PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL RECONOCIMIENTO DE CARACTERES BASADO EN LA RED NEURONAL PERCEPTRON

Tesis para optar el Título de **Ingeniero informático,** que presenta el bachiller:

Sammy Nahín Carranza Hernández

ASESOR: Juan Miguel Angel Guanira Erazo

Lima, junio de 2014

RESUMEN

El presente proyecto tuvo como objetivo final construir un sistema basado en el funcionamiento de redes neuronales para el reconocimiento de caracteres dibujados a mano. El proyecto se divide en 2 fases. La primera fase es la de entrenamiento. En esta fase se entrena al sistema con el algoritmo resilient backpropagation. Para esto se trabaja con una data de entrenamiento, los cuales son una seguidilla de dibujos de caracteres hechos a mano. Al final de la fase de entrenamiento se obtiene los parámetros del sistema de red neuronal, con los cuales se podrá configurar el sistema de red neuronal. La siguiente fase es la fase de testeo. En esta fase se busca saber cuan efectivo ha sido el proceso de entrenamiento del sistema de red neuronal. Para esto, se pone a prueba el sistema ingresándole nueva data la cual nunca ha sido vista por el sistema. A esta data, se le llama data de testeo. Al final de esta fase se obtiene el grado de efectividad del sistema en reconocer acertadamente cada carácter ingresado al sistema.

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL RECONOCIMIENTO DE CARACTERES BASADO EN LA RED NEURONAL PERCEPTRON**

ÁREA: CIENCIAS DE LA COMPUTACIÓN

PROPONENTE: Manuel Tupia

ASESOR: Juan Miguel Guanira Erazo

ALUMNO: Sammy Nahín Carranza Hernández

CÓDIGO: 20047001

TEMA N°: 532

FECHA: 09 de septiembre de 2014

DESCRIPCIÓN

Acciones de búsqueda, edición de un material literario digital y el procesamiento de este material por otro software especializado, como por ejemplo un software de lectura solo es posible si existe una capa de datos de caracteres codificados en el material literario.

Por otro lado, en la actualidad existe diversos enfoques con que se intenta transformar una imagen de un caracter en un caracter codificado, sin embargo ninguno de estos son óptimos.

El presente proyecto de fin de carrera abordará este problema desde una perspectiva ya conocida: las redes neuronales. Las redes neuronales artificiales son sistemas para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona. Todos los procesos del cuerpo humano se relacionan en alguna u otra forma con la (in)actividad de las neuronas. Las mismas son un componente relativamente simple del ser humano, pero cuando millares de ellas se conectan en forma conjunta se hacen muy poderosas.

En el mundo de las redes neuronales existen diversas arquitecturas. La arquitectura Perceptron, la cual será tema para el trabajo de esta tesis, distribuye las neuronas en tres capas: la capa de entrada, la capa oculta y la capa de salida. Los datos ingresan por medio de la capa de entrada, esta capa no los procesa, solo los transporta a la capa oculta, en esta última los datos son procesados en las diferentes subcapas que la conforman y transportados de una a otra, finalmente los pasan a la capa de salida, aquí son procesados nuevamente, obteniéndose una respuesta. El tipo de red neuronal Perceptron es usada para tareas de clasificación; y un sistema OCR (Optical Character Recognition) cabe muy bien dentro este campo de actividad. El poder de una MLP yace en su habilidad de representar relaciones no lineales.

El fin de este proyecto es arrojar nuevos datos para las estadísticas.

OBJETIVO GENERAL

Implementar un sistema de información para el reconocimiento de caracteres basado en la red neuronal Perceptron.

OBJETIVOS ESPECÍFICOS

1 Implementar el software de extracción de características del archivo de imágenes y normalización del archivo de objetivos.

2 Implementar el software de configuración y entrenamiento de la red neuronal para obtener los pesos y las bías de la red neuronal.

3 Diseñar el proceso de reconocimiento de dígitos dentro de la red neuronal.

4 Implementar el prototipo de sistema de información para el reconocimiento de dígitos.

ALCANCE

Este es un proyecto de investigación aplicada que busca entrenar e implementar un prototipo de reconocimiento de caracteres basado en redes neuronales, el cual se hará cargo de leer un archivo que contiene una secuencia de imágenes de dígitos de 28x28 píxeles en escala de grises.

El prototipo final no realizará la segmentación de caracteres de ningún documento.

El prototipo no tratará con ninguna otra extensión de archivo de entrada que no sea la base de datos de dígitos del MNIST.

Una parte de la fase del entrenamiento será hecha usando una función del programa MatLab.

HISTORIAL DE REVISIONES

Descripción de la revisión	Fecha de la revisión por parte del asesor
Problemática, Marco teórico, Estado del arte	2013/06/09
Problemática	2013/06/13
Capítulo 3,4 y 5	2014/05/09
Capítulo 3,4 y 5	2014/05/12

Tabla de contenido

CAPÍTULO 1

1 Problemática

Hoy en día no cabe duda que los sistemas digitales son de gran ayuda en los diferentes quehaceres del ser humano. Para que un dispositivo digital pueda realizar procesamientos matemáticos, financieros, fonéticos, contables, edición de texto y hasta una sencilla búsqueda de texto a través de software especializado, el archivo digital (en adelante se llamará solo archivo) debe contener una capa de datos, cuyos componentes más básicos deben ser caracteres codificados.

Hasta hace algunas décadas, los libros, boletas, certificados eran expedidos en su mayoría exclusivamente en papel. La invención del scanner trajo la ventaja de la digitalización de los libros, boletas y certificados existentes en ese momento en papel u otro material. Esto permitió su fácil traslado, visualización y almacenamiento. La digitalización de una imagen a través del scanner genera un archivo, cuyo componente básico es el pixel. El pixel indica la manera como se va a pintar un punto determinado de la imagen. Sin embargo; esta forma de representación no es suficiente para que un software especializado de cálculo matemático, financiero, fonético, contable, o de edición de texto pueda procesarlo. Hace falta una capa de datos de caracteres codificados.

El deterioro de material impreso como libros, revistas y periódicos a través del tiempo, debido a elementos externos o inherentes al papel, puede conllevar a la pérdida definitiva de valiosa información. Ante ello, existe la necesidad de conservar en algún otro medio la información. Una manera podría ser traspasar manualmente la información. Sin embargo, en muchas situaciones, este proceso puede ser muy demandante de tiempo y costoso monetariamente. Los dispositivos que capturan imágenes como las cámaras fotográficas y los escáneres pueden hacer que la información sea perdurable en el tiempo. Sin embargo, las imágenes siguen siendo un mapa de bits para un dispositivo digital y un ser humano lector no podría hacer una búsqueda de una frase que necesite encontrar en determinado momento.

Los seres humanos por naturaleza tenemos una necesidad de aprender debida a nuestra innata curiosidad. Los invidentes y los que tienen baja visión en el mundo según [OMS 2011] son alrededor de 285 millones de personas y siempre hubo una necesidad de ellos de acceder al conocimiento que puede brindar un material escrito. La escritura braille es una forma en la que los invidentes pueden leer, al poner sus dedos en la superficies agrietadas de símbolos que vienen a representar letras. Sin embargo, el alto costo que requiere producir estos libros hace que su disponibilidad sea muy escasa. Sin embargo, en la actualidad existe software capaz de leer un archivo que contenga una capa de caracteres codificados e imitar la voz humana.

En la actualidad hay una gran demanda en el mundo de material de literatura. En EEUU el 20% del total de libros vendidos en el 2012 son ebooks [Book Industry Group 2012], es decir, material digital de literatura. Esta cifra se ha elevado exponencialmente en comparación con años anteriores. Por otro lado, muchas veces el número de ejemplares de cada libro es limitado y en algunos momentos, la demanda puede superar a la oferta de ejemplares que posee por ejemplo una biblioteca. Ejemplares digitales que puedan ser

consultados, y sometidos a acciones de búsqueda y acciones avanzadas solo es posible si el ejemplar digital contiene una capa de caracteres codificados.

Finalmente, hoy en día no existe ninguna aplicación OCR basado en redes neuronales capaz de reconocer con total eficacia cualquier carácter. Las estadísticas de cuan buena es una configuración de una determinada red neuronal son muy escasas.

Ante esta problemática, diferentes investigaciones en el campo de la inteligencia artificial comenzaron a realizarse. En la actualidad, hay diversos enfoques con que se intenta transformar una imagen de un caracter en un caracter codificado, sin embargo ninguno de estos son totalmente óptimos. El presente proyecto de fin de carrera abordará el problema desde una perspectiva ya conocida: las redes neuronales, arrojando nuevos datos para las estadísticas.

2 Marco teórico

2.1 Marco Conceptual

2.1.1 Conceptos relacionados al problema

A continuación se explicará los conceptos más relevantes relacionados al problema y que el lector tal vez no pueda conocer.

Deterioro de papel	Los principales motivos para el deterioro del papel son: ❖ Altas temperaturas ❖ Humedad inferior al 40% y superior al 68% ❖ Moho ❖ Contaminantes atmosféricos como el dióxido de azufre, sulfuro de hidrogeno, dióxido de nitrógeno. ❖ La luz ❖ Insectos y roedores ❖ Componentes ácidos que contiene el papel, que son usados en el proceso de manufactura. Debido al último punto, el papel se deteriora inevitablemente debido a su composición auto degradante. La mayoría de papel proviene de la madera. Según el nivel de acides del papel de madera hay 2 tipos: papel ácido y papel alcalino (o no ácido). La diferencia en los procesos de manufactura de los papeles ácidos y alcalinos reside en los tipos de rellenos y agentes de encolado interno usados (los rellenos y los encolantes son ingredientes críticos en cualquier fórmula para producir papel). Un papel ácido tiene un pH menor a 5, mientras que un papel alcalino tiene un pH superior a 7.5.

	Un papel alcalino, o libre de ácido, durará aproximadamente 400 años en un ambiente saludable, mientras que un papel ácido durará 100 años en un ambiente saludable. Entre los años 1800 y 1900, todo el papel que procedió de la madera, fue papel ácido. Este problema de autodegradación no ocurre con el papel hecho de algodón o lino.
Página de códigos de caracteres	Es una lista en donde cada componente es un par: carácter y su representación numérica. Ejemplos: USA ASCII, ANSI, Unicode.
Pixel	Es la unidad más pequeña que compone una imagen digital. Cada pixel se codifica con una cantidad de bits determinada, definido al momento de la creación de la imagen digital. La longitud de bits en el pixel determinara la profundidad del color en una imagen, y en algunos modelos incluye el nivel de transparencia del pixel. Entre los modelos más usados para la presentación de imágenes se encuentra: RGB, CMYK y LAB.
RGB	Red Green Blue. Es un modelo de color conformado por 3 bytes, los cuales representan la intensidad de los 3 colores de luz primarios: el rojo, el verde y el azul. El primer, segundo y tercer byte indican la intensidad del color del rojo, verde y azul respectivamente.

2.1.2 Conceptos relacionados a la propuesta de solución

2.1.2.1 OCR (Optical Character Recognition)

Reconocimiento óptico de caracteres. Es una herramienta informática que identifica caracteres de un determinado alfabeto a partir de una imagen para luego almacenarlo como un documento de texto [Richard G. Casey, Erci Lecolinet 1996].

El proceso de OCR se define en el siguiente diagrama:

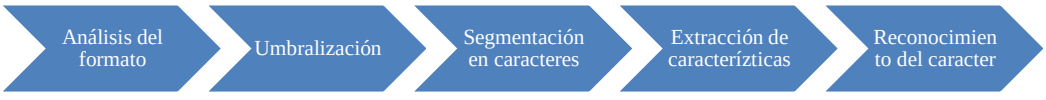


Fig. 2.1: Proceso OCR

- El primer paso consiste en distinguir los títulos, párrafos, columnas, tablas y gráficos e identificar la ubicación de ellos en el documento para efectuar una mejor segmentación

[Richard G. Casey, Erci Lecolinet 1996; José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa (apud Yvette Fiorella Guerrero Calle)].

- El segundo paso consiste en determinar si un pixel es parte de un trazo de carácter o no. Debido a que muchas veces el documento original se encuentra manchado o contiene fondos, separar el fondo de los caracteres facilitará los posteriores pasos del proceso [Richard G. Casey, Erci Lecolinet 1996; José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa (apud Yvette Fiorella Guerrero Calle)].

- El tercer paso consiste en identificar los límites y aislar los caracteres presentes en el documento en segmentos. Los algoritmos que realizan esta función se dividen en 4 grupos: por disección, por segmentación basada en el reconocimiento, segmentación híbrida y las estrategias integrales. También, puede incluir una etapa extra: la reasignación del tamaño del carácter para que haya un estándar que permite hacer más fácil la realización del siguiente paso [Richard G. Casey, Erci Lecolinet 1996; José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa (apud Yvette Fiorella Guerrero Calle)].

- El cuarto paso consiste en extraer características del carácter como por ejemplo el ancho, el alto y la orientación (qué tanto hacia arriba o hacia abajo, hacia la derecha o hacia la izquierda se encuentra con respecto de ciertas líneas de referencia); todos estos medidos en pixeles [Richard G. Casey, Erci Lecolinet 1996; José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa (apud Yvette Fiorella Guerrero Calle)].

- Y el último paso, consiste en encontrar la mejor coincidencia entre el carácter incógnita y un símbolo de un determinado alfabeto [Richard G. Casey, Erci Lecolinet 1996; José R. Hilera González, Juan P. Romero Villaverde, José A. Gutiérrez de Mesa (apud Yvette Fiorella Guerrero Calle)]. Para este fin, existen diversos algoritmos, entre los más destacados: la red neuronal, lógica difusa, las cadenas ocultas de Markov, el voto mayoritario, sistemas basados en microprocesadores, transformada de Fourier, transformada de wavelet, método estadístico de Bayes, árboles de decisión y el genético.[Manuel Alejandro Mongue Osorio, 2008]

2.1.2.2 Redes neuronales

"Una red neuronal es un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona. Todos los procesos del cuerpo humano se relacionan en alguna u otra forma con la (in)actividad de estas neuronas. Las mismas son un componente relativamente simple del ser humano, pero cuando millares de ellas se conectan en forma conjunta se hacen muy poderosas. "[Marco Antonio Valencia Reyes, Cornelio Yañes Márquez, Luis Pastor Sánchez Fernández 2006].

La sinapsis (Fig. 2.2) es una unión intercelular especializada entre neuronas o entre una neurona y una célula efectora (casi siempre glandular o muscular). El proceso de comunicación entre células es la siguiente: la célula emisora segrega compuestos químicos llamados neurotransmisores, los cuales son captados por la sinapsis de la célula receptora. La célula receptora es excitada o inhibida en cada una de sus múltiples entradas y cuando alcanza un cierto umbral, la célula receptora se dispara o activa, generando una corriente eléctrica que es enviada a través de su axón, el cual segrega, otra vez neurotransmisores, los cuales serán captados por otra célula vecina.

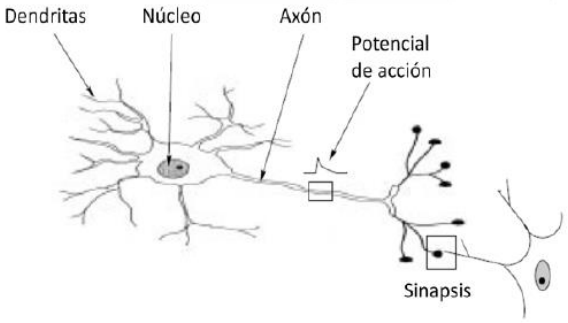


Fig. 2.2: Una neurona natural

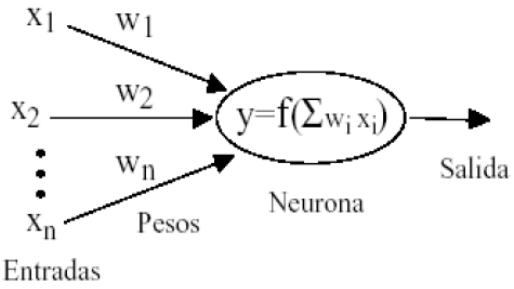


Fig. 2.3: Una neurona artificial

Las dendritas, sinapsis y el axón de la neurona biológica (Fig. 2.3) vienen a ser las entradas, los pesos y la salida, respectivamente, de la neurona artificial.

El poder de la red neuronal radica en la interacción entre las neuronas y en sus diversas formas de interconexión.

La forma y el conocimiento en cómo la red neuronal resuelve su tarea se encuentra almacenada en la forma de su arquitectura, los pesos y los umbrales de cada neurona, a los que a partir de ahora se llamarán los parámetros de la red neuronal.

2.1.2.3 Arquitectura de redes neuronales artificiales

La siguiente lista muestra las diversas arquitecturas de redes neuronales existentes:

a. Adaline
b. Adaptive resonance theory
c. BAM
d. Boltzmann machine
e. CPN
f. Elman SRN
g. Perceptron
h. Hopfield
i. Jordan SRN
j. NEAT
k. Radial basis function
l. Recurrent SOM
m. Kohonen/SOM
n. NeuroFuzzy perceptron
o. Hebbian network

2.1.2.4 Técnicas de entrenamiento

La lista siguiente muestra las diferentes técnicas de entrenamiento de redes neuronales existentes:

a. Annealing
b. Auto backpropagation
c. Backpropagation
d. Binary delta rule
e. Resilient propagation
f. Hebbian learning
g. Scaled conjugate gradient
h. Manhattan update
i. Instar/Outstar
j. Kohonen
k. Levenberg Marquardt (LMA)
l. Genetic
m. Instar
n. Outstar
o. ADALINE

2.1.2.5 Arquitectura Perceptron

Las neuronas en una arquitectura Perceptron pueden estar situadas en una de estas 3 capas:

i. Capa de Entrada: Es la que recibe estímulos externos del aparato sensorial (células glandulares o musculares).
ii. Capa oculta: Puede ser 1 o varias. Es la que no tiene relación con lo externo.
iii. Capa de salida: Es la que da respuesta al sistema externo.

Según Fig. 2.4, los datos ingresan por medio de la capa de entrada, esta no los procesa, solo los pasa a la capa oculta, esta los procesa y el resultado lo pasa a la siguiente capa oculta, esta lo procesa y el resultado lo pasa a la capa de salida, esta lo procesa y da una respuesta.

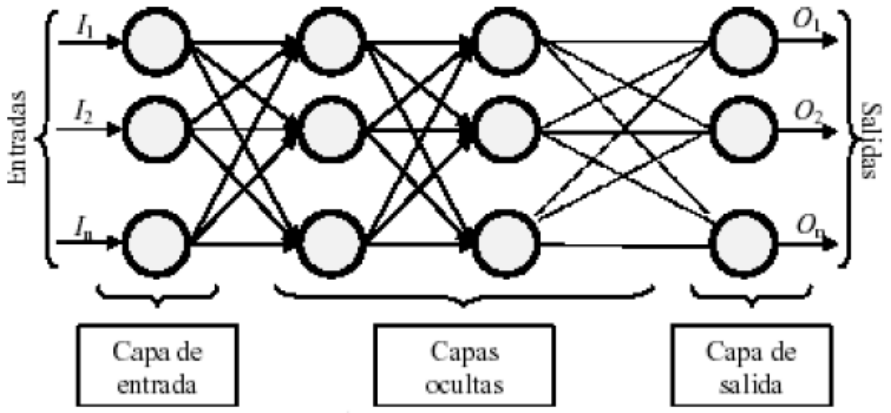


Fig. 2.4: Arquitectura Multilayer Perceptron(MLP).

El tipo de red neuronal Perceptron es usada para tareas de clasificación; y un sistema OCR cabe muy bien dentro este campo de actividad. “El poder de una MLP yace en su habilidad de representar relaciones no lineales” [Anna Hart 1992].

2.1.2.6 Entrenamiento de redes neuronales

El proceso de aprendizaje requiere de data de prueba (una muestra) que represente la población de los datos de entrada. De ahí, de que sea necesario de que la data de prueba sea de calidad (diversa), dado que influencia la manera en que la red neuronal aprende. Luego, de que haya terminado la etapa de aprendizaje, la red neuronal entrara a un proceso de prueba, en donde la red neuronal recibirá data que nunca antes ha visto, para verificar según un grado mínimo de eficacia, si la red es confiable o no en sus respuestas. De esto se desprende que la data de prueba no sea usada en la data de aprendizaje.

Dada un conjunto de datos de entrenamiento y una determinada arquitectura de red, puede dar como resultado diversos conjuntos de parámetros diferentes en cada entrenamiento, que puedan modelar la data de entrenamiento. Algunos consejos que se da en [Anna Hart 1992] es que antes que la data de entrada sea procesada por la red, sea limpiada de cualquier ruido, porque este podría degradar la calidad de la respuesta. Otro consejo que da es escoger buenos parámetros de inicio para que el resultado del aprendizaje sea una red neuronal más eficaz.

2.1.2.7 La técnica de entrenamiento Backpropagation

En las palabras de [Marco Antonio Valencia Reyes, Cornelio Yañes Márquez, Luis Pastor Sánchez Fernández 2006] el método de aprendizaje Backpropagation funciona de la siguiente manera:

“Primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada.

La importancia de la red backpropagation consiste en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones de entrada y sus salidas correspondientes. Es importante la capacidad de generalización, facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas

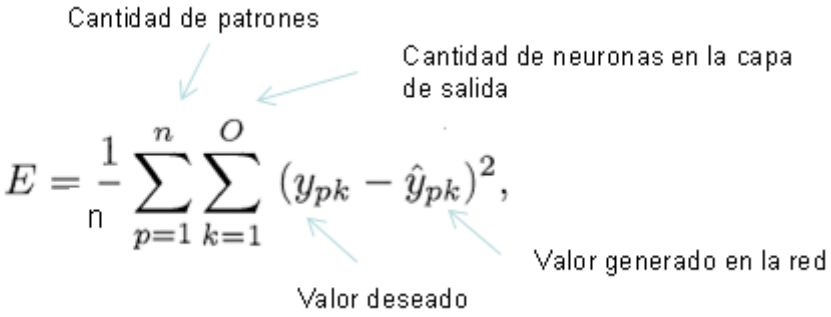
En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás (feedback) ni laterales entre las neuronas de la misma capa.

La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose éstos valores con la salida esperada para así obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primera capa. De esta manera se han modificado los pesos de las conexiones de la red. La técnica Backpropagation requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal".

La técnica de aprendizaje Backpropagation es como sigue:

Vamos a asumir que la red neuronal es de 2 capas: 1 capa oculta y 1 capa de salida.

La función de error es:



Lo que se quiere es acercar lo más próximo a 0 la función de error E.

Se inicializan los pesos de cada neurona por algún método aleatorio

mientras (1)

Calcular E;

si (E < errorMinimo) entonces

romper;

fin si

actualizar pesos:

Se actualizan pesos de la capa de salida: $w_{kj}^{t+1} = w_{kj}^{t} - \eta * \frac{dE}{dw_{kj}}$, w es el peso, k es la posición de neurona de la capa de salida, j es la posición de neurona de la capa oculta y η es la tasa de aprendizaje;

Se actualizan los pesos de la capa oculta: $w_{ji}^{t+1} = w_{ji}^{t} - \eta * \frac{dE}{dw_{ji}}$, w es el peso, j es la posición de neurona de la capa oculta, j es la posición de una entrada y η es la tasa de aprendizaje;

fin mientras

2.2 Marco regulatorio / legal

2.2.1 Legislación sobre propiedad intelectual

La legislación sobre propiedad intelectual es una legislación nacional. Los libros producidos legalmente en formatos alternativos en un país no pueden compartirse en otro país. Modificar las restricciones sobre los derechos de autor que impiden compartir obras en formatos alternativos más allá de las fronteras de cada país mejoraría notablemente el acceso a los libros de las personas con discapacidad para leer texto impreso.

3 Estado del arte

3.1. Productos no comerciales (de investigación) para resolver el problema

3.1.1 *OCR para DNI de Colombia*

La necesidad de controlar y registrar el acceso de personas a una local a través del DNI (los campos nombre, apellido y número), se implementó un OCR [C. Isaza, J. Vargas, C. Gaviria, L. Hernández], cuyo proceso se explica a continuación:

Se extrae la imagen a través de un scanner.

Se hizo un análisis previo que concluyó que los pixeles RGB que contaban con bastante intensidad de color rojo y verde eran caracteres y que el fondo eran pixeles con bastante intensidad en azul. Así, se convierte lo pixeles que tienen un umbral mínimo definido de rojo y verde a 1 (color negro), y los pixeles con bastante intensidad de azul a 0 (color blanco).

Luego se comienza a buscar con una plantilla que contiene la palabra “REPUBLICA”, la palabra “REPUBPLICA”, con el fin de que luego de encontrar esta sección, se pueda saber en qué zonas se podría encontrar los campos Nombre, Apellido y Número.

Debido a que algunos pixeles de los caracteres coincidían, en su forma original, con el fondo, quedaron algunas porosidades luego de realizar el paso de umbralización. Entonces, lo que se hizo fue convertir un pixel de blanco al negro siempre y cuando sus pixeles superior, inferior y laterales sean negros; y por otro lado, se convierte un pixel de negro a blanco, si existen 3 o más pixeles superior, inferior o laterales de color blanco (Fig. 3.1).

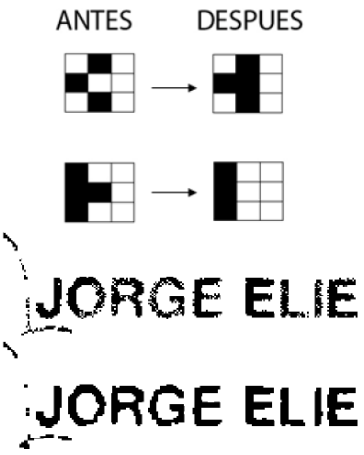


Fig. 3.1

Como ya se tiene las zonas tentativas donde se encuentran Nombre, Apellido Y Número, dentro de estas zonas se hace una proyección horizontal para encontrar exactamente en qué filas comienza y termina un renglón, y luego se realiza una proyección vertical para saber en qué columnas comienza y termina un carácter (Fig. 3.2)

Fig. 3.2: Arriba, proyección horizontal y abajo, proyección vertical [C. Isaza, J. Vargas, C. Gaviria, L. Hernández].

En (Fig. 3.2), según lo descrito, la letra EZ formaría un carácter. Esto se solucionó segmentando por la mitad los caracteres que tengan un ancho mayor a 40 pixeles.

El, último paso es el reconocimiento del carácter. Para esto se usó la librería OpenCV, la cual contiene 27 plantillas definidas que corresponden a cada letra del alfabeto español. Para cada carácter, la plantilla que arroje el mayor valor de correlación, define la letra que se tiene en la imagen.

Se hizo una prueba a 34 cédulas escaneadas a una resolución de 300 (PPP) pixeles por pulgadas con una profundidad de 24 bits por pixel. El método tomo 0.4 segundos en promedio por DNI para identificar el nombre, apellido y número. La eficacia de reconocimiento fue de 92.6%.

3.1.2 Diseño de una arquitectura para una red neuronal artificial perceptron multicapa sobre un FPGA aplicada al reconocimiento de dígitos

El trabajo hecho en[Manuel Alejandro Mongue Osorio, 2008] consistió en lo siguiente:

El primer paso realizado es el de encontrar la mejor arquitectura perceptron, quiere decir definir el número de capas, la cantidad de neuronas por capa y los pesos de cada neurona. Para esto hace uso de 2 herramientas: la base de datos de entrenamiento del MNIST y el Neural Networks Toolbox. La primera es una base de datos que contiene imágenes de caracteres de 28x28 píxeles en escala de grises. La segunda es una herramienta de la aplicación de software del MatLab para el entrenamiento de redes neuronales. Entonces, se umbraliza los píxeles de todas las imágenes de la base de datos del MNIST a un 1 binario o un 0 binario, dependiendo si el pixel alcanza o no un umbral especificado. Por lo tanto, cada imagen se reduce de 28 x 28 píxeles a 28 x 28 bits. Esta umbralización se realiza a través de un script en el MatLab.

La cantidad de neuronas en la capa de salida es 10, porque 10 son la cantidad de dígitos que hay [0-9], esto quiere decir, que cada neurona representa un dígito y se espera que por cada entrada de la base de datos del MNIST que procese la red neuronal recibida, solo se active una neurona en la capa de salida. Luego, se comienza a construir varias arquitecturas en el Neural Networks Toolbox. Lo que se busca es minimizar el peso de cada neurona, minimizar la cantidad de capas ocultas, minimizar la cantidad de neuronas de entrada y

minimizar el error cuadrático medio (mse). Por ejemplo, una arquitectura definida y que luego sería la elegida es la arquitectura MLP 49-25-10 (Fig. 3.3), red neuronal número 26 (Fig. 3.4), lo que quiere decir: 49 neuronas en la capa de entrada, 25 neuronas en la capa oculta y 10 neuronas en la capa de salida. Cada neurona de entrada, para esta arquitectura de red neuronal, recibió la suma binaria de un fragmento de 4 x 4 bits de una imagen umbralizada de 28 x 28 bits. Así la cantidad de neuronas de entrada es 49, pues 4 * 4 * 49 = 28 * 28, lo cual se definió antes de correr el Neural Networks Toolbox.

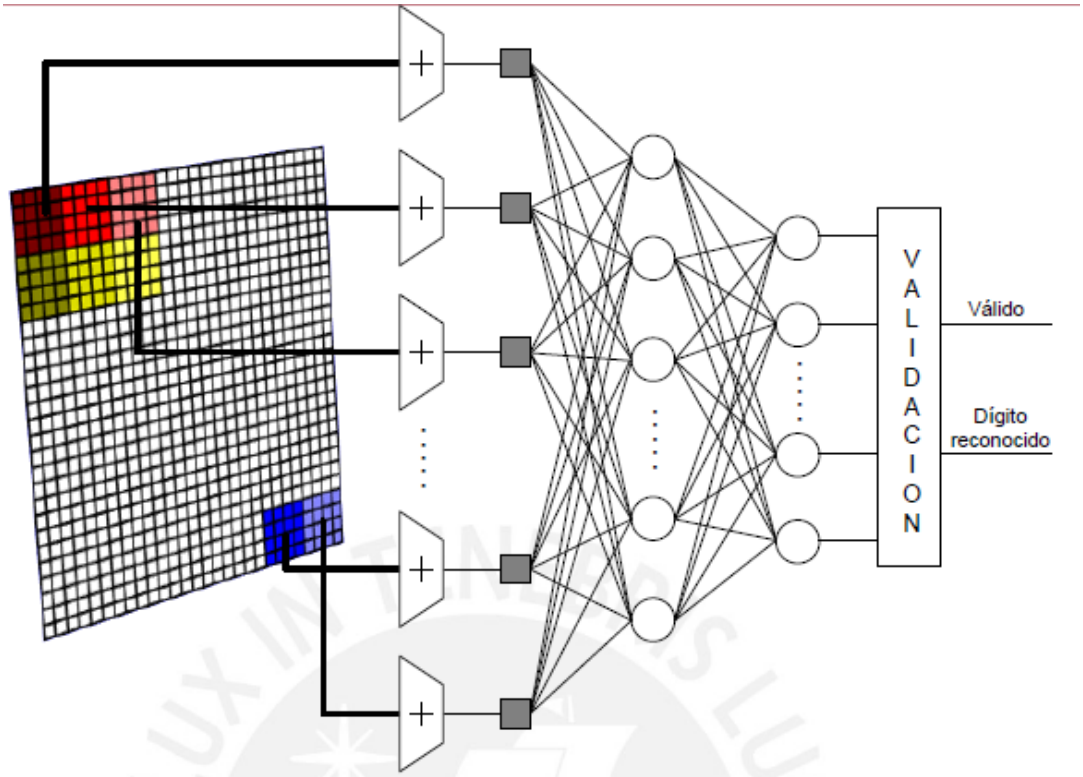


Fig. 3.3: Diagrama de la arquitectura 26[Manuel Alejandro Mongue Osorio, 2008].

Red Neuronal	#bits / entrada	Neuronas en la capa de entrada	Neuronas en la capa oculta	Neuronas en la capa de salida
23	4	196	50	10
24	16	49	15	10
25	16	49	20	10
26	16	49	25	10
27	16	49	30	10
28	16	49	35	10

Fig. 3.4: Atributos de algunas redes neuronales [Manuel Alejandro Mongue Osorio, 2008].

El segundo paso es realizar la prueba para cada arquitectura establecida (Fig. 3.6) con la base de test del MNIST, con la finalidad de verificar la eficiencia de cada arquitectura.

Red Neuronal	Epochs	Performance (mse)	Peso máximo
23	100	0.00615423	877
24	100	0.0167706	753
25	100	0.01682905	747
26	100	0.013892	767
27	100	0.0139489	784
28	100	0.0128555	787

Fig. 3.5. Performance en la fase de entrenamiento de algunas redes neuronales

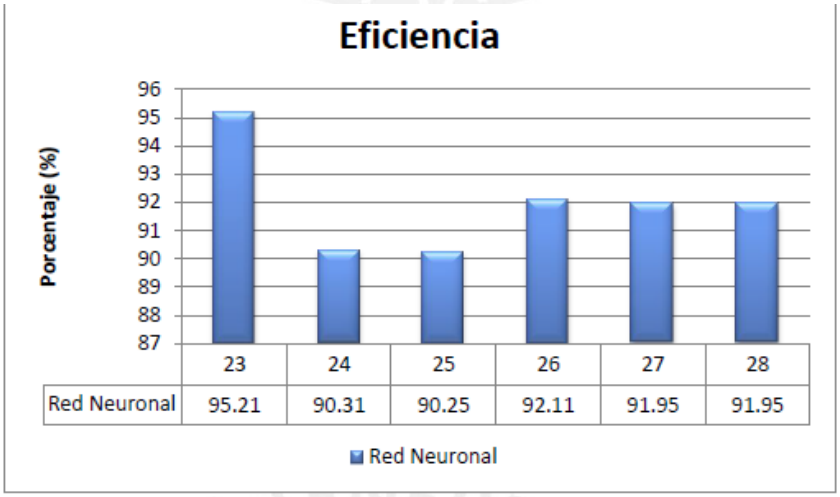


Fig. 3.6. Eficiencia en la fase de testeo de algunas redes neuronales

En este punto, se escoge la red neuronal número 26, MLP 49-25-10, como la más adecuada debido a que contiene una cantidad reducida de neuronas en la capa de entrada, una cantidad reducida de capas ocultas, un valor reducido de peso máximo en comparación a otras arquitecturas y una eficiencia aceptable.

Y como último paso, se implementa la red neuronal en el FPGA (una plataforma electrónica programable de uso genérico). En la implementación se agregó un módulo de validación, que se ejecuta luego de que se ha obtenido el resultado de cada neurona de la capa de salida, que consiste en validar el resultado siempre y cuando solo haya una neurona en la capa de salida con valor cercano a 100 y las demás con valor cercano a 0. En otras palabras, se válida que en la capa de salida solo se active una neurona y las demás se inhiban.

3.2 Productos comerciales para resolver el problema

3.2.1 *Adobe Acrobat XI Pro*

El pdf es un formato de archivo (una estructura definida) para la distribución de documentos electrónicos que funciona de manera independiente al hardware, sistema operativo o de la aplicación de software que fue usada para crearlo. Este formato preserva la fuente, el formato, gráficos y el color del documento. Un documento pdf es capaz de mezclar en un solo documento de manera íntegra diversas fuentes: páginas web, hojas con texto enriquecido, hojas de cálculo, documentos escaneados, fotos y gráficos. También soporta las firmas digitales, encriptación de datos, permisos de seguridad para que el creador del documento restringa ciertos privilegios sobre el documento y accesibilidad al contenido a discapacitados.

Cabe resaltar que cada versión de pdf es compatible hacia atrás a partir de la versión 1.0. Eso quiere decir que una aplicación de software que ha sido diseñada para leer documentos pdf en versión 1.0, podrá abrir y leer un documento pdf en versión 1.7, ignorando aquellas funcionalidades que no soporta la versión 1.0 y que están contempladas en la versión 1.7 [Adobe Systems Incorporated, 2003].

Un archivo de PDF es una colección de objetos. Cada objeto puede hacer referencia a otros objetos.

PDF soporta 8 tipos básicos de objetos:

- Boolean
- Integer
- String
- Name
- Array
- Dictionary
- Stream

- Null

La habilidad de que un pdf pueda representar en pantalla diferentes formas de gráficos subyace en el modelo de imagen de Adobe. Las 3 principales formas de gráfico son:

- Objeto sendero: Consiste de una secuencia de puntos conectados y desconectados, líneas y curvas unidas a través de operadores, cuyo resultado son las formas y sus posiciones. Los principales operadores son la S que pinta el sendero y la F que pinta el interior de un sendero.
- Objeto texto: Consiste en uno o varios glifos que representan caracteres de texto. La manera como el glifo va a ser dibujado está definido en un archivo aparte llamado fuente.
- Objeto imagen: Es un arreglo rectangular de píxeles. Típicamente, usado para representar fotografías.

Adobe Acrobat se especializa en el trabajo con archivos con formato pdf [Adobe Systems Incorporated, 2013].

Entre sus principales funcionalidades se tienen:

La funcionalidad de reconocimiento de texto analiza el mapa de bits de la imagen, conserva la imagen original, y coloca bajo de ella un lienzo invisible de texto con el fin de hacer búsquedas de texto o incluso modificarlo en el mismo Acrobat. Si al momento de generar la capa de texto se presenta una palabra incierta, se genera la palabra más cerca al óptimo y se marca la palabra como sospechosa. Luego, con la herramienta Buscar todos los sospechosos, se podrá corregir la palabra según el usuario vea conveniente.

Por otro lado la opción Guardar como permite exportar una documento pdf a un formato que permite la edición de texto como doc, txt, rtf, xls, xml. En el proceso, Adobe hace uso de su OCR para convertir los objetos de imagen (mapas de bits) y los objetos trayectoria, ambos, en texto, y lo que no pueda convertir a texto lo conserva como trozos de imágenes. Sin embargo, los objetos texto que pueden estar embebidos en el pdf son pasados directamente como texto al documento exportado, sin necesidad de un procesamiento intermedio, pues los objetos texto ya se encuentran como caracteres de texto.

Otra funcionalidad importante son las capas de PDF. Cada capa es un conjunto de datos (imágenes, texto, marcas de agua, vínculos) y al que se le puede dar un tratamiento diferenciado como: determinar su visibilidad o no visibilidad en pantalla; si se va a imprimir o no; y si al momento de exportar a otro formato, la capa se exportará o no se exportará. Se pueden importar capas desde un pdf o un archivo de imagen a un pdf destino.

La funcionalidad de formulario interactivo consiste en insertar objetos interactivos con el usuario como: campos de texto, cuadros de lista, lista desplegable, campo de firma digital, casillas de verificación, botones. Estos al ser seleccionados pueden ejecutar una acción. La data que es ingresada en el formulario puede ser enviada automáticamente a un correo electrónico o a un servidor como SharePoint o Carpeta de red. Los objetos interactivos pueden ejecutar lenguaje JavaScript, el cual puede desencadenar acciones automáticas de

aplicación de formato, cálculo, validación de datos o conexiones a base de datos mediante ODBC.

La herramienta de recorte puede ajustar el área de cuadro de una página que se va a mostrar. La información que esta fuera del margen del recorte no se descarta, solo se oculta, por lo que el recorte no reduce el tamaño del archivo.

La herramienta Leer en voz alta permite leer en voz alta texto en formato Unicode presente en un documento pdf. Esta funcionalidad está orientada a personas con discapacidad como problemas de movilidad, ceguera o visión disminuida.

También, permite insertar objetos multimedia como contenedores FLV, SWF, F4V, mp3 y, por otro lado, también los contenedores MOV, M4V, MP4, 3GP y 3G2 siempre y cuando estén codificados con H.264.

La funcionalidad codificación con contraseña permite generar una llave simétrica (en RC4 o AES) y encriptar el documento pdf para restringir su apertura, la impresión, edición o la copia del contenido. La funcionalidad de agregar ID (firma) permite generar un par de llaves asimétricas, las cuales puede ser usadas para firmar, codificar un documento pdf y para generar un certificado de seguridad.

3.3 Conclusiones sobre el estado del arte

En la actualidad hay una cantidad considerable de aplicaciones de OCR, basados en cualquiera de los algoritmos de clasificación (sección 2.1.2.1) que cualquier persona con un presupuesto de alrededor de los 200$ puede adquirir. Incluso, hay software libre en OCR. Sin embargo, hasta el momento no existe ninguna aplicación OCR completamente efectiva.

Las diversas configuraciones que puede tener una red neuronal MLP para afrontar el reconocimiento de un carácter son muchísimas, por no decir casi infinitas. Este proyecto de fin de carrera generará algunas configuraciones para una red neuronal MLP, analizará sus resultados y serán de utilidad para las estadísticas del estado del arte de OCR basado en redes neuronales, que en la actualidad son muy escasas.

CAPÍTULO 2

1 Objetivo general

Implementar un sistema de información para el reconocimiento de caracteres basado en red neuronal perceptron.

2 Objetivos específicos

Objetivo 1	Implementar el software de extracción de características del archivo de imágenes y normalización del archivo de objetivos
Objetivo 2	Implementar el software de configuración y entrenamiento de la red neuronal para obtener los pesos y las bías de la red neuronal
Objetivo 3	Diseñar el proceso de reconocimiento de dígitos dentro de la red neuronal
Objetivo 4	Implementar el prototipo de sistema de información para el reconocimiento de dígitos

3 Resultados esperados

Del Objetivo 1:

- Software de extracción de características del archivo de imágenes y normalización del archivo de objetivos
- Características extraídas del archivo de imágenes y archivo de objetivos normalizado.

Del Objetivo 2:

- Software de configuración y entrenamiento de la red neuronal para obtener los pesos y las bías de la red neuronal.
- Configuración de la red neuronal.

Del Objetivo 3:

- Documento del proceso de reconocimiento de dígitos de la red neuronal.

Del Objetivo 4:

- Prototipo del sistema de información para el reconocimiento de dígitos.

4 Herramientas, métodos y procedimientos

4.1 Mapeo

Resultados esperado	Herramientas a usarse
Prototipo del sistema de información de preprocesamiento, configuración y entrenamiento de la red neuronal.	- MatLab
Documento del proceso de reconocimiento de caracteres de la red neuronal	- Pseudocódigo
Prototipo del sistema de información para el reconocimiento de caracteres	- Visual Studio - C#

4.2 Visual Studio

- Descripción

Es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Es decir es un espacio de trabajo para desarrollar aplicaciones que funcionan en variados ambientes como de escritorio o Web; siempre y cuando trabaje junto a un sistema operativo Windows y tenga el framework .NET necesario. Visual Studio soporta los siguientes lenguajes de programación:

- C++

- C#

- Basic

- J#

- ASP.NET

4.3 C#

- Descripción

Es un lenguaje de programación orientado a objetos. Su funcionamiento se basa en el framework .NET. Es un lenguaje hibrido, ya que al momento de compilar el programa, el compilador genera un archivo en lenguaje Common Intermediate Language (CIL), el cual es un lenguaje legible por el ser humano pero de más bajo nivel. Luego, en la etapa de ejecución, la máquina virtual convertirá el archivo CIL en código binario. Su sintaxis está basada en el lenguaje C++.

4.4 Pseudocódigo

- Descripción

Es una descripción informal de un algoritmo informático, independiente de cualquier lenguaje de programación, y orientado para la lectura humana. Este también puede tener lenguaje natural y notaciones matemáticas compactas. Sin embargo, el pseudocódigo omite detalles que no son necesarios para la compresión humana del algoritmo como la declaración de variables y llamadas al sistema operativo. Se tomará en cuenta las convenciones de pseudocódigo usadas en [Thomas H. Cormen 2009].

4.5 MatLab

- Descripción

Es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación.

5 Limitación

5.1 Alcance

Este es un proyecto de investigación aplicada que busca entrenar e implementar un prototipo de reconocimiento de caracteres basado en redes neuronales, el cual se hará cargo de leer un archivo que contiene una secuencia de imágenes de dígitos de 28x28 píxeles en escala de grises.

El prototipo final no realizará la segmentación de caracteres de ningún documento.

El prototipo no tratará con ninguna otra extensión de archivo de entrada que no sea la base de datos de dígitos del MNIST.

Una parte de la fase del entrenamiento será hecha usando una función del programa MatLab.

5.2 Riesgos

Los riesgos que se han considerado para el proyecto son:

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Pérdida de la información del proyecto	Duración	Uso de un repositorio de datos en internet.
No encontrar una taza de aprendizaje adecuado para el algoritmo de redes neuronales.	Poca efectividad de reconocimiento de caracteres entregado por las redes neuronales.	Instanciar diversas redes neuronales con diferentes tazas de aprendizaje.

6 Justificación y viabilidad

6.1 Justificativa del proyecto de tesis

Uno de los beneficios que pretende obtener este proyecto es el servir de referencia para la conservación del material literario existente en papel. La gran parte de papel que ha sido producido hasta ahora, dependiendo de su proceso de fabricación, en mayor o menor medida, innatamente auto degradable. Además de esto, factores externos (sección 2.1.1) también cooperan a la degradación del papel. La digitalización de caracteres de un material literario permitirá que los lectores puedan tener un proceso de aprendizaje, al momento de leer, más llevadero y rápido, ya que podrá realizar acciones de búsqueda. Por otro lado, la digitalización de caracteres permitirá al lector poder editar el texto de una manera más rápida.

Hoy en día se están llevando a cabo investigaciones y ya existe aplicaciones capaces de leer y simular la voz humana con buenos resultados. Según [OMS 2011] son alrededor de 285 millones de personas los invidentes y los que tienen baja visión en el mundo. Sin embargo, el material en fuente alternativas en el mundo es aún muy bajo, y mucho más bajo en países con escasos recursos y desarrollo. La digitalización de los materiales literarios servirá de entrada para las aplicaciones existentes de lectura y simulación de la voz humana y esto a su vez permitirá que los invidentes y las personas que tienen paralizadas ciertas partes de su cuerpo puedan acceder y aprender del conocimiento que brinda un material literario.

En la actualidad, las estadísticas que existen acerca de cuan eficaz es una determinada configuración de red neuronal para el reconocimiento de caracteres es aun escasa. Mucha de esta información se maneja en ambientes aislados y no públicos. Además, no existe una aplicación de OCR, basada en cualquiera de los algoritmos aplicables (sección 2.1.2.1) que tenga una totalidad de efectividad. Este proyecto contribuirá al conocimiento de las estadísticas de la eficacia de las diversas configuraciones de las redes neuronales para el reconocimiento de caracteres.

6.2 Análisis de viabilidad del proyecto de tesis

6.2.1 Viabilidad técnica

Se utilizará la base de datos de entrenamiento de MNIST, de acceso libre y gratuito, para el entrenamiento de las diversas redes neuronales. Asimismo, se utilizará la base de datos de test del MNIST, también de acceso libre y gratuito, para la validación de las diversas redes neuronales generadas luego del entrenamiento.

Tanto en el pseudocódigo empleado en [Thomas H. Cormen 2009], en el lenguaje de programación C# y en el IDE Visual Studio, el alumno encargado del proyecto tiene un nivel suficiente de conocimiento y experiencia.

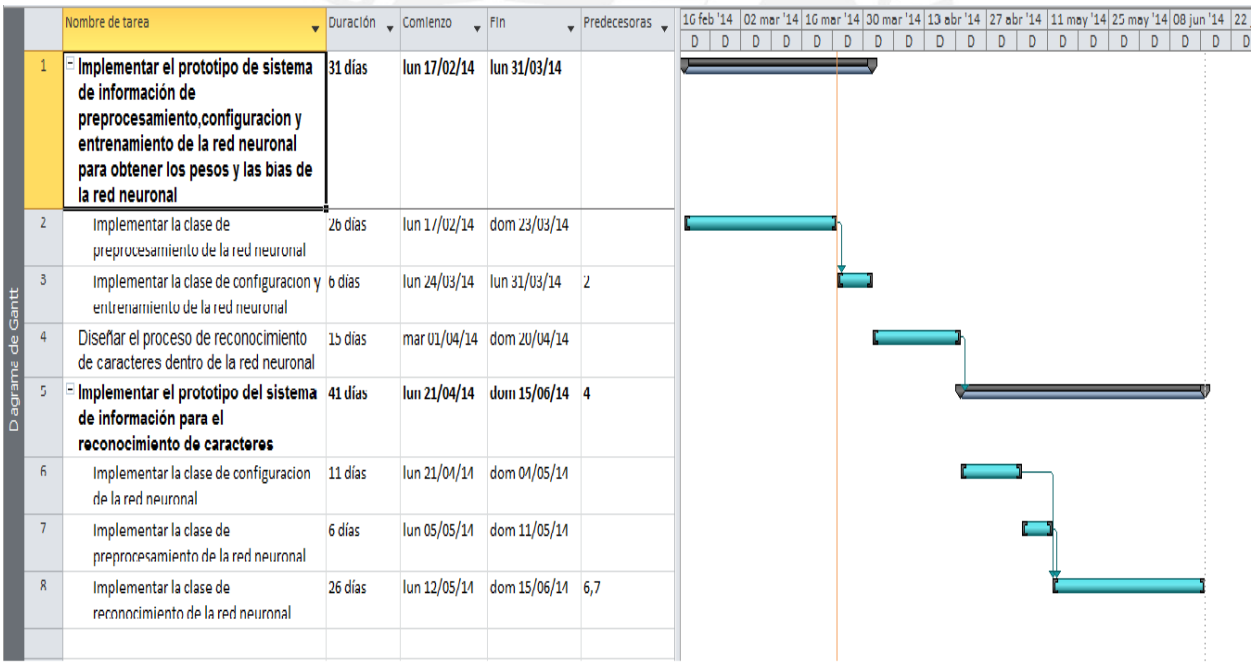
Las redes neuronales artificiales, el método backpropagation, y conceptos de cálculo numérico como gradiente, regla de la cadena, derivadas son de un nivel aún insuficiente por parte del alumno encargado del proyecto. Sin embargo, el alumno del proyecto se encuentra

en proceso de aprendizaje en estos 3 aspectos mencionados y cuenta con acceso a material literario para consultar y cuenta con personas que conocen del tema que están dispuestos a ayudarlo en cierta medida.

6.2.2 Viabilidad temporal

El tiempo que dispone el alumno para terminar con el proyecto es limitado y corto. El mes de Julio, es el tiempo en que el alumno tendrá más tiempo disponible para avanzar con el proyecto. Desde el mes de agosto hasta el mes de diciembre, el alumno dispondrá de poco tiempo para avanzar con el proyecto, ya que necesitará tiempo para trabajar en los aproximadamente 4 cursos más en los que estará matriculado en la universidad. Así que el proyecto, tendrá que ser avanzado en más de un 60% en el mes de julio, para minimizar el riesgo de que no se pueda culminar con el proyecto hasta el mes de diciembre del presente año.

7 Plan de actividades



CAPÍTULO 3

1 Implementación del software de extracción de características del archivo de imágenes y normalización del archivo de objetivos

Antes de explicar esta etapa, se explicará algunos conceptos:

- Data de entrenamiento: Es la data que se utilizará para la fase de entrenamiento de la red neuronal. Está compuesto por un archivo de imágenes (train-images.idx3-ubyte) y por un archivo de objetivos (train-labels.idx1-ubyte). La cantidad de pares: imágenes y objetivos son 60000.
- Data de testeo: Es la data que se utilizará para la fase de testeo de la red neuronal. Está compuesto por un archivo de imágenes (t10k-images.idx3-ubyte) y por un archivo de objetivos (t10k-labels.idx1-ubyte). La cantidad de pares: imágenes y objetivos son 10000.
- Archivo de imágenes: Contiene una secuencia de imágenes de dígitos. Cada píxel del dígito se encuentra en escala de grises y cada pixel ocupa 1 byte. Cada dígito en el archivo de imágenes ocupa 28 x 28 bytes, es decir un total de 784 bytes (pixeles).
- Archivo de objetivos: Contiene una secuencia de números entre [0,9]. Cada número ocupa 1 byte. Hay un total de 10000 números entre [0,9].

En primer lugar se busca extraer las características de la data de entrenamiento para que se conviertan en la entrada de la red neuronal. En otras palabras, hay que normalizar la data de entrenamiento para que la red neuronal lo entienda en la fase de entrenamiento.

Hay que normalizar (o extraer las características o preprocesar) 2 archivos: el archivo de imágenes y el archivo de objetivos.

Cada imagen dentro del archivo de imágenes, según su orden en el archivo, está asociado a un número dentro del archivo de objetivos, según su orden en el archivo.

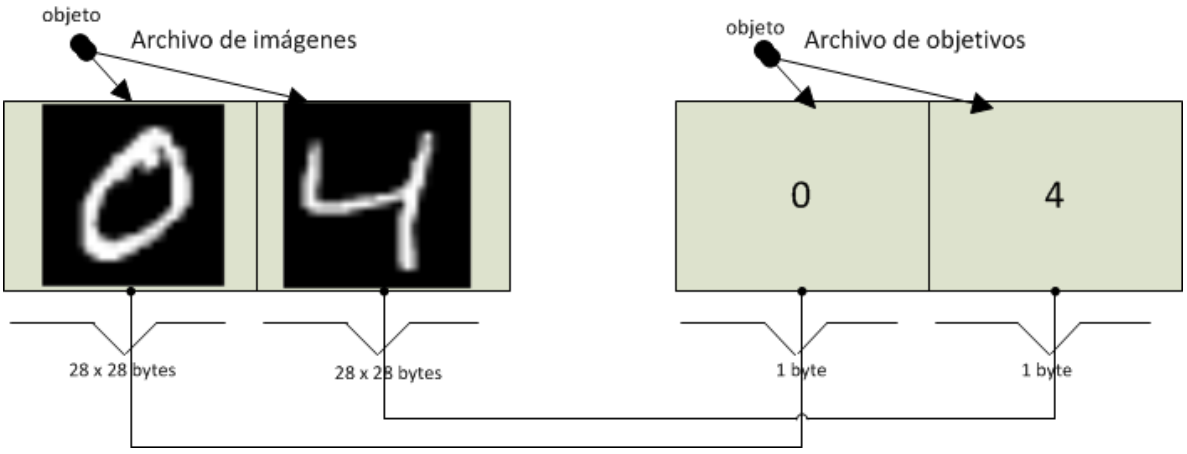


Fig. 11.1: Relación entre una imagen dentro del archivo de imágenes y un número dentro del archivo de objetivos.

Se busca reducir la data de entrada a través de la normalización de la data de entrada. Esto es debido a que mientras más grande sea la data de entrada, más grande la red neuronal

(más neuronas en la capa de entrada y en las capas ocultas), y a más neuronas por capa, más data de entrada se necesita para entrenarla [Jeff Heaton 2008].

El proceso de normalización va como sigue:

- Primero, se convierte la imagen a una imagen en blanco y negro, y a la vez se invierte los colores.
- Segundo, se elimina las líneas verticales y horizontales exteriores a la imagen del dígito, ya que estas líneas no contienen información.
- Tercero, se actualiza a una resolución de 21x21 y se bipolariza cada pixel: el 1 (pixel blanco) se deja como 1, y el 0 (pixel negro) se pasa a -1. Esto es debido a 2 razones: La red neuronal con data de entrada 0 no aprende, y segundo que es recomendable que la data este dentro del rango entre [–1,1] [Howard Demuth 2009].

Fig. 11.2: Como queda la imagen luego del tercer paso.

- Cuarto, se sectoriza la nueva imagen a un sector de 3 x 3 y para cada sector se suma sus pixeles y se divide entre 9. Este resultado se convertirá en una de las entradas de la imagen. Así en total, cada imagen dentro del archivo de imágenes tendrá un total de 49 entradas: (21/3) x (21/3) = 49; y cada entrada va a variar entre el rango [-1,1], lo cual es recomendable.

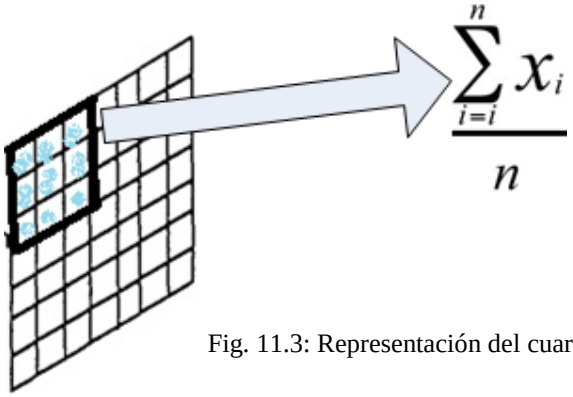


Fig. 11.3: Representación del cuarto paso

❖ Y quinto, por cada objeto del archivo objetivo, se crea un vector de 10 celdas. La celda del vector cuya posición sea igual al objeto del archivo objetivo correspondiente contendrá el valor de 1, el resto de las celdas contendrá el valor de cero.

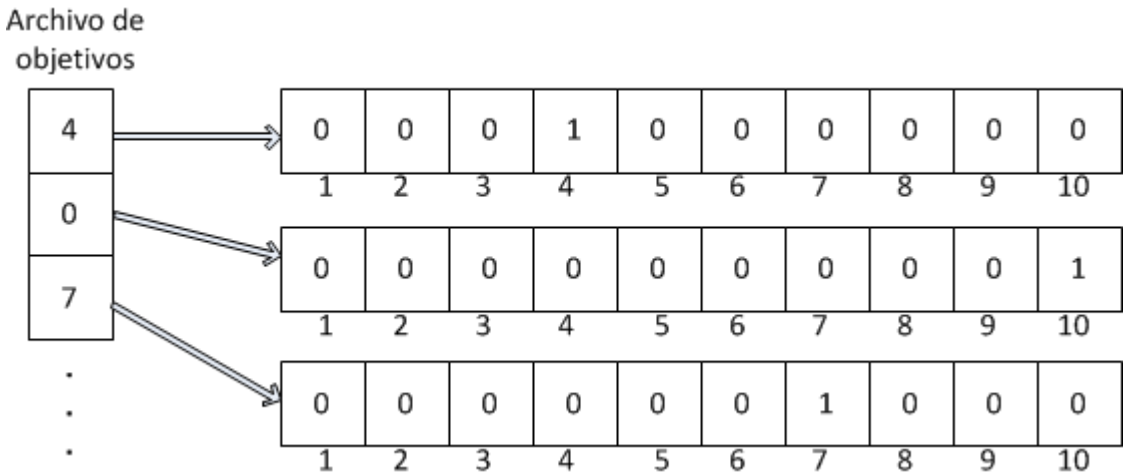


Fig. 11.4: Representación del quinto paso

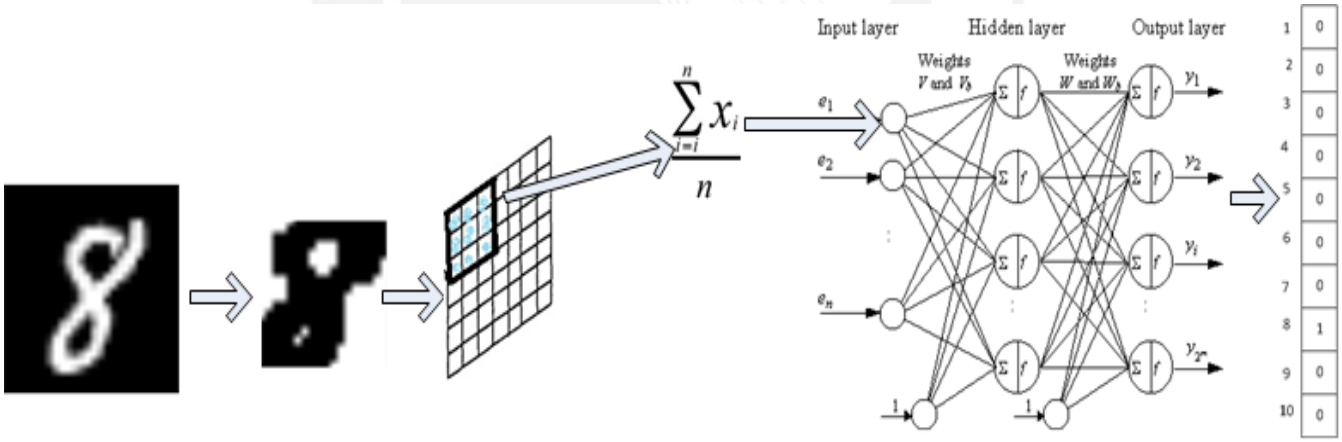


Fig. 11.5: Representación de todos los paso de la extracción de características.

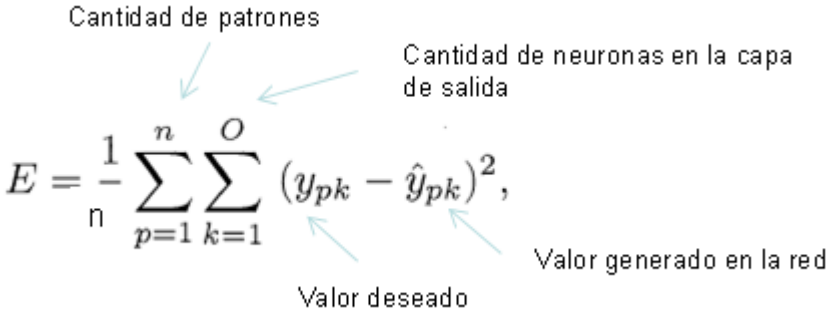
Cada imagen normalizada del archivo de imágenes se va guardando en una matriz en memoria principal. Cada número normalizado dentro del archivo de objetivos se va guardando en otra matriz dentro de memoria principal.

CAPÍTULO 4

1 Implementación del software de configuración y entrenamiento de la red neuronal para obtener los pesos y las bías de la red neuronal

1.1 Función de error

La idea principal del entrenamiento es ir reduciendo iterativamente el valor de una determinada función de error E. Para este proyecto, se escogió la función de diferencia cuadrática.



Mientras la función de error sea más cercana a 0, quiere decir que la arquitectura está generando menos error.

1.2 Pseudocódigo

La cantidad de neuronas en la capa de salida es 10. Cada una de estas neuronas representa un digito entre [0,9]. En cada iteración del entrenamiento, se va ingresando a la red neuronal una por una la totalidad de las imágenes normalizadas del archivo de entrenamiento. Cada imagen normalizada que es procesada por la red neuronal va a botar 1 valor por cada neurona en la capa de salida, es decir, en total, botará 10 valores. Lo que se espera es que la neurona de salida que representa el valor de la imagen normalizada se active, en otras palabras, que tenga el valor de 1, y las demás que tengan el valor de 0. Sin embargo, al inicio del entrenamiento no ocurrirá eso. La idea es que a través de las iteraciones del entrenamiento, las neuronas de la capa de salida se vayan acercando al valor deseado para cada imagen normalizada, y para lograr aquello, lo que se hace es ir modificando iterativamente los pesos (w) de la red neuronal.

El algoritmo en general es como sigue:

1. Inicializar por algún método los pesos y los bías;
2. cantVueltas = 0;
3. mientras (cantVueltas < CantMaxVueltas y error > ErrorMin)
    4. para cada imagen normalizada del archivo de entrenamiento
        5. procesar la imagen a través de la red neuronal y obtener las salidas de cada
        6. capa;
        7. para cada capa de la red neuronal, comenzado desde la última capa hacia
        8. abajo;
            9. para cada neurona de la capa
                10. para cada peso(W) de la neurona
                    11. ir calculando parte de la $\frac{dE}{dW}$;
                12. fin para
            13. fin para
        14. fin para
    15. fin para
    16. si (error <= ErrorMin) entonces
        17. romper;
    18. sino
        19. actualizar los pesos de la red neuronal
        20. ++ cantVueltas;
    21. fin si
22. fin mientras

1.3 Método Nguyen – Widrow para inicialización de pesos y bías

[MathWorks, 2014]

Este algoritmo busca distribuir la región activa del dominio de la función de activación entre los pesos y las bías de cada neurona.

Lo que se busca es evitar que los valores iniciales de cada peso y de cada bías tengan valores que hagan que la función de activación llegue a su límite horizontal inferior o a su límite horizontal superior. En el caso de este proyecto, la función de activación es la sigmoidal. Vemos según fig. 11.6, que la función sigmoidal se satura cuando el x es menor a -4 o mayor a 4. Y es esto, lo que se quiere evitar.

x	f(x)= $\frac{1}{1+e^{-x}}$
-5	0.00669285
-4	0.01798621
-3	0.04742587
-2	0.11920292
-1	0.26894142
0	0.5
1	0.73105858
2	0.88079708
3	0.95257413
4	0.98201379
5	0.99330715

Fig. 11.6: Comportamiento de la función sigmoidal

El pseudocódigo del método Nguyen – Widrow es como sigue:

para cada capa oculta y de salida

$$b = 0.7 * \frac{cantNeurCapa\hat{}(1/cantNeurCapaAnt)}{\lim_{x\to\infty} FuncAct(x) - \lim_{x\to-\infty} FuncAct(x)}$$

para cada neurona de la capa

para cada peso de la neurona, excepto el bías

$$w_{ij} = random(0, b)$$

fin para

bías = random(-b, b)

fin para

fin para

1.4 Función de activación

La red neuronal de este proyecto de tesis usará como función de activación una función sigmoidal para cada neurona: $f(x) = \frac{1}{1+ e^{-x}}$

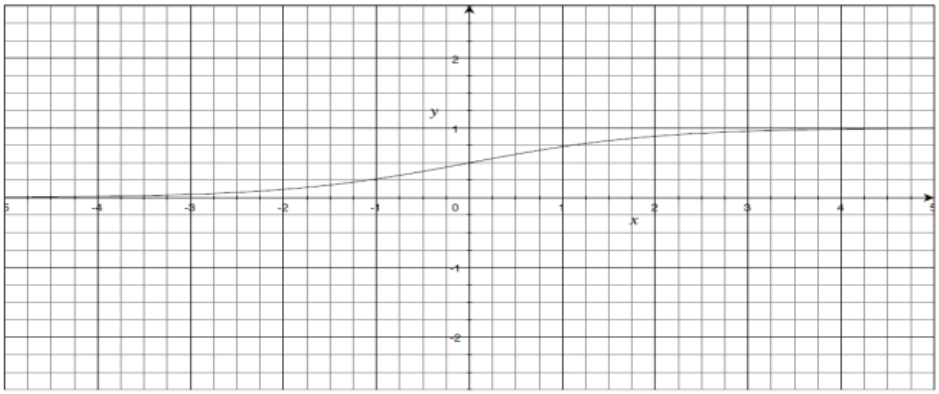


Fig. 11.7: Representación en el plano de la función sigmoidal [Jeff Heaton 2008].

Esta funcion comprime un rango de entrada infinito dentro de un rango de salida finito [-1, 1], y esto es deseable para este proyecto, ya que en la salida de una determinada neurona de salida se desea tener un 1 si la neurona se activa y 0 si la neurona se inhibe.

Otra razón por la cual se aplica una función de transferencia exponencial $f(y) = \frac{1}{1+ e^{-y}}$ a la sumatoria $y = \sum w_i * x_i$ en cada neurona de la red neuronal es para que la red neuronal sea capaz de separar el espacio de entradas en clústeres en forma de curva.

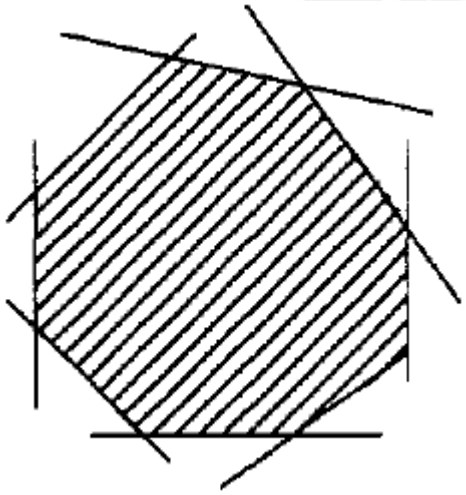


Fig. 11.8: La red neuronal separaría el espacio de entradas en clústeres lineales si solo existiera la sumatoria y no existiera una función de transferencia no lineal [R. Beale 1990].

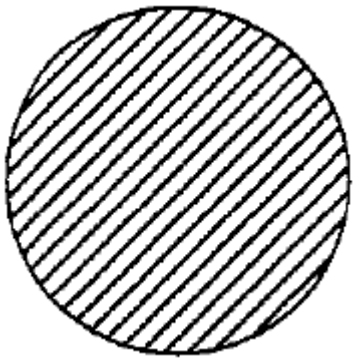


Fig. 11.9: La red neuronal separa el espacio de entradas en clústeres en forma de curva porque existe una función de transferencia no lineal, para el caso de este proyecto, una función exponencial [R. Beale 1990].

1.5 Algoritmo resilient backpropagation

El pseudocódigo de entrenamiento mostrado líneas arriba es la idea general del algoritmo backpropagation. Pero también puede ser la idea general de un algoritmo mejor, el algoritmo resilient backpropagation, la cual se usó para la etapa de entrenamiento de este proyecto.

El problema con el algoritmo backpropagation es que la derivada de la función sigmoidal se acerca a 0 cuando su entrada x es muy negativa o muy positiva. Esto ocasiona que el gradiente tenga una magnitud cercana a 0 y por consiguiente, causa pequeños cambios en los pesos, incluso aunque los pesos estén lejos de su valor óptimo.

En cambio en el algoritmo resilient backpropagation, la derivada solo es útil para determinar el signo de la actualización del peso. La magnitud de la derivada no tiene efecto en la actualización del peso. El tamaño del cambio del peso es determinado por un valor de actualización aparte. El valor de actualización para cada peso es incrementado por un factor delt_inc siempre que la derivada de la función de error con respecto al peso tenga el mismo signo por 2 iteraciones sucesivas. El valor de actualización es decrementado por un factor delt_dec siempre que la derivada con respecto al peso cambia el signo desde una previa iteración. Si la derivada es 0, el valor de actualización se mantiene el mismo. Siempre que el peso esté oscilando, el cambio el peso es reducido. Si el peso continúa cambiando en la misma dirección por varias iteraciones, la magnitud de la actualización del peso se incrementa [Howard Demuth 2009].

1.6 Receta para determinar la cantidad de neuronas por capas

Según [Jeff Heaton 2008], para determinar el número de neuronas en la capa oculta de una red neuronal de 2 capas hay 3 reglas:

1. El número de neuronas ocultas debería estar entre el tamaño de la capa de entrada y el tamaño de la capa de salida.
2. El número de neuronas ocultas debería ser 2/3 el tamaño de la capa de entrada más el tamaño de la capa de salida.
3. El número de neuronas ocultas debería ser menor que el doble del tamaño de la capa de entrada.

1.7 Data de entrenamiento

Y por último, es importante que la data de entrenamiento sea un subconjunto representativo de la data universal.

Típicamente, una nueva entrada dirige a una salida similar a la correcta salida para una entrada usada en el entrenamiento, la cual es similar a la nueva entrada. Esta propiedad de generalización hace posible entrenar a la red en un subconjunto representativo de pares

entradas/objetivos y conseguir buenos resultados, sin entrenar a la red en todo el conjunto universal de pares entrada/objetivo.

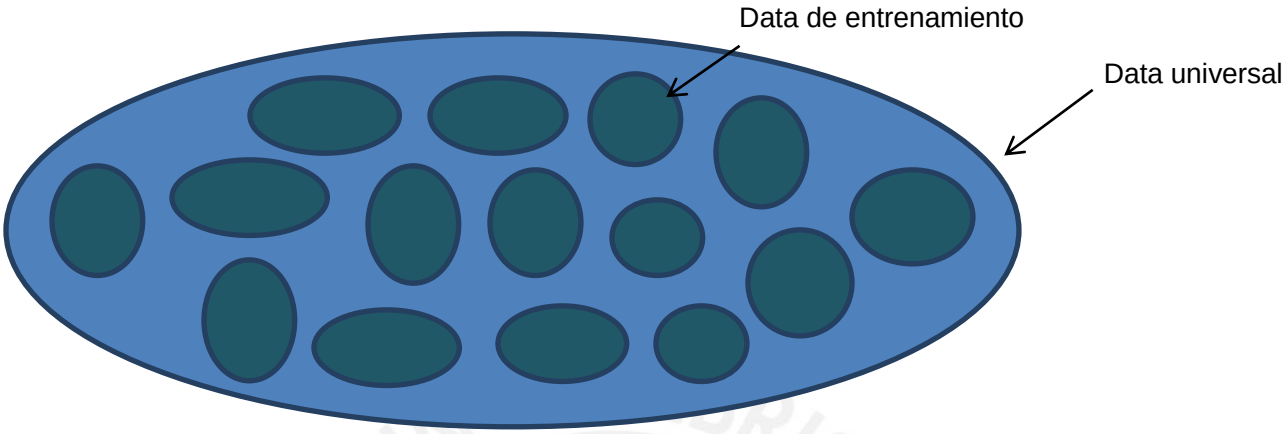


Fig. 11.10: Subconjunto representativo de la data de entrenamiento con respecto a la data universal.

CAPÍTULO 5

1 Diseño del proceso de reconocimiento de dígitos dentro de la red neuronal

1.1 Estructura de datos a usar

1. cantNeur_x_capa[i]: Cada celda del arreglo contiene la cantidad de neuronas de la capa i.
2. cantEntr_x_Cap[i]: Cada celda del arreglo contiene la cantidad de entradas de la capa i.
3. entradaCapa[i][j]: Cada celda j del arreglo contiene la entrada j para la capa i.
4. pesos[i][j][k]: Cada celda k del arreglo contiene el peso k de la neurona j de la capa i.
5. bias[i][j]: Cada celda j del arreglo contiene el bías de la neurona j de la capa i.
6. cantCap: Cantidad de capas que contiene la red neuronal.
7. NumImagTest: Es el número de imágenes de dígitos que hay en el archivo de imágenes de testeo. Su valor es igual a 10000.
8. contCorrect: Es el número de aciertos en la clasificación de una imagen.
9. contIncorrect: Es el número de errores en la clasificación de una imagen.
10. UmbralClasif: Es el umbral mínimo, el cual superado, se considera que una neurona en la capa de salida se activó. Su valor es igual a 0.8.
11. objetivoImag: Es el número que representa la imagen del dígito leída. Este valor se obtiene desde el archivo de objetivos.

1.2 Pseudocódigo

Inicializar pesos y bías desde el archivo de configuración;

1. numCorrect = 0;
2. numIncorrect = 0;
3. para cada imagen del archivo de imágenes de testeo
    4. Inicializar entradaCapa[0][j] desde archivo de imágenes de testeo;
    5. Leer objetivoImag desde el archivo de objetivos de testeo;
    6. para contCap = 0 a (cantCap - 1)
        7. para contNeur = 0 a cantNeur_x_capa[contCap]
            8. suma = 0;
            9. para contEntr = 0 a cantEntr_x_Cap[contCap]
                10. suma = pesos[contCap][contNeur][contEntr]*
                11.entradaCapa[contCap][contEntr];

12. fin para
13. suma = suma + bias[contCap][contNeur]
14. entradaCapa[contCap + 1][contNeur] = $\frac{1}{1+ e^{-suma}}$
15. fin para
16. fin para
17. validar(entradaCapa, contCorrect, contIncorrect)
18. fin para
19. inicio validar(entradaCapa, numCorrect, numIncorrect)
20. contUnos = 0;
21. para contNeurSal = 0 a 9
22. si (entradaCapa[cantCap][contNeurSal] >= UmbralClasif)
23. ++contUnos;
24. fin si
25. fin para
26. si (contUnos > 1)
27. ++ contIncorrect
28. sino
29. si (objetivoImag <> 0)
30. si (entradaCapa[cantCap][objetivoImag - 1] >= UmbralClasif)
31. ++contCorrec;
32. sino
33. ++contIncorr;
34. fin si
35. sino
36. si (entradaCapa[cantCap][9] >= UmbralClasif)
37. ++contCorrec;
38. sino
39. ++contIncorr;
40. fin si
41. fin si
42. fin si
43. fin validar

1.3 Explicación

La idea es primero inicializar los pesos y las bías de cada neurona de cada capa desde el archivo de configuración. Luego para cada imagen, lo que se hace es normalizar la imagen y guardar la imagen normalizada en cada celda del arreglo entradaCapa[0][j]. Este arreglo se convertirá en la entrada de la red neuronal. Esta entrada será procesada por la primera capa de la red neuronal (línea 6). Cuando la primera capa lo procese, cada neurona de la primera capa generará una salida que será guardada en cada celda del arreglo entradaCapa[1][j] (línea 14). Y así sucesivamente, hasta llegar a la última capa. Por lo tanto, la salida de la red neuronal, es decir la salida de la última capa de la red neuronal se guardará en cada celda del arreglo entradaCapa[cantCap][j]. Es ésta salida la que se valida.

A continuación, se detallará como es que la entrada de una capa determinada es procesada.

Toda la entrada entera de una capa es procesada individualmente por cada neurona de la capa en cuestión. Digamos que la entrada para una capa tal contiene R elementos, y digamos que la capa contiene S neuronas, entonces los R elementos son procesados individualmente por cada una de las S neuronas de la capa. Ahora, se explicará cómo es que procesa individualmente las S entradas una neurona cualquiera de la capa.

Para cada entrada p de las R entradas, una neurona tiene un número decimal, llamado peso. Cada peso respectivo es multiplicado por su respectiva entrada y el resultado es acumulado en una sumatoria. Al final, a la sumatoria de estas multiplicaciones se le suma un peso especial que no es multiplicado por ninguna entrada externa, sino que es multiplicado por 1, a este peso especial se le llama bías.

En la fig. 12.1, cada elemento de la entrada se ha guardado en un vector p, y cada peso de la neurona se ha guardado en un vector p. El b es un valor escalar.

Luego al resultado de W*p + b, se le aplica una función de activación, que para el caso de este proyecto, es una función sigmoidal (línea 14).

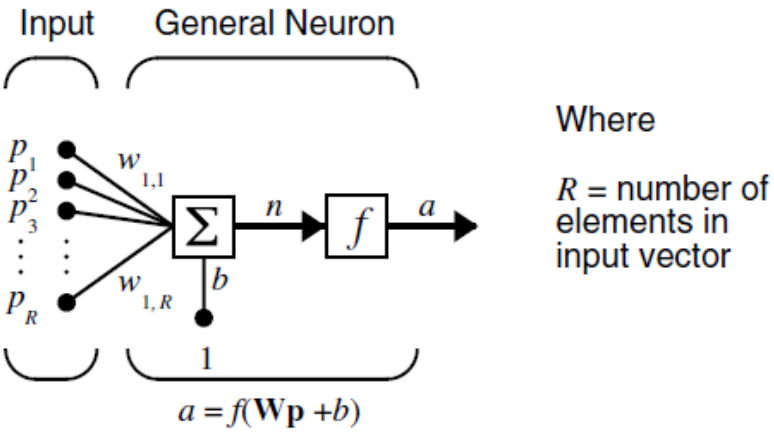


Fig. 12.1: La manera cómo procesa una neurona cualquiera la entrada entera de su capa respectiva [Howard Demuth 2009].

El resultado de esta función se convierta en la única salida de la neurona. Al final que la capa haya procesado la entrada, se tendrá S salidas, pues son S la cantidad de neuronas (Fig. 12.2).

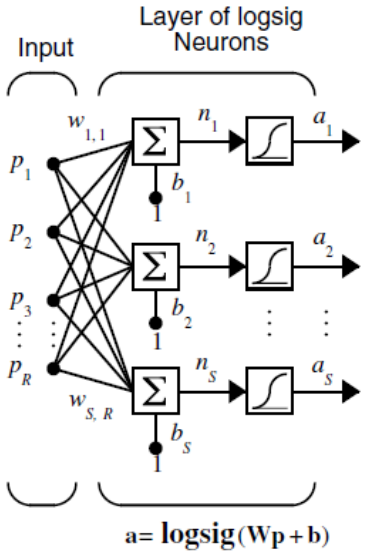


Fig. 12.2: La manera como una capa procesa su entrada

Estas salidas se convertirán en la entrada de la siguiente capa, y el proceso se repetirá así para cada capa.

CAPÍTULO 6

1. Implementación del prototipo de sistema de información para el reconocimiento de dígitos

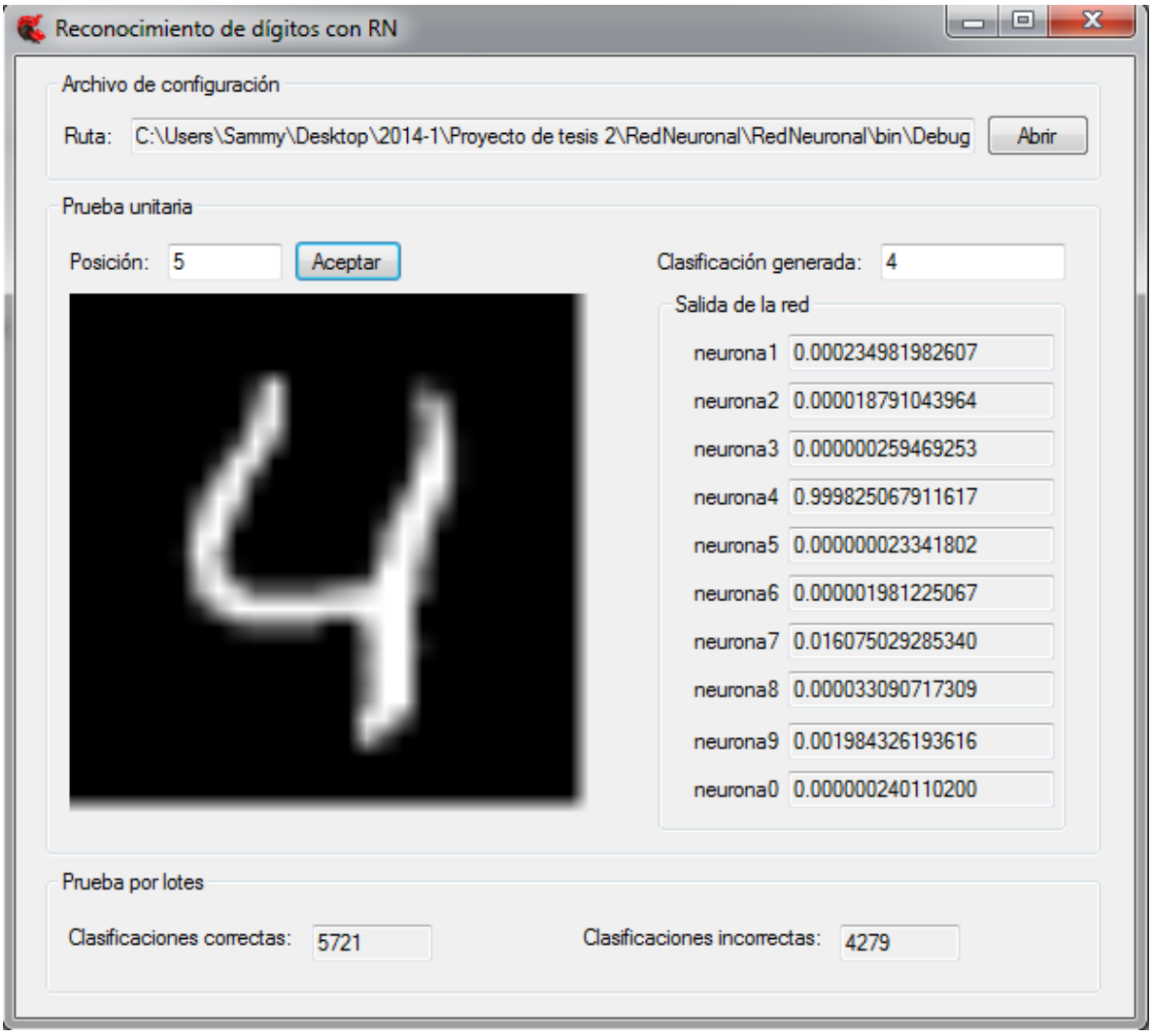


Fig. 13.1: Captura de pantalla del sistema de reconocimiento de dígitos con redes neuronales.

Las funcionalidades del sistema son las siguientes:

- **Realizar una clasificación por lote:**

1. El usuario selecciona la opción Abrir
2. El sistema muestra un árbol de archivos
3. El usuario selecciona el archivo de configuración de la red neuronal y selecciona la opción abrir.
4. EL sistema comenzará a poner a prueba cada imagen que yace en el archivo "t10k-images.idx3-ubyte". A la vez también ira leyendo cada número del archivo de objetivos "t10k-labels.idx1-ubyte" para comparar el valor generado por la red neuronal y el valor esperado para cada imagen. El sistema muestra en la caja de texto clasificaciones correctas la cantidad de dígitos en la cual la red neuronal acertó en clasificar. El sistema muestra en la caja de texto clasificaciones incorrectas la cantidad de dígitos en la cual la red neuronal erró en clasificar.

- **Realizar una clasificación unitaria**

1. El usuario ingresa, en el campo de texto posición, la posición del digito en el archivo de imágenes de testeo a querer clasificar. Este número debe estar en [0,100000]. El usuario, luego selecciona la opción aceptar.
2. EL sistema pone a prueba la imagen que yace en dicha posición a través de la red neuronal. El sistema muestra la imagen al lado izquierdo en un tono de grises. El sistema muestra en el campo de texto Clasificación generada la clasificación generada por la red neuronal para dicha imagen. Esta clasificación se mostrará en negro si es correcto o en rojo si es incorrecto. En la sección Salida de la red, el sistema muestra el valor de salida de cada neurona de la capa de salida.

Se obtuvo la mejor eficacia en la etapa de prueba (57% de acierto de un total de 10000 imágenes) en una arquitectura de 2 capas con 43 neuronas en la capa oculta y 10 neuronas en la capa de salida. A pesar de que este prototipo soporta el procesamiento con varias capas (más de 2), no se pudo superar la eficacia de la arquitectura antes mencionada, tal vez, porque no se halló en la literatura la cantidad necesaria óptima de neuronas por capa en las demás arquitecturas probadas.

CAPÍTULO 7

1 Conclusiones

Los 4 objetivos mencionados en el capítulo 2, punto 3, se alcanzaron en su totalidad. El resultado generó dos softwares, un documento de proceso, una configuración y un prototipo.

Al finalizar este proyecto, se concluye que las redes neuronales son útiles para cualquier problema de clasificación, como por ejemplo el reconocimiento de caracteres, siempre y cuando haya data suficiente para el entrenamiento y que esta data sea una buena representación de todo el universo de la data.

También se concluye que el método Nguyen - Widrow es un buen método para mejorar y acelerar la fase de entrenamiento.

Sin embargo, a pesar de que este prototipo soporta el procesamiento con varias capas (más de 2), no se pudo superar la eficacia de la arquitectura antes mencionada en otras arquitecturas de tres capas o más, tal vez, porque no se halló ni en la experimentación realizada en el proyecto ni en la literatura la cantidad necesaria óptima de neuronas por capa.

2 Recomendaciones

Este proyecto, sin mucho esfuerzo, puede ser adaptado para reconocer otros caracteres. El principal cambio sería la cantidad de neuronas de la última capa, la cual sería igual al total de grafemas del alfabeto español.

Según la bibliografía, se encontró la cantidad óptima de neuronas para una red neuronal de 2 capas. Sin embargo, no se encontró la cantidad óptima de neuronas para una red neuronal de 3 capas o más. Por tanto, una manera como se podría mejorar la eficacia de la red neuronal en la fase de testeo es hallando la cantidad optima de neuronas por cada capa para una red neuronal de 3 capas o más.

Otra manera de mejorar la eficacia de la red neuronal en la fase de testeo es cambiando la arquitectura de la red neuronal. Pero esto significaría rediseñar la fase de entrenamiento y rediseñar la fase de testeo.

Este proyecto no ha tratado con la segmentación de los caracteres dentro de una imagen de documento. Así, la red neuronal ya entrenada se podría incorporar dentro de un sistema mayor, el cual reciba un documento de texto, segmente los caracteres del documento, reconozca cada carácter y genere una nueva capa de texto.

Referencias bibliográficas

1. ADOBE SYSTEMS INCORPORATED
2013 *Adobe Acrobat XI: Ayuda y tutoriales.* Material de enseñanza
Consulta: 14 de abril de 2013
<http://www.adobe.com/la/>

2. ADOBE SYSTEMS INCORPORATED
2003 *PDF Reference.* Edición 1.5
Consulta: 14 de mayo de 2013
< http://www.adobe.com/devnet/pdf/pdf_reference_archive.html>

3. HART, Anna
1992 "Using Neural Networks for Classification Tasks -- Some Experiments on Datasets and Practical Advice". *The Journal of the Operational Research Society*, Vol. 43, número. 3, pp. 215-226.

4. BOOK INDUSTRY GROUP
2012

5. ISAZA C. y J. Vargas y C. Gaviria, L. Hernández
"Automatic OCR System for Colombian DNIs", *IEE*

6. GUERRERO CALLE, Fiorella Yvette
2009 *Diseño e implementación de una red neuronal para un sistema de reconocimiento óptico de caracteres (OCR).* Tesis en licenciatura en Ciencias e Ingeniería. Lima: Universidad Católica del Perú, Facultad de Ingeniería Electrónica.

7. DEMUTH, Howard y Mark BEALE y Martin HAGAN
2006 *Neural Network Toolbox 6 User's Guide*. Edición 6.0.2. The MathWorks, Inc

8. HEATON, Jeff
2008 *Introduction to Neural Networks for C#.* Edición 2. WordsRU.com.

9. HILERA GONZALES y José R. y Juan P. ROMERO VILLAVERDE y José A. GUTIERREZ DE MEZA
"Sistema de reconocimiento óptico de caracteres (OCR) con redes Neuronales". *Departamento de Ciencias de la Computación, Universidad de Alcalá de Henare*s.

10. MathWorks
2014 R2014a Documentation
Consulta: 5 de julio de 2014
<http://www.mathworks.com/help/nnet/ref/initnw.html>

11. MONGUE OSORIO, Manuel Alejandro
2008 *Diseño de una arquitectura para una red neuronal artificial perceptron multicapa sobre un FPGA aplicada al reconocimiento de caracteres*. Tesis en licenciatura en

Ciencias e Ingeniería. Lima: Universidad Católica del Perú, Facultad de Ingeniería Electrónica.

12. VALENCIA Marco Antonio y Cornelio YÁNEZ y Luis PASTOR
2006 "Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones". *Centro de investigación en Computación del IPN*

13. ORGANIZACIÓN MUNDIAL DE LA SALUD (OMS)
2011

14. CASEY Richard y LECOLINET Erci
1996 "A Survey of Methods and Strategies in Character Segmentation", *IEEE.* Vol 18, número 7, pp 690-705

15. CORMEN Thomas y Charles LEISERSON y Ronald RIVEST.
2009 *Introduction to Algorithms*. Edición 3. Mit Press.