UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

MODELO SUPERVISADO PARA LA CLASIFICACIÓN DE MANUSCRITOS DE NÚMEROS ARÁBIGOS UTILIZANDO RECONOCIMIENTO DE PATRONES PUNO – 2018

PRESENTADA POR:

CARLOS BORIS SOSA MAYDANA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2019

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

MODELO SUPERVISADO PARA LA CLASIFICACIÓN DE MANUSCRITOS DE NÚMEROS ARÁBIGOS UTILIZANDO RECONOCIMIENTO DE PATRONES PUNO – 2018

PRESENTADA POR:

CARLOS BORIS SOSA MAYDANA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE

.....
Dr. MARCO ANTONIO QUISPE BARRA

PRIMER MIEMBRO

.....
Dr. MARIO ANTONIO SUAREZ LÓPEZ

SEGUNDO MIEMBRO

.....
Dr. JOSÉ EMMANUEL CRUZ DE LA CRUZ

ASESOR DE TESIS

.....
Dr. IVÁN DELGADO HUAYTA

Puno, 20 de Agosto de 2019

ÁREA: Ciencias de la Computación
LÍNEA DE INVESTIGACIÓN: Ingeniería Computacional y Sistemas
TEMA: Reconocimiento de patrones

DEDICATORIA

A mi familia por el incondicional apoyo para el desarrollo del presente proyecto personal, gracias por su paciencia y comprensión.

A mis colegas y amigos con quienes compartimos la noble tarea de formar futuros profesionales en Ingeniería de Sistemas e Ingeniería Electrónica.

AGRADECIMIENTOS

- A la Universidad Nacional del Altiplano, por darnos la oportunidad de concretar sueños y desafíos profesionales.
- A cada uno de los integrantes del jurado de quienes cada día aprendemos a ser más humanos y profesionales de éxito.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En el presente proyecto implementamos un modelo supervisado para la clasificación de manuscritos de números arábigos utilizando reconocimiento de patrones basados en redes neuronales artificiales para reconocer los números de cualquier imagen de entrada. Separamos nuestro proyecto en dos partes, la segmentación de una imagen en caracteres individuales, y luego clasificamos estas imágenes en sus respectivas etiquetas de caracteres. Nuestro enfoque se basa en el reconocimiento de dígitos, y aplica las técnicas al reconocimiento de dígitos (0-9) propio. En particular, utilizamos arquitecturas que involucran redes neuronales superficiales y profundas que se expande en los clasificadores entrenados a partir de imágenes de manuscritos de números contenidos en MNIST. La particularidad de la investigación es el uso de una MATRIZ DE ORO para convertir la matriz de entrada original de 28 x 28 en su matriz equivalente de 14 x 14, sin que esta pierda su esencia ni utilizar métodos complejos de normalización de los datos de entrada, obteniendo un resultado del 95.9% de clasificaciones correctas para las 60000 imágenes en la etapa de entrenamiento y 94.4% de clasificaciones correctas de las 10000 imágenes en la etapa de pruebas, superando de esta forma la hipótesis planteada en el presente trabajo. Para la validación del modelo utilizamos curvas ROC, las mismas que están por encima del 0.9 validándolo como un test muy bueno para las 10 clases propuestas.

Palabras clave: clasificación supervisada, MATRIZ DE ORO, MNIST, redes neuronales, reconocimiento de patrones, ROC.

ABSTRACT

In this project we implemented a supervised model for the classification of manuscripts of Arabic numerals using pattern recognition based on artificial neural networks to recognize the numbers of any input image. We separate our project into two parts, the segmentation of an image into individual characters, and then we classify these images into their respective character labels. Our approach is based on the recognition of digits, and applies the techniques to the recognition of digits (0-9) of our own. In particular, we use architectures that involve superficial and deep neural networks that expand in trained classifiers from manuscript images of numbers contained in MNIST. The particularity of the investigation is the use of a GOLD MATRIX to convert the original 28 x 28 input matrix into its equivalent 14 x 14 matrix, without losing its essence or using complex methods of normalizing the input data , obtaining a result of 95.9% of correct classifications for the 60,000 images in the training stage and 94.4% of correct classifications of the 10,000 images in the test stage, thus exceeding the hypothesis proposed in this paper. For the validation of the model we use ROC curves, which are above 0.9 validating it as a very good test for the 10 proposed classes.

Keywords: GOLD MATRIX, MNIST, Neural networks, supervised classification, pattern recognition, ROC.

INTRODUCCIÓN

La presente investigación, tiene como principal motivación sumergirse en el mundo de la inteligencia artificial, visión por computador y, más concretamente, las redes neuronales artificiales (RNA). Dado que este campo es sumamente complicado e inalcanzable en su totalidad, decidimos comenzar por los conceptos fundamentales y una vez adquiridos, procedimos a generar un modelo de red neuronal artificial capaz de identificar y clasificar dígitos como caracteres manuscritos, en concreto las imágenes que se encuentran en la base de datos MNIST («MNIST database», 2018). Desde su generación aproximadamente en 1999, esta base de datos ha servido como referencia para desarrollar algoritmos de clasificación supervisada. Nuestra motivación personal es la de aplicar técnicas o métodos de la inteligencia artificial, los mismos que captaron mi atención debido a la extensa cantidad de aplicaciones conseguidas en este periodo de tiempo, relativamente corto. También nos motiva la idea de saber que al finalizar este proyecto compartiremos los resultados obtenidos al experimentar con nuestro modelo referido a clasificación supervisada de imágenes digitalizadas de manuscritos de números arábigos.

En la civilización árabe, éstos sostuvieron contacto cultural con los hindúes, los griegos del Imperio Bizantino y los egipcios, donde adquirieron el conocimiento por medio de las traducciones de las grandes obras de Euclides, Arquímedes, Aristóteles, Ptolomeo, Diofanto y otros a la lengua árabe. “El sistema numérico actual (llamado arábigo) no fue inventado por los árabes, sino por los hindúes, ellos recogieron este gran conocimiento y lo introdujeron en Europa, por ejemplo, al cero lo llamaron céfer, que en el idioma árabe significa vacío. Este nuevo sistema de numeración muy lentamente fue llegando a occidente reemplazando a los números romanos, que dominaron por muchos siglos. Aunque se sabe que el primer manuscrito europeo que utilizó los numerales árabes data aproximadamente del año 976 d.C., ya en el año 1500 d.C. la aritmética explicaba el sistema de numeración arábigo con todo lujo de detalles”.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1 Marco Teórico

1.1.1 Preprocesamiento

La etapa de preprocesamiento en las redes neuronales artificiales es una tarea necesaria para la preparación de los datos, los mismos que son utilizados para el análisis de datos. La justificación de este proceso previo al análisis de datos, radica generalmente en que los datos vienen sucios (con ruido) por diferentes motivos, entre las cuales podemos detallar:

- "Datos incompletos": valores incompletos o faltantes para algunos indicadores(atributos) o probablemente "sólo se tienen los datos agregados y no se cuenta con el detalle".
- "Ruido": datos erróneos. Por ejemplo, utilizar valores negativos en un campo o atributo que esta destinado al almacenamiento de salarios.
- "Inconsistencias": se refiere a discrepancias en los datos. Por ejemplo, "edad de un empleado = 35 y fecha de nacimiento = 08/07/1999".

Los algoritmos de aprendizaje computacional(machine learning), normalmente se utilizan para llevar a cabo ciertas tareas en el proceso de análisis de datos.

Frecuentemente, este preprocesamiento de datos tiene un alto impacto en el desempeño o desenvolvimiento de los algoritmos de aprendizaje supervisado. Aplicar estas técnicas permite que los algoritmos de aprendizaje logren ser más eficientes, por ejemplo, "si se reduce la dimensionalidad, los algoritmos de aprendizaje podrían actuar de forma más rápida y su efectividad podría mejorar".

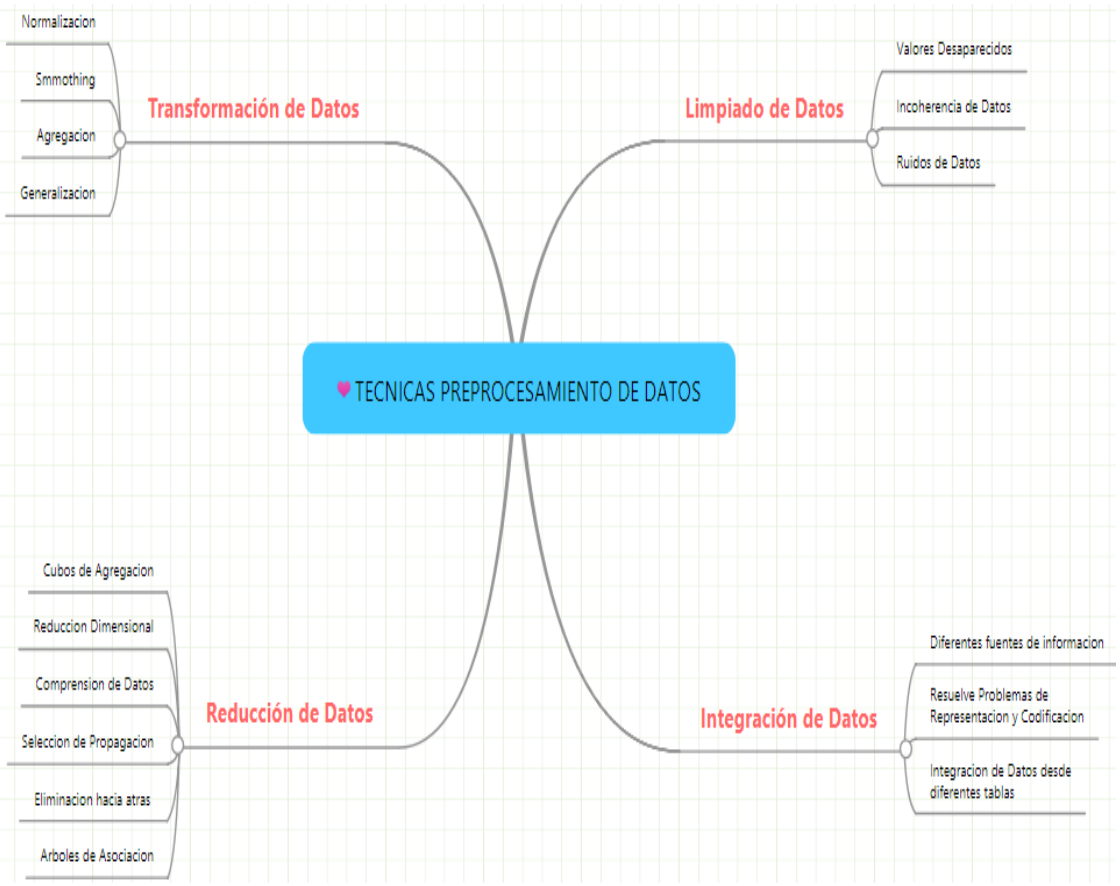


Figura 1. Técnicas de procesamiento de datos

Fuente : (*TECNICAS PREPROCESAMIENTO DE DATOS*, s/f)

1.1.2 Análisis estadístico

Sabemos que existen numerosas pruebas estadísticas a disposición para evaluar la significancia de las diferencias posibles encontradas en nuestros experimentos, muchas veces las utilizamos simplemente con criterios de filtrado de los datos basados en la variación del nivel de expresión de patrones identificados, es decir, en una asignación arbitraria de un umbral identificado. Sin embargo, resulta más apropiado hacer uso de otro tipo de aproximaciones que permitan distinguir y encontrar entre las diferencias significativas de las muestras de estudio y las diferencias puramente debidas al azar.

Una técnica estadísticas clásica como el Test de la T de Student; “evalúa la significancia de las diferencias encontradas entre dos muestras comparadas calculando el estadístico t y el P-valor asociado al contraste, el cual representa la probabilidad de que esas diferencias se deban al azar. De esta forma, los patrones de genes de un biochip pueden ordenarse en función de sus P-valores y posteriormente se podría seleccionar un umbral de corte apropiado dependiendo del número de falsos positivos que se consideren aceptables. Por otro lado, existe un método estadístico que correlaciona las distancias existentes en un conjunto de datos que se comparan entre sí en función de diferentes factores. Un ejemplo podría ser la correlación estadística que existe entre los datos obtenidos de microarrays hibridados con un determinado tipo de tumor y las variables clínicas de ese tumor. Las muestras cuya distancia en los datos de microarrays sea muy grande, también presentaran una larga distancia en cuanto a los datos clínicos y viceversa. Este tipo de aproximación, llamada medida de asociación lineal Mantel-Haenszel, indicaría en este caso que las diferencias clínicas encontradas en pacientes con un mismo tipo tumor están estadísticamente relacionadas con las diferencias en la expresión génica”.

1.1.3 Clustering

Métodos de clustering: hace mucho tiempo se han venido utilizando muchos y muy diversos métodos para agrupar los datos procedentes de microarreglos, incluyendo una simple inspección visual; pero se considera mejor aplicar métodos estadísticos robustos y fiables.

Clustering no supervisado: Es un conjunto de técnicas que usan la distancia de los datos para su agrupación sin utilizar ningún tipo de información externa para su organización. Vale decir que en función de la forma en la que los datos son agrupados, podemos distinguir dos tipos de clustering:

"Jerárquico: El clustering jerárquico aglomeratico es un método determinista basado en una matriz de distancias. Establece pequeños grupos de patrones de genes/condiciones que tienen un patrón de expresión común y posteriormente construye un dendograma (representación gráfica de un grupo de relaciones basadas en la cercanía o similitud entre los datos) de forma secuencial. El árbol o dendograma, establece una relación ordenada de los grupos previamente definidos y la longitud de sus ramas es una representación de la distancia entre los distintos nodos del mismo. En el desarrollo del clustering jerárquico se han utilizado diferentes algoritmos (UPGMA, Ward, etc.) aunque todos siguen la misma estrategia en general: separan cada patrón de gen en un nodo diferente, calculan la distancia entre los dos genes más próximos y los juntan en un cluster. Entonces se vuelve a calcular la matriz de distancias sustituyendo los dos patrones que se han unido por el promedio de ambos. En cada paso, los algoritmos son capaces de juntar los genes no solo de dos en dos sino muchos más a la vez. Muchos de ellos simplemente se diferencian en la forma en la que calculan la distancia del nuevo cluster formado al resto de los elementos de la matriz, y en este sentido, la aproximación del "Average linkage" (algoritmo que opera agrupando iterativamente los genes o clusters que presentan la distancia media más pequeña en cada paso sucesivo del cálculo de la matriz de distancias) es la más utilizada.

Por otro lado, existe el clustering jerárquico divisivo que es similar al anterior, pero agrupa los patrones de genes de forma inversa. Mientras que el clustering aglomerativo separa inicialmente todos los genes y posteriormente los va agrupando para construir clusters más grandes, el clustering divisivo agrupa inicialmente todos los genes en un único cluster y sucesivamente los va separando hasta que cada uno se encuentre aislado como una entidad. Es decir, el método divisivo va identificando aquellos genes con un patrón más diferente para separarlos en el espacio lo más posible. Este es el caso de SOTA (Self-Organizing Tree Algorithm)(Bose, s/f). A pesar de que no está exento de problemas, el clustering es una herramienta poderosa para la reducción de los datos obtenido de micorarrays y para el estudio de posibles hipótesis que relacionan los clusters de genes obtenidos con el fenotipo. Sin embargo, este tipo de relaciones deben ser formalmente validadas por otros experimentos adicionales".

No jerárquico: Para este caso los algoritmos comienzan por calcular la matriz de distancias a partir de un numero predefinido de grupos(clusters) y van de forma iterativa encontrando los clusters en los diferentes grupos hasta minimizar la dispersión interna de cada cluster. Los dos algoritmos más representativos de este tipo de clustering son:

"K-Medias: es un algoritmo que comienza con una muestra de "k" genes elegidos al azar de la matriz original de datos. Cada uno de ellos se utiliza como el centroide inicial de los "k" clusters que se van a formar. La matriz de distancias se calcula desde dicho centroide hasta cada uno de los genes de la matriz de datos y cada uno de ellos será asignado de esta forma al centroide más cercano. Entonces la matriz de distancias se recalcula reemplazando cada centroide por la media de los patrones de genes asignados a él y el algoritmo repite entonces el proceso anterior. El mapa de clusters que ofrece este algoritmo carece de topología".

"SOM: los mapas auto-organizados (Self-Organising Maps) son redes neuronales. El algoritmo permite, de forma iterativa, que los patrones más parecidos se vayan juntando entre sí y alejándose de aquellos otros que son más diferentes. Este tipo de algoritmos son más fiables y robustos puesto que se basan

en redes neuronales que por definición son capaces de trabajar con grandes cantidades de datos con ruido. Sin embargo, no carece de ciertos inconvenientes. SOM es una herramienta particularmente útil en el tratamiento de datos procedentes de series temporales.

El gran problema que presentan estos métodos no jerárquicos es que al no generar un dendograma no permiten hacerse una idea de la representación espacial de los patrones de genes, la cual suele ofrecer un conocimiento intuitivo de cómo analizar los datos de microarrays".

Clustering supervisado: se basa en la idea de que para la clasificación de la mayoría de muestras biológicas ya existe información preliminar que puede utilizarse para agrupación de nuevos datos en clusters. Los métodos supervisados aprenden de esta información previa, generalmente ofrecida por un conjunto de datos de "entrenamiento", la forma en que deben clasificar los nuevos datos (genes o condiciones) que se les presentan.

"SVM (Supported Vector Machines): es una técnica lineal que utiliza hiperplanos para separar los datos en el espacio como puntos negativos o positivos. Los datos de estudio son clasificados respecto de otro conjunto de datos previamente conocido".

"Pereceptrones: están basados en redes neuronales. Tienen algunas ventajas sobre las SVM como por ejemplo la capacidad de clasificar muchas muestras al mismo tiempo y discriminar entre varias clases diferentes".

1.1.4 Python.

Python ("Python", 2018) es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

1.1.5 Deep Learning.

Aprendizaje profundo (en inglés, deep learning) ("Aprendizaje profundo", 2018) es un conjunto de algoritmos de clase aprendizaje automático (en inglés, machine learning) que intenta modelar abstracciones de alto nivel en datos usando arquitecturas compuestas de transformaciones no lineales múltiples.

El aprendizaje profundo (Kumar & Malarvizhi, 2018) es parte de un conjunto más amplio de métodos de aprendizaje automático basados en asimilar representaciones de datos. Una observación (por ejemplo, una imagen) puede ser representada en muchas formas (por ejemplo, un vector de píxeles), pero algunas representaciones hacen más fácil aprender tareas de interés (por ejemplo, "¿es esta imagen una cara humana?") sobre la base de ejemplos, y la investigación en esta área intenta definir qué representaciones son mejores y cómo crear modelos para reconocer estas representaciones.

Varias arquitecturas de aprendizaje profundo, como redes neuronales profundas, redes neuronales profundas convolucionales, y redes de creencia profundas, han sido aplicadas a campos como visión por computador, reconocimiento automático del habla, y reconocimiento de señales de audio y música, y han mostrado producir resultados de vanguardia en varias tareas.(Cireşan et al., 2012)

1.1.6 GPU.

Unidad de procesamiento gráfico ("Unidad de procesamiento gráfico", 2018) o GPU (Graphics Processing Unit) "es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la unidad central de procesamiento (CPU) puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos)".

La GPU implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el antialiasing, que suaviza los bordes de las figuras para darles un aspecto más realista. Adicionalmente existen primitivas para dibujar rectángulos, triángulos, círculos y arcos, las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos.

1.1.7 Red Neuronal Artificial Perceptron Multicapa.

El perceptrón multicapa ("Perceptrón multicapa", 2018) es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón(Brain & Rosenblatt, s/f) (también llamado perceptrón simple). En el caso de perceptrón multicapa puede estar total o localmente conectado. A priori cada salida de una neurona de la capa "i" se convierte en entrada para todas las neuronas de la capa "i+1".

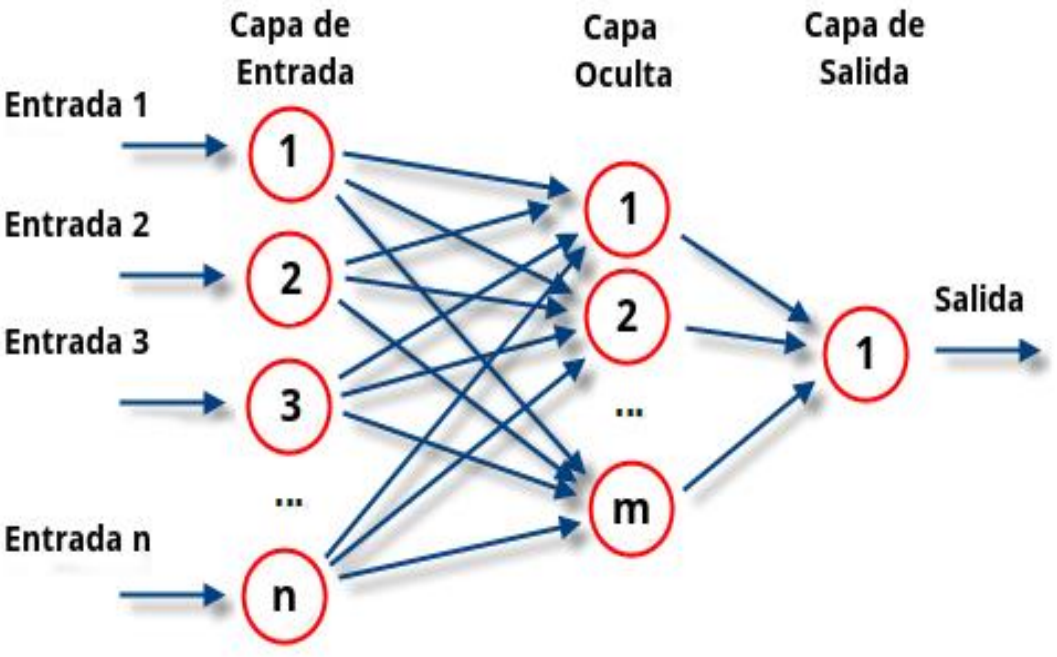


Figura 2. Arquitectura de una red neuronal artificial

Fuente ("Perceptrón multicapa", 2018)

1.1.8 MNIST DATABASE ("MNIST Database", 2018).

La base de datos MNIST (base de datos modificada del Instituto Nacional de Estándares y Tecnología) es una gran base de datos de dígitos escritos a mano que se usa comúnmente para entrenar varios sistemas de procesamiento de imágenes, creado al "volver a mezclar" las muestras de los conjuntos de datos originales de NIST. Los creadores sintieron que, dado que el conjunto de datos de capacitación del NIST se tomó de los empleados de la Oficina del Censo de los Estados Unidos, mientras que el conjunto de datos de las pruebas se tomó de los estudiantes de secundaria estadounidenses, no fue adecuado para los experimentos de aprendizaje automático. Además, las imágenes en blanco y negro de NIST se normalizaron para encajar en un cuadro delimitador de 28x28 píxeles y suavizantes, lo que introdujo niveles de escala de grises.

La base de datos MNIST contiene 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba. La mitad del conjunto de entrenamiento y la mitad del conjunto de prueba se tomaron del conjunto de datos de entrenamiento del NIST, mientras que la otra mitad del conjunto de entrenamiento y la otra mitad del conjunto de prueba se tomaron del conjunto de datos de prueba del NIST.

1.1.9 TRANSFERENCIA DE APRENDIZAJE

La transferencia de aprendizaje, se puede producir bien mediante la extracción de descriptores profundos, pesos internos de una capa, o mediante ajuste de pesos en una o varias capas internas de las redes convolucionales. Este ajuste se produce a través del entrenamiento y validación sobre el conjunto de imágenes del problema específico en cuestión, valiéndose del ajuste previo sobre imágenes de tipo generalista de un conjunto de imágenes.

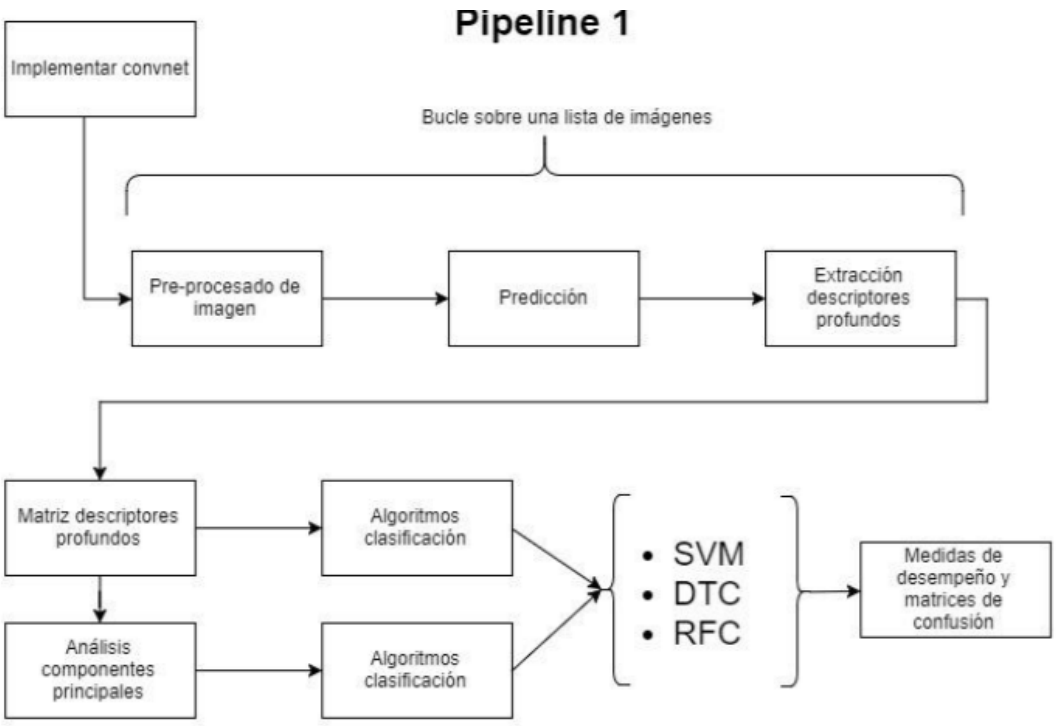


Figura 3. Funcionamiento de un descriptor profundo

Fuente: (Marturet Rodrigo, 2018)

1.1.10 ÁRBOLES DE DECISIÓN

Los árboles de decisión se presentan como una herramienta efectiva para la predicción de probabilidades de incumplimiento, no solo a nivel de capacidad de discriminación (potencia), estabilidad a través del tiempo, sino como una herramienta de fácil entendimiento que permite potencializar sus usos y servir además de la predicción, para la planeación de estrategias comerciales de venta de servicios, estrategias de cobranza entre muchas otras. Resulta de importancia el tener un modelo de cálculo de probabilidad que permita un incumplimiento confiable y con una alta capacidad de discriminación la que impacte considerablemente en el cálculo de provisiones, afectando directamente el balance y las utilidades que podría llegar a tener la entidad. “Adicionalmente como los modelos son empleados para otorgamiento de créditos, mantenimiento de cuentas, hacen parte fundamental de la gestión integral de riesgo, por tanto, un cálculo u operación inapropiada podría llevar a una institución financiera a situaciones de insolvencia”.(Ferri et al., s/f; Molnar, s/f)

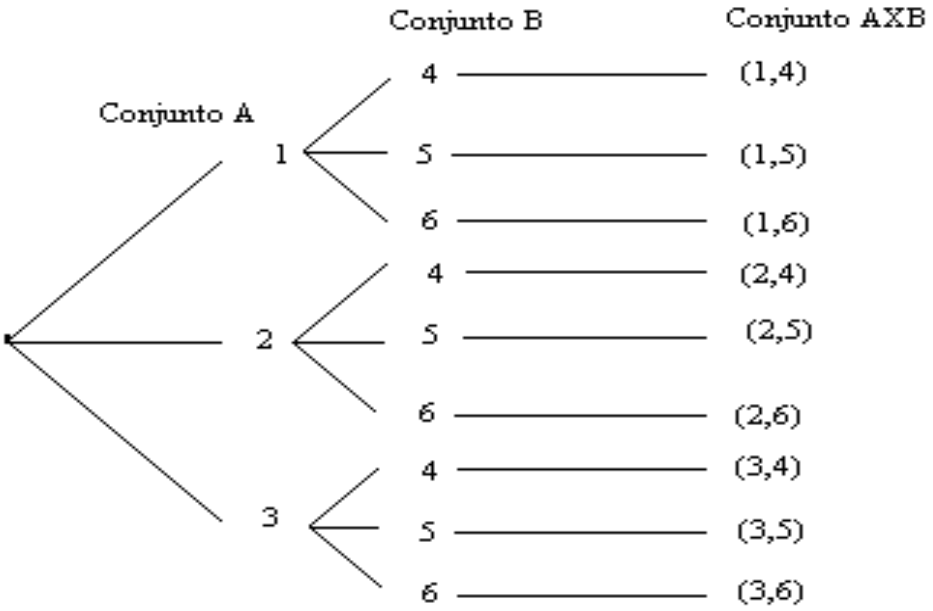


Figura 4 Ejemplo de árbol de decisión

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1 Planteamiento

En los últimos años, el uso de Redes Neuronales Artificiales ha sufrido un crecimiento exponencial a nivel global (internacional), tanto en su aplicación como en la precisión que estos pueden conseguir, y es rara aquella empresa que aún no ha tratado de integrarlas de alguna forma en el servicio prestado. Los usos de estas redes son muy variados, ya que, por ejemplo, se pueden utilizar para el reconocimiento de sonidos, tal y como hace Google con su aplicación Google Now(“Google Now”, 2018). Además de esto, se pueden usar Redes Neuronales para la traducción instantánea de texto, tal como hace Skype(Edje et al., 2013; “Skype”, 2018), o para la predicción de la siguiente palabra a escribir al mandar un mensaje como hace la aplicación de teclado Swiftkey(“SwiftKey”, 2018). Sin embargo, el uso más importante y extendido es el de reconocimiento de imágenes. En este ámbito, grandes empresas como, por ejemplo, Facebook(“Facebook”, 2018) han conseguido desarrollar algoritmos usando estas redes capaces de reconocer rostros humanos con una precisión superior al 97%. Microsoft, del mismo modo, ha implementado en Cortana(López et al., 2018; “Microsoft Cortana”, 2018), su asistente personal, una Red Neuronal capaz de clasificar especies animales, En argentina (García Navarro, s/f) en su trabajo de fin de grado hace uso de redes neuronales aplicando el lenguaje Python(“Python”, 2018) para hacer Deep Learning(“Aprendizaje profundo”, 2018; Pedregosa et al., 2011); también podemos citar que en la Universidad Politécnica de Valencia (Grau Moreso, 2014) implementa redes neuronales en Unidades de Procesamiento Gráfico (GPU) manejadas con tecnología CUDA bajo C++ para

efectuar el reconocimiento de cifras manuscritas. El sistema utilizado demuestra ser eficiente y aplicable a múltiples programas de Inteligencia Artificial. En el ámbito nacional (Garrido Rojas, s/f) experimenta con una base muy limitada de imágenes de manuscritos utilizando Matlab (Franc & Hlavaˇc, s/f; "MATLAB", 2018), también podemos citar a (MONGE OSORIO, s/f) quién hizo un diseño genérico y modular de una red neuronal artificial perceptron multicapa ("Perceptrón multicapa", 2018), orientada al reconocimiento de dígitos manuscritos en un FPGA("Field-programmable gate array", 2018) mediante el lenguaje de descripción de hardware VHDL(Shahdad, 1986; "VHDL", 2018), priorizando la implementación en hardware de las redes neuronales. En el ámbito local (regional) no se encontraron trabajos vinculados a procesamiento de manuscritos de números o texto, pero si trabajos sobre aplicación de redes neuronales cuyas aplicaciones son poco satisfactorias.

Determinada la importancia del reconocimiento de imágenes hoy en día, se plantea realizar en el presente proyecto un modelo de red neuronal artificial capaz de identificar imágenes de manuscritos de números con una precisión deseable por encima del 90%; el mismo que resumimos en la siguiente pregunta:

¿Cómo se relacionan el modelo de red neuronal supervisado y la clasificación de manuscritos de números arábigos para conseguir una precisión por encima del 90%?

2.2 Justificación

Esta investigación tiene como motivación adentrarse en el mundo de la inteligencia computacional y, más concretamente, las redes neuronales artificiales. En razón a que este campo es muy complejo e inabarcable en su totalidad, se ha decidido empezar por los conceptos fundamentales y, una vez adquiridos y comprendidos, se procederá a crear de una red neuronal artificial capaz de identificar y clasificar dígitos o caracteres manuscritos encontrados en la base de datos MNIST («MNIST database», 2018). Desde su lanzamiento en 1999, esta base de datos ha servido como referencia para la construcción de algoritmos de clasificación. Nuestra motivación personal es la de aplicar técnicas o métodos de la inteligencia computacional, la misma que ha captado mi atención debido a la extensa cantidad de logros conseguidos en un periodo relativamente corto de tiempo. Además de todo esto, me motiva la idea de saber que al final del proyecto comprenderé gran parte de los fundamentos que usan la mayoría de servicios del Smartphone, redes

sociales o incluso alguna que otra aplicación. Acciones que realizamos todos cada día, tras las cuales están presentes las redes neuronales sin ni siquiera percatarnos de ello.

Los árabes sostuvieron contactos culturales con los hindúes, los griegos del Imperio Bizantino y los egipcios, donde aprendieron que podían adquirir conocimiento por medio de las traducciones de grandes obras de autores como: Euclides, Ptolomeo, Arquímedes, Aristóteles, Diofanto, y otros al idioma árabe. El sistema numérico actual (llamado arábigo) no fue inventado por los árabes, sino por los hindúes, ellos recogieron este gran conocimiento y lo introdujeron en Europa en el que "al cero lo llamaron céfer, que en el idioma árabe significa vacío".

"Este nuevo sistema de numeración muy lentamente fue llegando a occidente reemplazando a los números romanos, que dominaron por muchos siglos. Aunque el primer manuscrito europeo que utilizó los numerales árabes data del año 976 d.C., ya en el año 1500 d.C. la aritmética explicaba el sistema de numeración arábigo con todo lujo de detalles".

2.3 Hipótesis

Es posible obtener una clasificación correcta por encima del 90% de manuscritos de números arábigos digitalizados a partir de un modelo supervisado de clasificación basado en redes neuronales.

2.4 Objetivos

2.4.1 General

Diseñar e implementar un modelo computacional que permita clasificar de manera automática manuscritos de números arábigos digitalizados utilizando reconocimiento de patrones.

2.4.2 Específicos

- Implementar una base de datos con manuscritos de números arábigos.
- Desarrollar un modelo computacional para la lectura automática de manuscritos de números arábigos digitalizados.

- Desarrollar un clasificador automático para el reconocimiento de manuscritos de números.
- Validar el modelo utilizando una matriz de consistencia y compararlo con curvas ROC de los resultados obtenidos en las diferentes etapas.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 Redes Neuronales Artificiales

El rendimiento o aprendizaje de los algoritmos de inteligencia computacional, depende en gran medida de la representación de datos que reciben y en muchas situaciones es difícil para el humano seleccionar las características que son más relevantes. Una probable solución a esta problemática es echar mano del aprendizaje automático para describir no sólo el esquema de la representación de salida, sino también su arquitectura y demás componentes, lo que se conoce como "representation learning". Combinando esta idea con otros conceptos, el aprendizaje profundo aprende de representaciones complejas en base a otras más sencillas. Para nuestra investigación utilizamos técnicas y métodos para reconocimiento de patrones basados en redes neuronales que describimos a continuación:

3.1.1 Redes feedforward

3.1.1.1 Estructura

Para establecer el mapeo entre entradas y salidas, las redes feedforward utilizan una arquitectura de red constituida por tres tipos de capas totalmente conectadas: capa de entrada, capas ocultas y capa de salida.

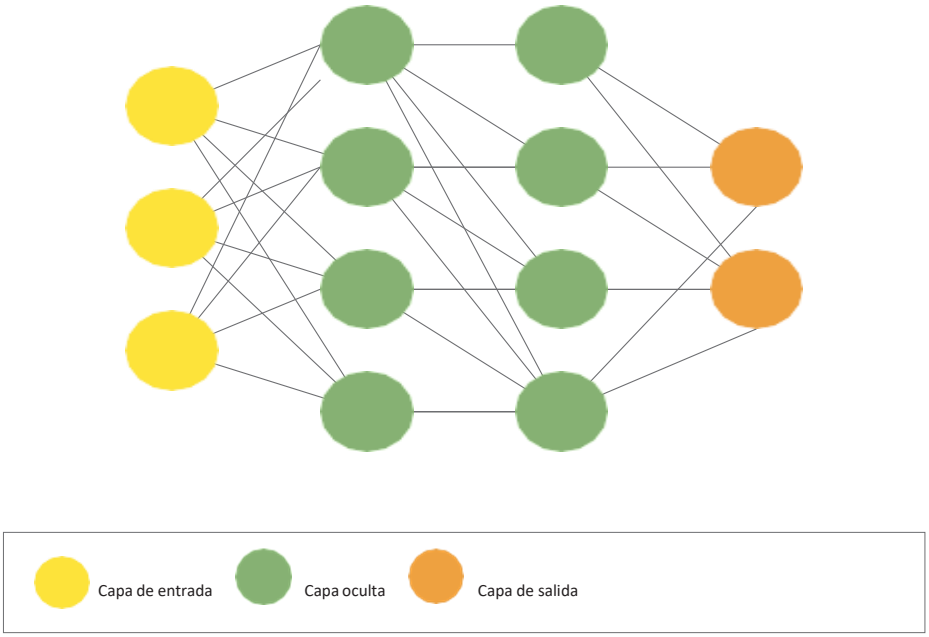


Figura 5. Esquema básico de una red feedforward

Capa de entrada: Se corresponde con el vector de datos de entrada. Si disponemos de un vector de datos, cada elemento del vector (x_1, x_2, ... , x_n) constituye una neurona de entrada.

Capa de salida: Es la capa en la que se realiza la operación objetivo, por ejemplo, la clasificación. En ese caso, la capa de salida tiene tantas neuronas como clases presente el problema a tratar. Cada neurona tiene asociado un peso y un bias.

Capa oculta: Son las que contienen todos los cálculos intermedios de la red. Están formadas por neuronas ocultas, cada una de las cuales tiene un peso y un bias, al igual que las neuronas de la capa de salida; mientras que las capas de entrada y de salida son únicas, el número de capas ocultas es variable y no determinado. De hecho, la capacidad para manejar un gran número de capas ocultas es lo que hace profundo a este tipo de aprendizaje y cabe mencionar aquí sus beneficios.

De acuerdo al teorema de aproximación universal, una red feedforward de una única capa oculta, con un número arbitrariamente grande de neuronas, se comporta como un aproximador universal. Pero esta capa oculta puede llegar a ser enorme, dificultando el aprendizaje y la generalización. En su lugar, el uso de modelos profundos permite reducir el número de neuronas necesarias para representar la función deseada, pudiendo completar el aprendizaje y respetando la capacidad de generalización de los modelos.

3.1.1.2 Función de activación

La función de activación es un elemento clave en las redes neuronales artificiales, pues es la que les otorga flexibilidad y el calculo del valor, proporcionándole la capacidad de estimar complejas relaciones lineales o no lineales en los datos.

"La elección de la función de activación es una decisión trascendente, dada que cada función se ajusta mejor a unos u otros problemas. Además, los pesos han de inicializarse de manera distinta según la función elegida. A este respecto, Glorot y Bengio han propuesto diferentes métodos de inicialización, basados en la longitud del vector de entrada, el número de neuronas de la capa oculta y el tipo de función de activación".

Aunque existe una gran variedad de funciones (identidad, binaria, lineal, sigmoidal, tangente hiperbólica, rectificadora, softplus, softmax y otros.), nosotros recomendamos el uso de la función tangente hiperbólica.

No obstante, estas funciones tienen un elevado coste computacional y en los últimos años se han visto reemplazadas por la función rectificadora (Rectified Linear Unit, ReLU), introducida en el año 2000 por Hahnloser (Agarap, 2018) con motivaciones biológicas y matemáticas. Esta función tiene un menor coste computacional, pues en lugar de hacer cálculos con exponenciales, utiliza una simple operación de máximo. Adicionalmente, su forma, lineal y no saturada, proporciona a las redes la capacidad para evitar los problemas de bajo gradiente y permite acelerar la convergencia del entrenamiento. Por todo ello, la función ReLU se ha convertido en la más empleada para la activación de las capas ocultas.

En lo que respecta a la capa de salida destaca el uso de la función identidad en problemas de regresión y la función softmax en problemas de clasificación. Esta última es una generalización de la función sigmoidal, encargada de normalizar la salida a valores comprendidos entre [0,1]. Se utiliza especialmente en los problemas de clasificación multiclase, donde la sigmoidal no permite distinguir más de dos clases.

3.1.1.3 Mecanismo de aprendizaje

Como ya hemos mencionado, la esencia de las redes neuronales artificiales radica en que no necesitan ser reprogramadas para resolver un problema, sino que estas arquitecturas son capaces de dar soluciones a nuevos casos sin la necesidad de ser reentrenadas con estos. Para poder conseguir esto, se recurre a un proceso de aprendizaje supervisado: "se presenta a la red un conjunto de patrones de ejemplo (una serie de muestras de entrada con sus correspondientes salidas esperadas) y los pesos de las neuronas se van modificando de manera proporcional al error que se produce entre la salida real y la salida esperada. El método de entrenamiento más utilizado para completar este proceso es el algoritmo de retropropagación o backpropagation".

Algoritmo de retropropagación o backpropagation

Este algoritmo es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen claramente dos fases: la fase de propagación, en la que se introduce un patrón de ejemplo a la red, que se va propagando desde la entrada hacia la salida; y la fase de aprendizaje, en la que los errores obtenidos a la salida de la red van propagándose hacia atrás, desde la salida hacia la entrada, con el objetivo de actualizar los pesos de las neuronas mediante el gradiente de la función de error.

Por tanto, el proceso de aprendizaje consiste en propagar hacia atrás el error entre la salida obtenida y la salida esperada, que se pretende vaya disminuyendo en el transcurso de las iteraciones. Para calcular dicho error, se recurre habitualmente a la función del error cuadrático medio. Una vez calculado el error en cada muestra, e(n), el objetivo es minimizarlo. Para

ello, aplicando el método del descenso del gradiente, cada peso de la red se actualiza para cada patrón de entrada, siendo esta la tasa de aprendizaje.

Dado que las neuronas de la red están agrupadas en niveles de capas ocultas, es posible aplicar de forma eficiente el método del descenso del gradiente, siguiendo para ello la regla delta generalizada.

Regla delta generalizada

"La regla delta generalizada no es más que una forma eficiente de aplicar el método de descenso del gradiente a los parámetros de la arquitectura de red. Su uso consiste en ajustar pesos y bias tratando de minimizar la suma del error cuadrático. Para ello, se modifican dichas variables en la dirección contraria al gradiente del error".

Entre las propiedades de la regla delta, destaca su capacidad de generalización. Por ello, las redes neuronales artificiales que son entrenadas con esta técnica dan respuestas más que razonables cuando el sistema recibe entradas que no ha visto antes.

Tasa de aprendizaje

La tasa de aprendizaje es el parámetro que determina la velocidad a la que cambian los pesos de la red y tiene rango [0,1]. Los valores más cercanos a cero hacen que los pesos cambien despacio, acercándose lentamente a la convergencia. Mientras, los valores cercanos a uno hacen que la red converja rápidamente, pudiendo oscilar demasiado una vez encontrado el peso óptimo final y terminar alejándose de él. Por esta razón, encontrar una tasa de aprendizaje óptima es uno de los factores clave para un buen entrenamiento.

Aquí es donde entra en juego un segundo concepto: el término momento, que pondera la influencia del cambio de los pesos en la iteración anterior. Así, cuando los pesos han cambiado mucho sabemos que aún queda lejos la tasa de aprendizaje óptima y hay que avanzar en su búsqueda más rápidamente.

Modo de entrenamiento

Finalmente, es necesario mencionar que el aprendizaje se produce mediante la presentación sucesiva del conjunto de entrenamiento, donde cada presentación completa recibe el nombre de época. Así, el proceso de aprendizaje se repite época tras época, de acuerdo al algoritmo de backpropagation, hasta que los pesos se estabilizan y el rendimiento de la red converge a un valor aceptable.

La forma en que se actualizan los pesos da lugar a dos modos de entrenamiento:

Modo secuencial: En este modo de entrenamiento la actualización de los pesos se produce tras la presentación de cada ejemplo de entrenamiento. Si el conjunto de entrenamiento consta de N ejemplos, el modo secuencial de entrenamiento tiene como resultado N correcciones de pesos durante cada época.

Modo batch(cola): En este modo la actualización de los pesos se produce una vez, tras la presentación de todo el conjunto de entrenamiento. Para cada época se calcula el error cuadrático medio producido por la red.

Si las muestras de entrenamiento se presentan a la red de manera aleatoria, el modo secuencial convierte la búsqueda de pesos en estocástica y disminuye la probabilidad de que el algoritmo de backpropagation quede atrapado en un mínimo local. Por su parte, el uso del modo de entrenamiento batch provee una estimación precisa del vector gradiente, garantizando así la convergencia hacia un mínimo local.

3.1.1.4 Sobre-entrenamiento

Una vez finalizado el proceso de aprendizaje, las redes neuronales han de ser capaces de abordar la tarea para la que hayan sido diseñadas, a partir de lo aprendido mediante un conjunto de datos de entrenamiento. No obstante, para abordar esa tarea de forma satisfactoria, las redes deben tener también

la habilidad de afrontar situaciones distintas a las presentes durante el entrenamiento. Es decir, han de ser capaces de generalizar.

El sobre-entrenamiento o sobreajuste u overfitting es la consecuencia de sobreentrenar el aprendizaje de las redes neuronales, de manera que estas pierdan su capacidad de generalización. Se trata de un problema habitual en estas arquitecturas y en la literatura se proponen distintos mecanismos de regularización para reducir su efecto(Ranzato et al., s/f).

Un mecanismo frecuente en el mundo del aprendizaje automático (el problema de sobreentrenamiento se da también en otros algoritmos de aprendizaje, no sólo en las RNAs) es el early stopping (Prechelt, s/f). Este método propone aislar un subconjunto de muestras de entrenamiento y utilizarlo a modo de validación. Cuando el error sobre el conjunto de validación empeora, se detiene el entrenamiento y se conserva así la capacidad de generalización de la red. Sin embargo, el uso de este método en las arquitecturas profundas no siempre aporta buenos resultados. Requiere un análisis detallado y, por ello, es habitual recurrir a otras opciones.

Uno de los mecanismos más sencillos para reducir el sobreajuste consiste en simplificar el aprendizaje: disminuir el número de capas, el número de neuronas por capa, el número de épocas, etc. Otra opción es incluir una serie de restricciones en los pesos de las neuronas. En ese sentido, destacan: la técnica conocida como weight decay, que consiste en incluir penalizaciones para los pesos grandes en función de sus valores al cuadrado (penalización L2) o absolutos (penalización L1); o la técnica max-norm, que propone restringir a un valor máximo la norma del vector de pesos.

Sin embargo, a veces estos mecanismos no son suficientes para prevenir el sobreajuste y resulta necesario recurrir a otras técnicas más elaboradas. En este sentido, destaca la técnica de dropout(“Dropout (Neural Networks)”, 2019), que propone transformar el entrenamiento de una red compleja en el entrenamiento de varias redes más simples, donde cada red simple es el resultado de ignorar parte de las neuronas de la red original.

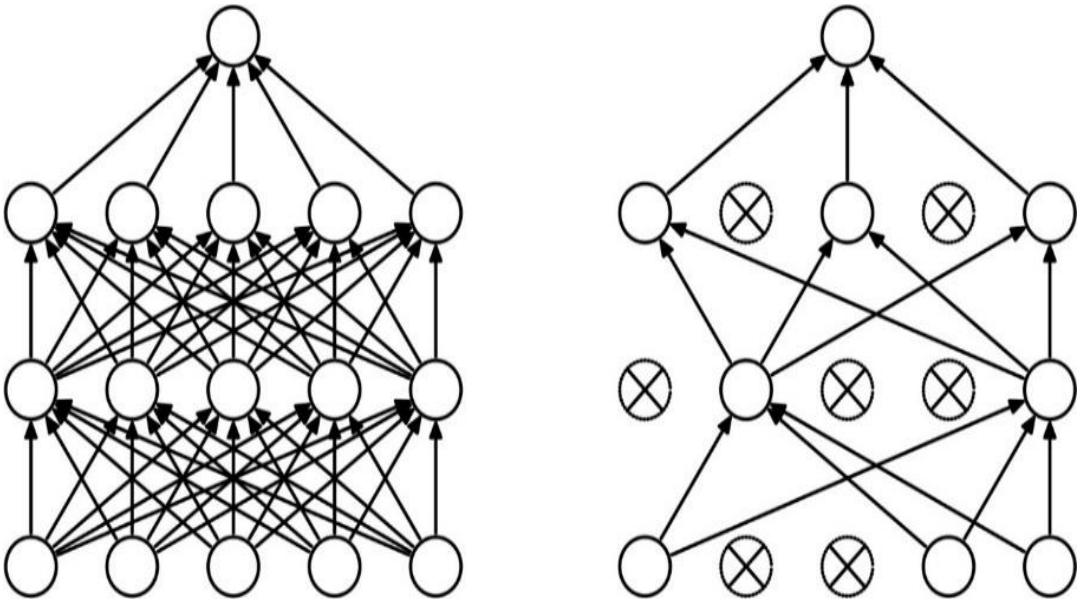


Figura 6. Ejemplo de dropout

Fuente: “Dropout (neural networks)”, 2019

Siguiendo esta idea, si se desactiva una neurona temporalmente, se deja fuera de la propia red y, por tanto, se pierden las conexiones entrantes y salientes asociadas. Esto conduce a un entrenamiento más uniforme y eficiente del modelo, ya que se simula el entrenamiento de varias subredes combinadas en una sola. Para llevar a cabo esta idea, se establece una probabilidad de retención de las neuronas y se genera una máscara aleatoria según esta probabilidad. La arquitectura de red final se obtiene multiplicando cada neurona por su probabilidad de retención correspondiente.

Ejemplo de dropout. A la izquierda se observa la red original y a la derecha esa misma red con dropout, como se muestra en la figura 4.

3.1.2 Redes convolucionales

Las redes convolucionales (Convolutional Neural Networks, CNN) (*Deep Learning - Libro online de IAAR*, s/f) son muy similares a las redes feedforward. Su principal diferencia radica en la inclusión de capas convolucionales, cuyas

neuronas no están totalmente conectadas: cada neurona de una capa no recibe conexiones entrantes de todas las neuronas de la capa anterior, sino sólo de algunas, lo cual favorece que cada neurona se especialice únicamente en una región de la capa anterior, reduciendo drásticamente el número de operaciones a realizar. De esta forma, las redes convolucionales dividen y modelan la información en partes más pequeñas, para combinar después esta información en las capas más profundas de la red.

Por ejemplo, en el caso del tratamiento de una imagen: las primeras capas de la red buscarían contornos, las siguientes tratarían de detectar formas a partir de los contornos, las próximas prestarían atención a la posición de las formas, etc. Por último, en las capas finales del modelo, se combinarían todos los patrones descubiertos para conseguir una predicción final de la suma de todos ellos. Así es como las redes convolucionales consiguen modelar una gran cantidad de datos: dividiendo el problema en partes para conseguir predicciones más sencillas y precisas.

“En los últimos años el Deep Learning ha producido toda una revolución en el campo del Machine Learning, con resultados notables en todos los problemas de percepción, como ver y escuchar, problemas que implican habilidades que parecen muy naturales e intuitivas para los seres humanos, pero que desde hace tiempo se han mostrado difíciles para las máquinas. En particular, el Deep Learning ha logrado los siguientes avances, todos ellos en áreas históricamente difíciles del Machine Learning.

- Un nivel casi humano para la clasificación de imágenes.
- Un nivel casi humano para el reconocimiento del lenguaje hablado.
- Un nivel casi humano en el reconocimiento de escritura.
- Grandes mejoras en traducciones de lenguas.
- Grandes mejoras en conversaciones text-to-speech.
- Asistentes digitales como Google Now o Siri.
- Un nivel casi humano en autos autónomos.
- Mejores resultados de búsqueda en la web.
- Grandes mejoras para responder preguntas en lenguaje natural.

En muchos sentidos, el Deep Learning todavía sigue siendo un campo misterioso para explorar, por lo que seguramente veremos nuevos avances en nuevas áreas utilizando estas técnicas. Tal vez algún día el Deep Learning ayuda a los seres humanos a hacer ciencia, desarrollar software y mucho más" (*Deep Learning - Libro online de IAAR*, s/f).

3.1.1.5 Estructura

Para establecer el mapeo entre entradas y salidas, las redes convolucionales utilizan una arquitectura de red constituida por 5 tipos de capas: capa de entrada, capas convolucionales, capas de pooling, capas ocultas y capa de salida.

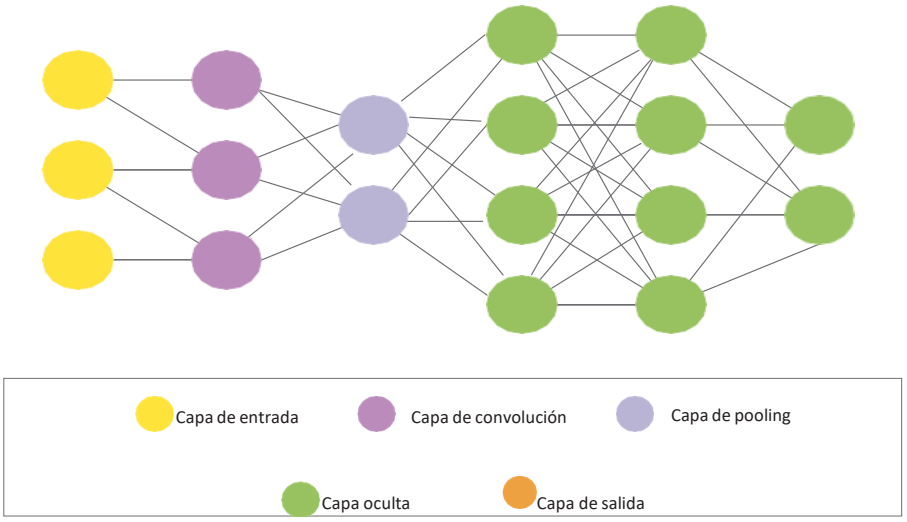


Figura 7. Esquema básico de una red convolucional.

Fuente: "Deep Learningâ□"Libro online de IAAR", s/f

Las capas de entrada y de salida se ajustan a las expuestas para el caso de las redes feedforward, mientras, el resto de capas son propias de este tipo de arquitectura:

Capas convolucionales: Son las capas que dan nombre a la red y se caracterizan por aplicar operaciones de convolución en lugar de productos entre matrices. En este caso, las matrices de pesos son filtros de un tamaño prefijado que se convolucionan con los datos de trabajo, devolviendo una señal filtrada de menor tamaño que la original. Estas capas quedan determinadas por el número de filtros que incorporan y por el tamaño de los mismos.(Jarrett et al., 2009)

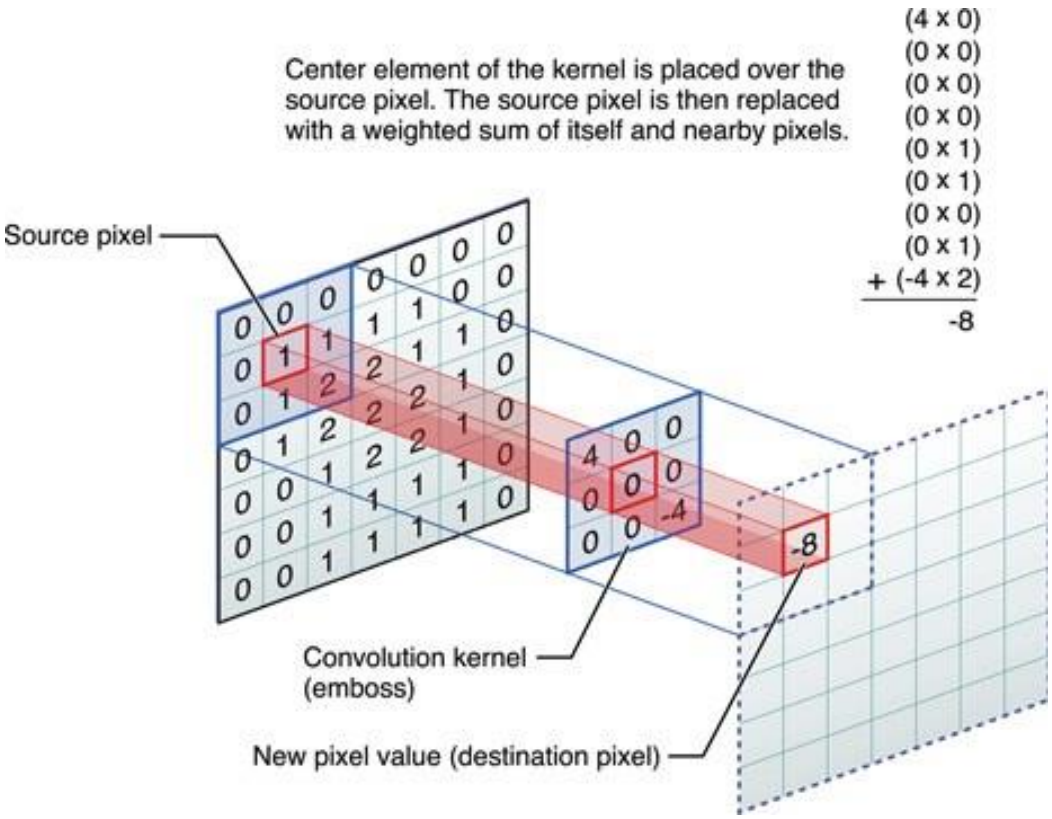


Figura 8. Capas convolucionales

Fuente: “Deep Learningâ□”Libro online de IAAR”, s/f

Capas de pooling o submuestreo: Estas capas se sitúan a la salida de las capas convolucionales y se utilizan para reducir la dimensionalidad de los datos. Esto implica la reducción del coste computacional de las siguientes convoluciones (si las hubiese), del número de parámetros a determinar, proporciona un grado de invarianza ante traslaciones y ayuda también a controlar el sobreajuste de la arquitectura. La operación de pooling requiere determinar el tamaño de la región sobre la que se desea aplicar el submuestreo y el tipo de pooling a realizar. Típicamente, se recurre al max-

pooling (se toma el valor máximo de la región) o al average-pooling (se toma el valor medio de la región).

Capas ocultas: Tienen las mismas características que en el caso de las redes feedforward. Se diferencian en que, en este caso, la primera capa oculta tiene una labor especial: es la que se conecta a la salida de las capas de convolución y pooling, encargándose de convertir la matriz de datos en un vector plano. Por ello, esta capa suele recibir el nombre de capa flatten. Cabe mencionar que las capas ocultas son las únicas totalmente conectadas en esta arquitectura.

Al igual que en las redes feedforward, las capas de entrada y salida son únicas, mientras que el número del resto de capas es variable. Esto da lugar a infinidad de posibilidades y la elección de la arquitectura definitiva dependerá del problema a tratar. Asimismo, deben elegirse otros parámetros, como el número de filtros de convolución y su tamaño, o el tipo y tamaño de las operaciones de submuestreo, los cuales tendrán una gran influencia en el comportamiento de la red.

3.1.3 MNIST

Para nuestra investigación utilizamos la base de datos MNIST, la cual contiene 60,000 imágenes que utilizamos para el entrenamiento de nuestro modelo de clasificación y 10,000 imágenes para la etapa de prueba; dichas imágenes las podemos observar en el ANEXO 01, para mayor referencia.

3.1.4 Curvas ROC

Las curvas ROC (“Curva ROC”, 2019), en la Teoría de detección de señales, una curva ROC (acrónimo de Receiver Operating Characteristic, o Característica Operativa del Receptor) es una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario según se varía el umbral de discriminación. “Otra interpretación de este gráfico es la representación de la razón o ratio de verdaderos positivos (VPR = Razón de Verdaderos Positivos) frente a la razón o ratio de falsos positivos (FPR = Razón de Falsos Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo)”. ROC también puede significar Relative

Operating Characteristic (Característica Operativa Relativa) porque es una comparación de dos características operativas (VPR y FPR) según cambiamos el umbral para la decisión. En español es preferible mantener el acrónimo inglés, aunque es posible encontrar el equivalente español COR. No se suele utilizar ROC aislado, debemos decir "curva ROC" o "análisis ROC".

El análisis de la curva ROC, o simplemente análisis ROC, proporciona herramientas para seleccionar los modelos posiblemente óptimos y descartar modelos subóptimos independientemente de (y antes de especificar) el coste de la distribución de las dos clases sobre las que se decide. La curva ROC es también independiente de la distribución de las clases en la población (en diagnóstico, la prevalencia de una enfermedad en la población). El análisis ROC se relaciona de forma directa y natural con el análisis de coste/beneficio en toma de decisiones diagnósticas.

"La curva ROC se desarrolló por ingenieros eléctricos para medir la eficacia en la detección de objetos enemigos en campos de batalla mediante pantallas de radar, a partir de lo cual se desarrolla la Teoría de Detección de Señales (TDS). El análisis ROC se aplicó posteriormente en medicina, radiología, psicología y otras áreas durante varias décadas. Sólo recientemente ha encontrado aplicación en áreas como aprendizaje automático (o machine learning en inglés), y minería de datos (data mining en inglés)"(Ferri et al., s/f).

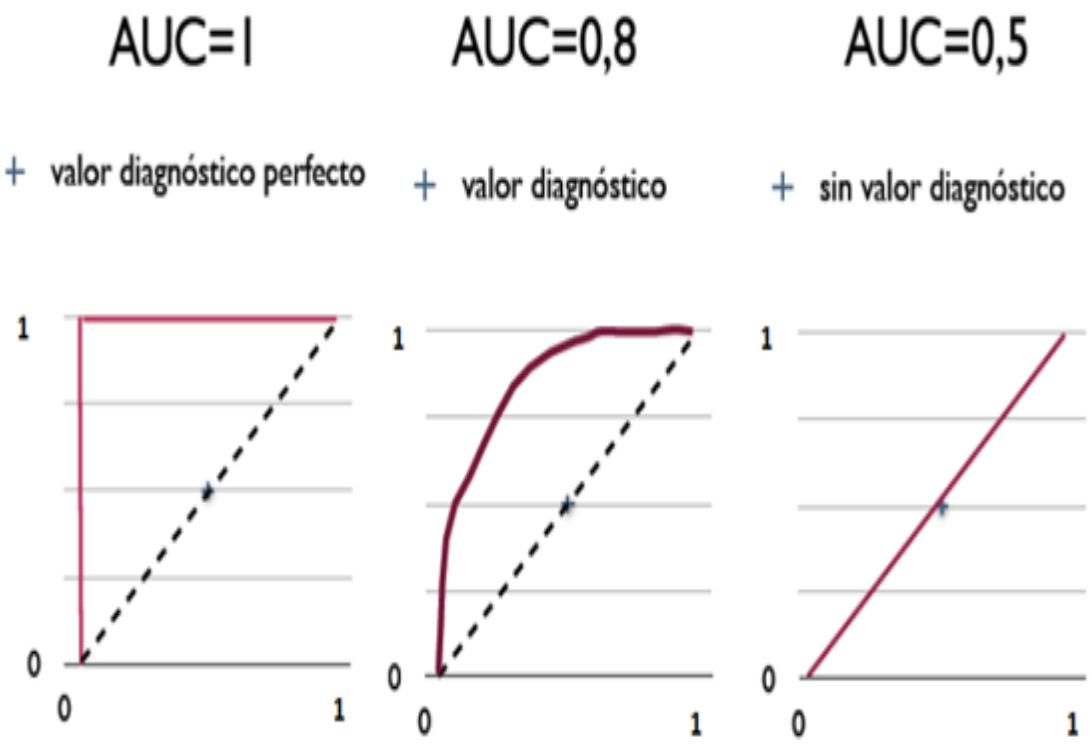


Figura 9. Ejemplos de curvas ROC

Fuente: ("Curva ROC", 2019)

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 Red Neuronal Propuesta

4.1.1 Modelo

El modelo propuesto para efectuar el reconocimiento de patrones de las imágenes propuestas es de una red neuronal paralela compuesta por una red neuronal que tiene 196 neuronas en la capa de entrada, 110 neuronas en la capa escondida y 10 neuronas para la capa de salida.

4.1.2 Datos de entrenamiento

Para realizar el entrenamiento de nuestro modelo utilizamos las 60000 imágenes contenidas en el repositorio de MNIST.

4.1.3 Tratamiento de Datos de entrenamiento

Los datos de las imágenes las procesamos en una matriz de 28 x 28, aplicando la función de escalón para su normalización.

Proceso de aprendizaje:

$$w(t+1) = w(t) + \Delta w(t), \text{ donde:}$$

$w(t+1)$: *valor actualizado al peso sináptico*

$w(t)$: *valor actual del peso sináptico*

$\Delta w(t)$: *variación del peso sináptico*

Aprendizaje supervisado:

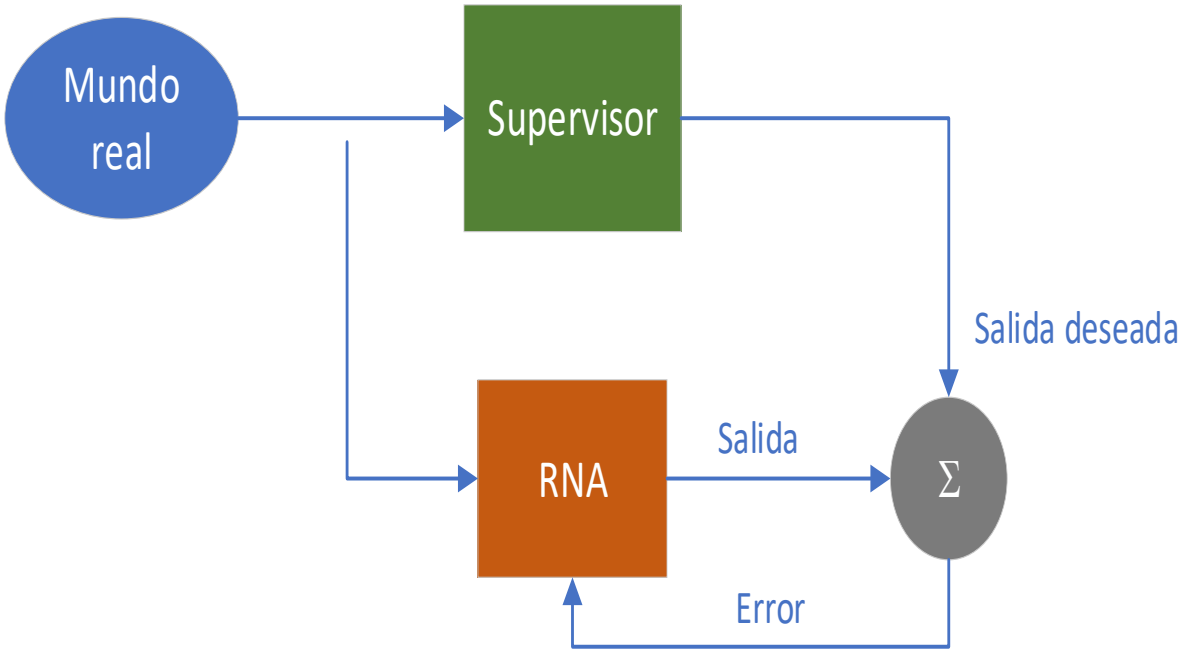


Figura 10. Aprendizaje supervisado

$error = d - y$, donde:

y: $salida\ de\ la\ red\ neuronal\ artificial$

d: $salida\ deseada$

Sea:

$X = \{x_1, x_2, \dots, x_p\}$ $conjunto\ de\ vectores\ de\ entrada$

$D = \{d_1, d_2, \dots, d_p\}$ $conjunto\ de\ vectores\ de\ salida$

Donde cada patrón de entrenamiento está constituido el par ordenado de vectores $\{x_p, d_p\}$, donde $p = 1 \dots 60000$; los cuales se pueden definir de la siguiente forma:

$x_p = \{x_{p1}, x_{p2}, \dots, x_{pN}\}$ *elementos de entrada del patrón de entrenamiento*

$d_p = \{d_{p1}, d_{p2}, \dots, d_{pM}\}$ *elementos de salida del patrón de entrenamiento*

El error global se calcula en función de la ecuación(Williams & Zipser, David, s/f):

$$E_p = \frac{1}{2p} \sum_{p=1}^{P} \sum_{j=1}^{M} (d_{pj} - y_{pj})^2$$

donde:

M: *número de neuronas en la capa de salida*

P: *número de patrones de entrenamiento*

Para el caso del modelo propuesto $M = 10$ y $P = 60000$.

Modelo de red neuronal:

Para la capa de entrada a nuestra red neuronal se utilizarán las imágenes de números arábigos contenidos en MNIST en forma de vector de la forma: $x_p = \{x_1, x_2, \dots, x_{784}\}$, luego generamos la matriz de oro utilizando la fórmula:

$$f_{n-2} + f_{n-1} - f_n = 0$$

formula para la generación de la matriz de oro

En nuestro caso los valores iniciales para la generación de la MATRIZ DE ORO son: $f_{n-2} = 0$ y $f_{n-1} = 1$.

Por otro lado, cabe mencionar que en el modelo utilizamos imágenes de manuscritos de números contenidos en el dataset de MNIST (Modified National Institute of Standards and Technology database) el cual costa de dos partes. La primera con 60000 imágenes de 28 x 28 pixels de tamaño, que usamos para el

entrenamiento del modelo. La segunda parte del conjunto de datos de MNIST tiene 10000 imágenes que usamos para las pruebas.

El conjunto de datos de MNIST puede ser obtenido de la página web: http://yann.lecun.com/exdb/mnist/ (*MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*, 2018) de aquí obtenemos los siguientes archivos comprimidos:

- train-images-idx3-ubyte.gz: imágenes de entrenamiento (9912422 bytes)
- train-labels-idx1-ubyte.gz: etiquetas de las imágenes de entrenamiento (28881 bytes)
- t10k-images-idx3-ubyte.gz: imágenes para la etapa de pruebas (1648877 bytes)
- t10k-labels-idx1-ubyte.gz: etiquetas de las imágenes de pruebas (4542 bytes)

En nuestro trabajo procesamos las imágenes y etiquetas contenidas en dichos archivos utilizando la herramienta MATLAB a través de dos funciones: loadMNISTImages y loadMNISTLabels cuya codificación la mostramos en el ANEXO 3.

Las imágenes tienen una estructura de matriz de 784 x 60000 para las imágenes y de 60000 x 1 para las etiquetas que se usan en la etapa de entrenamiento; mientras que para la etapa de pruebas las matrices son de 784 x 10000 y 10000 x 1 respectivamente.

Diseño de la red neuronal

Para su implementación utilizaremos la herramienta MATLAB (2018a), tratándolo como un problema de reconocimiento de patrones con la arquitectura que se muestra en la figura 9.

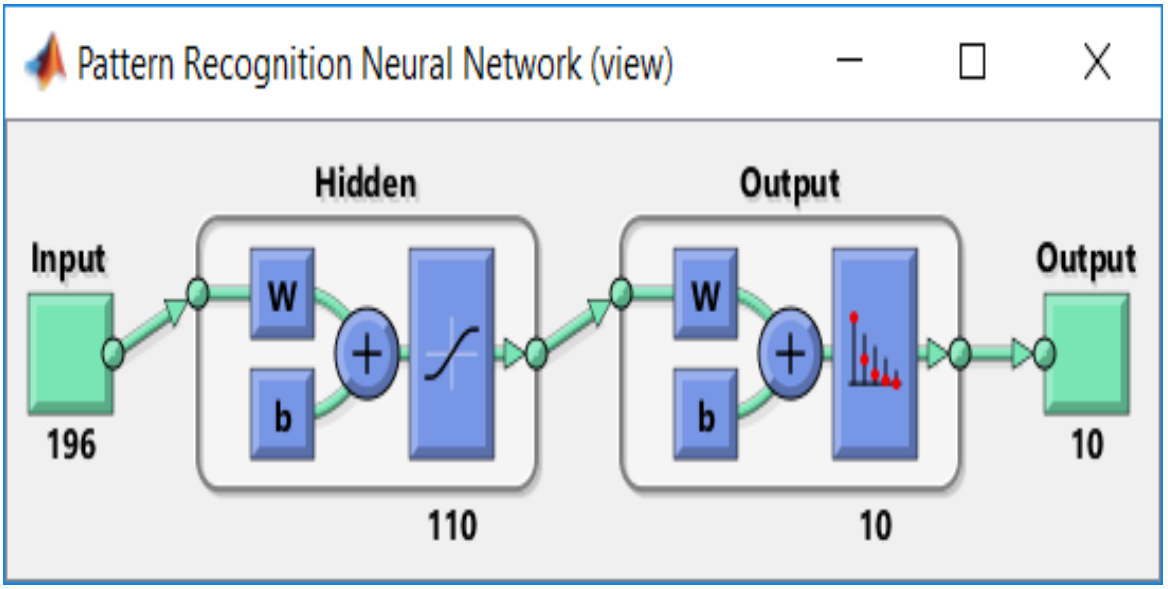


Figura 11. Arquitectura de la red neuronal propuesta

Como se puede apreciar las neuronas de entrada a la red neuronal son de 196, resultantes de una matriz de 14 x14 que difiere de la matriz original de 28 x 28, en razón a que esta sufrió una transformación resultado de la multiplicación de la matriz original por la matriz de oro de la forma siguiente:

$$\begin{bmatrix} x_{1,1} & \cdots & x_{1,28} \\ \cdots & \cdots & \cdots \\ x_{28,1} & \cdots & x_{28,28} \end{bmatrix} x \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,14} \\ \cdots & \cdots & \cdots \\ r_{14,1} & \cdots & r_{14,14} \end{bmatrix}$$

La nueva matriz resultante R es la que procedemos a entrenar, utilizando 110 neuronas en la capa escondida, para obtener una clasificación de acuerdo a la siguiente tabla:

Clase	*Etiqueta*	*Valor a obtener*
Clase 1	0	1000000000
Clase 2	1	0100000000
Clase 3	2	0010000000
Clase 4	3	0001000000
Clase 5	4	0000100000
Clase 6	5	0000010000
Clase 7	6	0000001000
Clase 8	7	0000000100
Clase 9	8	0000000010
Clase 10	9	0000000001

Tabla 1. Descripción de clases a clasificar

En el entrenamiento de la red neuronal utilizamos “scaled conjugate gradient backpropagation” (trainscg)(Ph.D Scholar, Computer Science Dept., Mohanlal Sukhadia University, Udaipur, India et al., 2014; Williams & Zipser, David, s/f) y la función de performance “crossentropy” (Nasr et al., 2002) , con una red de dos capas del tipo feed-forward cuya salida será de 10 neuronas con los valores descritos en la tabla 1.

Luego de un total de 1158 épocas, en un tiempo aproximado de 11 minutos y 18 segundos en una computadora con procesador core i5, utilizando “Scaled Conjugate Gradient”(Williams & Zipser, David, s/f), obtenemos el modelo de red neuronal apropiado cuyos detalles se aprecian en la figura 10.

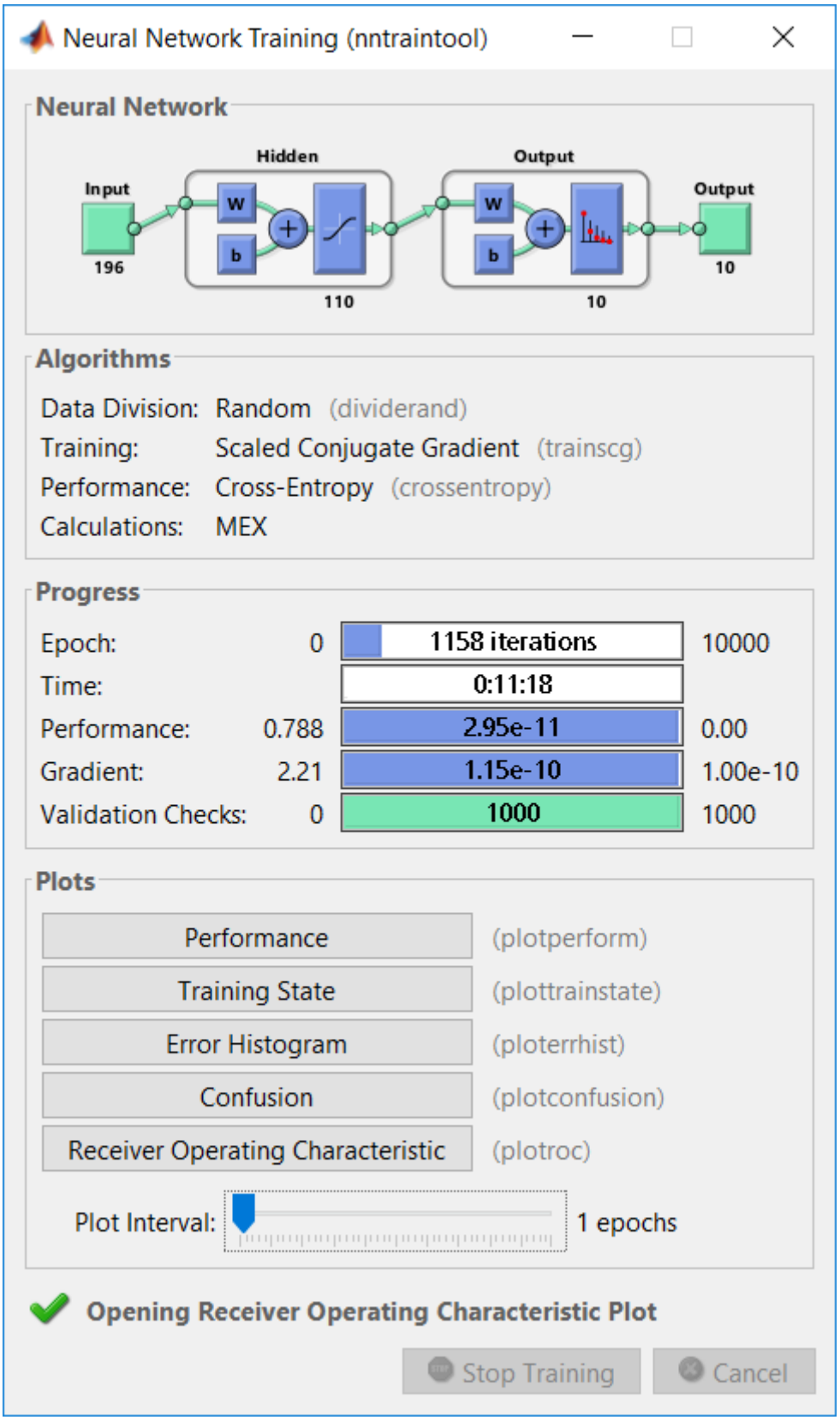


Figura 12. Resultados de entrenamiento de la red neuronal propuesta

Estos resultados como podemos apreciar en la figura 11 son satisfactorios ya que el porcentaje de clasificación está por encima del 90% de clasificaciones correctas tal como se planteó en la hipótesis del trabajo, cuyo detalle por cada clase también se puede apreciar en dicha figura; del mismo que podemos describir que conseguimos un 96.5% de clasificaciones correctas para la clase 1(0), 98.3% de clasificaciones correctas para la clase 2(1), 96.2% de clasificaciones correctas para la clase 3(2), 97.0% de clasificaciones correctas para la clase 4(3), 95.1% de clasificaciones correctas para la clase 5(4), 92.2% de clasificaciones correctas para la clase 6(5), 96.5% de clasificaciones correctas para la clase 7(6), 97.5% de clasificaciones correctas para la clase 8(7), 95.1% de clasificaciones correctas para la clase 9(8), 93.5% de clasificaciones correctas para la clase 10(9); obteniendo un promedio ponderado del 95.9% de clasificaciones correctas.

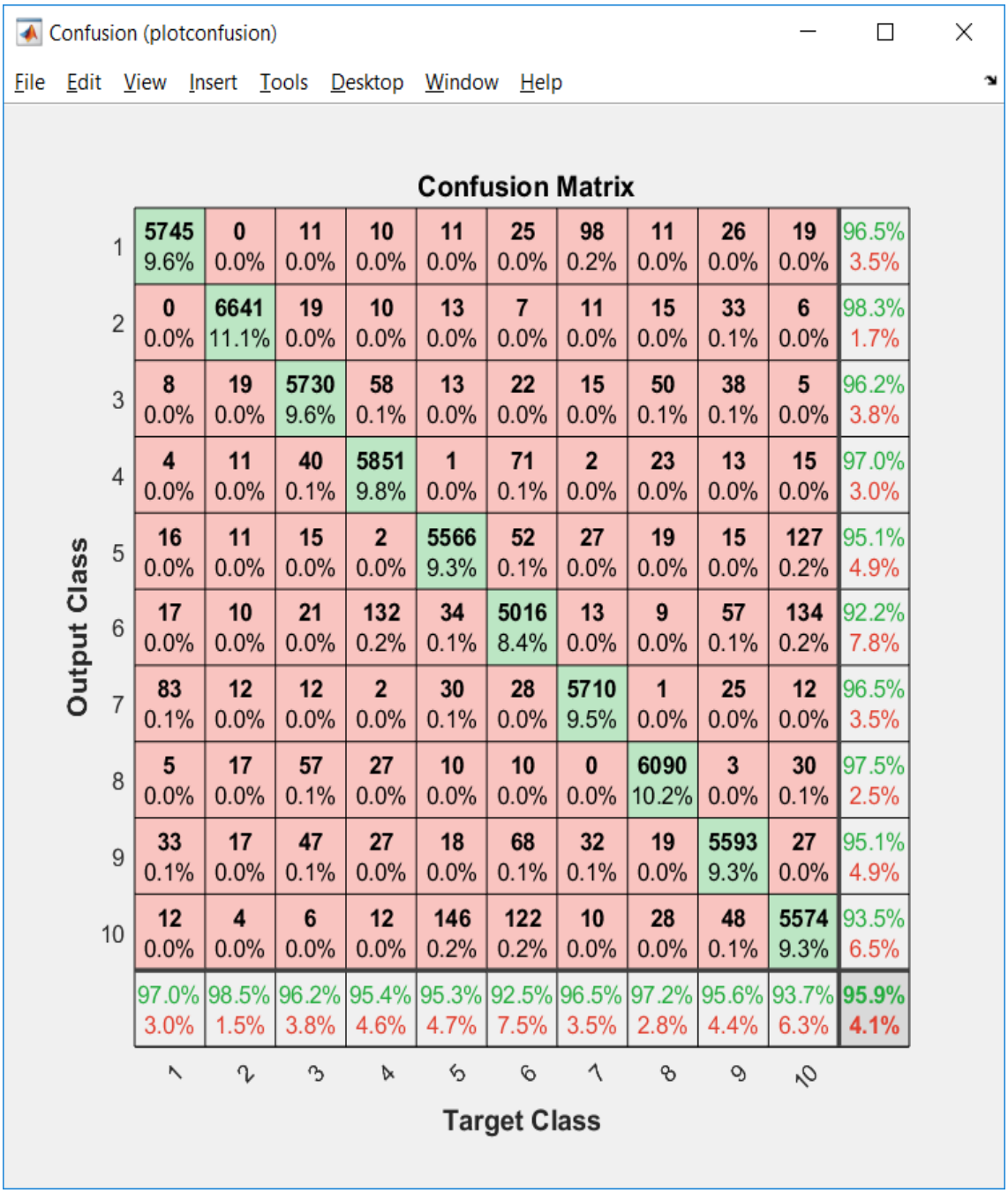


Figura 13. Matriz de confusión 60000 imágenes MNIST

Estos resultados los contrastamos con la representación de las diferentes etapas a través de curvas ROC, cuyos resultados son muy favorables cuyo detalle se aprecia en la figura 12.

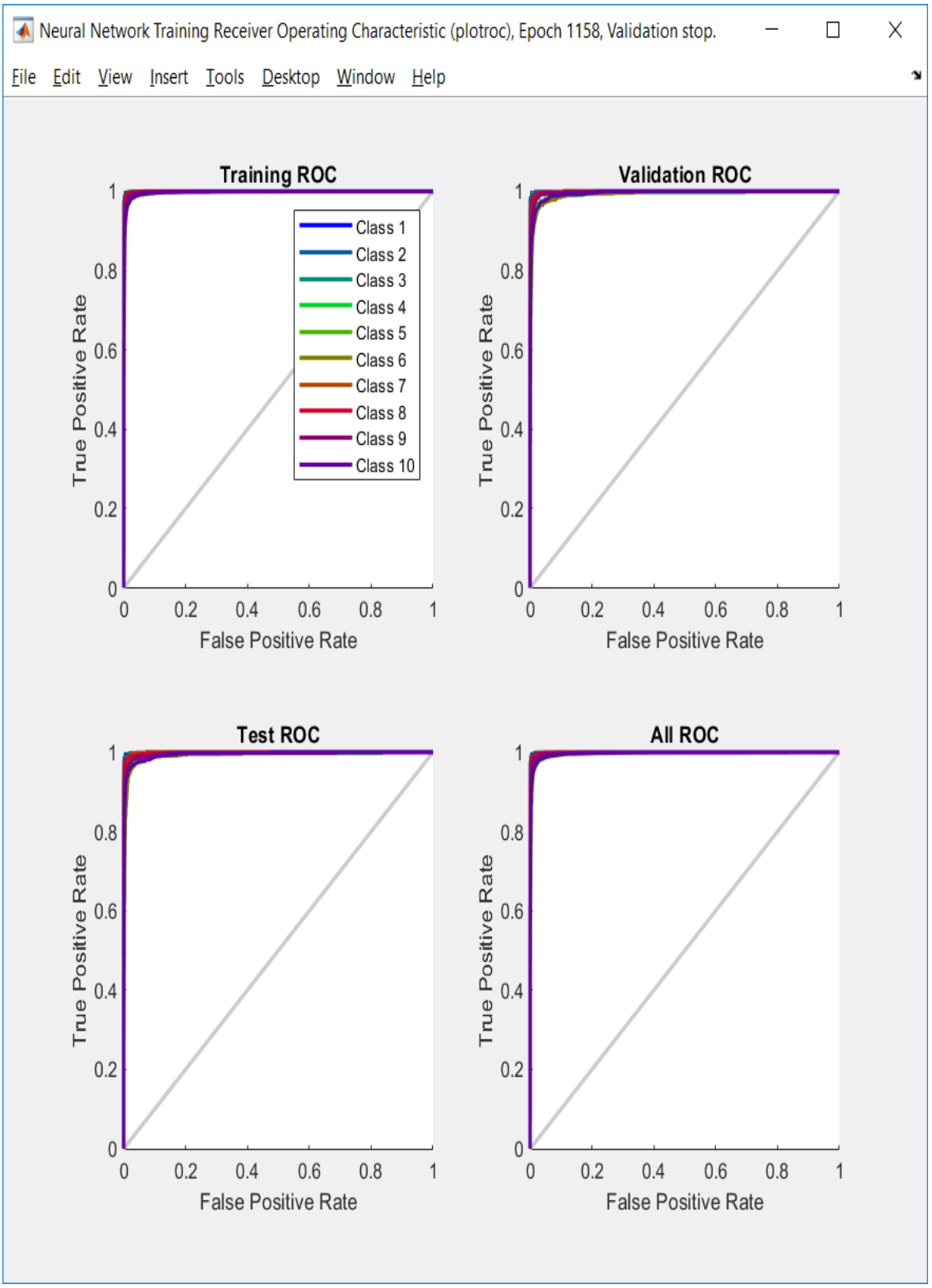


Figura 14. Resultados obtenidos expresados a través de curvas ROC

Resultado etapa de Pruebas

Para esta etapa utilizaremos el dataset de MNIST que corresponde a 10000 imágenes con sus correspondientes etiquetas, pa el mismo que a diferencia de otros autores no efectuamos la normalización de los datos de entrada dividiéndola entre 255 para convertirla a escala de grises, solamente necesitamos utilizar la matriz de oro para convertirla en una matriz de 14 x 14 y luego esta nueva matriz que corresponde a la matriz:

$$XT = \{x_1, x_2, \dots, x_{10000}\} \quad conjunto\ de\ vectores\ de\ entrada$$

$$DT = \{d_1, d_2, \dots, d_{10000}\} \quad conjunto\ de\ vectores\ de\ salida$$

Haciendo uso de la red neuronal entrenada, podemos observar que con las 10000 imágenes para las pruebas también obtenemos un porcentaje de clasificación superior al 90% cuyo detalle se muestra en la figura 13; así como la validación del mismo a través de la imagen de curva ROC representada en la figura 14.

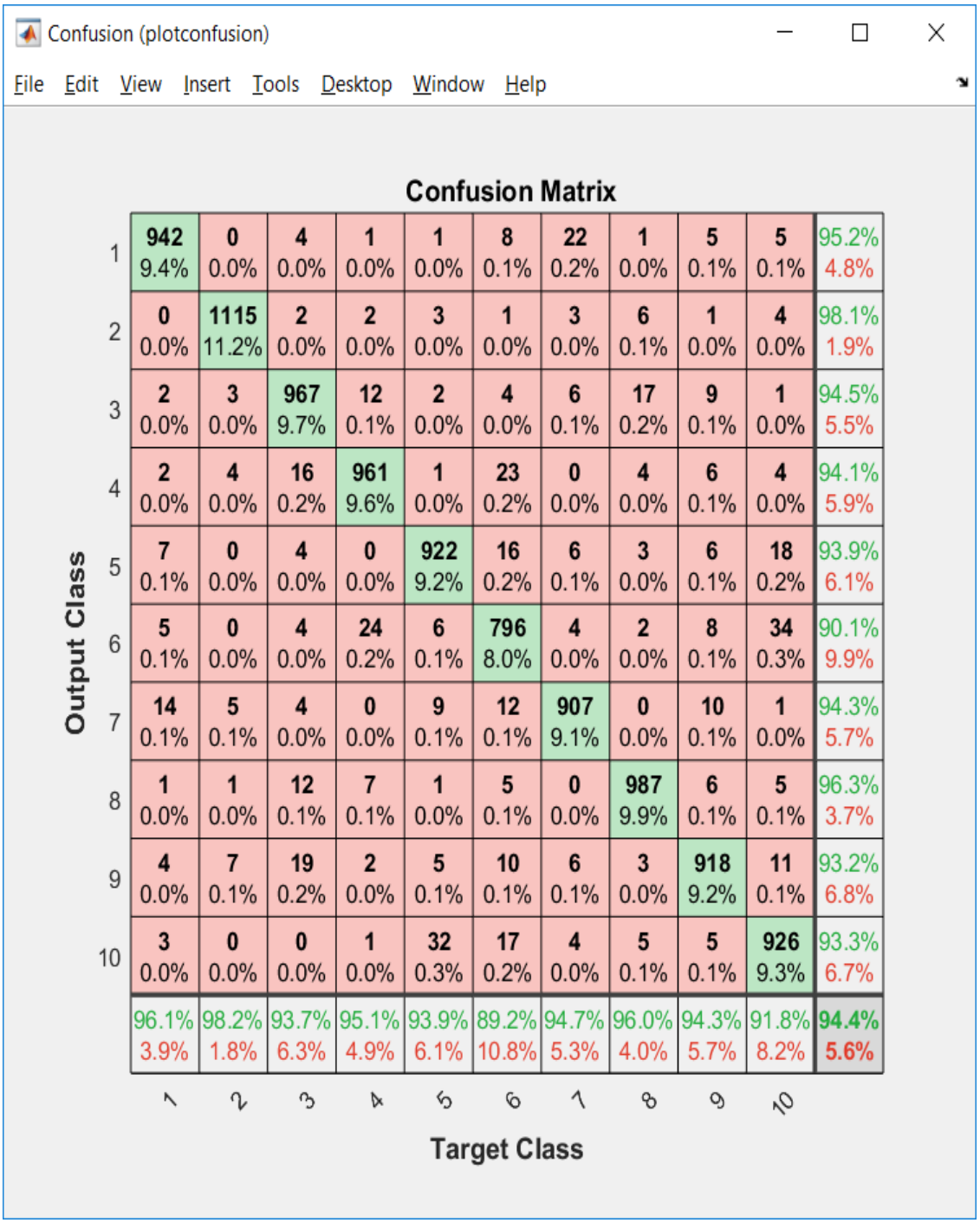


Figura 15. Matriz de confusión de los 10000 datos de pruebas

Como podemos apreciar se obtuvo un 95.2% de clasificaciones correctas para la clase 1(0), 98.1% de clasificaciones correctas para la clase 2(1), 94.5% de clasificaciones correctas para la clase 3(2), 94.1% de clasificaciones correctas para la clase 4(3), 93.9% de clasificaciones correctas para la clase 5(4), 90.1% de clasificaciones correctas para la clase 6(5), 94.3% de clasificaciones correctas para

la clase 7(6), 96.3% de clasificaciones correctas para la clase 8(7), 93.2% de clasificaciones correctas para la clase 9(8), 93.3% de clasificaciones correctas para la clase 10(9); teniendo un promedio ponderado del 94.4% de clasificaciones correctas.

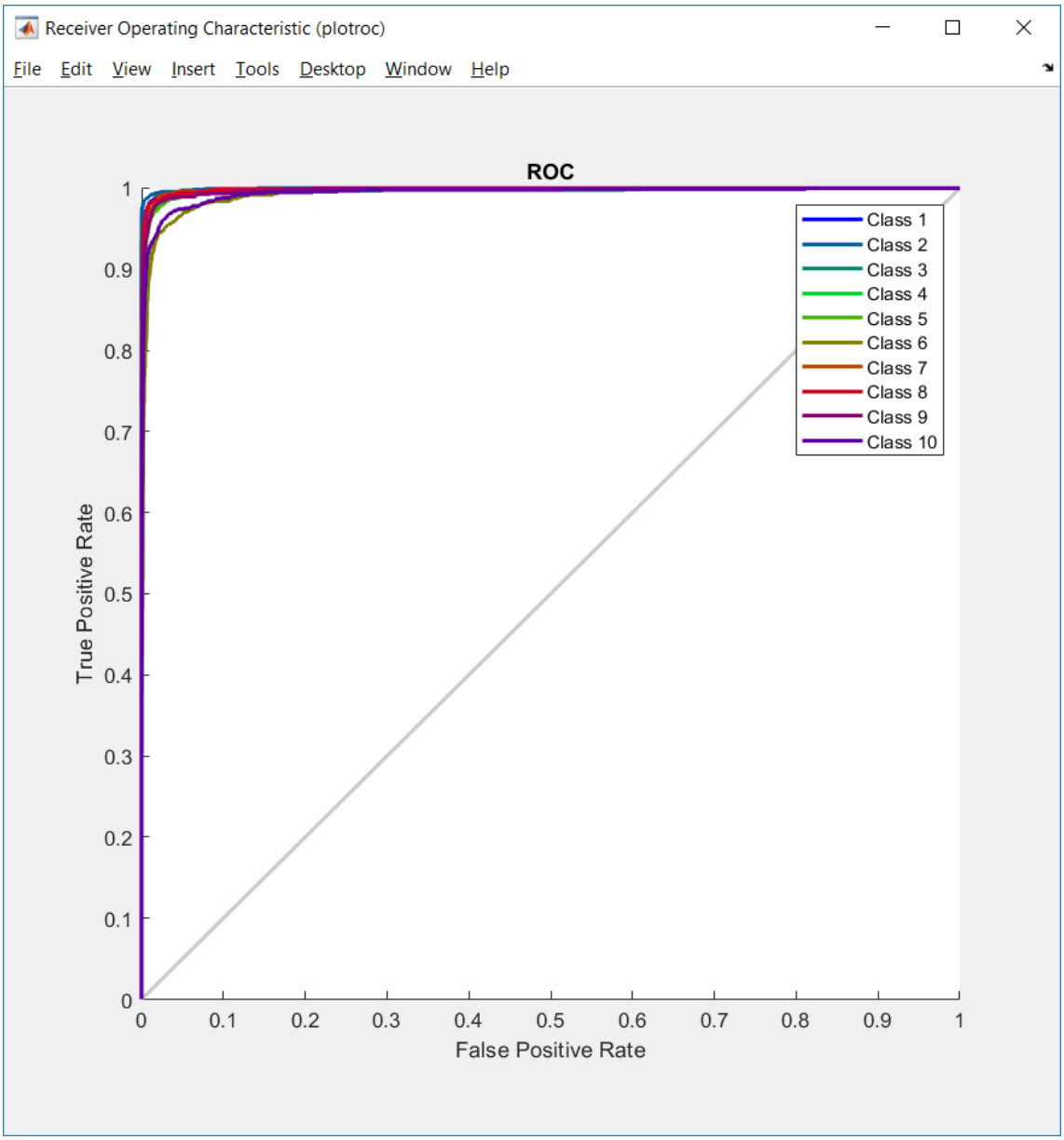


Figura 16. Curva ROC de la clasificación de 10000 imágenes de la etapa de pruebas

CONCLUSIONES

- Se logro diseñar e implementar un modelo computacional basado en redes neuronales artificiales para clasificar de manera automática manuscritos de números arábigos, que en su etapa de entrenamiento obtuvo un promedio ponderado del 95.9% de clasificaciones correctas y en la etapa de pruebas un promedio ponderado del 94.4% de clasificaciones correctas.
- La implementación de la base de datos de manuscritos se realizó utilizando el dataset propuesto por MNIST, considerando en una primera parte 60000 imágenes con sus correspondientes etiquetas para la etapa de entrenamiento y 10000 imágenes con sus etiquetas para la etapa de pruebas, utilizando para ello una matriz de 14 x 14, es decir 196 patrones para cada imagen.
- El clasificador automático se basa en un modelo de red neuronal artificial multicapa con 196 neuronas de entrada, 110 neuronas en la capa escondida y 10 neuronas en la capa de salida, el mismo que fue entrenado en 1158 épocas durante 11 minutos con 18 segundos.
- Para la validación del clasificador automático se utilizó curvas ROC, obteniendo un resultado por encima de 0.9 para cada una de las clases, considerándolo como muy bueno.

RECOMENDACIONES

- Se recomienda utilizar la matriz de oro para efectuar investigaciones relacionadas al procesamiento digital de imágenes.
- En la presente investigación utilizamos el dataset de MNIST con los 10 digitos diferentes de los números arábigos, sugerimos ampliar su aplicación a todo el alfabeto.
- No fueron necesarios normalizar los datos de entrada por no obtener diferencias significativas con estos, por lo que recomendamos ampliar su estudio.
- Recomendamos investigar sobre la aplicación del presente método en algoritmos de inteligencia colectiva.

BIBLIOGRAFÍA

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv:1803.08375 [cs, stat]*. http://arxiv.org/abs/1803.08375

Aprendizaje profundo. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Aprendizaje_profundo&oldid=110466517

Bose, N. K. (s/f). *NEURAL NETWORK FUNDAMENTALS WITH GRAPHS, ALGORITHMS, AND APPLICATIONS*. 6.

Brain, I. T., & Rosenblatt, F. (s/f). *The Perceptron: A Probabilistic Model for Information Storage and Organization*.

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column Deep Neural Networks for Image Classification. *arXiv:1202.2745 [cs]*. http://arxiv.org/abs/1202.2745

Curva ROC. (2019). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Curva_ROC&oldid=117636857

Deep Learningâ€ Libro online de IAAR. (s/f). Recuperado el 12 de abril de 2019, de https://iaarbook.github.io/deeplearning/

Dropout (neural networks). (2019). En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Dropout_(neural_networks)&oldid=890439084

Edje, L., Miller, C., Kiefer, J., & Oram, D. (2013). Using Skype as an Alternative for Residency Selection Interviews. *Journal of Graduate Medical Education*, *5*(3), 503–505. https://doi.org/10.4300/JGME-D-12-00152.1

Facebook. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Facebook&oldid=111613860

Ferri, C., Flach, P., & Hernández-Orallo, J. (s/f). *Learning Decision Trees Using the Area Under the ROC Curve*. 8.

Field-programmable gate array. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Field-programmable_gate_array&oldid=111196394

Franc, V., & Hlavaˇc, V. (s/f). *Statistical Pattern Recognition Toolbox for Matlab*. 99.

García Navarro, B. (s/f). *Implementacion de Tecnicas de Deep Learning.pdf*. Recuperado el 30 de octubre de 2018, de https://riull.ull.es/xmlui/bitstream/handle/915/1409/Implementacion%20de%20Tecnicas%20de%20Deep%20Learning.pdf?sequence=1

Garrido Rojas, E. (s/f). Garrido_Rojas_Eduart_Numeros_Manuscritos.pdf. Recuperado el 30 de octubre de 2018, de http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/903/GARRIDO_ROJAS_EDUART_NUMEROS_MANUSCRITOS.pdf?sequence=1

Google Now. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Google_Now&oldid=107324143

Grau Moreso, V. (2014). Adaptación de algoritmos de aprendizaje automático para su ejecución sobre GPUs. *Ingeniería del agua*, *18*(1), ix. https://doi.org/10.4995/ia.2014.3293

Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *2009 IEEE 12th International Conference on Computer Vision*, 2146–2153. https://doi.org/10.1109/ICCV.2009.5459469

López, G., Quesada, L., & Guerrero, L. A. (2018). Alexa vs. Siri vs. Cortana vs. Google Assistant: A Comparison of Speech-Based Natural User Interfaces. En I. L. Nunes (Ed.), *Advances in Human Factors and Systems Interaction* (pp. 241–250). Springer International Publishing.

Marturet Rodrigo, J. (2018). *Evaluación de redes neuronales convolucionales para la clasificación de imágenes histológicas de cáncer colorrectal mediante transferencia de aprendizaje*. http://openaccess.uoc.edu/webapps/o2/handle/10609/74105

MATLAB. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=MATLAB&oldid=109192691

Microsoft Cortana. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Microsoft_Cortana&oldid=109198248

MNIST database. (2018). En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=865560237

MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. (2018, agosto 20). http://yann.lecun.com/exdb/mnist/

Molnar, C. (s/f). *4.4 Decision Tree | Interpretable Machine Learning*. Recuperado el 11 de septiembre de 2019, de https://christophm.github.io/interpretable-ml-book/tree.html

Monge , M. A. (s/f). Diseño de una arquitectura de red neuronal artificial perceptron multicapa sobre un FPGA aplicada al reconocimiento de caracteres. 92.1

Nasr, G. E., Badr, E., & Joun, C. (2002). *Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand.* 381–384.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Perceptrón multicapa. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Perceptr%C3%B3n_multicapa&oldid=108448948

Ph.D Scholar, Computer Science Dept., Mohanlal Sukhadia University, Udaipur, India, Sharma, B., & K. Venugopalan, Prof. (2014). Comparison of Neural Network Training Functions for Hematoma Classification in Brain CT Images. *IOSR Journal of Computer Engineering*, *16*(1), 31–35. https://doi.org/10.9790/0661-16123135

Prechelt, L. (s/f). *Early Stopping | but when?* 15.

Python. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Python&oldid=111467527

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (s/f). *Efficient Learning of Sparse Representations with an Energy-Based Model*. 8.

Shahdad, M. (1986). An Overview of VHDL Language and Technology. *23rd ACM/IEEE Design Automation Conference*, 320–326. https://doi.org/10.1109/DAC.1986.1586107

Skype. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Skype&oldid=111179470

SwiftKey. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=SwiftKey&oldid=109828070

Tecnicas preprocesamiento de datos. (s/f). Recuperado el 30 de octubre de 2018, de MindMeister website: https://www.mindmeister.com/31824993/tecnicas-preprocesamiento-de-datos

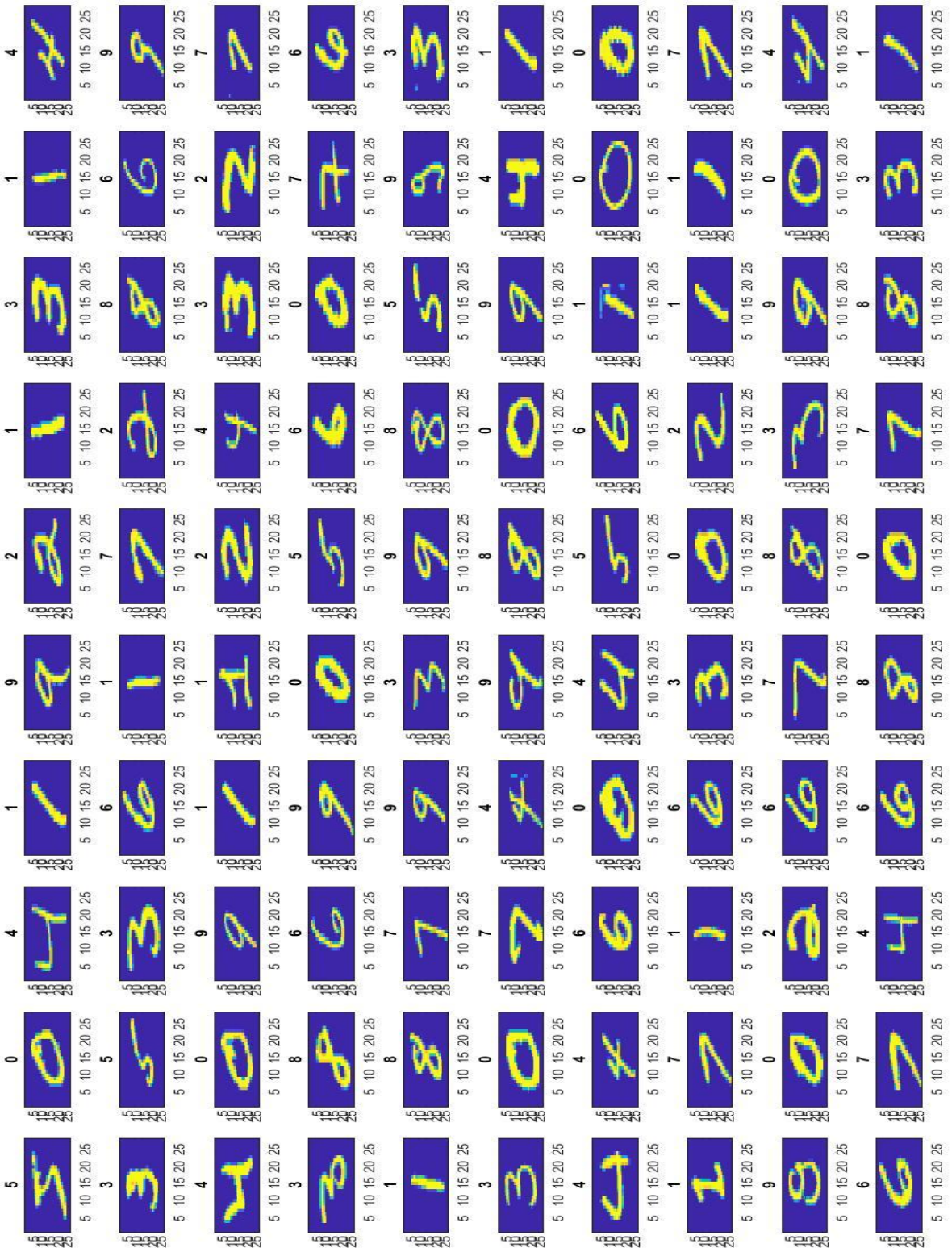
Unidad de procesamiento gráfico. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Unidad_de_procesamiento_gr%C3%A1fico&oldid=110720720

VHDL. (2018). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=VHDL&oldid=109679536

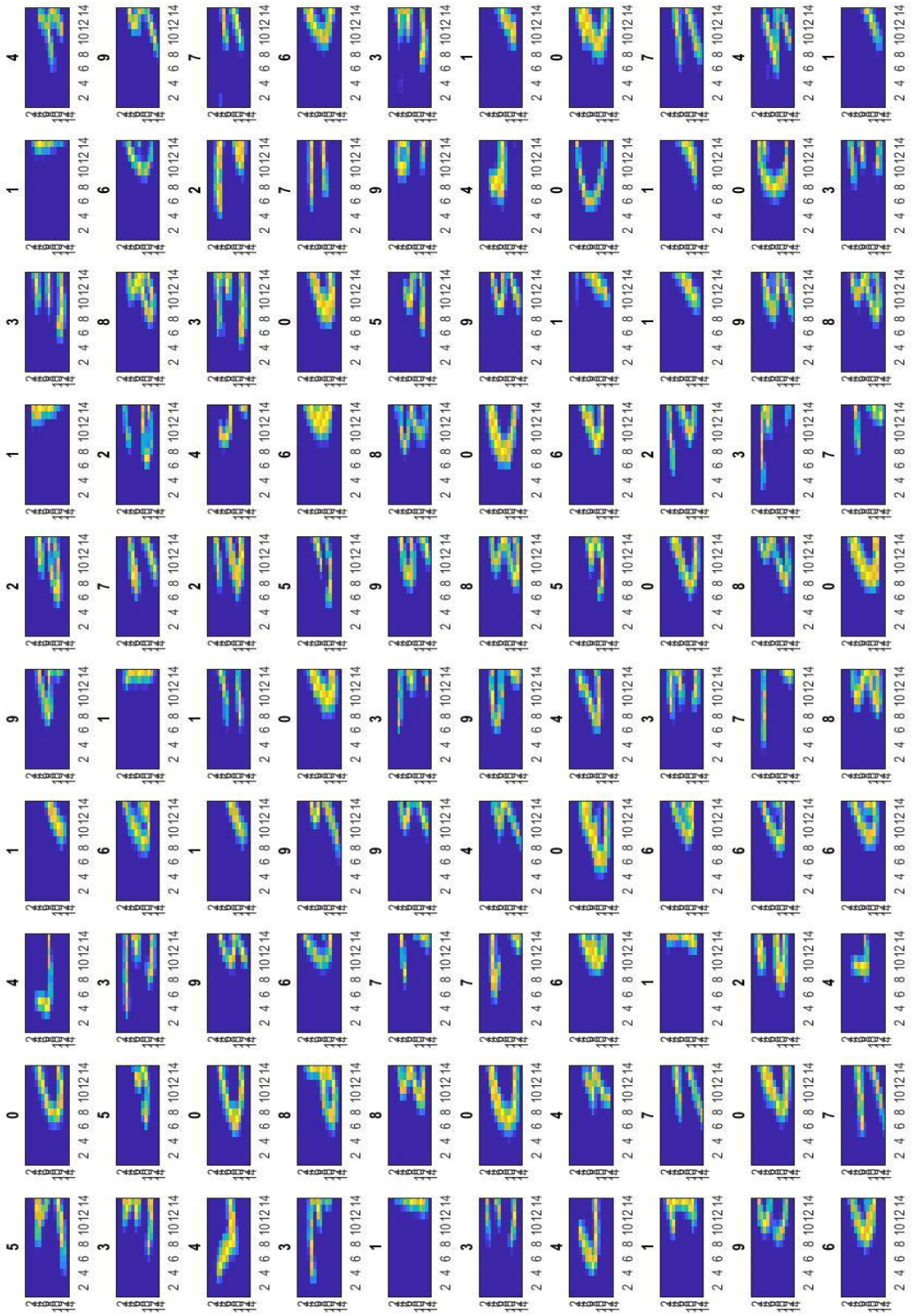
Williams, R. J., & Zipser, David. (s/f). *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*. 45.

ANEXOS

Anexo 1. Cien primeras imágenes para entrenamiento



Anexo2. Imágenes de la matriz equivalente a ser entrenadas



Anexo 3. CODIGO MATLAB

```
function images = loadMNISTImages(filename)

fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename,
'']);

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2051, ['Bad magic number in ',
filename, '']);

numImages = fread(fp, 1, 'int32', 0, 'ieee-be');
numRows = fread(fp, 1, 'int32', 0, 'ieee-be');
numCols = fread(fp, 1, 'int32', 0, 'ieee-be');

images = fread(fp, inf, 'unsigned char');
images = reshape(images, numCols, numRows,
numImages);
images = permute(images,[2 1 3]);

fclose(fp);

images = reshape(images, size(images, 1) *
size(images, 2), size(images, 3));
images = double(images) / 255;
end


function labels = loadMNISTLabels(filename)
fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename,
'']);

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2049, ['Bad magic number in ',
filename, '']);

numLabels = fread(fp, 1, 'int32', 0, 'ieee-be');

labels = fread(fp, inf, 'unsigned char');

assert(size(labels,1) == numLabels, 'Mismatch in
label count');

fclose(fp);

end
```