siguiente formula:

$$f(x) = \begin{cases} 0 & : x < 0 \\ 1 & : x \geq 0 \end{cases}$$

Su representación grafica queda de la siguiente manera:

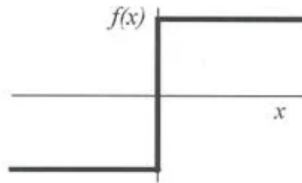


Figura 7.2: Función Escalonada

Las redes neuronales presentan demasiadas utilidades las cuales ayudan a resolver problemas como se muestra en el siguiente [4]: no linealidad, mapeo entrada-salida, aprendizaje robusto a errores en los datos de entrenamiento, entre otros. Existen varios tipos de Redes Neuronales tales como: Redes Neuronales de Perceptrón Multicapa, Redes Neuronales Convolucionales, entre otras, las cuales se describirán brevemente más adelante.

7.1.2. Redes neuronales de perceptrón multicapa

Las Redes Neuronales de Perceptrón Multicapa se pueden dividir en dos capas (las de entrada y salida), pero también en tres o más capas (la de entrada, una o más capas ocultas y la de salida). En las capas ocultas se pueden tener más de una fila de neuronas, las cuales son las encargadas de realizar las operaciones para eliminar la linealidad de los datos. También, como se comentó anteriormente, Sec. (7.1.1) la linealidad de los datos se elimina con las funciones de activación, las cuales modifican los parámetros de la red, permitiendo que se elabore un plano tridimensional, con el cual se puede encontrar la solución al problema planteado.

Además como se explicó anteriormente, no es muy recomendado trabajar con una sola neurona por los problemas presentados al resolver, tareas como el XOR donde se

requieren de dos líneas rectas para clasificar el problema correctamente.

Como se puede observar en la figura (7.3), para este caso se cuenta con una MLP que consta de 4 capas, 1 de entrada, 2 ocultas y 1 de salida. En las capas ocultas y de salida se lleva a cabo el procesamiento de las funciones de activación, donde cabe notar que no es así para la primera capa de entrada, donde únicamente sirve para representar las entradas al modelo, i.e. ahí no hay funciones de transferencia.

Así, las neuronas de color azul cuentan con una función (puede ser sigmoideal, escalonada, entre otras), y cuando se llegue a la capa de salida, cada función se va a sumar, de esta manera se obtendrá una función no lineal que de resolución a la tarea a resolver.

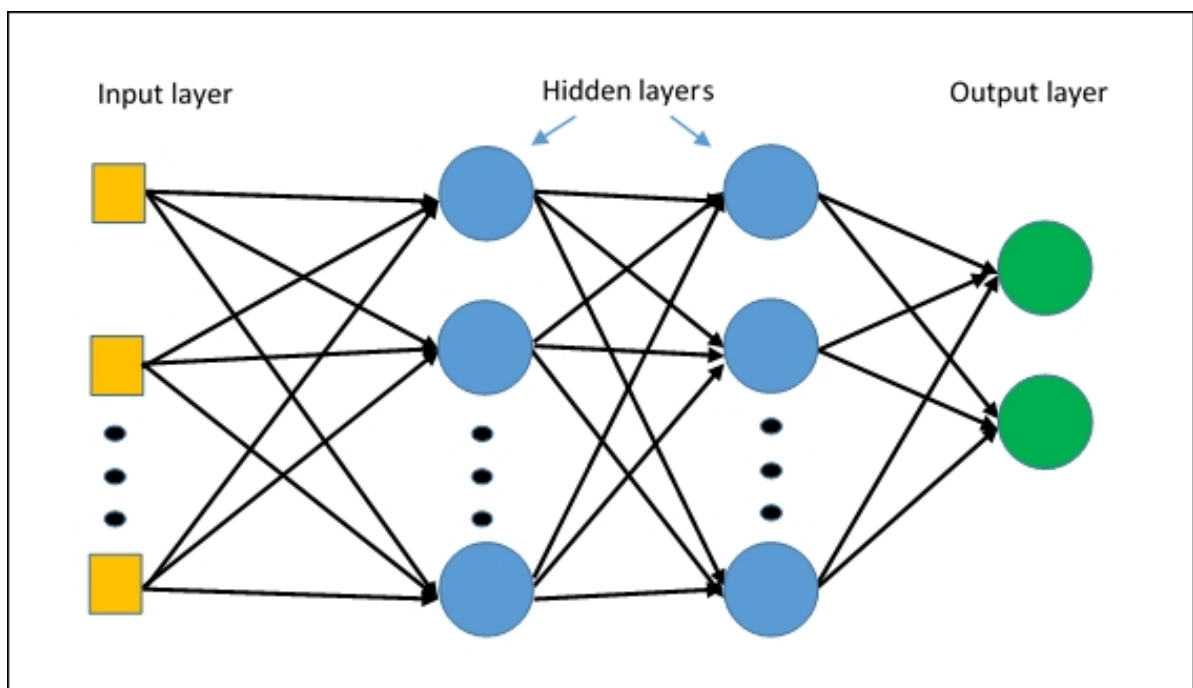


Figura 7.3: Perceptron Multicapa

7.1.3. Algoritmo backpropagation

El backpropagation es un algoritmo de aprendizaje que permite que una red neuronal pueda auto-ajustar todos sus parámetros para aprender una representación interna de la información que se está procesando. Llegó a dar solución a la limitante del perceptron, este resuelve los problemas lineales, la cual es que no se puede extender a redes más complejas, es decir, a problemas no lineales.

Usando este algoritmo se podrán obtener las derivadas parciales del gradiente y del peso, las cuales sirven para la optimización de la red neuronal.

Pero también se deben de calcular las derivadas del sesgo, donde se encuentra el número de capa donde esta la falla.

El uso de estas derivadas parciales permite encontrar el error, en otras palabras, lo que realiza dicho algoritmo es terminar un proceso y si se encuentra un error, este va a regresar hasta la neurona donde se encuentra este, pero va regresando desde la capa de salida hacia la primer capa oculta, este proceso se va a repetir hasta encontrar el error perfecto, el cual es donde el error disminuye a lo más bajo y el resultado de la red es lo más acertado.

Con lo anterior expuesto, se puede decir que esta metodología es muy útil en el uso de redes neuronales, es por eso que se usa en la investigación [1], para obtener un buen resultado en el aprendizaje incremental, como es explicado ahí es usado para que las redes obtengan una buena topología con buena actualización de pesos.

7.2. Aprendizaje incremental

Con el pasar de los años la tecnología a evolucionado, eso quiere decir que el Aprendizaje Automático se ha actualizado y que la cantidad de datos va aumentado con más frecuencia.

Se puede verificar como *Una tarea de aprendizaje es incremental si los ejemplos de entrenamiento usados para resolverla están disponibles en horas extras, generalmente uno a la vez* [5], si los resultados no se necesitan de manera urgente, este tipo de trabajos serán resueltos por algoritmos de aprendizaje no incremental.

Una área donde esto es de mucha utilidad es la *Robótica* porque este necesita estar en constante entrenamiento [5].

Dicha forma de aprender fue inspirada en la forma en que el humano aprende y esta más rápida, fue por esto que fue adoptada por el aprendizaje máquina.

Con el paso del tiempo se ha convertido en un paradigma del aprendizaje automático, aquí el aprendizaje toma el lugar de nuevos ejemplos para juntarlos y conforme van aprendiendo estos toman el lugar de los ejemplos ya aprendidos [4].

7.2.1. Algoritmos de aprendizaje incremental

El algoritmo de aprendizaje incremental puede definirse como aquel que cumple los siguientes criterios: 1) Ser capaz de aprender y actualizarse con cada nuevo dato etiquetado o no etiquetado. 2) Conservar los conocimientos adquiridos previamente. 3) No debe requerir el acceso a los datos originales. 4) Generar una nueva clase o cluster cuando sea necesario. Dividir o fusionar los clusters cuando sea necesario. 5). Ser de naturaleza dinámica con el entorno cambiante [9].

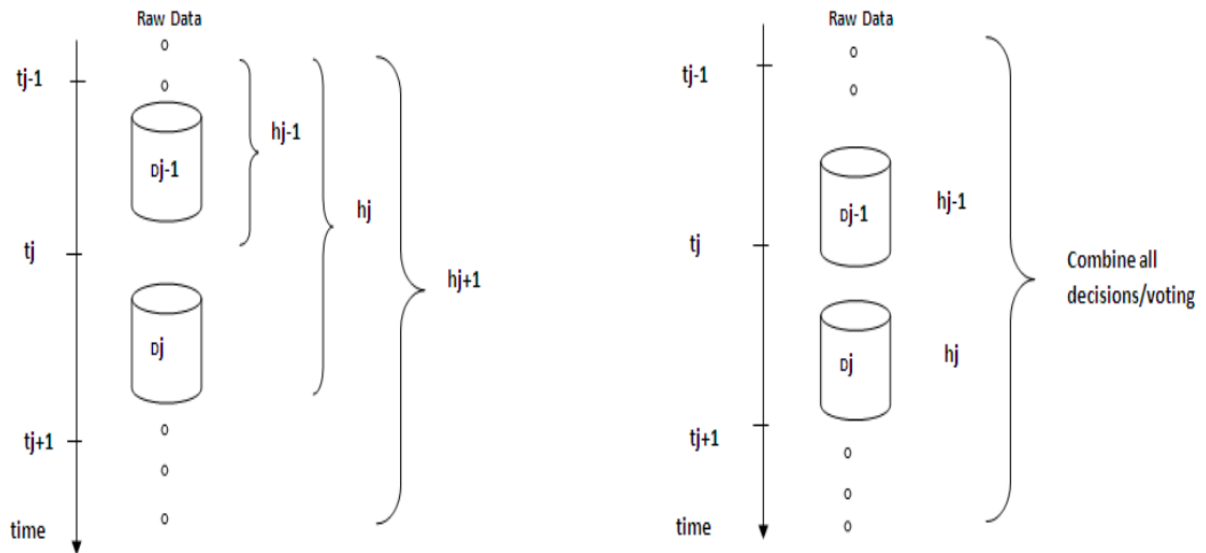


Figura 7.4: Dos enfoques tradicionales del aprendizaje incremental.

1 Metodología de acumulación de datos. 2 Metodología de aprendizaje por conjuntos.

Como se observa en la Figura 11, en el primer método, cuando se recibe una nueva porción de datos D_j , se descarta h_{j-1} y se desarrolla una nueva hipótesis h_j , basada en todos los datos disponibles acumulados hasta el momento. Y en el segundo método, cuando se recibe una nueva porción de datos D_j , se desarrolla una única hipótesis nueva o un conjunto de hipótesis nuevas basadas en los nuevos datos. Por último, se puede

utilizar un mecanismo de votación para combinar todas las decisiones de las diferentes hipótesis y obtener la predicción final.

Por ejemplo, si se deja que D_{j-1} represente la porción de datos recibida entre el tiempo t_{j-1} y t_j , y que la hipótesis h_{j-1} se desarrolle sobre D_{j-1} .

El sistema aprenderá información de forma adaptativa cuando se reciba una nueva porción de datos D_j . En el método de aprendizaje por conjuntos, se desarrolla una nueva hipótesis h_j o un conjunto de hipótesis $H: h_1, h_2, \dots, h_M$, basadas en los nuevos datos. A continuación, se utiliza el mecanismo de votación para combinar todas las decisiones de las diferentes hipótesis y llegar a la predicción final. La mayor ventaja de este enfoque es que no se requiere almacenar los datos vistos anteriormente, el conocimiento se ha almacenado en la serie de hipótesis desarrolladas a lo largo de la vida de aprendizaje.

Conocimiento en el momento t :

D_t es un trozo de datos con n instancias ($i=1, \dots, n$)

(x_i, y_i) es una instancia en el espacio de características m -dimensional X

$Y_i \in Y = 1, \dots, K$ clases

Función de distribución D_f

Una hipótesis h_t , desarrollada por los datos basados en D_t con P_t

La nueva entrada estará disponible en el momento $(t+1)$

Algoritmo de aprendizaje:

1. Encontrar la relación entre D_t y D_{t+1}
2. Actualizar la función de distribución inicial D_{t+1}
3. Aplicar la hipótesis h_t a D_{t+1} y calcular el pseudoerror de h_t
4. Refinar la función de distribución para D_{t+1}

5. Se desarrolla una hipótesis por los datos basados en D_{t+1} con P_{t+1}
6. Repetir el procedimiento cuando se reciba la siguiente porción del nuevo conjunto de datos.

Resultado: La hipótesis final.

Un algoritmo de aprendizaje es incremental si, para cualquier muestra de entrenamiento dada:

$$e_1, \dots, e_s$$

produce una secuencia de hipótesis

$$h_0, h_1, \dots, h_n$$

tal que

$$h_{i+1}$$

depende solo de

$$h_i$$

y del ejemplo actual e [5], como se observa, estos son algoritmos que permiten a la inteligencia artificial poder realizar actividades de predicción de una manera más eficaz.

Un ejemplo del uso de esta rama es el proyecto *COBWEB*, donde se trata de categorizar el número de Clúster y la pertenencia de dichas categorías por medio de una métrica probabilística global, esto lo realiza por medio de que se agrega una nueva categoría, este proceso lo que realizará es actualizar todas las probabilísticas con los nuevos datos recabados [10].

Capítulo 8

Metodología

El primer paso a realizar, es recrear el código mostrado en [1], el describe la implementación de una red neuronal multicapa usando el algoritmo de entrenamiento back-propagation en el lenguaje de programación Python. Así mismo, se utilizará el conjunto de datos de Optical Digits, en donde se tendrá que preprocesar los datos, para eliminar registros inválidos.

Posteriormente, se implementará una extensión del código, donde se experimentará con mas de dos capas de pesos duplicados para mejorar la tasa de olvido de información al momento de usar el aprendizaje incremental. Para ello se explorará incrementando gradualmente el número de capas de pesos duplicados.

Finalmente, cuando los resultados se obtengan se realizará una comparación, de los resultado del algoritmo base con los resultados del algoritmo extendido.

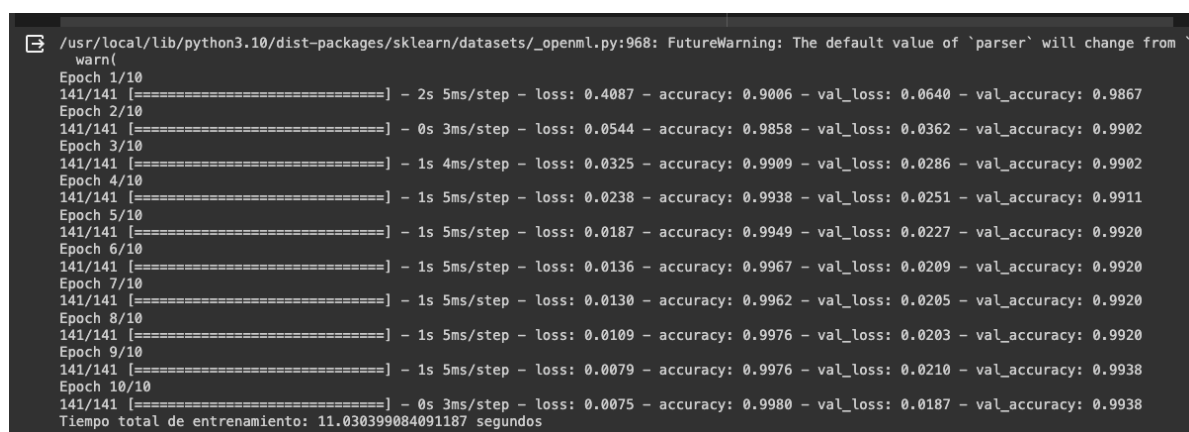
Capítulo 9

Resultados

En esta investigación, se expondrán dos redes neuronales, diferenciadas por el hecho de que en una de ellas se duplicaron los pesos. Este ajuste se llevó a cabo con el propósito de evaluar en qué medida el aprendizaje incremental impacta en el rendimiento de una red neuronal.

Para la construcción de estas redes, se emplearon datos del conjunto de datos Optical Digit, el cual proporciona la información necesaria para el entrenamiento de la red neuronal. Todo el proceso se llevó a cabo en el entorno virtual de Google Colab utilizando la biblioteca Tensorflow.

A continuación, se presentarán los resultados obtenidos de la red neuronal que no experimentó la duplicación de pesos. Esta red puede ser descrita como simple, al menos en comparación con su contraparte modificada.



```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from `
warn(
Epoch 1/10
141/141 [=====] - 2s 5ms/step - loss: 0.4087 - accuracy: 0.9006 - val_loss: 0.0640 - val_accuracy: 0.9867
Epoch 2/10
141/141 [=====] - 0s 3ms/step - loss: 0.0544 - accuracy: 0.9858 - val_loss: 0.0362 - val_accuracy: 0.9902
Epoch 3/10
141/141 [=====] - 1s 4ms/step - loss: 0.0325 - accuracy: 0.9909 - val_loss: 0.0286 - val_accuracy: 0.9902
Epoch 4/10
141/141 [=====] - 1s 5ms/step - loss: 0.0238 - accuracy: 0.9938 - val_loss: 0.0251 - val_accuracy: 0.9911
Epoch 5/10
141/141 [=====] - 1s 5ms/step - loss: 0.0187 - accuracy: 0.9949 - val_loss: 0.0227 - val_accuracy: 0.9920
Epoch 6/10
141/141 [=====] - 1s 5ms/step - loss: 0.0136 - accuracy: 0.9967 - val_loss: 0.0209 - val_accuracy: 0.9920
Epoch 7/10
141/141 [=====] - 1s 5ms/step - loss: 0.0130 - accuracy: 0.9962 - val_loss: 0.0205 - val_accuracy: 0.9920
Epoch 8/10
141/141 [=====] - 1s 5ms/step - loss: 0.0109 - accuracy: 0.9976 - val_loss: 0.0203 - val_accuracy: 0.9920
Epoch 9/10
141/141 [=====] - 1s 5ms/step - loss: 0.0079 - accuracy: 0.9976 - val_loss: 0.0210 - val_accuracy: 0.9938
Epoch 10/10
141/141 [=====] - 0s 3ms/step - loss: 0.0075 - accuracy: 0.9980 - val_loss: 0.0187 - val_accuracy: 0.9938
Tiempo total de entrenamiento: 11.030399084091187 segundos
```

Figura 9.1: Primeros Resultados

Tal como se puede apreciar en la imagen 9.1 previa, se evidencia que el tiempo de ejecución registrado fue de 11 segundos. Esta métrica temporal proporciona una indicación clara del rendimiento y eficiencia del proceso, siendo un factor relevante para evaluar la eficacia de la tarea realizada. La breve duración del tiempo de ejecución sugiere una ejecución rápida y eficiente de la operación en cuestión, lo cual puede ser un aspecto positivo en términos de eficacia y velocidad de respuesta del sistema o del algoritmo implementado.

A continuación, se procederá a examinar la precisión por época, la pérdida por época y cómo evoluciona su precisión a lo largo de las iteraciones. Este análisis proporcionará una visión detallada de la dinámica del modelo a medida que avanza en el proceso de entrenamiento. Observar la relación entre estas métricas a lo largo de las iteraciones permitirá entender mejor la capacidad de aprendizaje y la convergencia del modelo, ofreciendo insights cruciales sobre su desempeño en el conjunto de datos. Este enfoque detallado en las métricas temporales y su evolución es esencial para una evaluación integral del rendimiento del modelo a lo largo del tiempo.

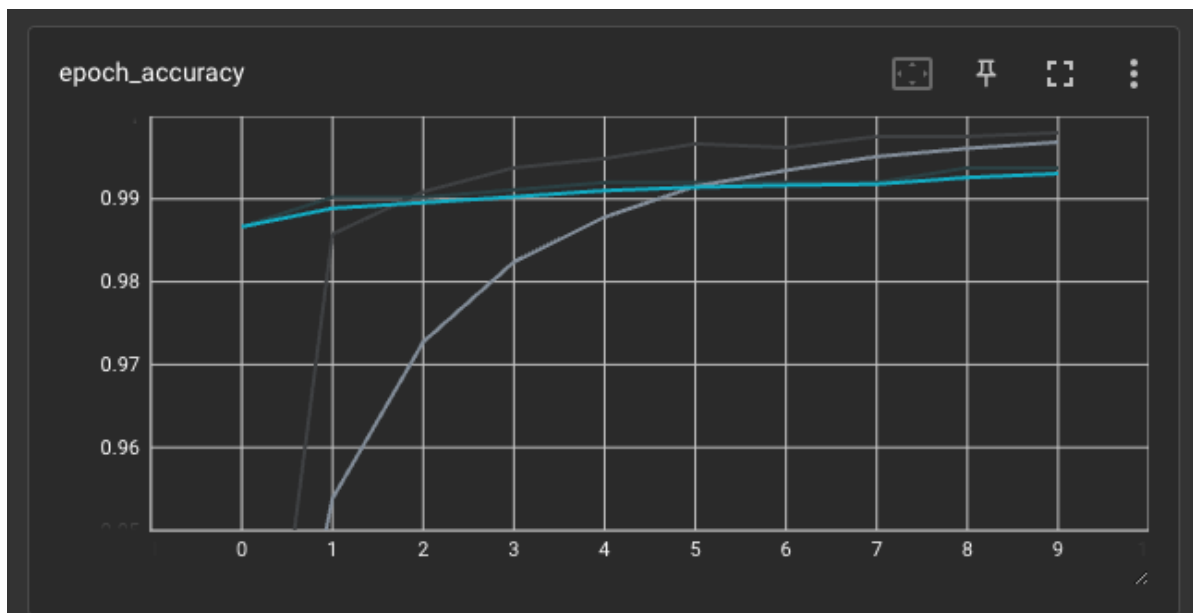


Figura 9.2: Precisión por época

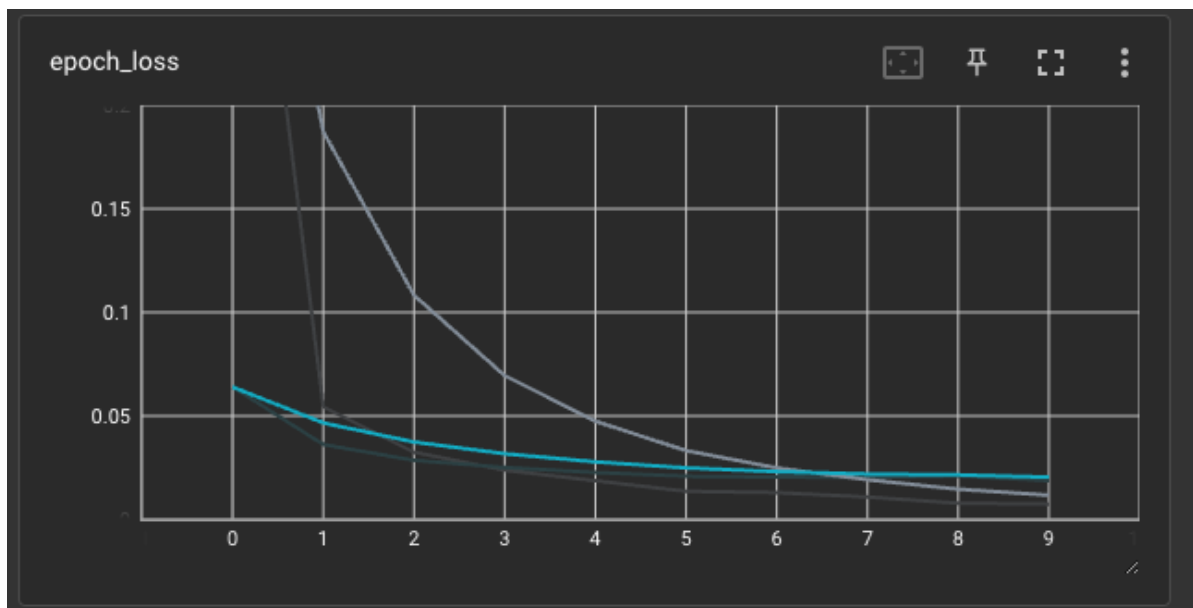


Figura 9.3: Pérdida por época

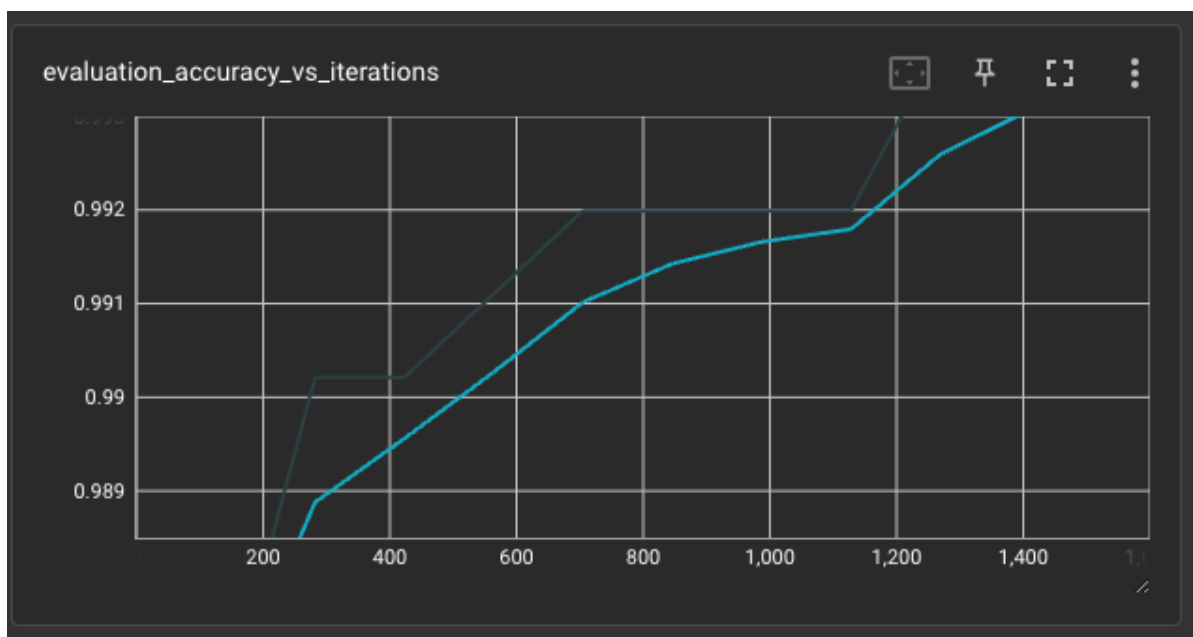


Figura 9.4: Evaluación en la precisión

La observación detallada de la evaluación de esta red neuronal revela un progreso positivo a lo largo del tiempo. Se aprecia una tendencia favorable en la mejora de las métricas, indicando un rendimiento en constante optimización. Es destacable notar que la pérdida de información exhibe una disminución constante, lo cual sugiere una eficaz capacidad de aprendizaje de la red. Este patrón descendente en la pérdida señala que la red está refinando su capacidad para representar y generalizar patrones, lo cual es

crucial para un rendimiento robusto en diversas situaciones. En resumen, los indicios visibles apuntan a una evolución positiva y a una mejora progresiva en la capacidad predictiva de la red neuronal.

En el próximo paso de la investigación, se llevará a cabo el mismo procedimiento, pero con la única variación de aumentar los pesos en la red neuronal. Esta modificación busca explorar y comparar las diferencias resultantes entre ambas configuraciones. Al introducir este ajuste específico, se espera identificar de manera clara y específica cómo el incremento de los pesos afecta las métricas de rendimiento, como la precisión y la pérdida. Este enfoque comparativo permitirá discernir el impacto singular de esta modificación en el comportamiento y la eficacia del modelo, proporcionando información valiosa sobre la influencia directa de los pesos en la capacidad de aprendizaje y generalización de la red neuronal.

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from `''`  
warn(  
Epoch 1/10  
141/141 [=====] - 4s 14ms/step - loss: 0.2776 - accuracy: 0.9415 - val_loss: 0.0456 - val_accuracy: 0.9920  
Epoch 2/10  
141/141 [=====] - 1s 10ms/step - loss: 0.0392 - accuracy: 0.9882 - val_loss: 0.0305 - val_accuracy: 0.9929  
Epoch 3/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0227 - accuracy: 0.9929 - val_loss: 0.0260 - val_accuracy: 0.9929  
Epoch 4/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0163 - accuracy: 0.9951 - val_loss: 0.0234 - val_accuracy: 0.9938  
Epoch 5/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0122 - accuracy: 0.9969 - val_loss: 0.0193 - val_accuracy: 0.9938  
Epoch 6/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0081 - accuracy: 0.9982 - val_loss: 0.0215 - val_accuracy: 0.9911  
Epoch 7/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0065 - accuracy: 0.9991 - val_loss: 0.0181 - val_accuracy: 0.9938  
Epoch 8/10  
141/141 [=====] - 1s 4ms/step - loss: 0.0061 - accuracy: 0.9982 - val_loss: 0.0174 - val_accuracy: 0.9947  
Epoch 9/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0041 - accuracy: 0.9993 - val_loss: 0.0182 - val_accuracy: 0.9947  
Epoch 10/10  
141/141 [=====] - 0s 3ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0166 - val_accuracy: 0.9947  
Tiempo total de entrenamiento: 9.532162189483643 segundos
```

Figura 9.5: Segundos Resultados

En el análisis de los resultados, se destaca una diferencia significativa, siendo notable que el tiempo de ejecución se reduce a dos segundos. Este cambio representa una ventaja considerable en términos de eficiencia temporal. La disminución en el tiempo de ejecución sugiere una mejora en la velocidad de procesamiento al aumentar los pesos en la red neuronal. Esta eficiencia temporal puede ser una consideración crucial, especialmente en entornos donde la rapidez en la obtención de resultados es esencial. La reducción del tiempo de ejecución a dos segundos podría traducirse en un beneficio operativo notable, optimizando el rendimiento general del modelo y acelerando el proceso de toma

de decisiones.

Ahora examinaremos detenidamente la presión, la pérdida de la época y también observaremos cómo evoluciona la precisión en relación con este resultado innovador. Nos enfocaremos en analizar minuciosamente cada aspecto, permitiéndonos comprender de manera más exhaustiva la influencia de estos elementos en el nuevo resultado que estamos obteniendo.

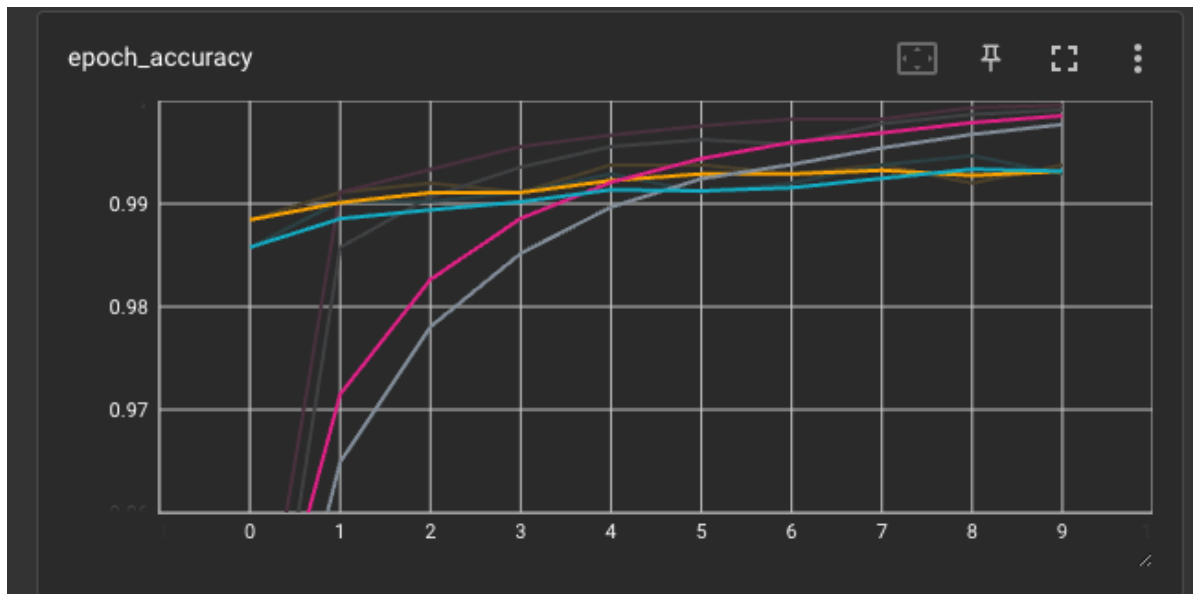


Figura 9.6: Precisión de época con pesos duplicados

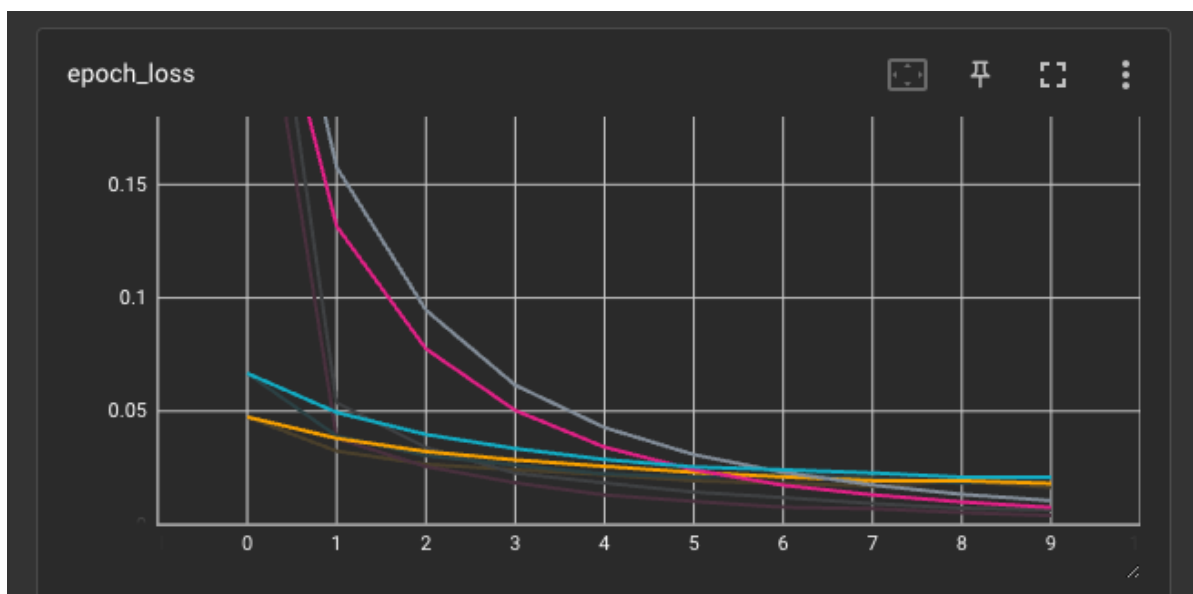


Figura 9.7: Pérdida de época

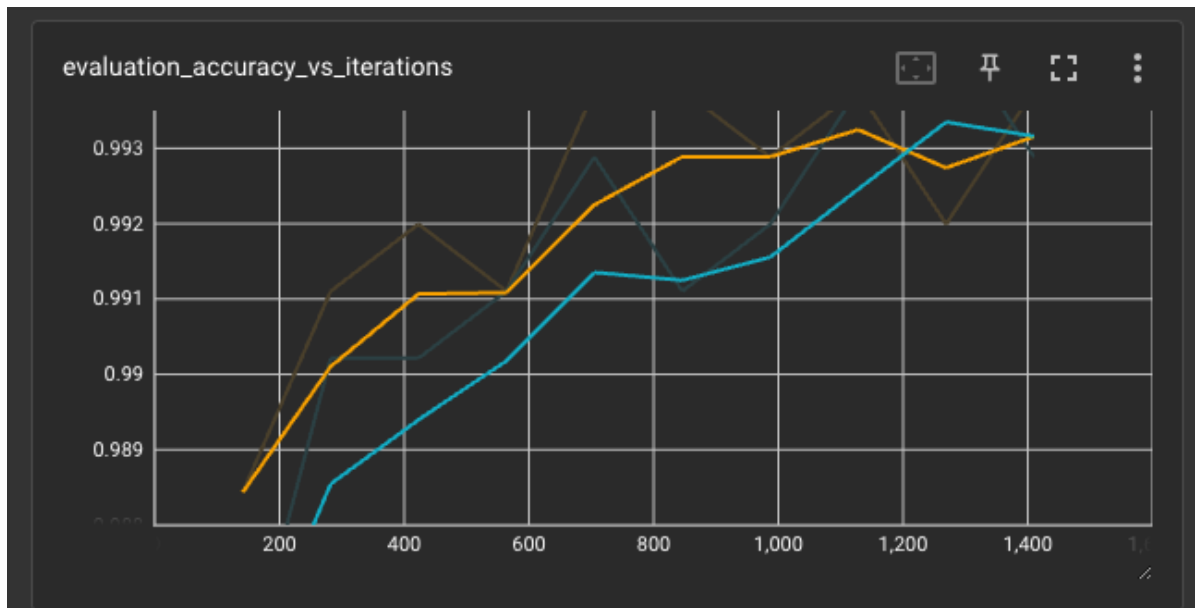


Figura 9.8: Evaluación en la precisión

Al analizar las representaciones gráficas proporcionadas a través de TensorBoard, se evidencian notables disparidades, siendo una de las más destacadas la presencia de un mayor número de capas ocultas. Estas capas adicionales se visualizan mediante líneas de colores distintos, y al evaluarlas, se observa una marcada diferencia. Se destaca que varias de estas capas adicionales exhiben una mayor precisión durante el proceso de entrenamiento.

En resumen, al duplicar los pesos de una red neuronal, se percibe una mejora sustancial en su rendimiento. Esta estrategia se revela como eficaz y eficiente, aunque conlleva un tiempo de entrenamiento ligeramente más prolongado en comparación con una red convencional. Al realizar la evaluación, se constata que la precisión de la red con pesos duplicados supera significativamente a la de una red convencional.

Capítulo 10

Organización del capituladoo

En el capitulo 2 se explicará lo que son y como funcionan las redes neuronales artificiales, asicomo la función de activación que es un método que utilizan estas0

Se describirán los tipos de redes neuronales, tanto de perceptron multicapa como convolucionales, y el algoritmo backpropagation. Se mencionará como es el aprendizaje en humanos, que este se divide en el aprendizaje activo y el aprendizaje con comprensión Y para terminar se describirá el aprendizaje incremental y su algoritmo.

En el capitulo 3 se implementará el algoritmo de John A. Bullinaria, se verificará su funcionamiento y los resultados que da al pasar los datos que dice para comprobar que es como menciona en su artículo.

En el capitulo 4 se explicará como se se extendió el algoritmo base permitiendo el uso de mas de dos pesos duplicados y se aplicaran los mismos datos de entrenamiento y de prueba que al algoritmo base.

Posteriormente, en el capitulo 5 con los resultados obtenidos, se mostrará una comparación de los resultados de ambos trabajos para notar si hubo una reducción significativa en las tasas de aprendizaje. En el capitulo 6 se verán las conclusiones y trabajo futuro.

Capítulo 11

Cronograma de actividades

| FASES | PRIMER SEMESTRE | | | | | | SEGUNDO SEMESTRE | | | | | |
|--|-----------------|---|---|---|---|---|------------------|---|---|---|---|---|
| | MESES | | | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
| Delimitación del tema | | | | | | | | | | | | |
| Preprocesamiento de datos | | | | | | | | | | | | |
| Implementación del algoritmo | | | | | | | | | | | | |
| Extensión del algoritmo | | | | | | | | | | | | |
| Comparación de ambas implementaciones | | | | | | | | | | | | |
| Preparación del trabajo para congresos | | | | | | | | | | | | |
| Escritura de tesis | | | | | | | | | | | | |
| Pre-examen | | | | | | | | | | | | |
| Atender observaciones | | | | | | | | | | | | |
| Examen profesional | | | | | | | | | | | | |

Bibliografía

- [1] J. A. Bullinaria, “Evolved dual weight neural architectures to facilitate incremental learning,” in *IJCCI*, pp. 427–434, 2009.
- [2] B. Zhao, T. Wu, F. Fang, L. Wang, W. Ren, X. Yang, Z. Ruan, and X. Kou, “Prediction method of 5g high-load cellular based on bp neural network,” in *2022 8th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pp. 148–151, IEEE, 2022.
- [3] L. A. C. Salazar, D. J. M. Aldana, and J. A. C. Montes, “Implementación de redes neuronales y lógica difusa para la clasificación de patrones obtenidos por un sónar,” in *2013 II International Congress of Engineering Mechatronics and Automation (CIIMA)*, pp. 1–6, IEEE, 2013.
- [4] Y. Liu, “Yuan liu incremental learning in deep neural networks,” 2015.
- [5] C. G. Giraud-Carrier, “A note on the utility of incremental learning,” *AI Commun.*, vol. 13, pp. 215–224, 2000.
- [6] A.-J. Li, “An improved algorithm for incremental learning learn++,” in *2008 International Conference on Wavelet Analysis and Pattern Recognition*, vol. 1, pp. 310–315, IEEE, 2008.
- [7] R. Elwell and R. Polikar, “Incremental learning of concept drift in nonstationary environments,” *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

- [8] P. Royo, “Qué son las redes neuronales y cuál es su aplicación en el marketing,” 2021.
- [9] R. Ade and P. Deshmukh, “Methods for incremental learning: a survey,” *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 4, p. 119, 2013.
- [10] D. H. Fisher, “Knowledge acquisition via incremental conceptual clustering,” *Machine learning*, vol. 2, no. 2, pp. 139–172, 1987.