APRENDIZAJE DE REDES CONVOLUCIONALES Y RECURRENTES PARA LA OPERACIÓN DE SECUENCIAS CON MÚLTIPLES DÍGITOS MANUSCRITOS

ANGÉLICA DANIELA QUEVEDO CORTÉS

FUNDACION UNIVERSITARIA KONRAD LORENZ

FACULTAD DE MATEMATICAS E INGENIERA

BOGOTA

2016

APRENDIZAJE DE REDES CONVOLUCIONALES Y RECURRENTES PARA LA OPERACIÓN DE SECUENCIAS CON MÚLTIPLES DÍGITOS MANUSCRITOS

ANGÉLICA DANIELA QUEVEDO CORTÉS

TRABAJO DE GRADO PARA OPTAR POR EL TITULO DE MATEMATICA

JUAN CARLOS CAICEDO RUEDA

FUNDACIÓN UNIVERSITARIA KONRAD LORENZ

FACULTAD DE MATEMATICAS E INGENIERA

BOGOTA

2016

NOTA DE APROBACIÓN

_____

_____

_____

_____

FIRMA DEL JURADO

_____

FIRMA DEL JURADO

Bogotá 02 de mayo del 2016

A mi mama Eliana Cortes y a mi papa Leonidas Quevedo por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como personal, por su incondicional apoyo perfectamente mantenido a través del tiempo,

Finalmente a mis maestros, aquellos que marcaron cada etapa de nuestro camino universitario, y que me ayudaron en asesorías y dudas presentadas en la elaboración de la tesis.

Todo este trabajo ha sido posible gracias a ellos.

TABLA DE CONTENIDO

LISTADO DE TABLAS.

LISTADO DE ILUSTRACIONES.

1. **INTRODUCCIÓN.**

Actualmente las Redes de Neuronas Convolucional (CNN) (Lecun, 1989), han permitido avanzar de forma sorprendente en el campo del reconocimiento de imágenes, ya que poseen una gran capacidad aprender de forma simultanea características de forma local y discriminativa, usando el álgebra lineal como principal herramienta, ya que convierte todo dato de entrada en representaciones vectoriales que facilitan su múltiple procesamiento [12].

Las Redes Convolucionales son uno de los métodos más populares para el reconocimiento de cualquier tipo de objetos, tales como el reconocimiento facial, dígitos manuscritos, el reconocimiento de sonidos bajo el entrenamiento con múltiples datos de entrada, y de esta misma manera los modelos de redes de neuronas recurrentes se establecen como un sistema completo para dar solución a ciertos problemas de clasificación de forma eficiente. Hasta el momento se han empleado con gran éxito cuando los datos de entrada son de tipología secuencial, cadenas de caracteres o de texto [14].

Tareas tales como el reconocimiento de texto y dígitos manuscritos pueden considerarse una tarea ya resuelta por una maquina u ordenador, pero la labor es mucho más compleja en cuanto al procesamiento de imágenes se trata, ya que involucra más variables, como color, bordes, texturas, iluminación, contraste entre otras.

Como ayuda para la tarea de reconocimiento de imágenes se han creado distintas bases de datos listas para ser procesadas, que contienen gran variedad de ejemplos ya sean imágenes, textos o dígitos. En este punto hacemos alusión a la base de datos MNIST ya que es la base de datos usada para el desarrollo del proyecto, la cual proporciona una variedad de datos presentados como dígitos de carácter escrito, y es fácil de conocer gracias a las Redes Convolucionales,

determinando las características aprendidas gracias a los datos de entrenamiento[16] .

Las CNN hacen el proceso de reconocimiento gracias a valores previamente ingresados durante todo el camino, además sus resultados en cuanto a aprendizaje no supervisado resultan ser más eficientes [17].

La propuesta del proyecto involucra la suma y la resta de dígitos manuscritos de una sola cifra, una tarea que significa un reto académico ya que es análoga a enseñar a sumar a un niño con tan solo una identificación previa de los dígitos. La tarea comienza a partir del reconocimiento de los dígitos y caracteres a través de una Red Convolucional. Para llevar a cabo esta tarea fue necesaria la creación de una base de datos de símbolos (+,-) de forma que se puedan concatenar los dígitos y así poder crear secuencias de tipo: digito, operador, digito.

Pero la tarea no yace en determinar o reconocer de forma individual cada caracter. Al tratarse de una secuencia se implementó una arquitectura que es capaz de incorporar dos sistemas de redes neuronales: uno para el reconocimiento de cada caracter (Red Convolucional, CNN) que realiza un proceso de clasificación y otro para recordar dichas salidas, es decir obtener esta secuencia a partir de una Red Recurrente (RNN) aprovechándonos de su característica de retroalimentación dinámica. Al final esta secuencia será obtenida y tratada para obtener una salida continua, es decir un numero resultante de la operación especifica aplicada a los dígitos de la secuencia, concretamente un proceso de regresión.

El desarrollo de este proyecto permite generar una primera evaluación de que tan eficiente puede ser crear mecanismos que sean capaces de procesar operaciones matemáticas, más adelante se podrá aumentar la complejidad de dichos procesos agregando operaciones de mayor orden y tamaño como sumas con dígitos de más cifras, derivadas, integrales, etc. Podemos estar hablando de un principio en el área del procesamiento matemático a través de redes neuronales, con la creación

de mecanismos capaces de resolver cualquier tipo de problema matemático operacional.

1.1 DESCRIPCIÓN.

En la actualidad, tareas que son simples y cotidianas para el hombre se constituyen como un gran reto de la tecnología, tales como el reconocimiento de imágenes, el entendimiento de ciertas situaciones, la toma de decisiones, entre otros[19].

Estas tareas requieren de un alto procesamiento en secuencia, que para el hombre tiene cabida dentro de unidades funcionales y estructurales denominadas neuronas, que en una interconexión masiva permiten un procesamiento de alto nivel de forma paralela con resultados inmediatos. La gran tarea de la comunidad académica es poder adaptar estos procesos a una forma computacional, la solución se estable en adaptar las redes de neuronas biológicas de forma artificial, implementando las ANN.

Las ANN (Artificial Neuronal Network) son modelos matemáticos inspirados en sistemas biológicos adaptados computacionalmente que comparativamente con otros métodos de simulación son más rápidos y producen salidas casi exactas, su tarea consiste en emular funciones propias del cerebro humano.

En el siguiente proyecto se hace uso de sistemas de ANN para el reconocimiento de patrones bajo el método de redes neuronales de tipo Convolucional propuesto por Yann Lecun las cuales se hicieron muy famosas para la década de los 80 [2]. Una de las características de este tipo de red neuronal es el ingreso de datos de entrenamiento que son sometidos a transformaciones de forma consecutiva [3]. El método de redes de neuronas Convolucionales se ha convertido en uno de los mejores para el reconocimiento de objetos, entre ellos los dígitos manuscritos.

Los recientes avances en el desarrollo de redes neuronales para reconocimiento de objetos demuestran que estos modelos pueden reconocer de forma muy precisa los dígitos manuscritos por diversos usuarios. Este problema puede resolverse con más del 90% de precisión en la colección de ejemplos de MNIST [4].

En este proyecto se diseña una arquitectura capaz de manipular una secuencia de dígitos extraídos de la base de datos MNIST y hacer operaciones entre estos usando una colección de operadores recolectados y procesados de forma similar a MNIST.

Este modelo procesa secuencias de 3 caracteres de tipo: digito, símbolo, digito, de los cuales conocemos el resultado. Se incorpora una Red Neuronal Recurrente que recordará cada caracter de la secuencia y al final producirá una salida que será entendida por sistema, los dígitos solo serán sumados o restados. En pocas palabras esta arquitectura no solo manipula los símbolos asociados a los dígitos manuscritos, sino que también recordará los patrones visuales y tomará una decisión al respecto.

Esta arquitectura se construye a partir de tres módulos:

1. El reconocimiento de caracteres a partir de una Red Convolucional, es decir la intervención de un proceso de clasificación. Dicha salida será conectada a una Red Recurrente.

2. La incorporación de la Red Recurrente para el procesamiento de la salida de la Red Convolucional. Es decir es almacenamiento de cada salida para así identificar la secuencia.

3. El procesamiento de la salida de la Red Recurrente es decir la operación entre los dígitos de la secuencia lo cual retorna uno o dos dígitos resultados de dicha operación, tras un proceso de regresión. En este último punto se evalúa la

eficiencia con la que el modelo acierta para cada iteración y así mismo el porcentaje de error que se comete (80 mil iteraciones).

1.2 ÁREA DE CONOCIMIENTO.

El proyecto abarca el área de la inteligencia artificial, dentro de esta las redes neuronales específicamente en los sistemas de redes de neuronas Convolucionales y Recurrentes, donde se establece la interacción en conjunto de áreas como el álgebra lineal, el cálculo diferencial, parcial. Todo esto de forma conjunta intenta emular funciones propias del cerebro humano, cuya tarea fundamental es la de aprender descriptores con el fin de reconocer y así mismo clasificar de forma total.

1.3 ALCANCES Y DELIMITACIONES.

Este proyecto replica la solución propuesta por las redes Convolucionales para la identificación de dígitos manuscritos, donde a partir de esto se generará un modelo en el cual se realicen operaciones entre secuencias compuestas por mismos números. A partir del resultado se determina el grado de efectividad de este modelo sometiéndolo a evaluación ingresando diversos ejemplos provistos por la base de datos MNIST y la base de datos de operadores creada en función del proyecto.

Las limitaciones se basan en el reconocimiento únicamente de dígitos manuscritos. El software con el cual se desarrolló el proyecto es Tensorflow, una biblioteca de código abierto para el cálculo numérico mediante grafos de flujo de datos y será realizado por una sola estudiante: Angélica Daniela Quevedo Cortes

de la carrera de matemáticas, bajo la supervisión de Juan Carlos Caicedo Rueda docente de Ingeniera de sistemas.

2. OBJETIVOS.

2.1 OBJETIVO GENERAL.

Diseñar y evaluar una estrategia computacional basada en redes convolucionales y Recurrentes para realizar operaciones con múltiples secuencias de dígitos manuscritos.

2.2 OBJETIVOS ESPECÍFICOS.

- Reproducir el trabajo de reconocimiento de dígitos manuscritos bajo el sistema de Redes Neuronales Convolucionales.
- Proponer e implementar un modelo para el análisis y procesamiento de secuencias de dígitos.
- Evaluar el resultado del procesamiento de secuencias de dígitos manuscritos experimentalmente.

3. MARCO REFERENCIAL.

Las redes neuronales artificiales ANN hacen uso de algoritmos inspirados en la mente humana y en nuestra comprensión de cómo el cerebro aprende: lo que se le enseña o lo que percibe.

Aspectos simples para la naturaleza humana, tales como el reconocimiento de voz, reconocimiento de objetos, la recuperación de imágenes y la capacidad para recomendar o poseer cierto tipo de criterio, basados en acercamientos previos han

sido el principal interés en la implementación de sistemas inteligentes que sean capaces de simular estas funciones.

Con el crecimiento de la tecnología y la computación, las redes neuronales se han constituido como la principal solución a dicha tarea, haciendo uso de nuevos métodos de aprendizaje mucho más eficientes y mucho más rápidos, convirtiéndose en el centro de la generación de dispositivos de reconocimiento de voz e imágenes, que han empezado a superar sistemas anteriormente implementados para el reconocimiento de objetos de cualquier tipo.

Esto hace parte de una rama denominada aprendizaje profundo (Deep learning) que ha tenido enorme impacto en la informática, por lo cual este marco referencial dará un breve recorrido sobre los temas más importantes en el desarrollo de las Redes de Neuronas Artificiales.

3.1 Inteligencia Artificial.

La Inteligencia Artificial es un área que intenta desarrollar paradigmas que requieren las máquinas para realizar ciertas tareas, que eventualmente los humanos llevan a cabo con más eficacia y control [7,8]. Un sistema que posea inteligencia artificial debe ser capaz de:

1. Guardar conocimiento.

2. Aplicar tal conocimiento guardado para resolver algún tipo de problema.

3. Adquirir nuevos conocimientos a través de la experiencia.

Como podemos notar, esta área multidisciplinaria en donde intervienen las Matemáticas, La Computación, La Física y la Lógica, intenta recrear lo que es posible realizar con el cerebro humano solo que artificialmente, haciendo uso de algoritmos computacionales complejos.

La Inteligencia Artificial posee tres componentes claves,

1. La representación.

2. El razonamiento.

3. El aprendizaje.

La representación se basa en el lenguaje de estructuras simbólicas que representen tanto el conocimiento general como el conocimiento específico a la hora de la resolución de un problema.

El razonamiento es simplemente la habilidad para resolver problemas de cualquier índole, y por último, el aprendizaje que se lleva a cabo desde el entorno, es el que suministra la información al elemento de aprendizaje, el cual usa esta información para inducir mejoras en la base del conocimiento, donde finalmente todo llega a un último bloque que es el que realiza la tarea suministrada.

3.2 Deep learning o aprendizaje profundo.

Todo ser vivo tiene la capacidad de aprender, de ver, percibir, actuar, comunicar e interpretar de forma rápida y eficiente que hasta el momento ningún método de aprendizaje de máquina ha podido simular. Esta tarea que yace en el cerebro es un proceso profundo ya que implica una extensa cadena de conexiones sinápticas que involucran capas y capas de operaciones y procesos [13].

La gran tarea de la comunidad académica es determinar algoritmos que se acerquen a un procesamiento eficiente tal que simulen tareas del cerebro animal. En este punto se destacan los algoritmos de aprendizaje sin supervisión[1] y

[1] Aprendizaje no supervisado es un método de Aprendizaje Automático ajustable a las observaciones. Es decir no existe conocimiento a priori.

supervisados[2] que determinan un enfoque diferente para la alimentación de datos que se aplican según la necesidad y el entorno. Por esta razón se deben generar algoritmos que sean capaces de seguir una arquitectura de jerarquías, es decir un modelo profundo de características para el reconocimiento [15].

Las técnicas de aprendizaje profundo se encuentran incorporadas dentro de las técnicas de aprendizaje automático, que construyen modelos jerárquicos desde unos datos de entrada que se procesan en múltiples capas que son una serie de aplicaciones y operaciones matemáticas. En un concepto de Redes de Neuronas cada una de estas operaciones se denomina neurona y cada nivel de neuronas determina el nivel de profundidad [12].

Gracias a las técnicas profundas se ha logrado potenciar el reconocimiento automático, el procesamiento de escritos, palabras e imágenes.

Los Avances en deep learning han implicado altos costes en cuanto a procesamiento computacional ya que implica el uso de múltiples parámetros, por lo cual una de las fijaciones ha estado en la reducción de dichos costes manteniendo los resultados obtenidos. Dentro del proceso de deep learning se encuentran las redes de neuronas profundas, que se basan en un pre-entrenamiento supervisado de datos que constan de ciertas etiquetas [12].

En este caso consideremos la siguiente función:

$$f(x) = \sqrt{x + \cos(a + b)}$$

[2] El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento

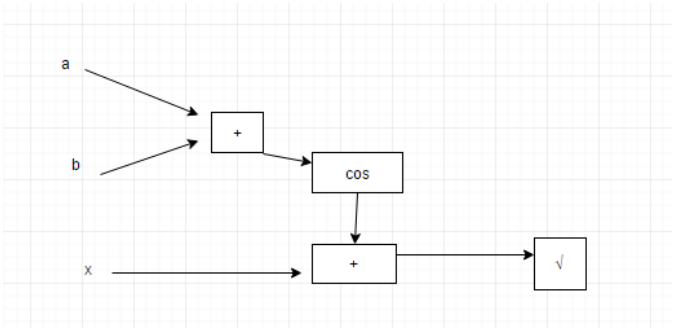

Ilustración 1. Arquitectura de una representación de profundidad 4, proveniente de una función.

Dentro del concepto de Redes Neuronales una neurona se constituye como un conjunto de operaciones entre pesos sinápticos que seguida de una transformación ya sea lineal o de mayor complejidad, en conjunto generan una arquitectura multicapa de neuronas de entrada, capas ocultas y de salida, es decir un modelo profundo [12].

Gracias a las arquitecturas profundas los algoritmos de inteligencia artificial pueden aprender múltiples niveles de representación y abstracción que les ayuden a crear sentidos visuales como las imágenes, o a reconocer textos, o secuencias como lo es el sonido.

El deep learning se basa en algoritmos computacionales complejos de aprendizaje automático, que como se mencionó anteriormente, intenta recrear o modelar abstracciones visuales y sensoriales usando arquitecturas que están compuestas de transformaciones no lineales múltiples. Es prudente acotar que las Redes Neuronales Convolucionales son usadas en este tipo de algoritmo. Algo interesante que busca el deep learning es el reconocimiento de ciertos patrones, sean aleatorios o no [12].

3.3 Red de Neuronas Artificiales.

En los últimos años, el estudio de las Redes Neuronales ha despertado un gran interés por parte de la comunidad científica. Su importancia radica en el hecho de que el cerebro humano trabaja de una manera distinta o como lo hace un computador actual.

El cerebro humano es una maquina altamente compleja y no lineal capaz de realizar y procesar información. El mismo tiene la capacidad de organizar sus estructuras constituyentes conocidas como Neuronas, para así poder llevar a cabo (o procesar) cierta cantidad de datos, que de hecho, en varios casos, la velocidad de procesamiento de la misma suele realizarse a una velocidad mayor que la de un computador.

Las redes neuronales, a diferencia del cerebro humano, están constituidas en su totalidad de neuronas artificiales, que tienen como función o están diseñadas para llevar a cabo ciertas tareas o funciones que van a depender exclusivamente del interés del problema que se quiera llevar a cabo. Ahora bien, la pregunta siguiente que deberíamos responder es: ¿Cómo construir una Red de Neuronas? La misma es implementada generalmente usando dispositivos electrónicos o bien simulándolas en mediante un software digital [7,8].

El propósito fundamental de la construcción de estas redes es poder llevar a cabo múltiples actividades, todas de la mano con el proceso de aprendizaje, que es el objetivo final cuando se tienen bien armadas o diseñadas. Podemos resumir todo lo comentado anteriormente, diciendo que las Redes Neuronales son un distribuidor masivo (compuesto de unidades llamadas neuronas) hechas con la finalidad de procesar información para su posterior uso en el proceso de aprendizaje. Las Redes Neuronales, al igual que el cerebro humano, adquieren el conocimiento de su entorno mediante el proceso de aprendizaje. El procedimiento

usado para llevar a cabo este proceso se conoce como Algoritmos de Aprendizaje [8].

Una de las aplicaciones de las Redes Neuronales en el campo del reconocimiento es el de dígitos manuscritos mediante programas computacionales.

Ya en el campo de del reconocimiento de dígitos manuscritos se pueden utilizar redes neuronales para abarcar el problema de una forma diferente, tomando ejemplos de números manuscritos existentes y utilizándolos como datos de entrenamiento para posteriores reconocimientos. Como resultado de entregarle más datos de números manuscritos a las redes neuronales, aumenta la precisión en el reconocimiento de nuevos datos.

Una red neuronal podría ser entrenada para reconocer miles de millones de dígitos manuscritos con un porcentaje incluso de 99% de exactitud [9].

De forma general una neurona artificial es un modelo matemático simple de una neurona (Mc Culloch y Pitts, 1943) que de forma general es de la siguiente manera:

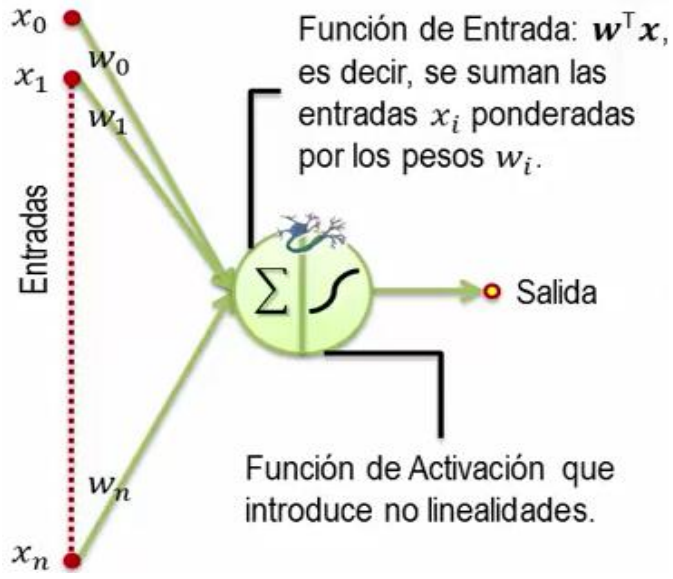

Ilustración 2. Concepto general de una red neuronal artificial.

La neurona recibe un número de entradas determinado y a cada una de estas entradas se le aplica una función de carácter matemático retornando una salida. Hablando de una arquitectura más compleja, una red neuronal se compone de la integración de miles de estructuras neuronales, haciendo de una red neuronal un modelo de arquitectura profunda.

3.4 Red de Neuronas Convolucionales

Las Redes Neuronales Convolucionales están compuestas por neuronas multicapas, que se constituyen como la unidad básica de inferencia en forma de discriminador lineal y permite entonces seleccionar, mediante un algoritmo determinado, un cierto criterio de un sub grupo proveniente de un grupo de componentes más grande [7, 8,9].

Las Redes Neuronales Convolucionales están específicamente diseñadas para reconocer cualquier tipo de forma bidimensional con un alto grado de invariancia trasnacional, de escala y otras formas de distorsión. Con invariancia trasnacional nos referimos a cualquier cambio en la longitud del sistema de coordenadas de dos dimensiones ya fijado.

Estas redes están constituidas por esta estructura:

1. Extracción de características.
2. Mapeo de las características.
3. Submapeo o pooling.

Describamos cada una de ellas. Al referirnos a la extracción de las características, se está pensando en que cada neurona toma su entrada sináptica desde un campo receptor local de la capa previa. En el mapeo de características de cada

capa computacional de la red está compuesta de múltiples mapas de características, donde cada neurona individual está obligada a compartir el mismo conjunto de peso sináptico. En el submapeo o Pooling cada capa Convolucional es seguida por una capa computacional que lleva a cabo un submuestreo o Pooling y un promedio, por lo que la resolución de la característica del mapa es reducida. Algo importante a mencionar es que estas Redes Neuronales Convolucionales están actualmente siendo usadas para resolver o explicar modelos en Neurobiología [7].

3.4.1 Arquitectura de una Red Convolucional

Las Redes Convolucionales o CNN dentro del deep learning son un tipo de redes de neuronas cuya función es aprender características (features) incorporadas en imágenes, permitiendo implementar de forma más eficiente algoritmos de reconocimiento generando la reducción de parámetros requeridos. Las CNN se basan en las redes de neuronas convencionales de carácter no lineal [11].

El término de Red Convolucional fue presentado por prima vez por Kunihiko Fukushima en 1980 bajo el modelo de “Neocognitron”, que se basan en la descripción y funcionamiento de las cedulas C o células complejas y las células S o simples, lo cual consiste en una conexión en secuencia de las mismas en la corteza primaria de visión, donde se sigue una estructura de cascada en donde las características locales son extraídas por las células S y las características complejas son detectadas por las células C. Las características locales obtenidas en la entrada son integradas gradualmente y clasificadas en capas superiores en todo el camino [19].

Posterior a esto fue Yann Lecun con su propuesta de Backpropagation para el reconocimiento de dígitos y textos bajo el aprendizaje supervisado, y el uso de la Convolución que se aplicaron a S mapas de funciones de forma simultánea [20].

Las operaciones importantes en la formación de las redes pueden ser vistas como circunvoluciones entre pares de matrices, que puede representar mapas de entrada y de salida, donde se hace uso del gradiente descendiente con respecto a la función de mapas, o bajo la identificación de pesos entre matrices.

El enfoque dominante de las Redes Convolucionales es que han permitido proporcionar un rendimiento excepcional en el campo del reconocimiento, ya que son capaces de aprender características de forma local y generar a partir de estas vectores de imágenes.

Una Red Convolucional funcionalmente realiza la alimentación desde adelante y de forma general consiste en lo siguiente [2][21]:

1. Se genera una fase de entrenamiento el cual es supervisado y se lleva a cabo utilizando un proceso de aprendizaje estocástico[3].

2. En las primeras capas (capas Convolucionales) se extraen características generales y elementales de la imagen de forma sectorizada, las cuales están conectadas a las capas de entrada, estas capas Convolucionales procesan los píxeles de las imágenes en forma de matrices multidimensionales extraídas localmente en grupos de valores que poseen una correlación, y donde las estadísticas locales son invariantes a lo particular.

3. En las capas posteriores o capas de Pooling, se encargan de clasificar las características provistas por las fuentes anteriores. Este es un proceso que se da de forma secuencial y bajo un proceso de propagación hacia adelante y atrás. Acá se aplica una serie de filtros en cada etapa.

[3] El aprendizaje estocástico consiste en realizar cambios asignaciones aleatorias en los valores de los pesos y posteriormente evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

De forma general se puede visualizar de la siguiente manera:

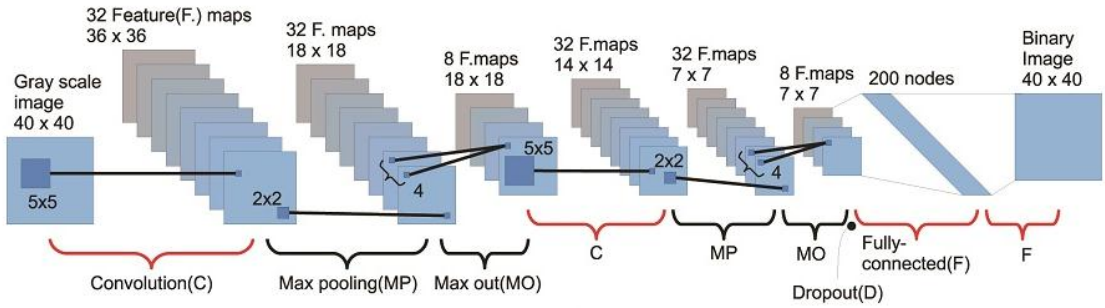

Ilustración 3. Aprendizaje profundo, [Documento gráfico], "Reconocimiento de objetos y la extracción región por Deep Convolucional Neural Network", 2014. Extraído de:http://mprg.jp/research/cnn_j.

La Redes Convolucionales se componen de varias capas que se aplican a distintos procesos, donde cada capa contiene un módulo de convolución, y después de esta se encuentra una capa de sub muestreo y una de normalización [36].

3.4.2 Capas de una Red Neuronal Convolucional.

Este tipo de redes se caracteriza por tener varios tipos de capas, como son: Capa Convolucional, muestreo o Pooling, y capa de conexión completa [37].

- Capa Convolucional

En esta se realiza una Convolución de la imagen de la capa previa, donde los pesos especifican el filtro de Convolución. Consiste en rejillas rectangulares de neuronas, donde cada neurona toma entradas de una rejilla rectangular de la capa anterior; los pesos para esta sección rectangular son los mismos para cada neurona en la capa Convolucional. Además, cada rejilla toma entradas provenientes de todas las rejillas en la capa previa, usando diferentes filtros [35].

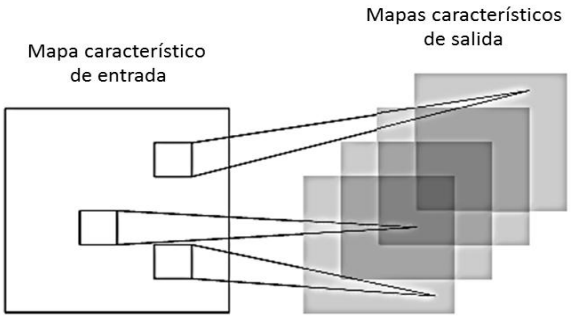

Ilustración 4. Ejemplo de una capa convolucional sencilla.

- Capa de Sub-muestreo o Pooling

La idea de realizar un Pooling sobre los mapas característicos obtenidos en capas previas se debe principalmente al ruido y las distorsiones. La función de esta capa es reducir progresivamente el tamaño espacial de los features. Este proceso de reducción de la resolución puede ser realizado de diferentes formas, pero la más conocida y la que usamos en el proyecto es el max o el Max Pooling.

La capa de Pooling funciona independientemente en cada rebanada donde se aplica un filtro distinto y se cambia la profundidad.

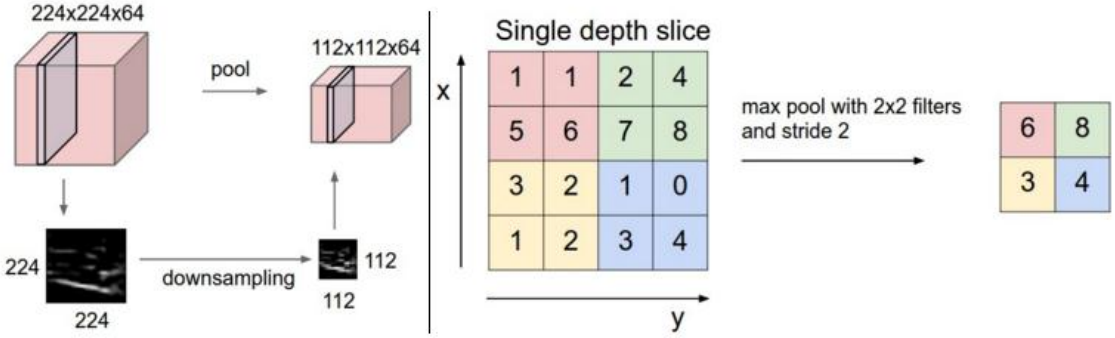

Ilustración 5. Ejemplo de capa de pooling, se puede evidenciar que los características se reducen de dimensión, además al aplicar el max pooling se obtiene el máximo valor por cada filtro de 2x2.

Para efectos del proyecto la técnica de submuestreo usada es el Max Pooling que extrae los valores máximos por cada filtro de 2x2 reduciendo la imagen de su tamaño inicial considerando las características de mayor importancia [40].

- Capa totalmente conectada

Las neuronas en la capa completamente conectada tienen conexiones completas a todas las entradas de la capa anterior, por lo cual su activación o función se puede calcular con una función lineal entre matrices [40].

3.5 El problema de clasificación.

Las Redes Neuronales son mecanismos capaces de resolver algunos problemas muy interesantes, una vez estos sistemas hayan sido capacitados ya sea bajo algoritmos supervisados o no. Se establecen como una solución óptima para la resolución de problemas de reconocimiento de patrones y de esta misma manera en procesos de clasificación.

Los procesos de permiten generar una asignación de etiquetas de clase para los datos de entrada, esto bajo una precisión arbitraria. Estos sistemas son particularmente adecuados para los problemas de decisión bajo la concepción de múltiples variables. Algunas de las funciones que ayudan a resolver estos procesos de clasificación se mencionan a continuación:

3.5.1 ¿Qué es Softmax?

Softmax es un modelo clásico, este tipo de regresión se da de forma natural y sencilla, es usado para asignar probabilidades a un objeto de pertenecer a cualquier posibilidad de una gran variedad de clases [5].

Ya más enfocados en el procesamiento de imágenes sabemos que al usar la base de datos de MNIST cada una de las imágenes corresponde a un digito, por lo cual necesitamos un modelo que determine la probabilidad de que esa imagen sea un digito. Tomemos como ejemplo el siguiente: tenemos un número 5 tenemos 85 % de probabilidad de que es un 5 de forma acertada pero podemos tener un 10% de que es un 6, y probabilidades más pequeñas de que sea el resto de dígitos.

Por esta razón se hace uso de la función Softmax que es un modelo natural y sencillo de usar, el cual tiene como objetivo asignar una probabilidad a un objetivo de ser distintas cosas diferentes, esta función es realmente eficiente y es usado desde los más simples hasta los más sofisticados en cuanto a modelos de redes neuronales .

Como se mencionaba anteriormente, este tipo de regresión logística es de uso frecuente en problemas de clasificación multiclase donde cada clase es mutuamente excluyente, ya basados en su uso en la interpretación de dígitos manuscritos donde la salida debe ser un único numero de 1 a 9 donde esta probabilidad debe sumar 1 [23].

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1 \middle| x^{(i)}; \theta\right) \\ p\left(y^{(i)} = 2 \middle| x^{(i)}; \theta\right) \\ \vdots \\ p\left(y^{(i)} = k \middle| x^{(i)}; \theta\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Dentro del modelo la salida es un vector que asigna un valor para cada uno de los dígitos manuscritos, el primer componente es la probabilidad de que una de las imágenes de entrada de un digito sea el numero 1 o que pertenece a la etiqueta o clase 1, Para lograr normalizar cada componente de tal manera que sume 1, la función Softmax aplica una función exponencial de sus entradas y luego las normaliza como se muestra en la siguiente expresión:

$$j(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$$

Lo interesante de esta función es que sólo posee una entrada en el vector con un valor cercano a 1, mientras que las entradas restantes estarán cerca de 0. En una predicción no tan eficiente se obtendrán variadas clases posibles, que tendrán más o menos la misma probabilidad.

Entonces tenemos que al dividir cada componente por la suma de todos los componentes iniciales se asegura que la suma es 1 ya que también partimos del fundamento de que las clases son mutuamente excluyentes.

$$j(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$$

3.5.2 ¿Qué es el Cross Entropy?

Es una función de error durante el proceso de aprendizaje, cuyo fin es disminuir el error en el intercambio de información entre neuronas. Este proceso de reducción del error puede ser extremadamente lento, lo cual puede retrasar los tiempos de aprendizaje. EL Cross Entropy Resulta de la simulación utilizando una función de error con mayor rendimiento, proporcionando un número reducido en cuanto a tiempos de estancamiento en el aprendizaje bajo la sustitución de las funciones cuadradas convencionales. La función del cross entropy se expresa de la siguiente manera [38]:

$$E_{m=}\frac{1}{m}\sum_{k=1}^{m}[t_k\, lny_k + (1-t_k)\, ln\,(1-y_k)]$$

3.5.3 ¿Qué es el Gradiente Descendente?

El Gradiente Descendente es un método cuyo fin es buscar la dirección en la que una pequeña variación del vector de pesos hace que el error decrezca de forma más rápida [39].

Para encontrar esta dirección se debe tener en cuenta, la salida que genera el gradiente en función al error con respecto al vector de pesos, esto se expresa de la siguiente manera:

$$\nabla E\left(\underset{w}{\rightarrow}\right) = \left[\frac{\partial E}{\partial W_0}, \frac{\partial E}{\partial W_1} \ldots\ldots \frac{\partial E}{\partial W_n}\right]$$

Acá la flecha permite reconocer la dirección en la cual pequeñas variaciones en los pesos producen las mayores variaciones en la función de error, que es la misma dirección que la que indica el gradiente de la función de error con respecto del vector de pesos [39].

3.6.4 ¿Qué es Relu?

Relu es una función lineal rectificadora cuya función es la siguiente:

$$f(x) = \max(0, x)$$

Esta se utiliza en lugar de una función de activación lineal para añadir no linealidad a la red, de lo contrario la red sería de forma lineal. Se caracteriza por ser una función lineal por tramos, que elimina la parte negativa a cero, y retiene la parte positiva [40].

3.5.5 ¿Qué es Backpropagation?

El algoritmo Backpropagation tiene como fin la actualización de pesos y ganancias con base en el error medio cuadrático, por lo cual actúa de forma iterativa, este algoritmo corre bajo el aprendizaje supervisado es decir necesita de datos de entrenamiento. La idea del mismo es reducir la función de coste al reducir el error en la clasificación de un ejemplo, este error se puede formular de la siguiente manera:

$$E = \frac{1}{2}\sum_{K}(d_k - y_k)^2$$

Para esto hay que modificar los pesos W_i en la dirección opuesta al gradiente es decir:

$$\Delta W_{ij} = -\mu\left(\frac{\partial E}{\partial W_{ij}}\right)$$

3.6 Aplicaciones matemáticas de carácter regresivo.

3.6.1 ¿Qué es la regresión lineal?

La regresión lineal es un modelo matemático cuyo fin es ajustar o determinar relaciones de dependencia de unas variables dependientes y unas variables independientes de forma aleatoria, de esta manera se seleccionan una función que realice dicho ajuste de la siguiente manera:

$$f_{\emptyset}(x) = \mathrm{w} \cdot \mathrm{x} + \mathrm{b}$$

$$\emptyset = \{w, b\}, w \in \mathbb{R}^d, b \in \mathbb{R}$$

Donde tenemos que $\boldsymbol{w}$ es un vector de pesos y $\boldsymbol{b}$ es un bias, donde se realiza un producto punto entre $\boldsymbol{w}$ y $\boldsymbol{x}$, de esta manera se trata de encontrar un valor para cada $\boldsymbol{w}$ y $\boldsymbol{b}$ que permitan generar la relación más exacta determinando un valor esperado.

Normalmente para calcular el error resultante de la aplicación de una regresión lineal se usa el error cuadrático medio, que corresponde a un estimador que mide el promedio de los errores al cuadrado, en pocas palabras la estimación calculada menos estimación real.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(Y_i - y_i)^2$$

Donde $\boldsymbol{Y}$ es el vector que contiene todas las predicciones y $\boldsymbol{y}$ es el vector de valores verdaderos, pero este se constituye como un valor esperado que puede ser muy variante cuando se tratan de valores realmente desconocidos [43].

3.7 Redes Neuronales Recurrentes.

Continuando con el estudio de Redes Neuronales, se llega al punto de las Redes Neuronales Recurrentes. Las mismas son un tipo de red neuronal con uno o más ciclos o loops de retroalimentación que puede ser de tipo local o global.

Este tipo de red realiza retroalimentación desde las neuronas de salida hasta la capa de entrada. Este tipo de redes se consideran recurrentes debido a que sus entradas son dependientes de las salidas de procesamientos anteriores. Otra forma a la cual se le presta gran atención es a la retroalimentación global cuando proviene de las neuronas escondidas de la red hasta la capa de entrada. Tendremos dos tipos de funcionalidades de estas Redes Neuronales Recurrentes, una de ella es la de las memorias asociativas y la del mapeo entrada y salida de

las redes, lo cual les permite generar predicciones. Sin embargo, deben ser entrenadas a partir de grandes conjuntos de datos [7, 8, 9].

La Red Neuronal Recurrente obtiene el resultado de uno o varios procesamientos anteriores y como se expone a continuación, se puede descomponer el proceso en una secuencia completa de entradas y salidas utilizadas en los siguientes ciclos.

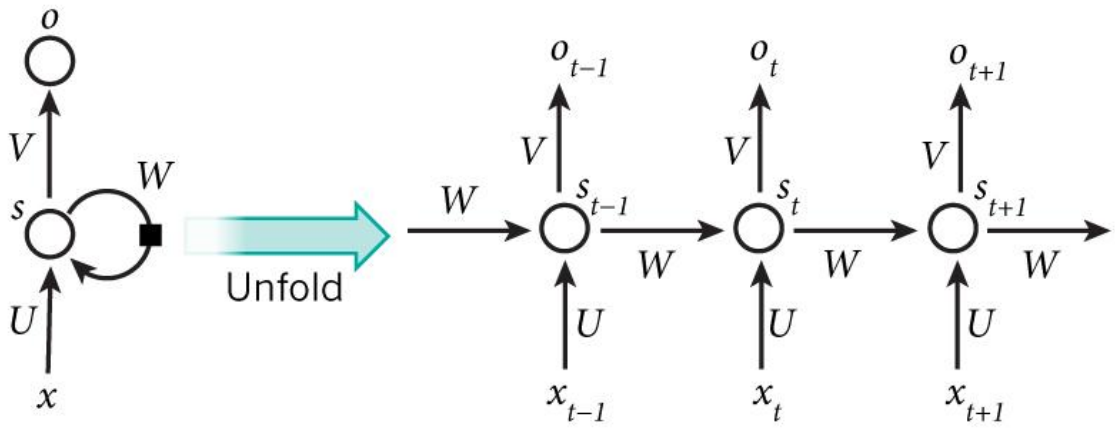

Ilustración 6. Modelo de arquitectura de una red neuronal recurrente.

Las Redes Neuronales Recurrentes pueden aprender datos a partir de entradas anteriores utilizando conexiones Recurrentes, lo cual significa que en el reconocimiento de patrones pueden crear relaciones entre dichas entradas y la actual, y a su vez permite determinar cuáles de los datos se encuentran más cercanos y más lejanos unos de otros [28].

Esta clase de procesamiento ha resultado ser particularmente efectivo cuando se trata de analizar uno a uno los pixeles en una imagen que no presentan dependencias espaciales específicas entre ellos, sino que dichas dependencias se crean en los recorridos secuenciales en diferentes direcciones y la red encargada de su aprendizaje aprenderá progresivamente las características de cada dato recibido y lo relacionará con los datos anteriores. El resultado del aprendizaje será

el aprendizaje de las distintas regiones de cada imagen y sus relaciones con entradas de imágenes similares de forma posterior.

4. MARCO TECNOLÓGICO

4.1 Tensor flow.

Tensorflow es un sistema de aprendizaje basado en el cálculo numérico, de código abierto, que hace uso de derivación automática, el uso de grafos y demás herramientas para el procesamiento de múltiples tareas. Tensorflow es una herramienta flexible y portátil.

Extraída de: http://techcrunch.com/2015/11/09/google-open-sources-the-machine-learning-tech-behind-google-photos-search-smart-reply-and-more/

La principal razón para seleccionar Tensorflow como herramienta base en el desarrollo del proyecto es que esta se especializada en tareas de machine learning, además de contar con un API extenso y completo que permite su fácil exploración. Es una herramienta fuertemente usada por empresas de gran tamaño como lo es Google, para el procesamiento de imágenes y de voz. Tensorflow se creó con fines académicos, inicialmente de carácter privado hasta el año 2015 donde fue liberado su código.

Tensorflow posee una alta experiencia que habla de un software especializado para tareas de aprendizaje, por lo cual satisface de forma completa todos los retos implicados en el proyecto, además de poseer librerías completas que contienen funciones de cálculos complejos implementando funciones para Redes Convolucionales, Recurrentes, algoritmos como el Softmax y Gradiente Descendiente, que son fundamentales para el desarrollo del proyecto, además de permitir hacer uso de una o más GPUs de procesamiento, está escrito en C++ y se puede programar con Python[5].

4.2 Datos de Minist.

El proceso de Deep learning tiene como objetivo el reconocimiento de patrones, que pueden ser imágenes de objetos, de letras o de dígitos en este caso dígitos manuscritos. MNIST es la más famosa base de datos para el procesamiento de imágenes cuyo contenido son dígitos escritos a mano o dígitos manuscritos dibujados de diversas formas, la cual es de carácter libre con más de 55.000 ejemplos de entrenamiento, 10.00 de prueba y 5,000 de validación. Cada una de las imágenes MNIST dispone de su respectiva etiqueta la cual indica el numero al cual hace referencia, dispuesta en el sitio web de Yannn LeCun's [6].

MNIST es una gran herramienta cuando se trabajan procesos de aprendizaje automático y métodos de reconocimiento de patrones en los datos del mundo real, ya que las imágenes están integradas de forma dimensionada listas para usar.

En este sitio se encuentran 4 tipos de archivos:

- tren-imágenes-IDX3-ubyte.gz: Imágenes del conjunto de entrenamiento (9912422 bytes)
- tren-etiquetas-IDX1-ubyte.gz: Etiquetas del conjunto de entrenamiento (28881 bytes)
- t10k-imágenes-IDX3-ubyte.gz: Imágenes aparatos de prueba (1648877 bytes)
- t10k-etiquetas-IDX1-ubyte.gz: Establecimiento de las etiquetas de prueba (4542 bytes).

Donde solo se dará una representación en un rango de 0 a 1 donde se determinará [24] la intensidad del pixel, con lo cual cada imagen ha quedado representada como arreglo de 28x28 = 784. Sin embargo, hay que tener en cuenta que no se trata únicamente de una sola imagen, se trata de 55.000 imágenes (imágenes de entrenamiento) es decir 55.000 X 784 lo cual se construye dentro

del concepto de un tensor. Por otro lado se tienen las etiquetas que son los dígitos naturales es decir del 0 al 9 que hacen referencia al número al cual corresponde la imagen [4].

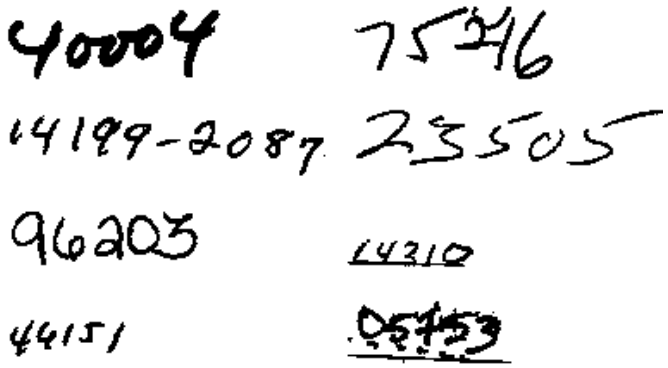

Ilustración 7. De datos de test - Ye Lecun hardwriten digit with a back-propagation network las imágenes poseen un tamaño de 28X28 pixeles que se representan como una matriz.

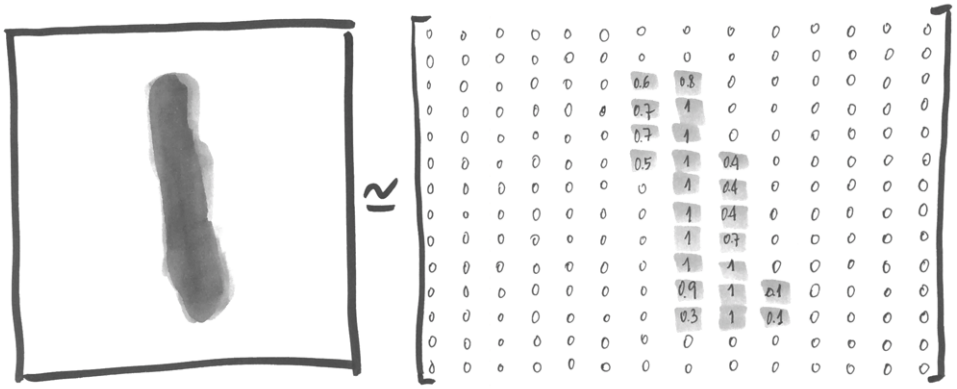

Ilustración 8. Representación de una imagen de forma matricial.

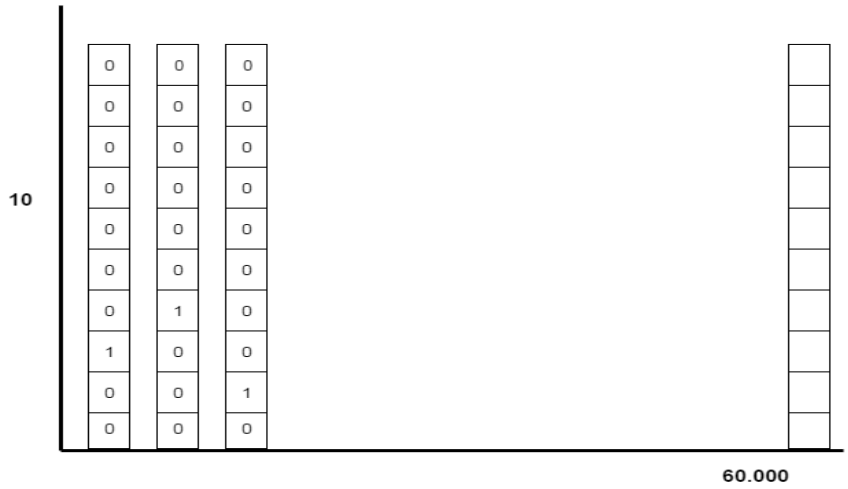

Ilustración 9. Main tarvel MNIST: Datos de entrenamiento.

4.3 Python.

Python es un lenguaje de programación de alto nivel, pero para poder desarrollar claramente el concepto hay que comprender la existencia de lenguajes de bajo nivel y de alto nivel. Los primeros también llamados lenguajes de máquina o de ensamble, son el único medio de interpretación comprensible por los sistemas computacionales, y poseen una sintaxis un poco más robusta y poco intuitiva para el ser humano. Sabiendo esto, para que una maquina comprenda el lenguaje de alto nivel este mismo debe ser procesado antes de ejecutarse en la máquina, lo cual se constituye como un problema en tiempo de compilación para los lenguajes de alto nivel.

Los lenguajes de alto nivel son de fácil entendimiento y toman menos tiempo de escribirse, además son portables y multiplataforma, es decir, funcionan en distintos ordenadores sin ninguna o con mínimas modificaciones. Gracias a estas ventajas los programas que traducen lenguajes de alto nivel se agrupan en dos categorías: los compiladores y los intérpretes, el intérprete tiene como función leer un programa de alto nivel y a su vez ejecutarlo, este se ejecutada por bloques de código, en cambio el compilador lee el programa y lo traduce por completo antes

de que el programa lo ejecute. En este contexto al programa de alto nivel se le llama código fuente y lo que se tradujo se denomina código ejecutable, después de compilar un programa se puede ejecutar repetidamente, los programas realizados en Python son ejecutados por un intérprete, de dos formas modo interactivo o por línea de comandos [10].

4.4 Anaconda

Extraída de: https://www.continuum.io/blog/developer-blog/anaconda-24-release

Anaconda es una distribución del lenguaje Python especializada en computación científica, manejo de big data, análisis y procesamiento de datos. Entre las utilidades más destacadas disponibles en sus versiones para Python 2.7 y Python 3.5, se encuentran librerías que facilitan el manejo de operaciones algebraicas como NumPy, SciPy, NumExpr y varias otras que utilizan el kernel de matemáticas de Intel (MKL). Otra de las grandes ventajas de Anaconda, utilizadas para el presente trabajo, es que su complemento IPhyton permite mostrar las imágenes ingresadas directamente en consola, lo cual permite comprobar la forma en la que el reconocimiento tanto de dígitos como se símbolos se ha realizado [18, 22].

5. ESTADO DEL ARTE

Es importante resaltar los trabajos más sobresalientes durante el tiempo con respecto a los temas tratados: Redes Convolucionales, Redes Recurrentes y bases de datos de dígitos manuscritos, por lo cual se hará un breve recorrido teniendo en cuenta los resultados más significativos en cada campo.

5.1 Redes de neuronas Convolucionales.

5.1.1 Reconocimiento de dígitos manuscritos (Yann Lecun, 1989).

El termino Red de Neuronas Convolucionales fue introducido en el año 1989 por el señor Yann Lecun (LeCun et al., 1989) precedido por el término Chip de Redes Neuronales. Inicialmente se buscó una solución para poder crear conocimiento en un sistema sobre dígitos a partir del reconocimiento de imágenes a través de la extracción de sus píxeles uno a uno. Los experimentos se iniciaron intentando hacer reconocimiento de dígitos escritos por humanos y códigos de área (zip codes) entregados por una empresa local de envíos. Sin embargo, en ese momento el reconocimiento fue solamente efectivo para los números diseñados adecuadamente más no para los dígitos escritos de forma más corriente, pues no se lograba el reconocimiento adecuado con un aprendizaje considerable. A pesar de ello, el trabajo iniciado por el señor Lecun permitió visualizar algunos de los escenarios que llegarían más adelante como la aplicación del algoritmo de Back Propagation [42].

5.1.2 Clasificación con redes convoluciones profundas (Krichevsky, 2012).

En el año 2012 es decir 23 años después de la primera implementación por Yan Lecun, Krichevsky generó una red de convolución más profunda que la de Lecun, centralizando su trabajo en el reconocimiento de objetos a partir de aprendizaje supervisado y bajo la necesidad de ese reconocimiento, las Redes de Neuronas Convolucionales se constituyeron como el modelo más eficiente para dar solución a la necesidad planteada [25].

El trabajo de Krichevsky se realizó con la base de datos IMAGEnet que consiste en un conjunto de datos de 15 millones de imágenes de alta resolución

pertenecientes a aproximadamente 22.000 categorías. Krichevsky tuvo que redimensionar las imágenes usadas, por lo cual las recortó de tal forma que ajustó sus dimensiones a 256 x 256.

Krichevsky utilizó una red de 8 capas aprendidas Convolucionales ordenadas según una estimación de importancia, donde se hizo uso del Relu para evitar la linealidad lo cual consiste en rectificar las unidades lineales de salida. La ventaja de este modelo es que no requería de la normalización de los datos de entrada de la red por lo cual el sistema era rápido de entrenar. En la arquitectura de Krichevsky, las 5 primeras capas eran convolucionales y las demás eran de simple conexión. Entre la segunda capa convolucional y la quinta existían relaciones con los mapas del núcleo de la capa anterior, que son capas de respuesta a la normalización.

Los resultados obtenidos que fueron presentados en la competencia ILSVRC y se pueden apreciar de forma más clara con la siguiente imagen:

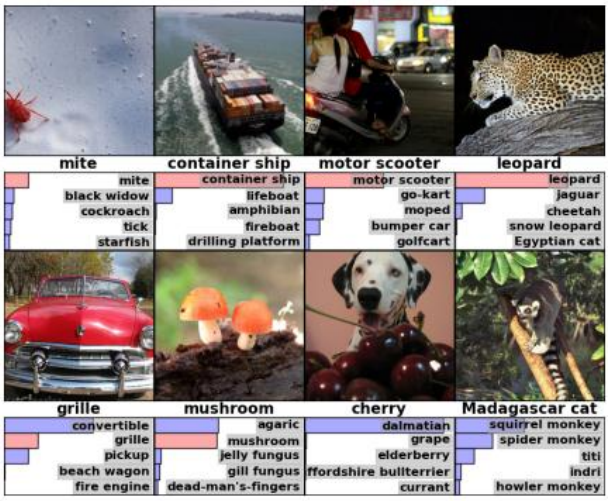

Ilustración 10. En la imagen es posible apreciar 8 imágenes de prueba y las cinco clases más probables consideradas por el modelo. En la parte de abajo se muestra la probabilidad asignada a la etiqueta correcta.

Para la imagen se tienen las 5 mejores predicciones sobre 8 imágenes de entrada o prueba, el sistema determino las 5 etiquetas que más probabilidad tenían de ser cada una [25].

5.1.3 Haciendo aprendizaje profundo con Convoluciones (Zeiler, 2013).

Posterior al trabajo de Krichevsky apareció Zeiler un año después (2013) con una Red Neuronal Convolucional con mayor profundidad. Aunque esto sea una gran avance no se ha estudiado hasta el momento a profundidad el comportamiento interno de estos modelos, sin un entendimiento basto del funcionamiento, el desarrollo de mejores modelos se reduce a ensayo y error [26].

La propuesta de Krichevsky es usada en la propuesta de Zeiler cuyo fin era explorar más conjuntos de datos. Este modelo utiliza el modelo de Redes Convolucionales estándar completamente supervisado. Se establece una probabilidad del vector y sobre la diferencia de las categorías, y cada capa consiste en la salida de la capa de convolución anterior, con un conjunto de filtros aprendidos, que han pasado por la función Relu, el uso del max pooling que normaliza la salida a través de mapas de características y en la parte superior de las capas está el Softmax donde se usa un conjunto de N imágenes, y un conjunto N de etiquetas, haciendo uso de la función Cross Entropy para determinar la pérdida en cuanto a la selección de la imagen adecuada. Cada capa está conectada a las otras haciendo uso de back propagation que es la derivada de la pérdida con respecto a un parámetro establecido en toda la red y la actualización de los parámetros a través del descenso hacia atrás por el gradiente descendiente.

Este modelo es muy parecido al de Krichevsky pero las conexiones dispersas (refiere a que Krichevsky utlizo dos GPUs) son reemplazadas por conexiones de peso. Este modelo se alimenta de los datos IMAGEnet 2012.

De forma general la construcción de esta red consistía en la separación de cada sector de la imagen, y en cada paso por cada píxel de la imagen se develaban las diferentes estructuras que existían en el mapa de características y a partir de estas características se determinaba el grupo de imágenes correspondientes.

Tras esto se obtuvo que los Modelos de Redes Convolucionales se pueden estructurar de distintas formas tales que permitan una clasificación eficiente, donde sus resultados dependieron de la detección de características de forma zonal. Este modelo es aplicable a cualquier conjunto de datos de entrenamiento como Caltech-101 y Caltech-256 [26].

5.1.4 Redes Convolucionales Profundas (Google, 2014).

Ya para el año 2014 se propone una arquitectura de red neuronal con un gran alcance visual de tal forma que revolucione el procesamiento de las redes neuronales, esto haciendo un mejor uso de recursos computacionales dentro de la red, teniendo en cuenta el tamaño de la misma, manteniendo el gasto computacional constante. Este modelo es generado por Google y es llamado GoogLeNet con 22 capas de profundidad [27].

Esta red se basó en la incepción como una visión diferente a un red Convolucional, donde la arquitectura general se basa en bloques de construcción local que se repiten de forma espacial donde en cada capa se sugiere hallar las estadísticas y las correlaciones, y donde las agrupaciones forman las unidades de la capa siguiente y están conectados a las unidades de la capa anterior.

5.1.5 Aprendizaje residual para el reconocimiento de imágenes (Microsoft 2015)

El objetivo de este entrenamiento hecho por Microsoft consistía en conseguir que la red reconozca de forma consistente conjuntos de imágenes que nosotros los humanos conocemos con facilidad, como por ejemplo, la imagen de un perro (Raiko, 2012). Esto se parece mucho a la forma en que un niño aprende qué es un perro, mediante la observación de los detalles de la forma de la cabeza, el comportamiento y otras características en animales peludos y que ladran, conocidos por las personas como perros.

De acuerdo a los trabajos realizados previamente, se encuentran las representaciones residuales. En este caso, el reconocimiento de imágenes, VLAD es una representación que codifica por los vectores residuales con respecto a un diccionario, y mediante un vector Fisher se puede formular como una versión probabilística de VLAD (Jegou, 2012). Ambos son representaciones superficiales de gran alcance para la recuperación de imágenes y clasificación. Acá también se hace uso del Multigrid es una alternativa jerárquica que se basa en variables que representan los vectores residuales entre dos escalas (Srivastava, 2015).

Experimentalmente, se evaluó el método en la clasificación IMAGEnet 2012. Se evaluaron las redes residuales con una profundidad de hasta 152 capas. La profundidad de reconocimiento visual es tal que obtuvo los primeros lugares en las tareas de detección IMAGEnet [29].

5.2 Bases de datos de dígitos.

5.2.1 MNIST base de datos de dígitos manuscritos.

MNIST se constituyó como una base de datos para entrenamiento de máquina (como por ejemplo, redes neuronales) para el reconocimiento de dígitos escritos en caligrafía (LeCun et al., 1998) la cual es un modelo modificado de la base de datos original NIST cuyas siglas significan “National Institute of Standard and Technology“. Se ofrece una pequeña comparación con algunas bases de datos similares como el TIDigit que une la base de datos similar creada por Texas Instrument. El link donde se puede encontrar la base de datos MNIST es el siguiente: http://Yann.lecun.com/exdb/MNIST/ [30].

Su objetivo es ayudar a estudiantes e investigadores a ahorrar tiempo en pre-procesamientos de textos y formateo de textos al momento de llevar a cabo entrenamiento de máquina o reconocimiento de patrones. También se le compara con la base de datos TIMIT creada por el MIT Massachusetts Institute of Technology la cual tiene como objetivo ayudar en el entrenamiento máquina para el reconocimiento de voz. En su página web, actualmente se encuentran 68 estudios diversos realizados al MNIST para analizar su efectividad, organizados en 6 categorías: clasificadores lineales, vecino k-esimo más cercano, decisión “stump”, clasificadores no lineales, soporte de máquinas vectoriales (SVM) y redes neuronales sin Convolución.

5.2.2 Red Neuronal Recurrente para la generación de imágenes con MNIST.

Con el pasar del tiempo el amplio campo de aplicaciones de redes inteligentes ha despertado el desarrollo tecnológico de estas orientado a distintas áreas de conocimiento, bien sea reconocimiento de objetos, de voz, traducciones,

aplicaciones comerciales, generación de imágenes. Por esta razón, en los últimos años el desarrollo de ordenadores potentes ha aumentado la efectividad de la inteligencia artificial. Sin duda alguna, se esperan resultados prometedores a futuro.

Particularmente el equipo de google para el 2015 (Google DeepMind) combinó un nuevo mecanismo de redes que imita la atención espacial del ojo humano, con una secuencia que varía dentro del marco de auto-codificación que permite la construcción iterativa de imágenes complejas.

Así mismo, en este estudio se demostró la capacidad de generar imágenes de gran realismo tales como fotografías de números de casa, así como la mejora en los resultados para la generación binarizada de MNIST más conocida. Además, establece que la atención diferenciable del mecanismo de dos dimensiones incorporado en el sorteo es beneficiosa no sólo para la generación de imágenes, sino también para la clasificación de imágenes. A continuación se muestra mediante la Fig.1, dígitos generados por una red MNIST, siendo básicamente lo que se muestra a lo largo de esta investigación, imágenes generadas por distintas redes neuronales [31].

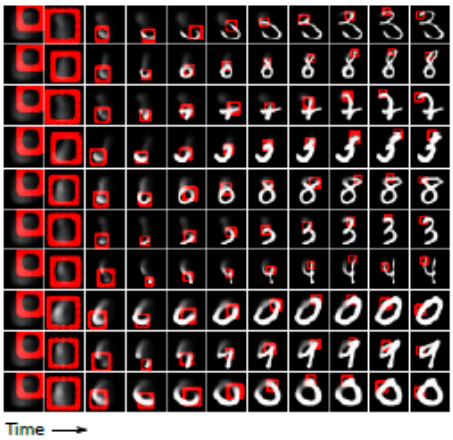

Ilustración 11. Dibujo de generación de dígitos mediante una red entrenada MNIST. Cada fila muestra etapas sucesivas en la generación de un solo dígito. Las líneas que componen las cifras

parecen ser "Dibujadas" por la red. El rectángulo rojo delimita la zona de la red en cada paso de tiempo, con la precisión focal indicada por el ancho del borde del rectángulo.

La estructura básica de una red de dibujo es similar a la de otras variaciones auto-codificadoras, puesto que una red codificadora determina una distribución códigos latentes que capturan información saliente sobre datos de entrada; por su parte, una red decodificadora recibe muestras de la distribución de código y los utiliza para condicionar su propia distribución sobre las imágenes (Larochelle, 2011).

Como conclusión se obtuvo que al desarrollar aún más este tipo de estudios se podrán generar imágenes complejas pudiendo ser utilizado en aplicaciones futuras.

5.3 Redes de Neuronas Recurrentes.

5.3.1 Almacenamiento a corto y largo plazo con sistemas Recurrentes.

Una de las tareas fundamentales era aprender a almacenar por largos periodos de tiempo por lo cual las memorias a corto plazo puede tomar mucho tiempo. El método LSTM se introduce como una mejora al análisis hecho por Hochreiter’s en el 91, empleando el gradiente de manera eficiente. Al compararlo con diversos métodos similares se han obtenido mejores resultados y se han podido resolver problemas complejos que no habían sido resueltos con anterioridad. Esto se logra disminuyendo la propagación del error [32].

En los métodos tradicionales, Back Propagation Through Time (BPTT) o Real-time Recurrating Learning (RTRL), la propagación en sentido inverso de la señal de error tiende a dos resultados posibles: estallar o desaparecer. Si estalla se puede deber a que los pesos oscilaban, mientras que para que desaparezcan, se pueden

necesitar cantidades de tiempo demasiado grandes de las que no se dispone o sencillamente no funciona.

El novedoso método LSTM corrige éstos problemas sin disminuir las capacidades de la memoria a corto plazo y sin afectar la trayectoria de la señal de error de las memorias a largo plazo.

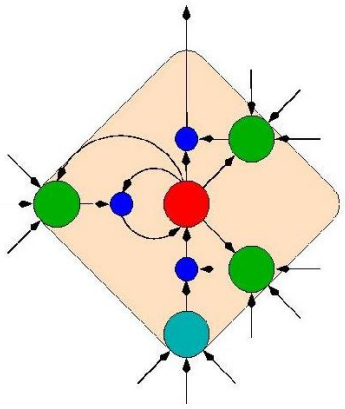

Ilustración 12. Arquitectura general de células LSTM Se muestra en rojo las unidades lineales, en verde las puertas sigmoideas y en azul las operaciones de multiplicación.

Aquí se hace referencia a varios experimentos realizados con Redes de Neuronas Recurrentes cada uno de los experimentos realizados paso a paso. Es recomendable referirse a cada uno de ellos para ya que cada uno tiene como objetivo un área diversa de estudio.

Experimento 1: Gramática de Reber embebida.

Experimento 2: Secuencias con y sin ruido de símbolos.

Experimento 3: Secuencia y ruido en la misma entrada con retrasos de tiempo prolongado.

Experimento 4: Problema de la suma.

Experimento 5: Problema de la multiplicación.

Experimento 6: Orden temporal.

Donde a partir de los resultados de cada uno se concluye que cada celda de memoria garantiza un flujo de error constante dentro de su entrenamiento. [33] Ésta es la base para poder afrontar el problema central que por el momento no se habían podido resolver de manera eficiente; sobrepasar retrasos de tiempo prolongados. Dos puertas de enlace, una de entrada y una de salida, y la entrada del error constante en el momento correcto[32].

5.3.2 Reconocimiento de voz con Redes Recurrentes.

Uno de los retos más complejos y apasionantes a los que se enfrenta el mundo científico es sin duda el de intentar imitar el comportamiento del ser humano por medio de aparatos y sistemas avanzados. Históricamente, se han investigado intensamente los fundamentos y aplicaciones de las redes neuronales recurrentes y se ha invertido un gran esfuerzo en el área del reconocimiento automático del habla (Mohamed, 2012).

Este trabajo de investigación estudiaba la combinación de múltiples niveles de representación que han demostrado ser eficaces en redes profundas proporcionando un alcance flexible que refuerza el potencial de las Redes Neuronales Recurrentes Profundas (RNNs) aplicadas al reconocimiento de voz [34].

Por otra parte, mediante el método de Redes Neuronales Recurrentes Bidireccionales (***BRNNs***) se ejecuta el procesamiento de los datos en ambas direcciones con dos capas ocultas separadas, como se muestra en la siguiente figura.

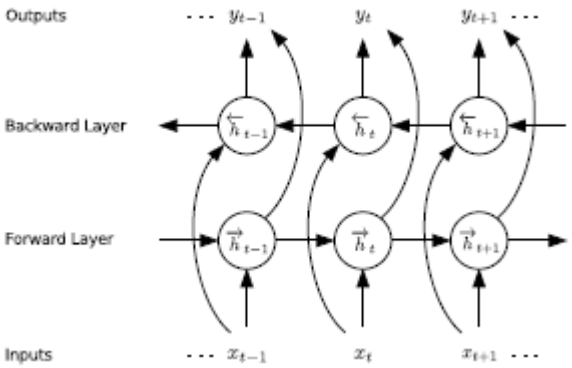

Ilustración 13. RNN Bidireccional.

Ahora bien, el estudio explica claramente que realizar una combinación entre BRNNs y LSTM proporciona al LSTM bidireccional acceso a contextos de largo alcance en ambas direcciones de entrada (Schuster, 1997). Por esta razón, señalan que el éxito del híbrido Hidden Markov Models (***HMM***) en sistemas de redes neurales reside en el uso de arquitecturas de profundidad, pues son capaces de construir representaciones de nivel progresivamente más altas mediante datos acústicos (Bourland, 1994).

De acuerdo a este enfoque, si el LSTM se utiliza para las capas ocultas se obtiene un LSTM bidireccional profundo, siendo este el punto principal de este estudio, pues representa la primera vez que se ha utilizado este método (LSTM profundo) al reconocimiento de voz, produciendo una gran mejora sobre el método LSTM de una sola capa [34].

6. DESARROLLO DEL PROYECTO.

6.1. Red Convolucional para la clasificación de dígitos manuscritos.

Para llevar a cabo el desarrollo del proyecto se hizo uso de las siguientes herramientas:

- Python 2.7
- Anaconda 4.0.0
- TensorFlow

Este objetivo consistió replicar el modelo de Red Convolucional propuesto por Yann Lecun y poder reproducir sus resultados, el objetivo de este proceso era poder reconocer los caracteres manuscritos ingresados, en este caso a través de la base de datos de MINIST constituida por 55000 imágenes de tamaño 28x28 de entrenamiento (MNIST_data/train-images), en 10 etiquetas o clases que corresponden a los dígitos naturales (0-9) (MNIST_data/train-labels).

Es interesante reconocer que la Red Convolucional es un sistema análogo al sistema visual animal adaptado computacionalmente, por lo cual este tipo de red hace un reconocimiento de forma local de cada mapa de características de la imagen de entrada.

Ya para la construcción de la Red Convolucional se implementaron 2 capas convolucionales y dos capas totalmente conectadas (Véase capítulo 3. Núm. 3.4.3.Capas de una Red Neuronal Convolucional).

El objetivo de esta red era poder reconocer y clasificar los datos que se suministraron como entrada, en este caso el conjunto de datos de MNIST en cada una de las etiquetas disponibles.

Hablando matemáticamente es necesario definir este proceso como una función

$$F(x) \longrightarrow y$$

Que transforma una entrada x(en este caso imágenes de dígitos manuscritos) del espacio de imágenes de 28x28 en una salida y del espacio de etiquetas del 0-9 es decir:

$$x \in \mathbb{R}^{28x28} \longrightarrow y \in \mathbb{R}^{10}$$

El objetivo de la Red Convolucional era determinar esa función f que permitiera hacer dicha asignación de forma eficiente.

Sabiendo que una Red Convolucional aplica una función a la salida de la capa anterior y esta a su vez produce una salida que será procesada por la capa siguiente, nos estamos refiriendo a una función compuesta. Es decir, una Red Convolucional es una función compuesta.

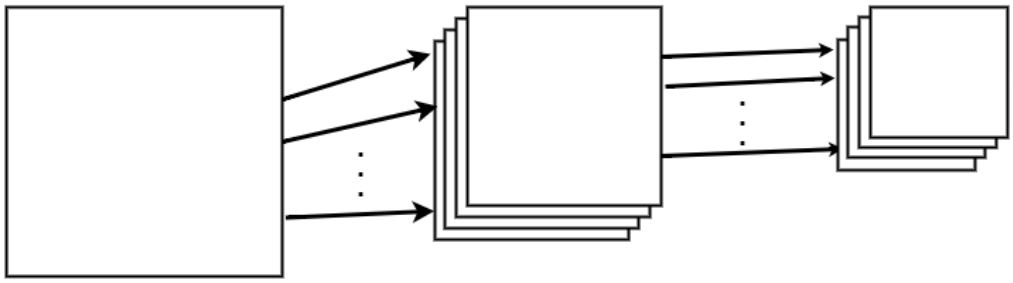

Ilustración 14. Procesamiento general de una red convolucional.

Como se mencionaba anteriormente, nuestro modelo de Red Neuronal Convolucional se compone de 2 capas de convolución y 2 capas totalmente conectadas. Por lo cual la función f se ve de la siguiente manera:

$$F : f_{FC4}(f_{FC3}(f_{CONV2}(f_{CONV1}(x))))$$

Donde el subíndice CONV se refiere a las capas convolucionales y el subíndice FC a las capas totalmente conectadas, donde:

$$f_{CONV1}(x) = P\big(R(w * x + b)\big)$$

$$x \in \mathbb{R}^{28\times 28}$$

$$f(x) \in \mathbb{R}^{14\times 14\times 32}$$

$\boldsymbol{f_{CONV1}}$ Corresponde a la primera capa Convolucional donde se establece un tensor de pesos $\boldsymbol{w}$ de **5x5x1x32** es decir, la primera convolución computa 32 características para cada sector de **5x5** (feature map), donde a cada entrada se le asigna un $\boldsymbol{b}$ (bias) que es de tamaño 32, los cuales se inicializan desde 0.1 bajo el movimiento de un pixel por cada ventana de **5x5**, es decir 25 neuronas de la capa de entrada.

Nótese que $\boldsymbol{f_{CONV1}}$ aplica dos tipos de funciones $\boldsymbol{P}$ y $\boldsymbol{R}$ a una función Convolucional (*) entre pesos $\boldsymbol{w}$ y a los valores $\boldsymbol{x}$ de entrada más un bias $\boldsymbol{b}$.

$\boldsymbol{P}$ Hace referencia a la función Max Pooling que disminuye la resolución del volumen espacial de cada salida de la red, seleccionando el máximo valor en cada rebanada que no se solapa, **R** es la función Relu o función de rectificación (Véase capítulo 3. Núm. 3.6.4. ¿Qué es Relu?) Que elimina la linealidad de la salida, por lo cual cada resultado o feature map se reduce a la mitad de tamaño de entrada, siendo la salida de $\boldsymbol{f_{CONV1}}$ la entrada de la segunda capa convolucional:

$$f_{CONV2}(x) = P\big(R(w * x + b)\big)$$

$$x \in \mathbb{R}^{14\times 14\times 32}$$

$$f(x) \in \mathbb{R}^{7\times 7\times 64}$$

$\boldsymbol{f_{CONV2}}$ Corresponde a la 2 capa Convolucional que recibe una matriz de pesos $\boldsymbol{w}$ de **5x5x32x64** es decir que la convolución 2 computó 64 características para cada sector **5x5** (feature map) de cada salida de $\boldsymbol{f_{CONV1}}$. Aplicando una función Relu a la función de convolución entre cada $\boldsymbol{x}$ de entrada y cada peso $\boldsymbol{w}$ más un bias $\boldsymbol{b}$ que es un vector de 64 posiciones, a esta salida se le aplica una función de Max Pooling que disminuye la resolución de la imagen inicial a la mitad, la salida de $\boldsymbol{f_{CONV2}}$ es la entrada de la siguiente capa, es decir la capa totalmente conectada:

$$f_{FC3}(x) = R(w \cdot x + b)$$

$$x \in \mathbb{R}^{7\times7\times64}$$

$$f(x) \in \mathbb{R}^{1024}$$

$\boldsymbol{f_{FC3}}$ corresponde a la 3 capa del sistema, en este caso una capa totalmente conectada, esta capa recibe todas las salidas de la capa anterior es decir $\boldsymbol{f_{CONV2}}$ asignando un vector de pesos $\boldsymbol{w}$ de **7x7x64** y un bias $\boldsymbol{b}$ de 1024 posiciones, es decir en esta capa se permite el procesamiento de la imagen de forma completa, pero al no tratarse una capa convolucional no se aplica Max Pooling, esta capa solo toma los datos entrantes y los trata linealmente, posterior a esto se aplica Relu para modificar dicha linealidad, la salida de esta capa es la entrada de la última capa totalmente conectada que hace la asignación final a alguna de las 10 clases o etiquetas finales:

$$f_{FC4}(x) = S(w \cdot x + b)$$

$$x \in \mathbb{R}^{1024}$$

$$f(x) \in \mathbb{R}^{10}$$

f_{c4} Corresponde a la 2 capa totalmente conectada y a la última capa del modelo, esta capa recibe las 1024 características extraídas de la capa anterior, aplicando Softmax $\boldsymbol{S}$ a la función lineal entre cada uno de los $\boldsymbol{x}$ de entrada por el vector de pesos $\boldsymbol{w}$ que corresponde **1024x10** y la suma de un bias $\boldsymbol{b}$ de tamaño 10, la función Softmax (Véase capítulo 3. Núm. 3.5.1 ¿Qué es Softmax?) Envía cada salida a una de las 10 clases o etiquetas asignando una probabilidad numérica de pertenecer a cada de una de las etiquetas o clases, a su vez normalizando los datos para que estos no sumen más de uno. Se determina cómo la etiqueta real, la salida con el valor más alto entre todas las probabilidades, retornando un $\boldsymbol{y}$ en un espacio en $\mathbb{R}^{\mathbf{10}}$, pero esta es una probabilidad generada o resultado de un procesamiento computacional, por lo cual el modelo debe ser evaluado con respecto a las etiquetas reales $\boldsymbol{y}$ que corresponden a los labels de los datos de MNIST.

Este sistema de capas compuesto por funciones se repite por cada imagen de entrada, es decir el proceso anteriormente enunciado corresponde al procesamiento de una sola imagen, pero nuestra colección de datos corresponde a 55000 imágenes, por lo cual hay que repetir este proceso de forma iterativa, haciendo a su vez una estimación numérica de la media de la suma de las probabilidades generadas por el sistema por las probabilidades reales haciendo uso del cross entropy (Véase capítulo 3. Núm. 3.6.4. ¿Qué es el Cross Entropy?), lo que busca esta función es reducir la métrica de qué tan malo es el modelo intentando elegir los $\boldsymbol{w}$ y lo $\boldsymbol{b}$ más apropiados para el sistema. Esto solo se logra tras el entrenamiento, enviando este resultado a una función de optimización, en este caso una función que minimiza el Cross Entropy y entrena el modelo de

forma descendente por cada iteración en el sistema bajo el uso de la AdamOptimizer.

Este modelo de evaluación se imprime dentro de un rango de 5000 que es un valor variante, a un paso de 100, seleccionando un conjunto de ejemplos de 50 imágenes en cada paso.

En este modelo se evalúa con respecto a una función de desempeño que determina cuando la probabilidad del sistema y la probabilidad real son iguales, retornando un valor booleano o descriptivo y una estimación numérica derivable que determina la media entre los índices del tensor que retorna el valor máximo resultado de la función de desempeño.

6.2. Modelo Recurrente para procesamiento de secuencias con sumas y restas.

1. Creación de secuencias Para el desarrollo de la Red Neuronal Recurrente cuyo fin es el procesamiento de secuencias de dígitos fue necesaria la creación de una base de datos de símbolos (+,-).

Para la creación de esta base de datos se llevó a cabo el siguiente proceso:

1. Recolección de 50 caracteres escritos a mano por distintas personas.

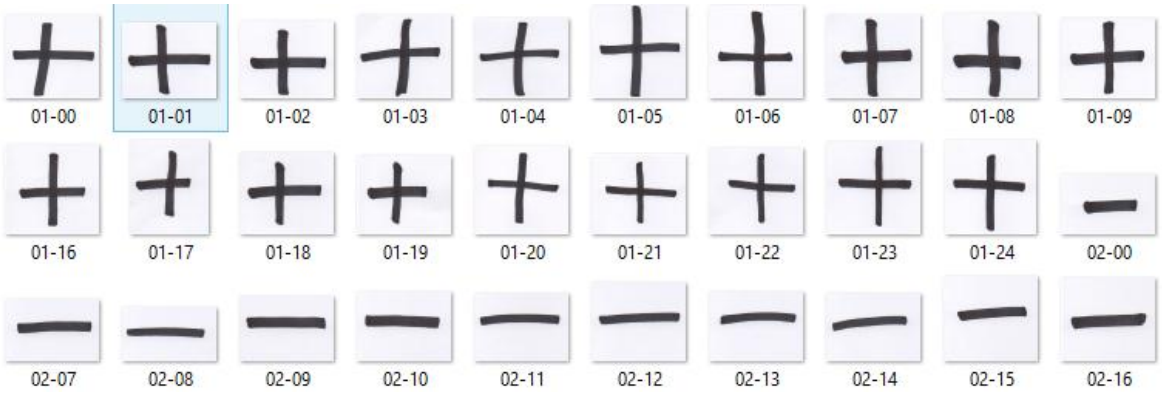

Ilustración 15. Símbolos manuscritos recolectados.

2. Se redimensionaron los símbolos obtenidos de la siguiente manera:

2.1. Se modificó la el tamaño de cada imagen en formato de 28x28.

2.2. Se transformaron bajo un 1 solo canal de color para que estas mismas quedaran en blanco y negro.

2.3. Cada una de las imágenes se almacenó como un arreglo Numpy de 28X28.

2.4. Estas mismas se almacenaron como posiciones de un arreglo de mayor tamaño: 50x28x28 es decir 50 posiciones de imágenes de 28x28, guardadas dentro de un archivo llamado DATA_TRAIN_IMAGES.npy.

2.5. Se crearon las etiquetas correspondientes es decir, un arreglo Numpy para las imágenes de entrenamiento de 50x2 almacenadas dentro de un archivo llamada DATA_TRAIN_LABELS.npy, ya que cada imagen pertenece a una única clase ya sea un más o un menos, asignando un 1 a la posición que corresponde de la siguiente manera:

Posición 0: +.

Posición 1: -.

Ejemplo: El arreglo (0,1).

2.6. Visualización de imágenes:

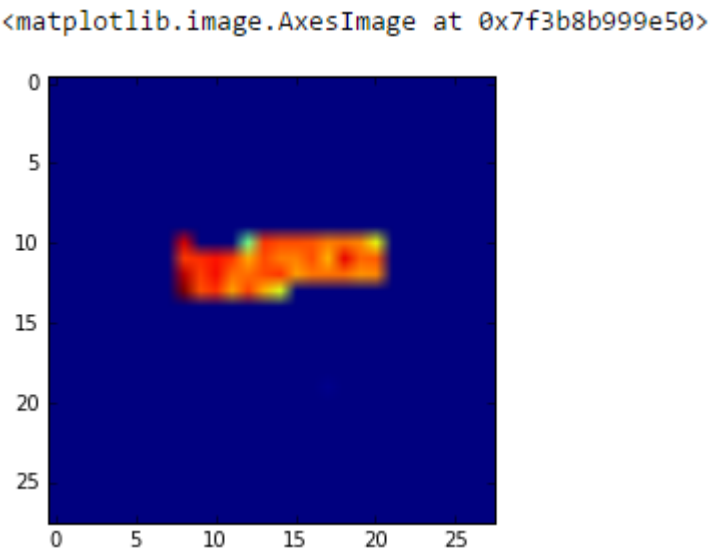

Ilustración 16. Visualización de símbolo menos de la base de datos DATA_TRAIN_IMAGES.npy

2.7. Creación de la secuencia

Para crear las secuencias se debió crear una función que toma un elemento de una posición de forma aleatoria entre los datos de entrada de dígitos (mnist.train.images), un elemento de una posición aleatoria de la base de datos de símbolos (DATA_TRAIN_IMAGES.npy) y otro elemento de una posición aleatoria de la base de datos de dígitos (mnist.train.images), disponiendo cada uno en la posición adecuada dentro de la cadena de 3 posiciones donde se generaran k secuencias que se almacenan en un tensor de tamaño kx3x28x28x1, donde k es el número total de secuencias, 3 es el número de caracteres de cada secuencia, 28x28 es el tamaño de cada carácter de la secuencia y 1 es el canal de color para cada carácter. La secuencia es construida de la siguiente manera:

[Posición 0 Posición 1 Posición 2]

[Digito, Símbolo, Digito]

Esta función además reconoce las etiquetas de los dígitos y de los símbolos es decir, dentro de esta función se definirá la operación. Cabe aclarar que la red recurrente hace operaciones a partir de imágenes por lo cual en ningún momento hace la manipulación de símbolos, la operación se define a través de la lectura de los labels de los símbolos contenidos en DATA_TRAIN_LABELS.npy. En este punto se identifica si la operación es una suma o una resta, identificando el valor máximo del arreglo, que en este caso solo puede ser 1 trayendo su posición dentro del arreglo.

Ejemplo:

Tenemos el vector (1,0) su máximo valor es 1 y se encuentra en la posición 0 por lo cual corresponde a una suma, si el vector es (0,1) su máximo valor se encuentra en la posición 1 lo que significa que se está trabajando una resta. Visualicemos un ejemplo: esta corresponde a una secuencia creada a partir de aplicada de la función crear secuencia.

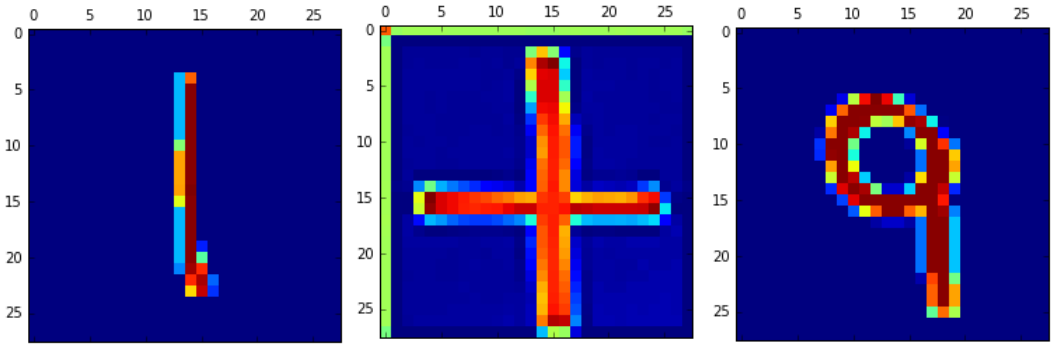

Ilustración 17. Representación de una secuenca generada por el sistema.

Ahora veamos otro ejemplo:

Se describe una secuencia obtenida de forma vectorial:

(0,1,0,0,0,0,0,0,0,0) (1,0) (0,0,0,0,0,0,0,0,0,1)

El primer elemento corresponde a un 1 ya que posee un 1 en su segunda posición:

(0,1,0,0,0,0,0,0,0,0)

El segundo elemento corresponde a un más ya que en su primera posición posee un 1:

(1,0)

El tercer elemento corresponde a un 9 ya que posee un 1 en su última posición

(0,0,0,0,0,0,0,0,0,1)

Lo cual producirá una salida:

(10)

Podemos evidenciar que se conoce el resultado previamente por lo cual es un modelo entrenado, en este caso 1 + 9 = 10.

Es importante aclarar que este proceso solo se explica para una sola secuencia, y para la arquitectura en general se crearan $\boldsymbol{K}$ secuencias.

Descripción de la arquitectura recurrente:

Nuestro modelo se forma general se visualiza de la siguiente manera:

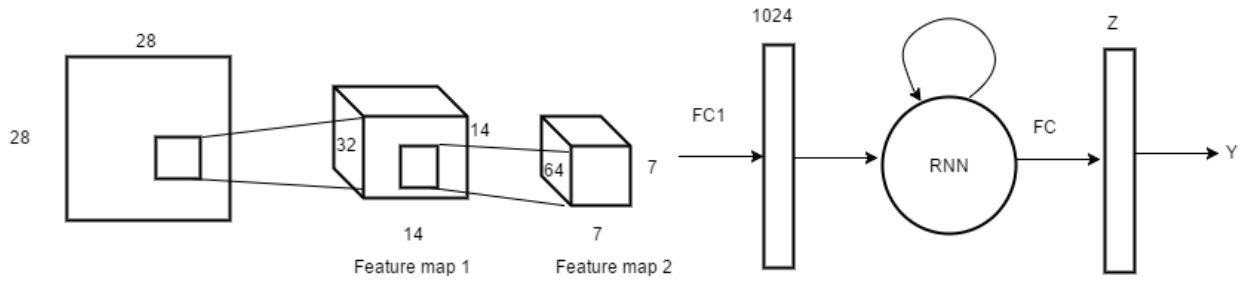

Ilustración 18. Representación gráfica de la arquitectura implementada a partir de un modelo convolucional y recurrente.

Esta arquitectura de divide o se implementa bajo 3 módulos

1. Módulo 1: Módulo de Convolución, en este primer momento de la arquitectura se incorpora la Red Neuronal Convolucional, pero esta se corta en la última capa es decir la $f_{FC3}(x)$ ya que no es necesario que esta retorne una clase para un dígito, es necesario que la salida de esta sea conectada a la red recurrente y tratada por la mismas es decir la entrada de la red recurrente sera de 1024, recordemos la definición de $f_{FC3}(x)$:

$$f_{FC3}(x) = R(w \cdot x + b)$$

$$x \in \mathbb{R}^{7\times 7\times 64}$$

$$f(x) \in \mathbb{R}^{1024}$$

Es decir, la red recurrente recibirá un vector de 1024 características.

2. Módulo 2: implementación de un modelo Recurrente:

Definamos una función $\boldsymbol{Re(x)}$ una función Recurrente donde $\boldsymbol{x} = (\boldsymbol{x_t}, \boldsymbol{h_{t-1}})$ es decir $\boldsymbol{x}$ es una dupla que recibe un $\boldsymbol{x}$ en un tiempo $\boldsymbol{t}$ y un $\boldsymbol{h}$ en un tiempo $\boldsymbol{t-1}$, en pocas palabras la entrada de una Red Neuronal Recurrente siempre es una dupla compuesta por un estado actual y un estado anterior.

En este caso la arquitectura Recurrente recibirá como entrada un:

$$f(x) \in \mathbb{R}^{1024}$$

Resultado de las transformaciones efectuadas por la Red Convolucional. Pero no se trata de un solo carácter se trata de una secuencia, la cual debe ser construida a partir de la red Recurrente.

Es importante identificar que la Red Neuronal Recurrente aplica una función Recurrente, así como para la implementación de la CNN se aplicaron funciones de Convolución, la ventaja del uso de Tensorflow radica en que estas funciones ya están implementadas por lo cual solo se deben aplicar a los datos de entrada, acá como en el modelo de CNN siempre se buscara un $\boldsymbol{w}$ y $\boldsymbol{b}$ tales que produzcan una salida casi exacta del modelo, por lo cual siempre $\boldsymbol{w}$ y $\boldsymbol{b}$ serán nuestras incógnitas del sistema.

Ahora, sabiendo que la Red Recurrente procesa un dupla de un estado actual y un estado anterior se puede decir que se recibe como entrada la siguiente dupla

$$(x_t \in \mathbb{R}^{1024}, h_{t-1} \in \mathbb{R}^{512})$$

Donde x_t es la salida de la Red Convolucional y h_{t-1} es un estado ya procesado por la red Recurrente, es decir en un espacio 512, un aspecto importante de la red Recurrente es que almacena sus salidas como marcaciones en memoria, cada entrada procesada se fundamenta en que la red recurrente solo aprende las características más relevantes reduciendo una entrada a la 6 sexta

parte de su tamaño inicial , almacenándola como una característica en memoria en este caso de tamaño 512.

De forma general la función Recurrente se expresa de la siguiente manera:

$$Re(x) = ((w_a \cdot x_t + b_a) + (w_b \cdot h_{t-1} + b_b))$$

Retornando un estado final:

$h_t \in \mathbb{R}^{512}$ (512 siendo una cantidad ajustable)

Veamos mas a profundidad un modelo de de funcion Recurrente, el modelo Recurrente es dinamico, es decir varia con respecto al tiempo entonces:

$t = 0$	$h_0 = \emptyset^{\rightarrow}$
$t = 1$	$h_1 = (w_a \cdot x_1 + b_a) + (w_b \cdot h_0 + b_b)$
$t = 2$	$h_2 = (w_a \cdot x_2 + b_a) + (w_b \cdot h_1 + b_b)$
$t = 3$	$h_3 = (w_a \cdot x_3 + b_a) + (w_b \cdot h_2 + b_b)$

Tabla 1. Proceso detallado del proceso de alimentación de la red recurrente.

Donde se evidencia la variacion entre los parámetros que entran a la Red Recurrente según la actualizacion de estados.

Este tipo de procesamiento no se implementa de forma manual ya que Tensorflow implementa la funcion Recurrente, haciendo de forma implicita cada uno de estos pasos , recibiendo tan solo un conjunto de vectores.

Simplemente la función recurrente almacena una salida como un estado en memoria y recordará los estados anteriores, lo cual es necesario para el procesamiento de las secuencias, ya que al procesar este primer dígito el mismo se almacena y esperara el procesamiento del segundo caracter. La red recurrente lo recordará y lo almacenará como un nuevo estado en memoria.

La red recurrente almacenara la transformacion de una primera entrada de la red convoluvional, y posteriormente a esto recibira un simbolso , este simbolo es almacenado y convertido a través de la función recurrente , posteriormente se recibira una segunda entrada de la red convoluvional, aca la red convoluvional no hara tratmiento del simbolo este simplemente se reconocera por el label asignado.

Al final se transformará de forma conjunta con la última entrada, la red recuerda las salidas anteriores por lo cual la salida final sera la construccion de la secuencia. De esa forma obtenemos un conjunto de datos almacenados como estados de memoria construyendo así dichas secuencias de forma volátil en un índice de memoria.

3. Modulo 3: Módulo de salida, etapa final.

Esta última etapa lee las salidas de la red recurrente y se recodifica a un espacio de memoria de 512 , esto con el fin de hacer la asignacion de un unico numero resultado de una operación:

$$h_3 \in \mathbb{R}^{512} \rightarrow z \in \mathbb{R}^{512}$$

$$f_s(x) = R(w \cdot z + b)$$

$$z \in \mathbb{R}^{512}$$

$$f(x) \in \mathbb{R}^1$$

Donde $\boldsymbol{w}$ es de tamaño 512x1 y $\boldsymbol{b}$ es un vector de tamaño 1, acá simplemente se aplica una funcion lineal $\boldsymbol{f_s}$ a la salida de la Red Recurrente y a esta misma un Relu, el resultado de la aplicación de dichas funciones retorna un único dígito por

lo cual se habla de un modelo de regresión ya que retorna un valor resultado de una aplicación lineal.

6.3. Evaluación de la arquitectura.

La arquitectura propuesta se evaluara por modulos:

1. Modulo 1.

Implementanción de la Red Convolucional.

Este proceso consistió en un problema de clasificación que pretende a partir de un conjunto de datos (imágenes de dígitos manuscritos,MNIST) encontrar la etiqueta correcta, es decir clasificar los dígitos de entrada en alguna de las 10 etiquetas existenes, en este caso los números naturales del 0-9.

Donde se obtuvo que sistema Convolucional ha aprendido de forma muy rápida.La precisión final después de llevar a cabo el proceso es de aproximadamente 99,2%, donde se evalúa el modelo ingresando digitos de test, por que implica que este modelo ha aprendido de forma muy acertada.

```
step 0, training accuracy 0.1
step 100, training accuracy 0.86
step 200, training accuracy 0.82
step 300, training accuracy 0.98
step 400, training accuracy 0.96
step 500, training accuracy 0.92
step 600, training accuracy 0.94
step 700, training accuracy 0.98
step 800, training accuracy 0.98
step 900, training accuracy 1
step 1000, training accuracy 0.96
step 1100, training accuracy 1
step 1200, training accuracy 0.94
step 1300, training accuracy 0.98
step 1400, training accuracy 0.94
step 1500, training accuracy 0.96
step 1600, training accuracy 1
step 1700, training accuracy 0.98
step 1800, training accuracy 0.98
step 1900, training accuracy 0.98
step 2000, training accuracy 0.98
step 2100, training accuracy 0.98
step 2200, training accuracy 0.98
step 2300, training accuracy 1
```

Ilustración 19. Resultados obtenidos tras cada iteración en la Red Convolucional.

Es posible observar que el modelo es casi exacto ya que esta obteniendo un resultaro no menor al 90% en el acierto, bajo una probailidad para el manejo de la tasa de perdida de 0,5.

Podemos visualizar algunos de los ejemplos resultantes del reconcimiento de la base de datos de MNIS:

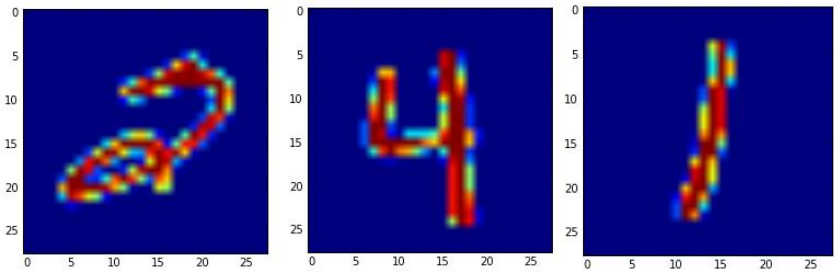

Ilustración 20. Ejemplo de imágenes de MNIST.

2. Modulo 2 y 3.

Este experimiento consistía en el procesamiento de secuencias, donde se realizaron 80.000 iteraciones en el proceso de optimización. Aquí se obtuvo un error inicial de 220.572205 lo cual habla de un margen de error bastante elevado el cual decrece de forma simultanea con cada iteracion ejecutada, mientras los pasos de la iteración transcurren. Es importante aclarar que pese a que el error es algo elevado y genera una salida con alto ruido se usa una función de redondeo que aproxima el digito de salida al dígito mas cercano.

Ahora observe:

Es posible ver cómo el error disminuye de forma lenta pero continua, claro está de forma global:

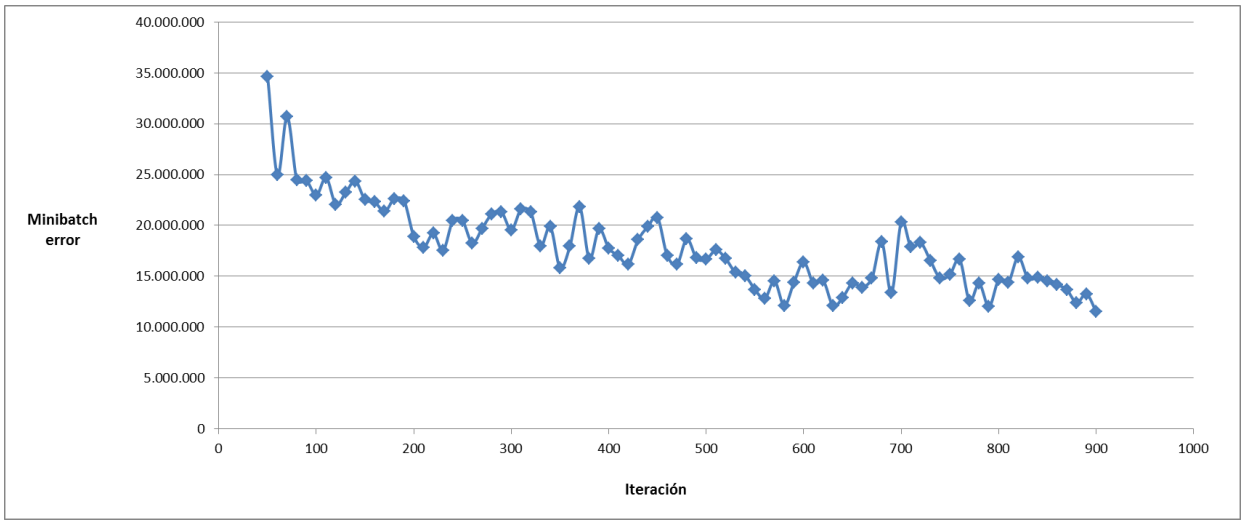

Ilustración 21. Visualización grafica de los resultados obtenidos tras la implementación de la arquitectura de la red recurrente, donde se evidencia la tendencia del reducción del error.

Y además se observa con el pasar de las iteraciones la presición o acurracy aumenta:

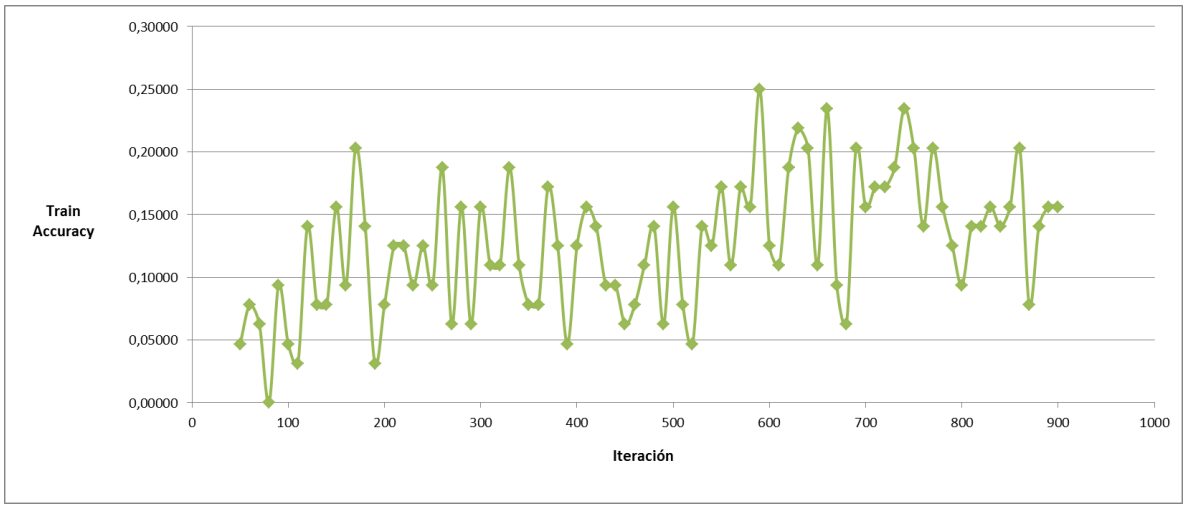

Ilustración 22. Representación gráfica de la tendencia del accuracy precisión obtenido en cada interacción, es posible observar una ligera tendencia hacia el incremento.

Es posible apreciar sobre los resultados que la arquitectura propuesta puede aprender a sumar dígitos manuscritos de una sola cifra siguiendo una estrategia secuencial, con error menor a 1 (una unidad), y con presión mayor al 95%.

7. CONCLUSIONES

1. Se ha replicado de forma satisfactoria el modelo de Red Convolucional para el procesamiento de dígitos manuscritos, gracias al uso de herramientas provistas por tensor flow que ha llegado a aplicar funciones de forma eficiente.

Se ha logrado optimizar el tiempo de desarrollo e implementación de mecasnismo que fueran capaces de leer y entender la base de datos de MNIST ya que estas funciones están integradas dentro de Tensorflow.

Las Redes de Neuronas convolucionales son modelos eficientes cuando al tratamiento de imágenes se trabaja ya que de forma secuencial se obtienen las características más relevantes de cada imagen, este modelo ha obtenido un 99,2% de acierto o coincidencias al final del modelo.

2. La arquitectura implementada permite conectar la red Recurrente y la Red Convolucional de forma efectiva. El concepto de poder recortar la Red Convolucional para preservar sólo las 3 primeras capas f_{FC3},f_{CONV2},f_{CONV1} y conectarla con la Red Recurrente funciona de forma eficiente, la facilidad que ofrece Tensorflow para este tipo de trabajos, reduce tiempos de desarrollo y líneas extensas de codigo.

Se puede considerar que los modelos de Redes Recurrentes son eficientes no sólo en el procesamiento de textos que es el campo donde más se involucran, son muy eficientes recordando estructuras de mayor complejidad.

3. Al evaluar la Arquitectura implementada se obtiene una presición mayor del 95%, que corresponde a una estimación muy buena, claramente este modelo

puede mejorarse, implementando nuevas funciones, agregando estados de incio y modificando el número de iteraciones.

Aunque este trabajo pretende hacer un primer acercamiento en la integración de arquitecturas en este caso Convolucional y Recurrente, y generar un salida casi exacta, no es un modelo final, este se puede conciderar como la base para muchos mas retos académicos, por lo cual de forma general se puede concluir que es posible hacer operaciones más complejas que no sólo involucren un dígito sino múltiples de estos, que no sólo sean sumados o restados si no que también sean multiplicados, divididos, sea posible derivarlos y construir operaciones mucho más complejas integrando las Redes Convolucionales y las Redes Recurrentes para el procesamiento de caracteres de tipo manuscrito.

8. BIBLIOGRAFIA

[1] Bataineh, M.; Marler, T.; Abdel-Male, K. and J. Arora. (April, 2016). Neural network for dynamic human motion prediction. Expert Systems With Applications, 48, 26-34.

[2] Lauer, F.; Ching S. and G. Bloch (June, 2007). *A trainable feature extractor for handwritten digit recognition*. Pattern Recognition. Volume 40, issue 6, pages 1816-1824.

[3] Sermanet, P.; Chintala, S. y Y. LeCun (2012). Convolutional Neural Networks Applied to House Numbers Digit Classification. New York University: The Courant Institute of Mathematical Sciences.

[4] LeCun Y. y C. Corinna (2012). The MNIST database of handwritten digits. Citado el 29 de enero de 2016, de The Courant Institute of Mathematical Sciences. Sitio web: http://yann.lecun.com/exdb/mnist/

[5] https://www.tensorflow.org/

[6] LeCun, Y.; Cortes, C.; and C.J. Burges. (2012). The MNIST database of handwritten digits. Site web: http://yann.lecun.com/exdb/mnist/

[7] Haykin, S. (1999). Neural Networks: A Comprehensive Foundation. Ontario, Canada: 2nd Edition.

[8] Laster, Madlon T. (2009). Teach the Way the Brain Learns: Curriculum Build Neuron Networks. North America: R&L Education.

[9] Artificial Neural Networks. S Lek, J.L. Giraudel, J.F. Guegan, Prof.
Sovak Lek, Dr. Jean Francois Guegan.

[10] Downey, A. (2012). Think Phyton: How to think like a computer scientist. Needham, Massachusetts: Green Tea Press.

[11] Pusiol, Pablo D. (2014). Redes Convulsionales en Comprensión de Escenas. Tipo de documento: Trabajo final de grado. Licenciado en ciencias de la computación. Universidad Nacional de Córdoba, Argentina.

[12] Chandra B. (1 January 2016). Fast learning in Deep Neural Networks. Neurocomputing, Volume 171, Pages 1205–1215. 10 August 2015, de http://www.sciencedirect.com/ Base de datos.

[13] De la Rosa Montero, Erick D. (2014). El aprendizaje profundo para la identificación de sistemas no lineales. Tipo de documento: Tesis final de grado. Maestro en ciencias. Centro de Investigación y de estudios avanzados del instituto politécnico nacional. México, D.F.

[14] Visin, F.; Kastnerm, K.; Cho, K.; Matteucci, M.; Courville, A. y Y. Bengio (July, 2015). ReNet A recurrent Neural Network based alternative to convolutional networks, Cs CV. 23 Jul 2015, de http://arxiv.org/abs/1505.00393v3

[15] http://Yannn.lecun.com/

[16] Sermanet, P.; Chintala, S. y Y. LeCun (2012). Convultional Neural Networks Applied to House Numbers Digit Classification. Cs CV, 18 Apr 2012, de http://arxiv.org/pdf/1204.3968.pdf

[17] LeCun, Y. y S. Pierre (2011). Traffic Sign Recognition with Multi-Scale Convolutional Networks. The Courant Institute of Mathematical Sciences. Sitio web: http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf

[18] Barat, C. y D. Christophe (January 2016). String representations and distances in deep Convolutional Neural Networks for image classification. Pattern Recognition. Volume 54, pages 104-115

[19] Fukushima, K. (January 2013). Artificial visions by multi-layered neural networks: Neocognitron and its advances. Neural Networks. Volume 37, pages 103-119.

[20] Mathieu, M; Hernaff, M y Y. LeCun (2014). Fast Trainig of Convolutional Networks Through FFTs. Cs CV, 6 Mar 2014, de http://yann.lecun.com/exdb/publis/pdf/mathieu-iclr-14.pdf

[21] Pérez-Carrasco, J. A; Serrano, C; Acha, B.; Serrano-Gotarredona, T y B. Linares Barranco (2011). Red neuronal convolucional rápida sin fotogramas para reconocimiento de dígitos. Universidad de Sevilla, España: ETSIT..

[22] Wu, David J. (2012) End-To-end- Text Recognition with convolutional neural networks. Tipe of document: Final Thesis. Department of computer science of Standford University.

[23] McCormick, C. (June 13, 2014). DEEP LEARNING TUTORIAL – SOFTMAX REGRESSION. Site web: https://chrisjmccormick.wordpress.com/2014/06/13/deep-learning-tutorial-softmax-regression/

[24] Torres, J. (Friday, 27th November 2015). Introducción Práctica al Deep Learning con tensorflow de google. Sitio web: http://www.jorditorres.org/introduccion-practica-al-deep-learning-con-tensorflow-de-google-parte-6/

[25] Krizhevsky, A.; Sutskever, I. and G. E. Hinton. (s/f). ImageNet Classification with Deep Convolutional Neural Networks. Sitio web: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[26] Zeiler, M. D. and R. Fergus. (2013). Visualizing and Understanding Convolutional Networks. Cs. Cv, 28 nov 2013, de http://arxiv.org/pdf/1311.2901.pdf

[27] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D; Vanhoucke, V.; A, Rabinovich. (2015). Going Deeper with Convolutions. Site web: http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

[28] Zuo, Z; Shuai, B.; Wang, G.; Liu,X.; Wang, X.; Wang, B. and Y. Chen (2015). Convolutional Recurrent Neural Networks: Learning Spatial Dependencies for Image Representation. NanYanng Technological University, Singapore.

[29] He, K.; Zhang X.; Ren, S. and J. Sun. (2015). Deep Residual Learning for Image Recognition. Cs. Cv, 10 Dec 2015, de http://arxiv.org/pdf/1512.03385.pdf

[30] Deng, L. (2015). The MNIST Database of Handwritten Digit Images for machine Learning Research. Site web: http://research.microsoft.com/pubs/204699/MNIST-SPM2012.pdf

[31] Gregor, K.; Danihelka, I.; Graves, A.; Rezende Jimenez, D. y D. Wierstra. (2015). DRAW: A Recurrent Neural Network For Image Generation. Cs. CV, 20 may 2015, de http://arxiv.org/pdf/1502.04623v2.pdf

[32] Hochreiter S. and J. Shmidhuber (1997). Long Short Term Memory. Neural Computation 9, pages 1735-1780. Massachusetts Institute of Technology.

[33] Cho, K.; Bahdanau, D.; Bougares, F. and Y. Bengio. (2014). Learning Phrase Representatios using RNN Encoder-Decoder for Statistical Machine Translation. Cs. CL, 3 Sep 2014, de http://arxiv.org/pdf/1406.1078.pdf

[34] Graves, A.; Mohamed, A. and G. Hinton. (2013). Speech Recognition With Deep Recurrent Neural Networks. Cs. NE, 22 Mar 2013, de http://arxiv.org/pdf/1303.5778.pdf)?

[35]http://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks

[36]http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

[37] Dettmers, T. (2015). Understanding in deep Learning. Site web: http://timdettmers.com/2015/03/26/convolution-deep-learning/

[38] G.E. Nasr; E.A. Badr and C. Joun. (2002). Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand. School of Engineering and Architecture Lebanese American University

[39] http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

[40] http://cs231n.github.io/neural-networks-1/

[42] LeCun, Y.; Guyon I.; (1989). Handwriten digit recognition: Applications of Neural Network Chips and Automatic Learning. Sitio web: http://yann.lecun.com/exdb/publis/pdf/lecun-89c.pdf

[43] Pascal Vincent; (2009). Machine Learning from linear regression to Neural Networks. Sitio web: http://www-labs.iro.umontreal.ca/~vincentp/ift3395/cours/FromLinearRegressionToNNets_print.pdf

9.ANEXOS.

1. CD con código fuente del desarrollo del trabajo dentro de un archivo RAR llamado sequentialOperations que contiene:

Red_Convolucional_Tesis.py, implementación de la red convolucional

Cargar_rnn.py, implementación de la arquitectura recurrente

results.txt, archivo con los resultados obtenidos en el procesamiento de la red recurrente

convnet.pkl, estados de la red Convolucional.