UNIVERSIDAD POLITÉCNICA DE PUEBLA

PROGRAMA ACADÉMICO DE
INGENIERÍA EN INFORMÁTICA

Compresión de imágenes usando una red neuronal artificial

Norma Angélica Pichón Posada

Reporte Técnico PII-03-04-09

COMITÉ EVALUADOR

Dr. Jorge de la Calleja Mora (*Asesor*)
C. Dr. Pedro Vargas García (*Sinodal*)
Dra. María Auxilio Medina Nieto (*Sinodal*)

PROFESOR(A) DE PROYECTO DE INVESTIGACIÓN II

Dra. María Auxilio Medina Nieto

Juan C. Bonilla, Puebla
Mayo 2009

ÍNDICE

RESUMEN

El siguiente proyecto tiene como objetivo la reducción de imágenes usando técnicas de inteligencia artificial como lo son las redes neuronales. Este proyecto pretende dar pauta para la resolución del problema que se tiene al manejar imágenes con gran tamaño y que se presentan en grandes cantidades. Para la implementación y experimentación de la red neuronal se uso el programa Matlab 7.0 y el código propuesto por Hinton G. y Salakhutdinov R [1]. Durante el entrenamiento de la red neuronal y la compresión de imágenes, se arrojaron resultados de error favorables cuando el tiempo de entrenamiento fue mayor al ser probada con configuraciones de 10 y 20 en la variable *maxepoch* del archivo *mnistdeepauto.m* , así mismo fueron desfavorables cuando se les dio un valor de 1 y 2.

CAPÍTULO 1. PLANTEAMIENTO DEL PROBLEMA DE INVESTIGACIÓN

1.1 INTRODUCCIÓN

Actualmente uno de los problemas con el manejo de imágenes es la gran cantidad y tamaño de éstas que se tiene para su procesamiento, ya sea si se trata en tareas de clasificación, visualización o almacenamiento. Una imagen es una matriz numérica en memoria donde cada componente contiene un número entero que a su vez representa un determinado color [3], así una imagen digital de 200 X 200 píxeles, se representa lógicamente por una matriz numérica de igual tamaño, donde cada color de los 40,000 píxeles que la componen es descrito por un número. Sin embargo, las tareas relacionadas con el manejo de imágenes, no procesan sólo una imagen, sino que se pueden administrar miles de ellas. Es entonces que el problema deriva generalmente en la dimensión de las imágenes. Lo anterior porque entre más grandes sean las dimensiones, los recursos y el tiempo para el procesamiento aumentarán. Es por ello que se han desarrollado técnicas para la reducción de datos, particularmente para la compresión de imágenes. Estas técnicas, que en los últimos años han involucrado a la inteligencia artificial, intentan manejar el menor número de atributos posibles conservando la información relevante que representa a cada imagen.

Una de las técnicas de compresión de imágenes más utilizadas es el Análisis de Componentes Principales (*Principal Component Análisis,* PCA), que encuentra las direcciones y vectores de las variaciones más grandes sobre un conjunto de datos [1]. Sin embargo, se puede realizar la compresión de imágenes mediante un método de aprendizaje

automático como redes neuronales, que por sus características, las hacen bastante apropiadas para problemas de clasificación y reconocimiento de patrones de voz, imágenes, señales, entre otros. Éste trabajo da un enfoque distinto al uso de las redes neuronales, que generalmente es de clasificación.

Una red neuronal se compone de unidades llamadas neuronas, donde cada una recibe una serie de entradas a través de interconexiones y emite una salida [2]. Este mecanismo hace posible que las redes "aprendan" si se expone el sistema a un conjunto de ejemplos. Los ejemplos consisten en parejas formadas por un posible valor de variables de entrada junto con la acción que se debe tomar para dicho valor [2].

En el presente trabajo se propone utilizar una red neuronal artificial para comprimir imágenes, de tal manera que tareas posteriores como la clasificación de objetos, pueda realizarse utilizando menores recursos.

1.2 OBJETIVO GENERAL

Investigar e implementar el funcionamiento de una red neuronal artificial para comprimir imágenes.

1.3 OBJETIVOS ESPECÍFICOS

- Poner en funcionamiento el código de la red neuronal propuesta por Hinton G. y Salakhutdinov R [1] sin modificar la estructura de la RNA.
- Modificar el código de la red neuronal para modificar las configuraciones originales.
- Probar diferentes configuraciones de redes neuronales para realizar la compresión de las imágenes.
- Realizar comparaciones entre el método PCA y el propuesto con redes neuronales.

1.4 JUSTIFICACIÓN

La compresión de imágenes, y en general de cualquier conjunto de datos, facilita la realización de diversas tareas como clasificación, visualización, comunicación y almacenamiento. Esto es porque al trabajar con un conjunto de datos menor pero

representativo, se reduce el tiempo de procesamiento y se obtienen resultados similares que al usar imágenes completas. Por otro lado, los dispositivos de almacenamiento pueden alojar cierta cantidad de información, por lo tanto, si se reduce el tamaño de las imágenes, se puede mejorar el rendimiento de estos dispositivos independientemente de su tecnología. Al realizar este trabajo, se espera que el uso de redes neuronales para la compresión de imágenes brinde mejores resultados que la utilización del método de Análisis de Componentes Principales.

1.5 CRONOGRAMA DE ACTIVIDADES

Tarea	Septiembre				Octubre				Noviembre				Diciembre				Enero				Febrero				Marzo				Abril			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Asignación de asesor																																
Elección del tema de investigación																																
Revisión de la literatura																																
Definición de hipótesis																																
Elaboración de la propuesta de solución																																
Aprendizaje del manejo de Matlab																																
Instrucción en imágenes digitales																																
Aprendizaje de redes neuronales																																
Entendimiento de PCA																																
Revisión del código en Matlab de la red neuronal propuesta por Hinton																																
Estudio del código de PCA																																
Implementación de pruebas del código revisado																																
Modificación del código revisado y probado de la red neuronal propuesta por Hinton																																
Comparación de resultados de la red neuronal contra el método PCA																																

1.6 ALCANCES Y LIMITACIONES

Alcances:

- Se utilizará una red neuronal multicapa para comprimir imágenes, lo cual ofrece un enfoque y uso distinto a las tareas tradicionales de una RNA tal como la clasificación.

Limitaciones:

- El método propuesto sólo se comparará contra PCA.
- Se experimentará con conjuntos pequeños de imágenes, algunas decenas.
- Sólo se pondrá en funcionamiento la red neuronal, dejando para trabajo a futuro la compresión de otro tipo de imágenes.

1.7 RECURSOS DE HARDWARE Y SOFTWARE

Software:

- Código fuente escrito en Matlab de una red neuronal, desarrollado por Hinton G. y Salakhutdinov [1].
- Programa Matlab versión 7.0
- Toolbox escrito en Matlab de análisis de componentes principales (PCA).

Hardware:

- Espacio mínimo en disco duro de 1 Gb.

Para hacer las pruebas de implementación se usarán dos computadoras:

- Computadora de 3 Gb. en memoria RAM, 500 Gb. en disco duro y un procesador Intel Core 2 de 2.4 Ghz.
- Computadora de 256 Mb. En RAM, 120 Gb. en disco duro y un procesador Pentium IV de 3.0 Ghz.

CAPÍTULO 2. MARCO TEÓRICO

2.1 IMÁGENES DIGITALES

2.1.1 DEFINICIÓN

Existen diferentes definiciones dadas a consecuencia del tipo de imagen digital. [Ortega 2008] indica que las imágenes digitales se dividen en dos grandes grupos: imágenes vectoriales e imágenes de mapa de bits o bitmap. Una imagen de mapa de bits, según [Gómez 2005] es una matriz numérica en memoria, donde cada componente de la matriz alberga un número entero que a su vez representa un determinado color. En [Microsoft Encarta 2006] se define a una imagen bitmap como un conjunto secuencial de bits en memoria, donde cada bit corresponde a un píxel en la pantalla. En una imagen vectorial "la información de cada uno de los puntos se recoge en forma de ecuación matemática que lo relaciona con el resto de los puntos que forman la imagen [Ortega 2008]."

La obtención de una imagen digital puede ser mediante dispositivos de conversión analógica-digital tal como cámaras digitales y escáneres, o de forma directa con la utilización de programas informáticos. El almacenamiento de una imagen digital se hace en un dispositivo de grabación de datos como un disco duro.

2.1.2 REPRESENTACIÓN DE IMÁGENES

Es importante la estructura y representación de las imágenes digitales, ya sean tanto imágenes vectoriales o de mapa de bits. En la página Web de la [Biblioteca de la Universidad de Cornell -USA 2003], menciona que al realizarse una muestra de una imagen digital tipo bitmap, se confecciona un mapa en forma de cuadrícula de puntos llamados píxeles. Cada píxel tiene asignado un valor tonal como el negro, blanco, matices de gris o color. Un píxel se representa por un código binario, que se almacenan en una secuencia.

Según [Gómez 2005], una imagen digital bitmap de m x n píxeles se representa con una matriz numérica de igual tamaño, donde el color de cada píxel es descrito por un código numérico. Es así que un número en la matriz representa un píxel de la imagen, donde se tiene coincidencia de posición tanto en fila y columna en la matriz y en la imagen, tal y como se ejemplifica en la Figura 1.

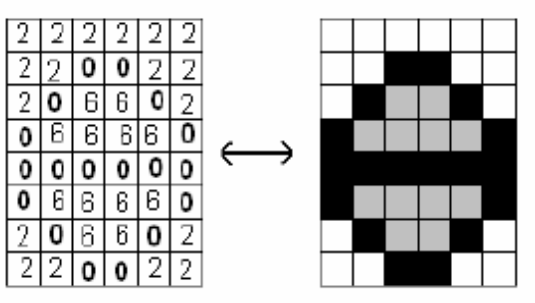

Figura 1. Relación entre imagen de píxeles y matriz numérica

Con respecto a las imágenes vectoriales, [Ortega 2008] indica que debido a su definición matemática, una imagen vectorial ocupa un espacio menor en comparación a las bitmap, ya que una fórmula matemática es suficiente para representar todos los puntos que componen a una imagen, tal y como se muestra en la Figura 2. Las imágenes vectoriales tienen la ventaja de que la calidad de la imagen no varía al modificar el tamaño, ya que la información de cada punto o píxel no es absoluta sino relativa al resto de la imagen.

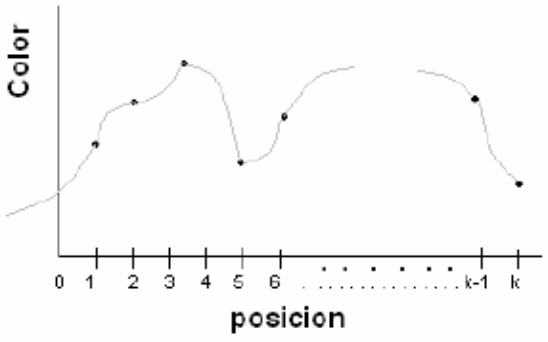

Figura 2. Curva de una función matemática que representa a una imagen

Para la visualización de una imagen digital se requiere de un programa de visualización que convierta la información binaria en píxeles y de un dispositivo que permita su despliegue tal como lo hace un monitor. Cabe hacer mención, que el presente proyecto de investigación está dedicado a la compresión de imágenes digitales, la cual se utiliza para reducir el tamaño del archivo de una imagen para un mejor almacenamiento, procesamiento y transmisión.

2.2 REDES NEURONALES

2.2.1 DEFINICIÓN

A fin de entender la metodología que se planteará más adelante, es necesario que se tenga una comprensión clara sobre el concepto de una red neuronal. Según citan [Hilera y Martínez 1995], al profundizar en los principios de las redes neuronales artificiales y observar continuamente el término neurona, no es de extrañar que se piense en el cerebro humano, este hecho quizás se deba a que las redes neuronales artificiales están basadas en la inspiración biológica. El hombre posee cerca de 10 000 000 000 de neuronas masivamente interconectadas, la neurona es una célula especializada que puede propagar una señal electroquímica. Las neuronas tienen una estructura ramificada de entrada, las dendritas y una estructura ramificada de salida, los *axones*. El cuerpo de la neurona o *soma* contiene el núcleo, que se encarga de todas las actividades metabólicas de la neurona y recibe la información de otras neuronas vecinas a través de las conexiones sinápticas. Esta estructura se muestra en la Figura 3. Los axones de una célula se conectan con las dendritas de otra, por vía de la sinapsis, la neurona se activa y excita una señal electroquímica a lo largo del axón. Esta señal transfiere la sinapsis a otras neuronas, las que a su vez pueden excitarse. Las neuronas se excitan sólo sí la señal total recibida en el cuerpo de las células por conducto de las dendritas, es superior a cierto nivel, conocido como *umbral de excitación*. Las redes neuronales artificiales tratan de imitar este principio de funcionamiento cerebral.

Figura 3. Estructura de una neurona biológica

Las redes neuronales artificiales, denominadas también como RNA, son un paradigma de aprendizaje y procesamiento automático, inspiradas en la forma en que funciona el cerebro. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir una salida [Nilsson 2001].

Una red neuronal, según [Freman y Skapura 1993], es un sistema de neuronas paralelas conectadas entre sí, donde cada neurona de la red se representa como un nodo. Estas

conexiones establecen una estructura jerárquica que tratando de emular la fisiología del cerebro, busca nuevos modelos de procesamiento para solucionar problemas concretos del mundo real. “Lo importante en el desarrollo de la técnica de las RNA es su útil comportamiento al aprender, reconocer y aplicar relaciones entre objetos y tramas de objetos propios del mundo real [Nilsson 2001].”

En [Villanueva 2002], se destaca que las propiedades que hacen especialmente atractivas a las redes neuronales para ser usadas en una gran cantidad de problemas prácticos son las siguientes:

a) *Simulación de sistemas distribuidos no lineales*. Una neurona es un elemento no lineal por lo que una interconexión de ellas también será un dispositivo no lineal.

b) *Son sistemas tolerantes a fallos*. Una red neuronal, al ser un sistema distribuido, permite el fallo de algunos elementos individuales sin alterar significativamente la respuesta total del sistema.

c) *Adaptabilidad*. Una red neuronal tiene la capacidad de modificar los parámetros de los que depende su funcionamiento de acuerdo con los cambios que se produzcan en su entorno de trabajo.

2.2.2 ESTRUCTURA

Según [Villanueva 2002], las RNA imitan la estructura del sistema nervioso, con la intención de construir sistemas de procesamiento de información que puedan presentar un comportamiento “inteligente”.

[Hilera y Martínez 1995], describen que las redes neuronales artificiales están formadas por una gran cantidad de neuronas, éstas no suelen denominarse neuronas artificiales sino nodos o unidades de salida. Un nodo o neurona cuenta con una cantidad variable de entradas que provienen del exterior $(X_1, X_2, ..., X_m)$. A su vez dispone de una sola salida (X_j) que transmitirá la información al exterior o hacia otras neuronas. Cada X_j o señal de salida tiene asociada una magnitud llamada *peso*, éste se calcula en función de las entradas, por lo cual cada una de ellas es afectada por un determinado peso $(W_{jo}...W_{jq+m})$. Los pesos

corresponden a la intensidad de los enlaces sinápticos entre neuronas y varían libremente en función del tiempo en cada una de las neuronas que forman parte de la red. La Figura 4, ilustra la estructura de una neurona, según lo citado por [Hilera y Martínez 1995].

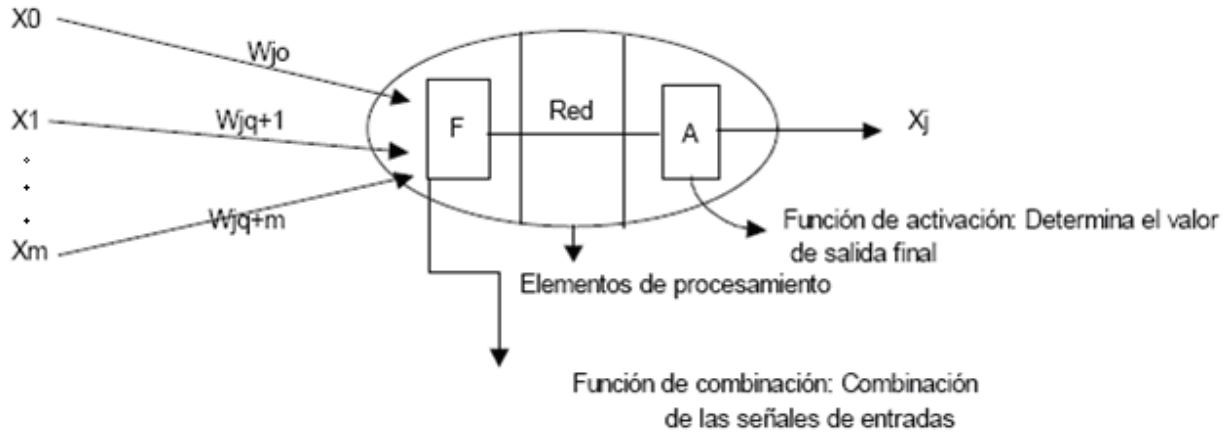

Figura 4. Esquema de una neurona.

[Nilsson 2001] explica que en todo modelo de redes neuronales se tienen cuatro elementos básicos:

a) Un *conjunto de conexiones, pesos o sinapsis* que determinan el comportamiento de la neurona. Estas conexiones pueden ser excitadoras (presentan un signo positivo), o inhibidoras (conexiones con signo negativo).
b) Un *sumador* que se encarga de sumar todas las entradas multiplicadas por las respectivas sinapsis.
c) Una *función de activación no lineal* para limitar la amplitud de la salida de la neurona.
d) Un *umbral exterior* que determina el umbral por encima del cual la neurona se activa.

La Figura 5 muestra las partes en que está dividida una red neuronal. Como [Villanueva 2002] explica, cada neurona realiza una función matemática. Las neuronas se agrupan en capas, está diseñada y entrenada para llevar a cabo una labor específica. Finalmente, una o varias redes, más las interfaces con el entorno, conforman un sistema neuronal.

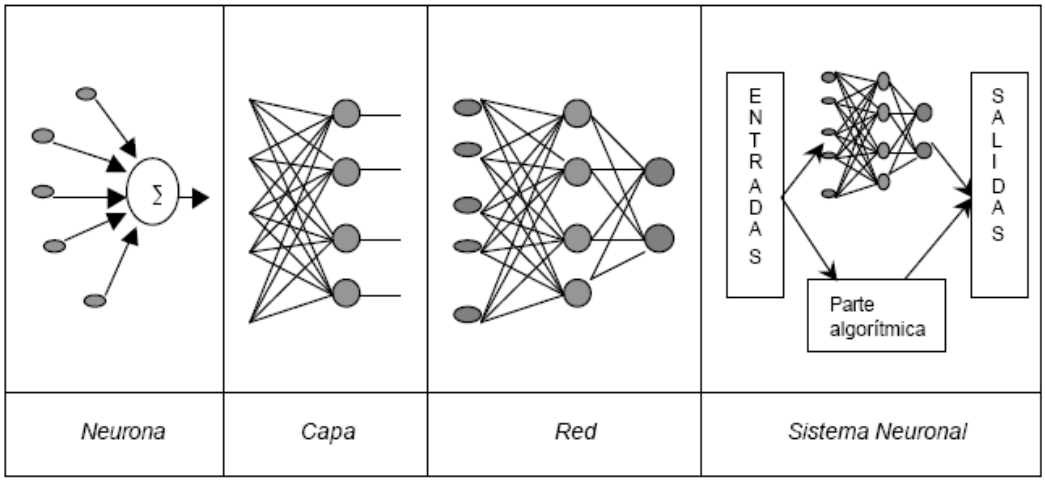

Figura 5. Estructura jerárquica de un sistema basado en RNAs

En [Freman y Skapura 1993], se señala que el proceso de aprendizaje consiste en hallar los pesos que codifican los conocimientos. Una regla de aprendizaje hace variar el valor de los pesos de una red hasta que éstos adoptan un valor constante, cuando esto ocurre, se dice que la red ya "ha aprendido" o ha sido "entrenada". "En una RNA entrenada, el conjunto de los pesos determina el conocimiento de esa RNA y tiene la propiedad de resolver el problema para el que la RNA ha sido entrenada [Nilsson 2001]."

2.2.3 TIPOS DE REDES NEURONALES

Según [Freman y Skapura 1993], al conectar varias neuronas de un determinado modo, se consigue una red. Existen variaciones de topologías que se clasifican según tres criterios: número de niveles o capas, número de neuronas por nivel o según las formas de conexión.

Según el número de capas que posea la red neuronal, [Villanueva 2002] las clasifica en monocapa y multicapa. Las RNA monocapa son consideradas las más sencillas, ya que se tiene una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan diferentes cálculos. Por otra parte, las RNA multicapa, a diferencia de las monocapa, tienen un conjunto de capas intermedias o capas ocultas entre la entrada y la salida. Las Figuras 6 y 7, ejemplifican esta clasificación.

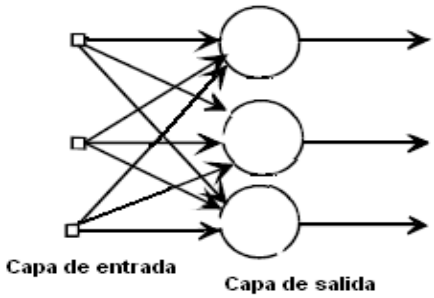

Figura 6. Red neuronal monocapa

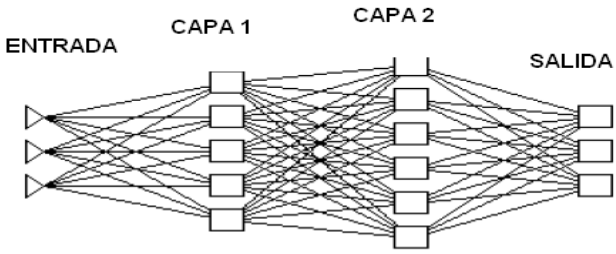

Figura 7. Red neuronal multicapa

[Hilera y Martínez 1995] clasifica a las RNA según el tipo de conexiones en no recurrentes y recurrentes. En las no recurrentes, la propagación de las señales se produce en un sentido solamente, no existe la posibilidad de retroalimentación. Por otro lado, las RNA recurrentes son caracterizadas por la existencia de lazos de retroalimentación. Estos lazos pueden ser entre neuronas de diferentes capas, neuronas de la misma capa o entre una misma neurona. La Figura 8 representa el esquema de una red recurrente.

[Freman y Skapura 1993] dividen a las RNA según el grado de conexión en redes neuronales totalmente conectadas y parcialmente conectadas. En la primera clase, todas las neuronas de una capa se encuentran conectadas con las de la capa siguiente o con las de la anterior. En las redes parcialmente conectadas, no se da la conexión total entre neuronas de diferentes capas.

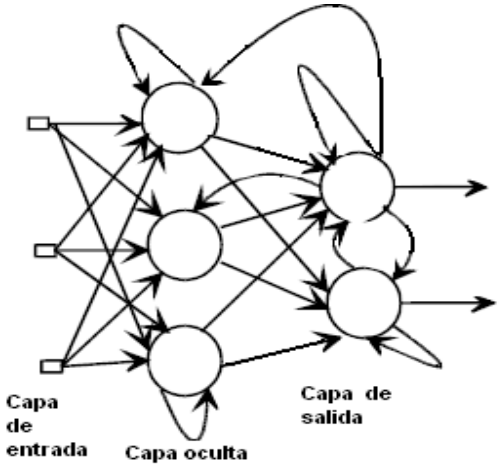

Figura 8. Red neuronal recurrente

2.3 ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

2.3.1 DEFINICIÓN

Según [Terrádez 2000], PCA es una técnica estadística de síntesis de la información o reducción de la dimensión en número de variables. Es decir, ante un banco de datos con muchas variables, el objetivo será reducirlas a un menor número perdiendo la menor cantidad de información posible. Los nuevos componentes principales o factores serán una combinación lineal de las variables originales, y además serán independientes entre sí.

[Smith 2002] explica que en estadística, PCA es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos. Técnicamente, PCA busca la proyección según la cual los datos queden mejor representados en términos de mínimos cuadrados que expliquen la variabilidad de dichos datos. Este método permite representar los datos originales en un espacio de dimensión inferior del espacio original, mientras limite al máximo la pérdida de información.

Una de las ventajas de PCA para reducir la dimensionalidad de un grupo de datos, es que retiene aquellas características del conjunto de datos que contribuyen más a su varianza. PCA disminuye el número de variables hasta un número deseado, de modo que se mantenga máxima la cantidad de varianza en los datos de entrada. “Esta reducción de dimensión ayuda a eliminar la información redundante y el ruido de los datos [Terrádez 2000].”

PCA se emplea en muchas disciplinas científicas y en aplicaciones de ingeniería como reconocimiento de patrones, procesamiento de imágenes, análisis espectral de alta resolución para la estimación de frecuencias, en la reducción de sistemas de control, entre otras.

2.4 COMPRESIÓN DE IMÁGENES MEDIANTE REDES NEURONALES

2.4.1 TRABAJO DE HINTON

La reducción de dimensiones facilita la clasificación, visualización, comunicación y almacenamiento de grandes dimensiones de datos. Un método simple ampliamente usado es PCA, que encuentra las direcciones de gran variación en un conjunto de datos y representa cada punto por sus coordenadas a lo largo de cada dirección. [Hinton y Salakhutdinov 2006] dicen: “Nosotros describimos una generalización no lineal de PCA que usa una red neuronal multicapa adaptable llamada *encoder* para transformar los datos de grandes dimensiones en código de pequeña dimensión. Además se usa otra red neuronal similar llamada *decoder* para recuperar los datos del código”.

Pueden convertirse los datos de grandes dimensiones a pequeñas dimensiones mediante el entrenamiento de dos redes neuronales multicapa con una pequeña capa central para reconstruir los vectores de entrada, tal y como se ejemplifica en la Figura 9. Empezando con pesos aleatorios en las dos redes neuronales, éstas pueden ser entrenadas juntas para minimizar la diferencia entre los datos originales y su reconstrucción. Para esto, según [Hinton y Salakhutdinov 2006] se usa el método de retropropagación; primero a través de la red neuronal *decoder* y después a través de la red neuronal *encoder*. El sistema entero se llama *autoencoder*.

[Hinton y Salakhutdinov 2006] afirman que la utilización de redes neuronales, es una manera eficaz para reducir la dimensión de los datos, en comparación que el análisis de componentes principales.

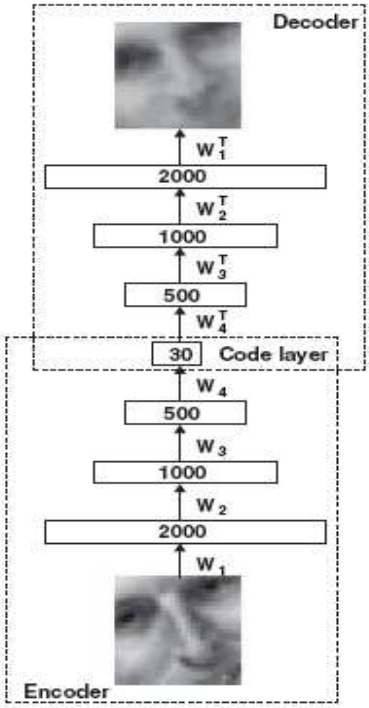

Figura 9. Interacción entre las redes neuronales *decoder* y *encoder* que tienen en común una pequeña capa central.

El trabajo de [Hinton y Salakhutdinov 2006], se acerca a lo que se quiere lograr en este proyecto de investigación, ya que se emplean las redes neuronales como medio principal para la compresión de imágenes, sin embargo, los autores mencionados entrenaron su red neuronal propuesta con un conjunto conformado de miles de imágenes. Esto significa una diferencia, ya que el presente proyecto tiene como limitación usar en la fase de entrenamiento un número reducido a decenas.

2.4.2 PROPUESTA DE GÓMEZ

[Gómez 2005] planteó un método para la reducción de imágenes mediante el uso de redes neuronales multicapa con la finalidad de conseguir un menor almacenamiento y obtener un software con fines iguales pero con mejores resultados que los ofrecidos por *winzip* y *winrar*. Esto supone una diferencia en la presente investigación, ya que uno de los objetivos del trabajo es la comparación de resultados contra PCA y no con una aplicación como *winzip*.

Otra diferencia que se encuentra con respecto al trabajo de [Gómez 2005], es que el desarrollo se hizo en C++ junto con la librería allegro, mientras que este proyecto utilizará el

código de una red neuronal escrita en Matlab. Sin embargo [Gómez 2005], menciona que un problema en su red neuronal fue el alto consumo de tiempo para el entrenamiento. Por este motivo, el autor concluye que una manera de mejorar su sistema sería el uso de un sistema experto junto con la red neuronal. Se recalca que este proyecto no contempla el uso de sistemas expertos.

CAPÍTULO 3. DISEÑO DE LA RED NEURONAL

3.1 INTRODUCCIÓN

En el presente capítulo se define el esquema del sistema utilizado como base para el desarrollo del proyecto de investigación.

3.2 REQUERIMIENTOS

Los requerimientos describen las características de la red neuronal y son los siguientes:

a) Requerimientos funcionales

1. Entrenar la red neuronal
2. Comprimir imágenes con la red neuronal entrenada

b) Requerimientos no funcionales

Plataforma. La red neuronal se ejecuta bajo la plataforma de Microsoft Windows Vista.

Costo. El código de la red neuronal es libre, no tiene ningún costo.

Desempeño. El desempeño de la red neuronal es de importancia significativa en el proyecto, ya que por lo general el entrenamiento tiene una duración de hasta de 7 horas. Se compara la calidad de la imagen obtenida con la original.

Herramienta. Para manejar a la red neuronal, se usa Matlab 7.0, el cual proporciona el acceso al manejo del código, así como el entrenamiento.

3.3 CASOS DE USO

Los casos de uso, tienen el propósito de representar las actividades que el usuario puede realizar en la red neuronal. A continuación se describen los principales.

1) *Nombre del caso de uso*: Entrenamiento de la red neuronal.
   Descripción: En este proceso, la red neuronal toma como entrada un conjunto de imágenes para que la red neuronal aprenda sus características y ajuste los pesos

apropiadamente para lograr una correcta compresión. Para una mejor ejemplificación vea la Figura 10.

Datos de entrada: Conjunto de 10,000 a 60,000 imágenes de dígitos de 28 X 28 píxeles.

Datos de salida: Red neuronal ajustada para comprimir imágenes digitales.

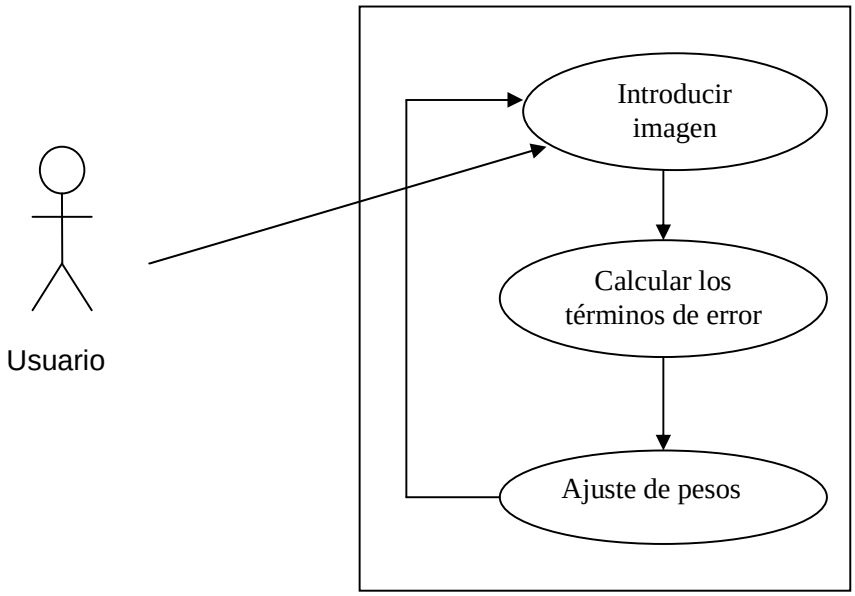

Figura 10. Caso de uso entrenamiento de la red neuronal

2) *Nombre del caso de uso:* Compresión de una imagen digital

Descripción: En este proceso, se recibe como entrada una imagen de 28 X 28 pixeles, la cual se procesa por la red neuronal previamente entrenada que arroja como resultado otra imagen pero de menor tamaño que la original, esperando también que la calidad del resultado sea buena. Véase Figura 11.

Datos de entrada: Red neuronal entrada y una imagen digital como la Figura 12.

Datos de salida: Una imagen digital de formato *jpg* con un tamaño menor que la original.

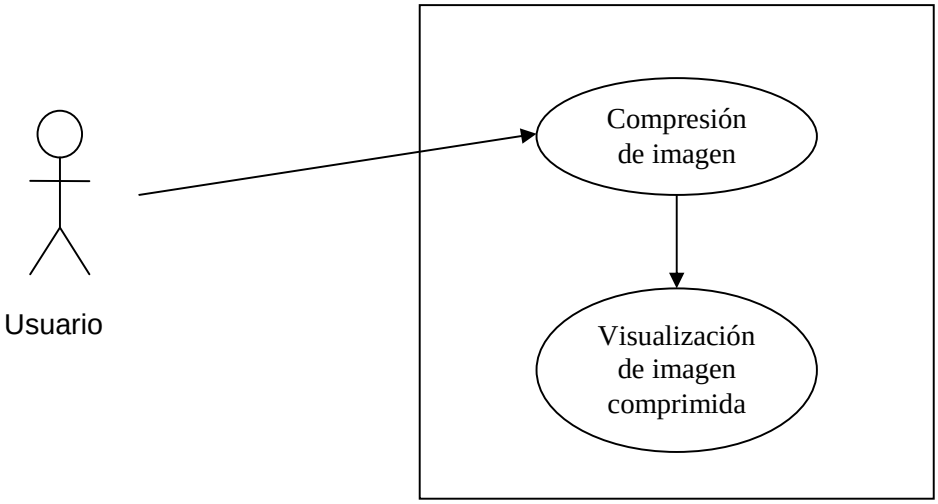

Figura 11. Caso de uso compresión de imagen digital

3.4 CASOS DE PRUEBA

1) *Nombre del caso de prueba:* Inserción de imágenes para entrenamiento

Nombre del caso de uso que se desea probar: Entrenamiento de la red neuronal.

Datos de entrada: Conjunto de 10,000 a 60,000 imágenes de dígitos de 28 x 28 píxeles.

Figura 12. Ejemplo de imagen original de dígitos

2) *Nombre del caso de prueba:* Inserción de imagen para compresión

Nombre del caso de uso que se desea probar: Compresión de imagen digital

Datos de entrada: Red neuronal entrenada con imagen

Datos de salida: Imagen digital comprimida en formato jpg. La calidad de la imagen menor en comparación a la original. La Figura 13 muestra un ejemplo de una compresión de baja calidad.

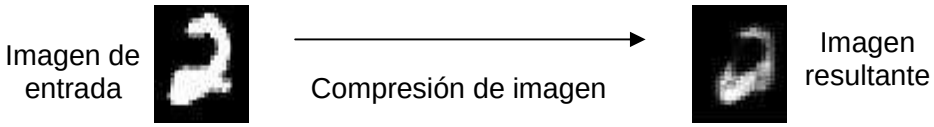

Figura 13. Proceso de compresión de imagen

3.5 RECOLECCIÓN DE DATOS Y SELECCIÓN DE PRUEBAS

Los datos necesarios para hacer funcionar a la red neuronal es un conjunto de imágenes de dígitos que sirven como base para el entrenamiento. Las imágenes están disponibles en el sitio Web de Geoffrey E. Hinton[1].

3.6 ESPECIFICACIÓN DE ACTORES

La única persona que podrá tener acceso a los datos, en este caso las imágenes y a la red neuronal, será el usuario, que se define como aquella que monitorea la fase de entrenamiento de la red neuronal y la compresión de imágenes. Así mismo, esta persona se encarga de las modificaciones en la red neuronal.

[1] Sitio Web con la dirección *http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html*

CAPÍTULO 4. IMPLEMENTACIÓN

Para el entrenamiento de la red neuronal, así como para la compresión de imágenes, se usan siete archivos escritos en Matlab, obtenidos del sitio Web señalado en el capítulo 3.

A continuación se da una breve explicación del funcionamiento de los archivos:

1. *mnistdeepauto.m* Archivo principal para el entrenamiento de la red neuronal llamada "Autoencoder". En este archivo se puede establecer el número máximo de épocas para el preentrenamiento de cada capa de la red neuronal. Se le llama época a cada iteración de la red por el lote de entradas creadas por *makebatches.m*. Se modifica el valor de la variable maxepoch.
2. *converter.m* Convierte las imágenes en formato reconocible para Matlab, para utilizarlas durante el entrenamiento.
3. *makebatches.m* Crea lotes para el entrenamiento de una RBM (Restricted Boltzmann Machine), que es una red neural de dos capas.
4. *rbm.m* Entrena a un RBM con unidades visibles (píxeles) y ocultas (vectores detectores de figuras). Las unidades visibles son conectadas a las unidades ocultas usando conexiones con pesos simétricos.
5. *rbmhidlinear.m* Entrena a un RBM. Conecta las unidades visibles con las ocultas redefiniendo los valores de las unidades ocultas.
6. *backprop.m* Contiene el código necesario para ajustar los pesos de la red neuronal durante la fase de entrenamiento utilizando el método de retropropagación. Los pesos ajustados obtenidos se almacenan en el archivo mnist_weights.mat. Se puede establecer el máximo de épocas modificando el valor de la variable *maxepoch*.
7. *mnistdisp.m* Despliega el progreso durante la etapa de ajuste que realiza el archivo *backprop.m.* Las imágenes resultantes de la compresión son mostradas por medio de este archivo.

Para ejecutar el archivo *mnistdeepauto.m* es necesario que se encuentre en la misma carpeta junto los archivos antes mencionados, además de los archivos que contienen las imágenes que se utilizan para el entrenamiento y la compresión. La Figura 14 describe la interacción de los diferentes archivos invocados desde el archivo principal *mnistdeepauto.m.*

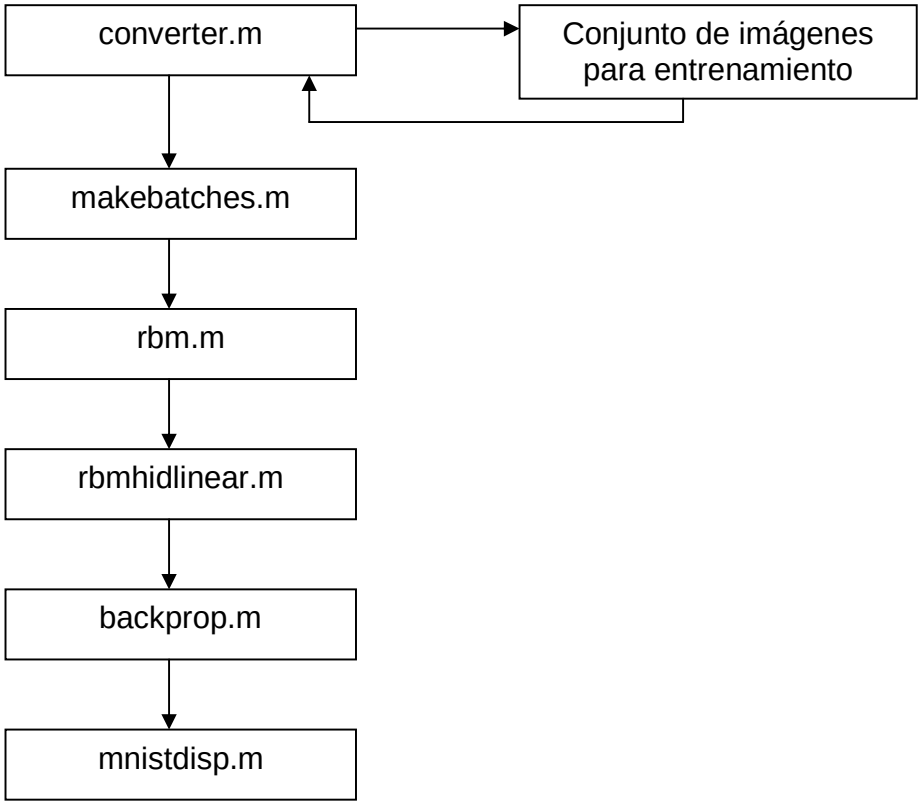

Figura 14. Flujo de información del archivo principal mnistdeepauto.m

Es también relevante explicar la configuración de la red neuronal, la cual es 1000 - 500 - 250 – 30 para la utilización de todas las pruebas. Esto se muestra en la Figura 15, donde se puede observar las 4 capas de la red neuronal, así como también el número máximo de neuronas para cada capa.

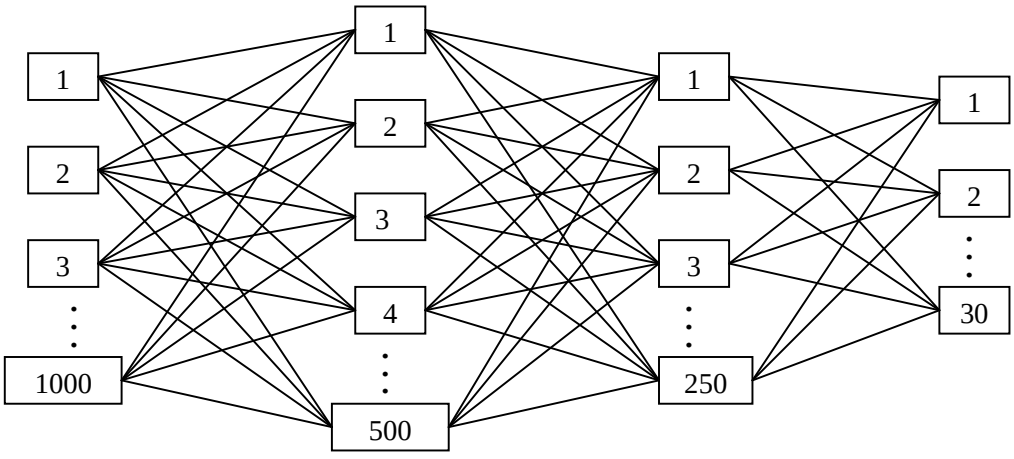

Figura 15. Estructura de la red neuronal

CAPÍTULO 5. RESULTADOS

Este capítulo tiene como propósito mostrar los resultados obtenidos mediante las pruebas realizadas durante la ejecución de la red neuronal. El término ejecución engloba las fases entrenamiento y compresión.

5.1 EJECUCIÓN DE LA RED NEURONAL

Para una mejor comprensión del lector acerca del funcionamiento de la red neuronal, se procede a explicar por medio de pantallas la ejecución. El ejemplo considera el valor 2 de la variable *maxepoch* del archivo *mnistdeepauto.m* y el valor de 5 a la variable *maxepoch* del archivo *backprop.m.*

El primer rectángulo que se muestra en la Figura 15 es la conversión de los archivos que contienen las imágenes a un formato reconocible por Matlab. Todos los experimentos se realizaron con un conjunto de 10,000 a 60,000 imágenes de dígitos de 28 X 28 píxeles (ver las elipses de la Figura 16).

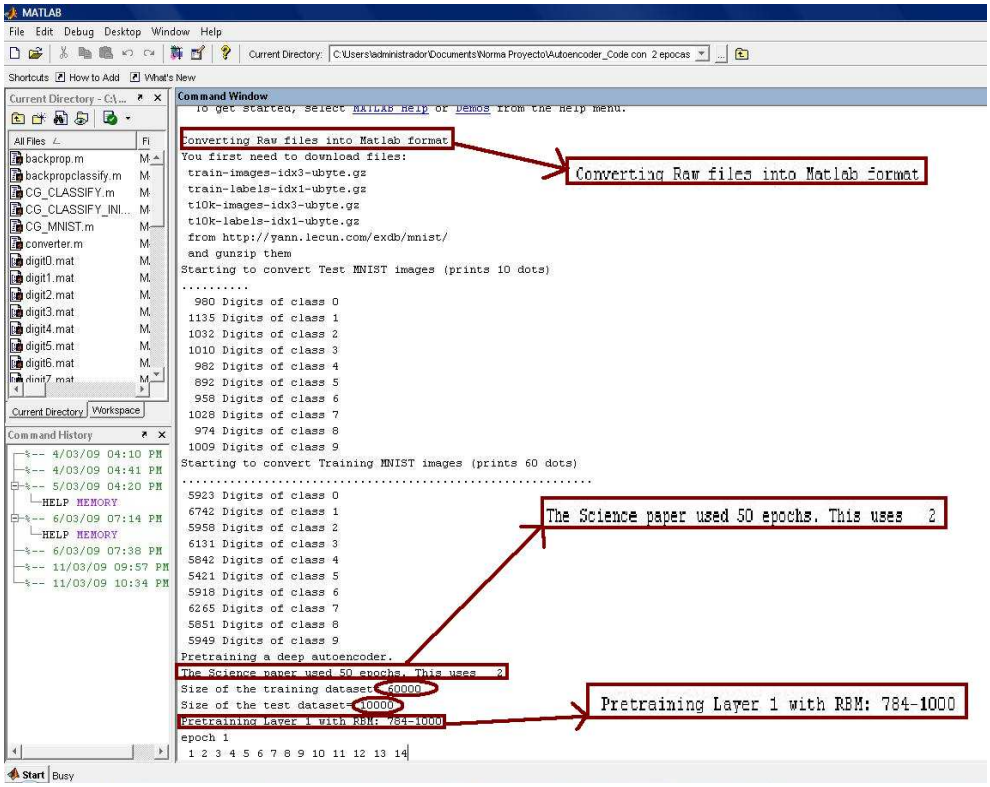

Figura 16. Comienzo de la ejecución de la red neuronal

El número máximo de épocas utilizadas por el archivo *mnistdeepauto.m,* en este caso 2, se muestra en el segundo rectángulo de la Figura 16. Desde el tercer rectángulo comienza el entrenamiento de las 4 capas de la red neuronal con sus respectivos números de neuronas.

En las elipses de las Figuras 17 y 18 se muestra cómo el número máximo de épocas es respetado durante el entrenamiento de cada una de las capas.

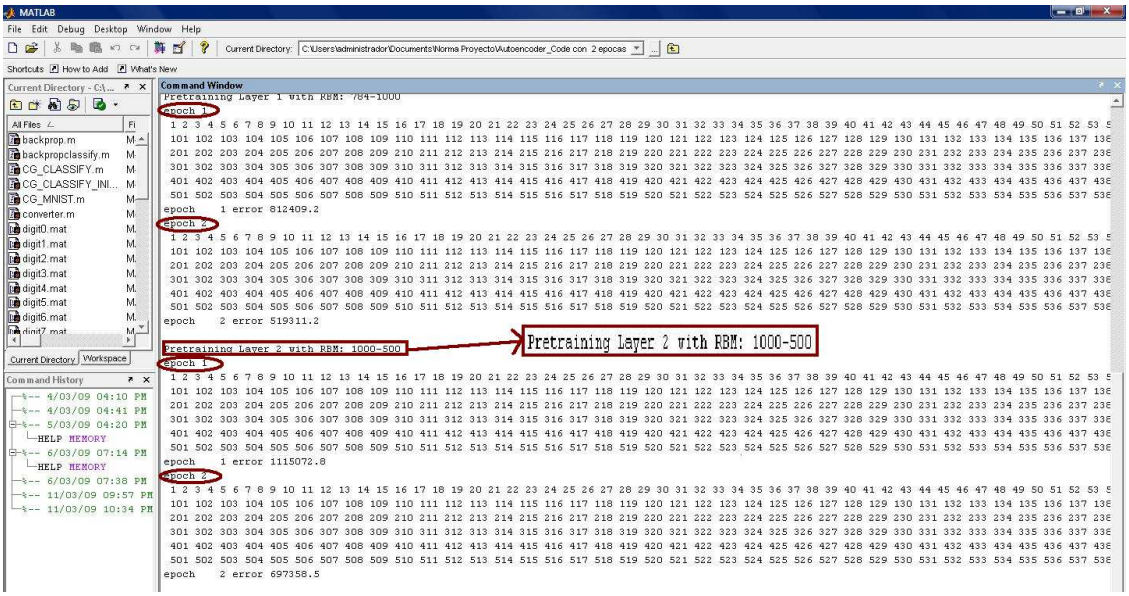

Figura 17. Entrenamiento de la capa 1 y 2 de la red neuronal

Se puede apreciar en la Figura 18 cómo termina el entrenamiento de la red neuronal en la capa 4, dejando como resultado los **30 componentes** que se utilizaron para la **reconstrucción** de las imágenes en el proceso de compresión.

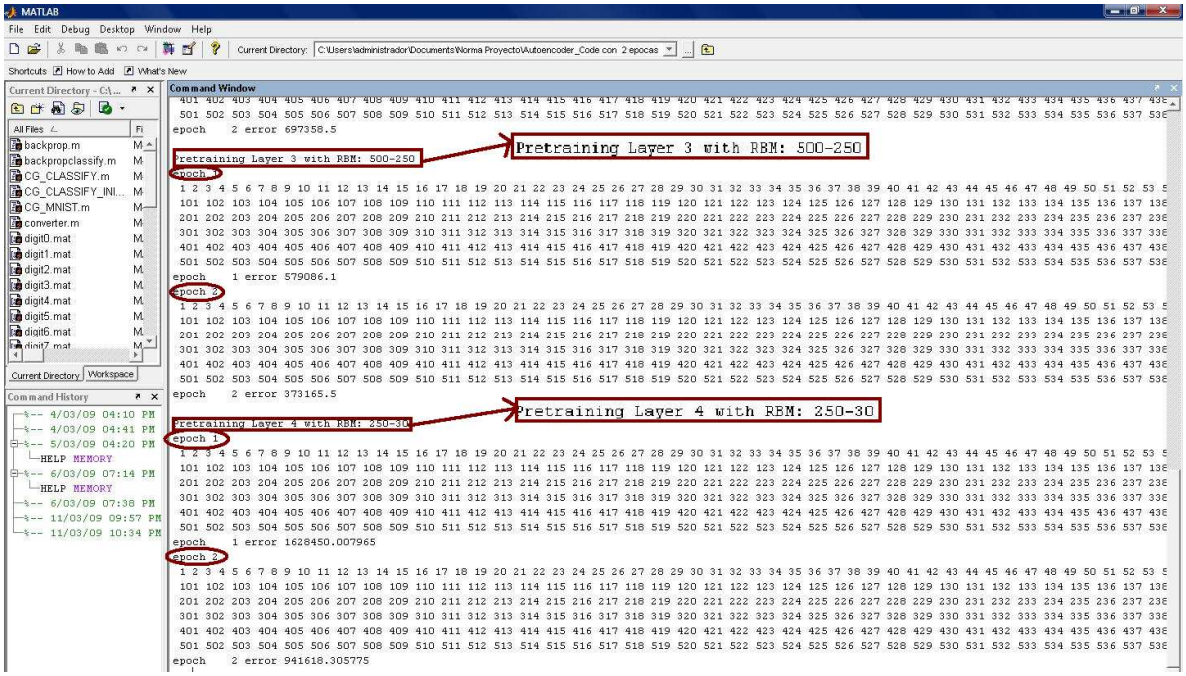

Figura 18. Entrenamiento de la capa 3 y 4 de la red neuronal

Después del entrenamiento se procede al proceso de compresión, donde se muestra el error obtenido en cierto número de época, según el valor establecido en la variable *maxepoch* del archivo *backprop.m* (en este caso 5). La Figura 19 ejemplifica el mensaje obtenido para la época 1 de reconstrucción. Es similar para las 4 épocas restantes. El apéndice A describe un ejemplo completo de la compresión.

```
Displaying in figure 1: Top row - real data, Bottom row -- reconstructions
Before epoch 1 Train squared error: 25.436 Test squared error: 25.093
```

Figura 19. Mensaje de error obtenido en la época 1 de reconstrucción

Simultáneamente al mensaje de error, se despliega una ventana como la de la Figura 20, la cual muestra las dos filas de imágenes de dígitos reconstruidas después del mensaje de la Figura 19. La primera fila, representa a las imágenes originales, mientras que la segunda fila muestra las imágenes obtenidas por la red neuronal.

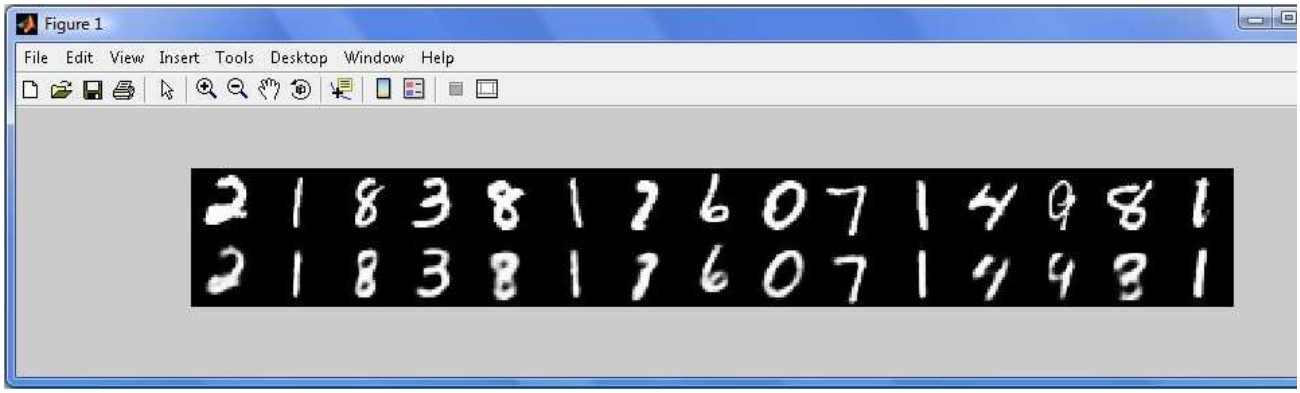

Figura 20 Ventana que muestra el resultado de la Figura 19

5.2 ANÁLISIS COMPARATIVO

Para medir el desempeño de la red neuronal con respecto al tiempo de entrenamiento y compresión, se modificó el valor de la variable *maxepoch* del archivo *mnistdeepauto.m*, con los valores mostrados en la Tabla 1. Por otro lado, se mantuvo el valor de *maxepoch* del archivo *backprop.m*. El tiempo obtenido está en minutos, tanto en el entrenamiento como en la compresión de imágenes de dígitos, además de la duración total de la ejecución de la red neuronal.

Tabla 1. Tiempos obtenidos por la red neuronal

Valor *maxepoch* del archivo *mnistdeepauto.m*	Valor *maxepoch* del archivo *backprop.m*	Entrenamiento (min)	Compresión (min)	Ejecución (min)
1	5	11	294	305
2	5	19	288	307
6	5	53	290	343
10	5	84	264	348
20	5	161	267	428

La Tabla 1 muestra que al aumentar los valores de *maxepoch* del archivo *mnistdeepauto.m*, se incrementa también el tiempo de ejecución, (tiempo mínimo 5 horas 5 min, máximo 7 hrs 8 min).

Ahora bien, otra forma de medir el desempeño de la red neuronal es a través de los errores y la calidad de las imágenes obtenidas. Como el valor de *maxepoch* del archivo *backprop.m* se

mantuvo siempre en 5, significa que existieron 5 resultados de error correspondientes a cada una de las épocas. La Tabla 2 muestra estos errores.

Tabla 2. Errores obtenidos en las diferentes épocas

Valor *maxepoch* del archivo *mnistdeepauto.m*	Archivo backprop.m				
	Época 1	**Época 2**	**Época 3**	**Época 4**	**Época 5**
1	35.438	12.528	11.194	10.468	10.008
2	25.093	10.515	9.544	9.100	8.731
6	17.382	8.150	7.511	7.154	6.909
10	13.04	6.996	6.472	6.141	5.926
20	11.61	6.344	5.853	5.583	5.395

En la Tabla 2 se puede notar que el error obtenido en la época 1 con respecto a los de la época 2 disminuye considerablemente, de 45.35% (cuando *maxepoch* vale 20) a 62.36% (cuando *maxepoch* vale 1). En contraste, la diferencia de error obtenido entre la época 2 a la época 5 es de tan sólo unas cuantas décimas.

También se observa en la Tabla 2 que los errores de la época 1 son los errores iniciales, mientras que los pertenecientes a la época 5 son los finales. Durante el paso de las cinco épocas, se hacen los ajustes necesarios para minimizar el error y así obtener una imagen de mejor calidad. El menor error inicial es 11.61, pertenece al valor 20 de *maxepoch* de *mnistdeepauto.m* y el mayor fue de 35.438 (valor 1 de *maxepoch*). La diferencia de calidad de imagen se puede observar en la Figura 21, donde la primera fila pertenece a las imágenes originales, la segunda cuando existe un error de 35.438 (error máximo) y la tercera fila cuando es de 11.61 (error mínimo).

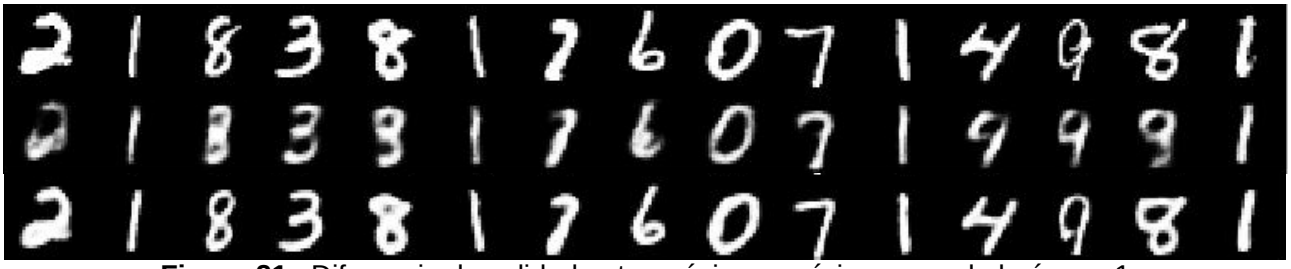

Figura 21. Diferencia de calidad entre máximo y mínimo error de la época 1

Por otro lado, a lo que se refiere con los errores finales, se destaca que el menor fue de 5.395 correspondiente al valor 20 de *maxepoch* de *mnistdeepauto.m* y el mayor fue de 10.008 referente cuando *maxepoch* vale 1. La Figura 22 muestra las diferencias de

reconstrucción, la primera fila pertenece a las imágenes originales, la segunda al error máximo y la tercera fila al error mínimo de la época 5.

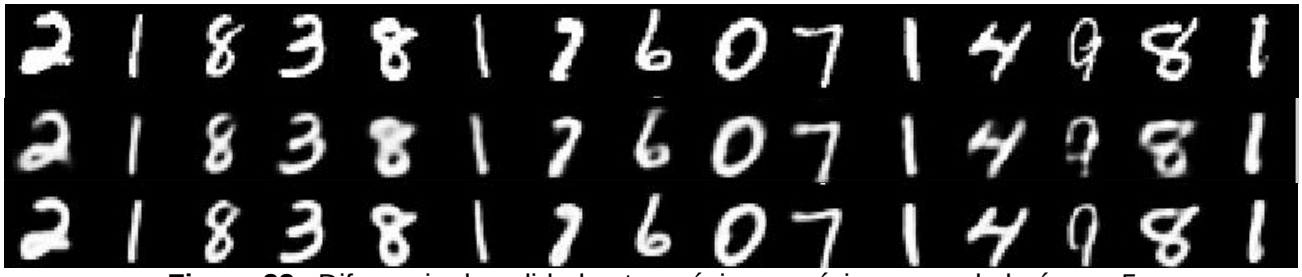

Figura 22. Diferencia de calidad entre máximo y mínimo error de la época 5

En el apéndice B se puede observar y comparar las imágenes obtenidas para cada valor de *maxepoch* del archivo *mnistdeepauto.m* en cada una de las 5 épocas de *backprop.m.*

5.3 COMPARACIÓN ENTRE LOS RESULTADOS OBTENIDOS POR RNA Y PCA

Para cumplir uno de los objetivos planteados, se hace la comparación entre los resultados obtenidos por la red neuronal contra los que obtuvieron Hinton G. y Salakhutdinov R [1] al utilizar PCA.

La Figura 23 muestra las imágenes obtenidas por una red neuronal y las reconstrucciones obtenidas utilizando PCA, donde cada técnica de compresión utiliza 30 componentes. En la primera fila se muestran las imágenes originales, la segunda cuando se obtiene un resultado de error de 3.0 usando RNA y la última fila un resultado de error de 13.87 utilizando PCA.

Figura 23. Comparación entre las reconstrucciones usando RNA y PCA

CAPÍTULO 6. CONCLUSIONES

El utilizar una técnica de inteligencia artificial como redes neuronales para la compresión de imágenes, generó imágenes reconstruidas tanto de buena como de mala calidad. Las imágenes de mala calidad resultantes se debieron al menor entrenamiento de las capas de la red neuronal, en contraste con las de buena calidad. Para la obtención de imágenes más semejantes a las originales, se debe de disminuir los resultados de error. Esto conlleva a dos notables desventajas: 1) el incremento del tiempo de entrenamiento y 2) la utilización de mayores recursos, sobretodo memoria. Aún así, los resultados obtenidos utilizando redes neuronales para comprimir imágenes, son mejores que los obtenidos con PCA

Con respecto a los recursos, la red neuronal necesitó ser implementada en una máquina de 3 Gb en RAM y un procesador Intel Core Duo, ya que se intentó hacerlo en computadoras Intel Pentium 4 de 256 Mb y 512 Mb, pero en éstas la red neuronal no logró ejecutarse porque consumía en su totalidad los recursos y terminaba bloqueándolas.

Con base en los resultados, se recomienda que para obtener imágenes de calidad aceptable en un tiempo "razonable", usar en la red neuronal la configuración donde la variable *maxepoch* del archivo *mnistdeepauto.m* equivalga a 10 y un valor de 5 para la variable *maxepoch* del archivo *backprop.m,* la cuál se puede modificar a 3 porque la reducción de error a partir de este valor es mínimo, tal y como se explicó en el capítulo 5, lo que reduciría un tiempo de ejecución a aproximadamente 2 horas.

Todos los objetivos planteados en la investigación fueron alcanzados. Como trabajo a futuro se propone modificar el código de la red neuronal para que ésta pueda ser entrenada con otros conjuntos de imágenes como galaxias o rostros, con las configuraciones ya probadas en esta investigación.

REFERENCIAS

1. Hinton G., Salakhutdinov R. 2006. Reducing the dimensionality of data with neural networks. *Science*. Vol. 313. 28 July No. 5786, pp. 504 - 507.

2. Nilsson N. 2001. Inteligencia artificial, una nueva síntesis. McGrawHill.

3. Gómez J.D. 2005. Comprensión de imágenes digitales utilizando redes neuronales: multicapa (backpropagation) y RBR's. Scientia et Technica. No. 29. Diciembre, pp 100-106.

4. Biblioteca de la Universidad de Cornell / Departamento de Investigación. 2000-2003. http://www.library.cornell.edu/preservation/tutorial-spanish/toc.html

5. Microsoft® Encarta®. 2006. Imagen de bits. Microsoft Corporation.

6. Ortega F. 2000-2009. http://www.fotonatura.org/revista/articulos/32/1/ . Fotonatura © Accesada el 7 de octubre del 2008.

7. Freeman J.A. , Skapura DM. 1993. Redes Neuronales. Algoritmos, aplicaciones y técnicas de programación. México. Addison-Wesley.

8. Hilera J., Martínez V. 1995. Redes neuronales artificiales: fundamentos, modelos y aplicaciones. Madrid. RA-MA.

9. Villanueva M. 2002. Las Redes neuronales artificiales y su importancia como herramienta en la toma de decisiones. Lima. Trabajo de Investigación - Universidad Nacional Mayor de San Marcos. Facultad de Ciencias Matemáticas. EAP de Investigación Operativa.

10. Smith L. 2002. A tutorial on Principal Components Analysis. John Wiley & Sons Inc.

11. Terrádez M. 2000. Análisis de componentes principales. MECD.

APÉNDICE A: PROCESO DE COMPRESIÓN

Este es un ejemplo completo del proceso de compresión donde se considera el valor 2 de la variable *maxepoch* del archivo *mnistdeepauto.m* y el valor de 5 para la variable *maxepoch* del archivo *backprop.m*

Se puede ver en las siguientes figuras cómo para cada época (en total 5 épocas debido al valor de *maxepoch* del archivo *backprop.m*) se muestra un resultado de error, así como la imagen respectiva resultante. La primer columna identifica a los dígitos originales y la segunda columna a los dígitos reconstruidos con el margen de error. En el primer caso, el resultado de error obtenido es 25.093, el cuál se puede ver en la elipse de la Figura A1.

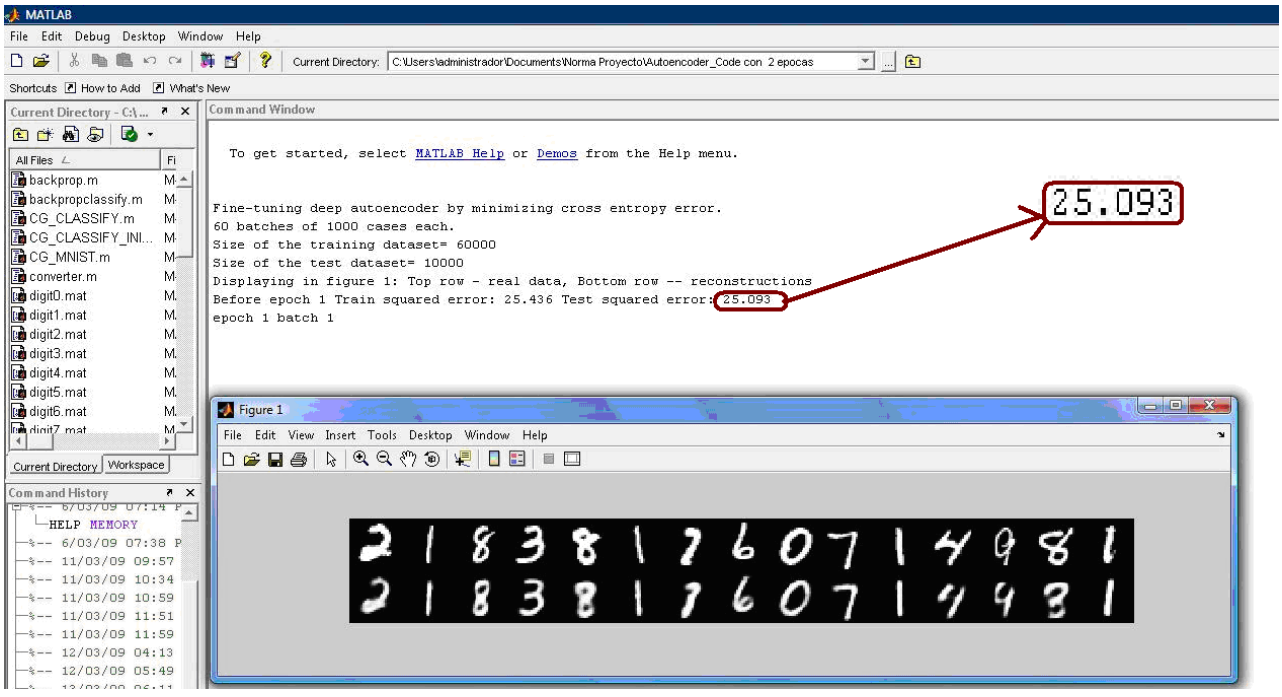

Figura A1. Resultado de error obtenido en la época 1

El error resultante de la época 2 fue de 10.515, el cual se muestra con su imagen respectiva en la Figura A2.

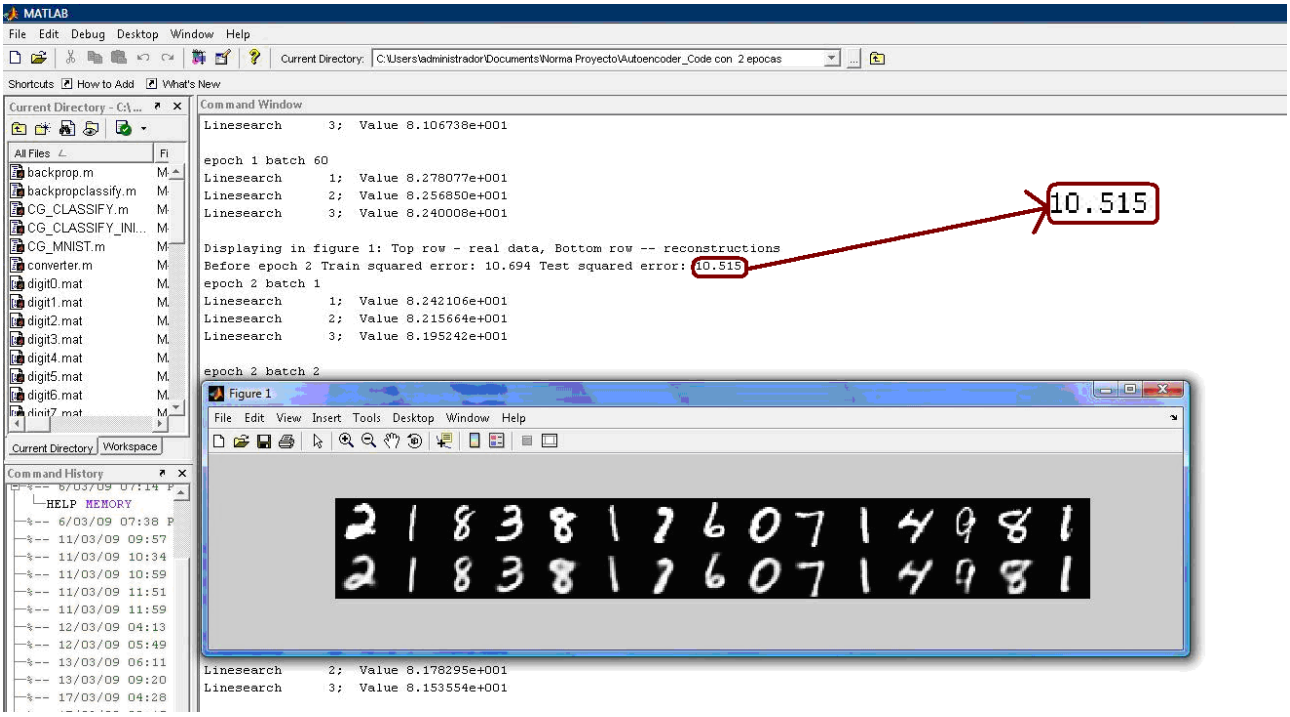

Figura A2. Resultado de error obtenido en la época 2

El tercer resultado de error fue de 9.544 tal como muestra la Figura A3.

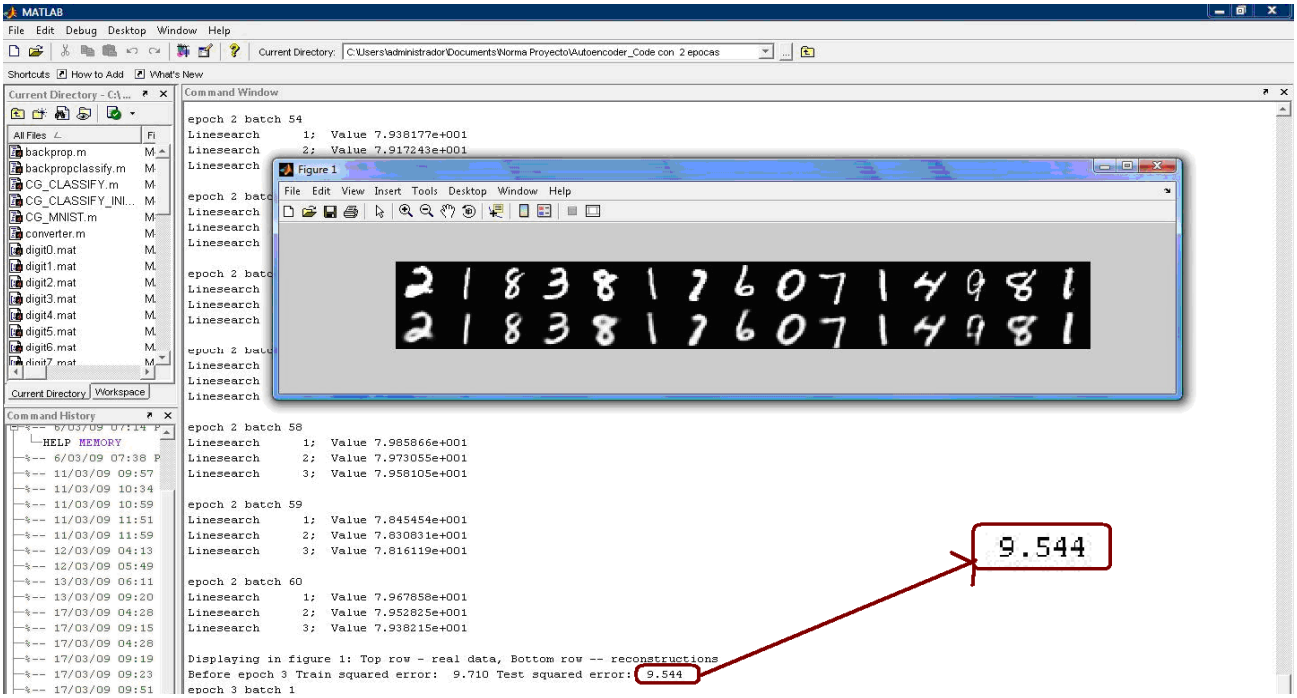

Figura A3. Resultado de error obtenido en la época 3

Para la época 4 se obtuvo un error de 9.1, se muestra la Figura A4.

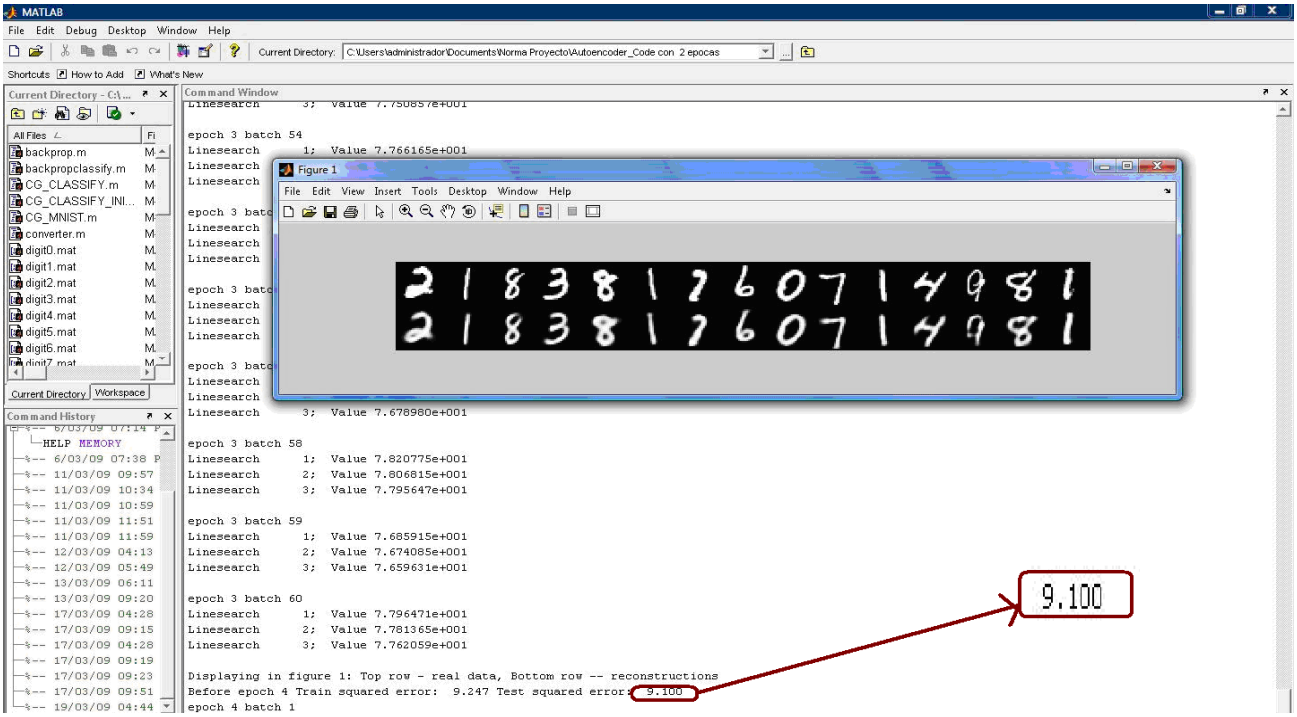

Figura A4. Resultado de error obtenido en la época 4

Finalmente, para la época 5 el error fue de 8.731, con el cual se termina el proceso de compresión. Este resultado se muestra en la Figura A5.

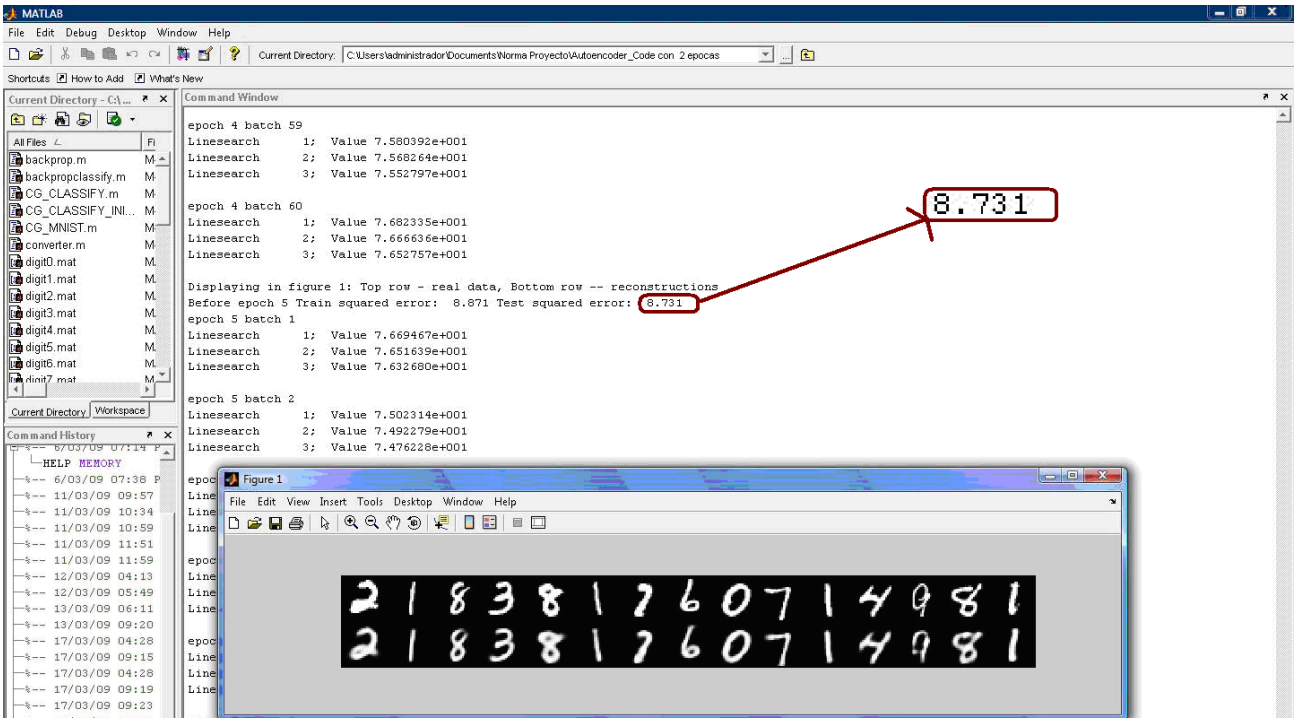

Figura A5. Resultado de error obtenido en la época 5

APÉNDICE B: COLECCIÓN DE IMÁGENES RESULTANTES

En este apéndice se puede observar y comparar la colección de imágenes obtenidas para cada valor de *maxepoch* del archivo *mnistdeepauto.m* en cada una de las 5 épocas de *backprop.m.* En cada figura, la primer fila corresponde a los dígitos originales y la segunda a los dígitos reconstruidos, siendo las figuras de la época 5 las que poseen una mejor calidad. La Tabla B1 muestra los resultados de error obtenidos para facilitar la comparación entre las imágenes y su respectivo error.

Tabla B1. Errores obtenidos en las diferentes épocas

Valor *maxepoch* del archivo *mnistdeepauto.m*	**Archivo backprop.m**				
	Época 1	**Época 2**	**Época 3**	**Época 4**	**Época 5**
1	35.438	12.528	11.194	10.468	10.008
2	25.093	10.515	9.544	9.100	8.731
6	17.382	8.150	7.511	7.154	6.909
10	13.04	6.996	6.472	6.141	5.926
20	11.61	6.344	5.853	5.583	5.395

Cuando el valor de *maxepoch* del archivo *mnistdeepauto.m* equivale a 1, las imágenes resultantes de las 5 épocas de *backprop.m* se muestran en las Figuras B1 a B5.

Figura B1. Imagen obtenida en la época 1 de *backprop.m* cuando *maxepoch* vale 1

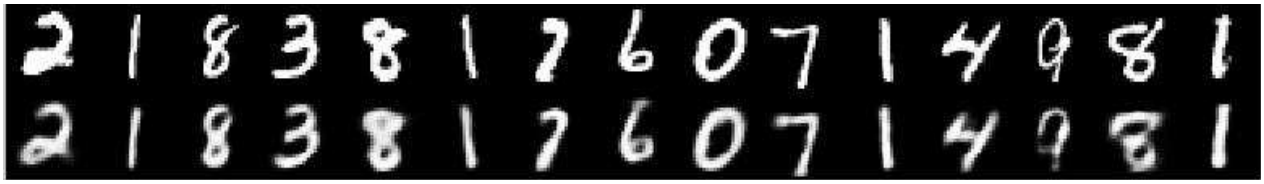

Figura B2. Imagen obtenida en la época 2 de *backprop.m* cuando *maxepoch* vale 1

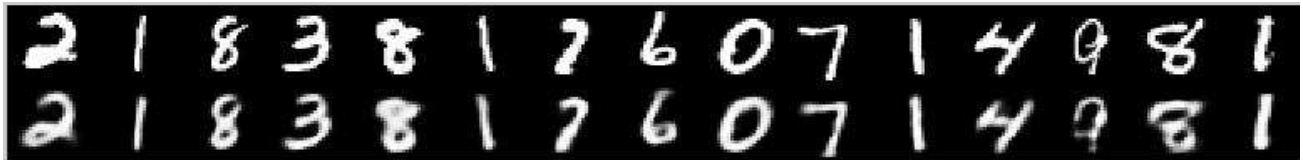

Figura B3. Imagen obtenida en la época 3 de *backprop.m* cuando *maxepoch* vale 1

Figura B4. Imagen obtenida en la época 4 de *backprop.m* cuando *maxepoch* vale 1

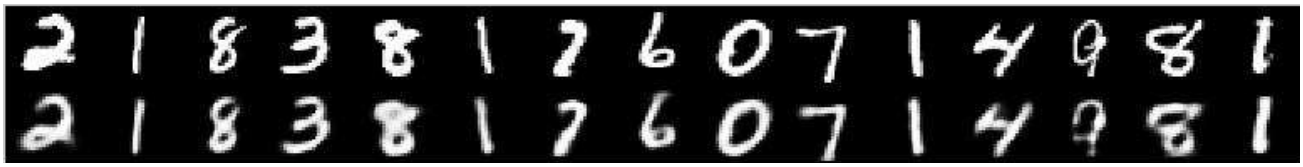

Figura B5. Imagen obtenida en la época 5 de *backprop.m* cuando *maxepoch* vale 1

Cuando el valor de *maxepoch* del archivo *mnistdeepauto.m* equivale a 2, las imágenes resultantes de las 5 épocas de *backprop.m* se muestran en las Figuras B6 a B10.

Figura B6. Imagen obtenida en la época 1 de *backprop.m* cuando *maxepoch* vale 2

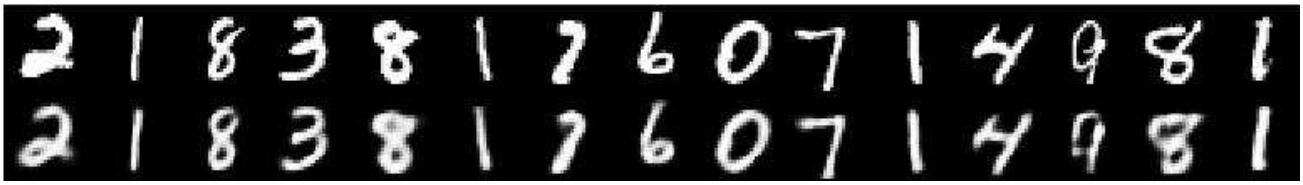

Figura B7. Imagen obtenida en la época 2 de *backprop.m* cuando *maxepoch* vale 2

Figura B8. Imagen obtenida en la época 3 de *backprop.m* cuando *maxepoch* vale 2

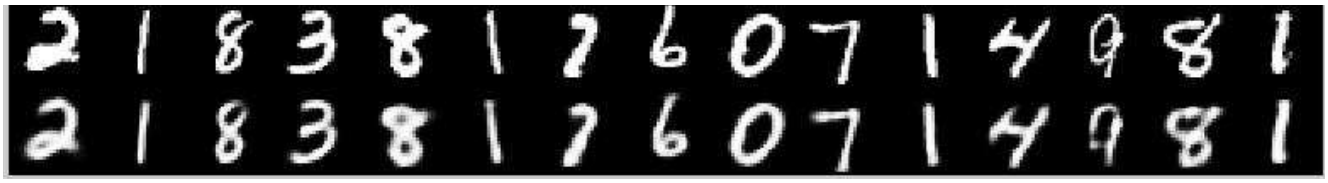

Figura B9. Imagen obtenida en la época 4 de *backprop.m* cuando *maxepoch* vale 2

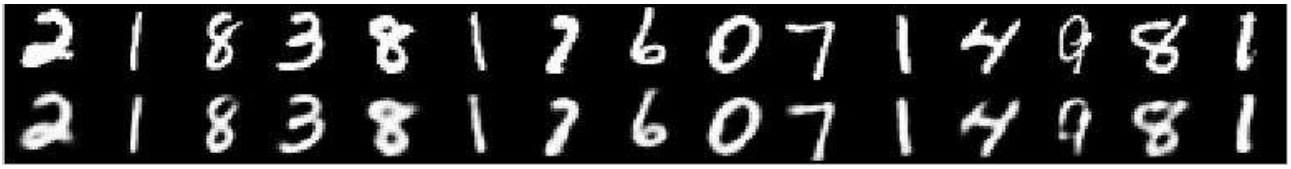

Figura B10. Imagen obtenida en la época 5 de *backprop.m* cuando *maxepoch* vale 2

Las Figuras B11 a B15 muestran las imágenes resultantes de las 5 épocas de *backprop.m*, cuando el valor de *maxepoch* del archivo *mnistdeepauto.m* equivale a 6.

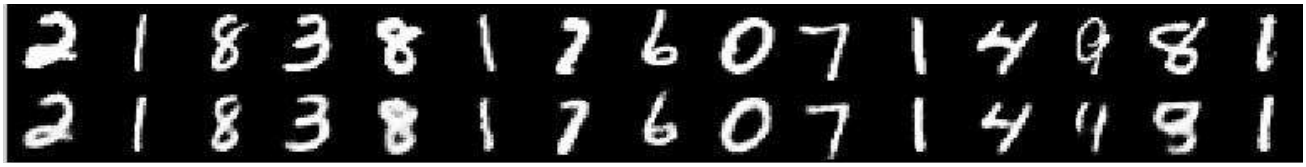

Figura B11. Imagen obtenida en la época 1 de *backprop.m* cuando *maxepoch* vale 6

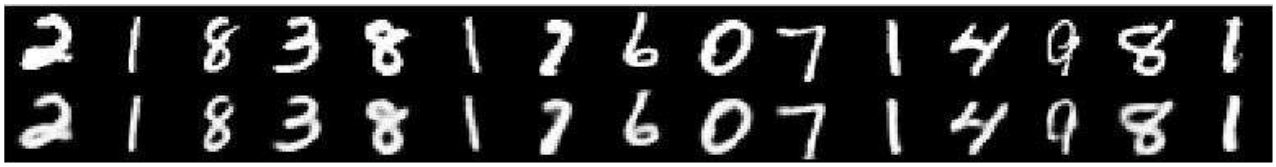

Figura B12. Imagen obtenida en la época 2 de *backprop.m* cuando *maxepoch* vale 6

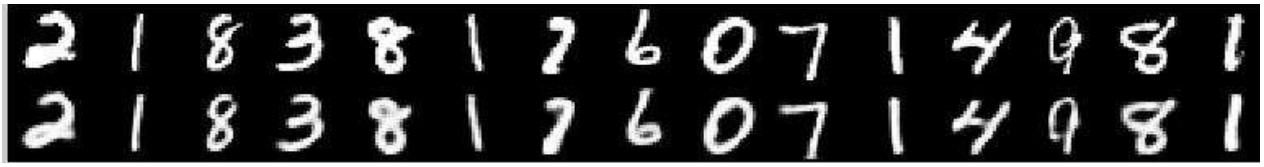

Figura B13. Imagen obtenida en la época 3 de *backprop.m* cuando *maxepoch* vale 6

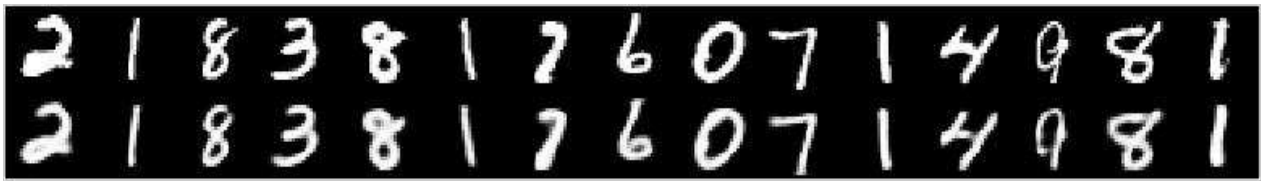

Figura B14. Imagen obtenida en la época 4 de *backprop.m* cuando *maxepoch* vale 6

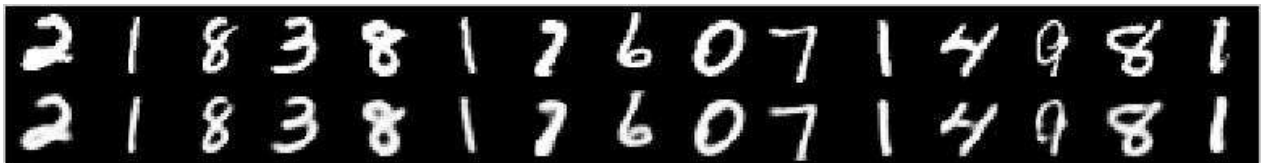

Figura B15. Imagen obtenida en la época 5 de *backprop.m* cuando *maxepoch* vale 6

Las Figuras B16 y B17 muestran las imágenes resultantes de la época 1 y 5 de *backprop.m*, cuando el valor de *maxepoch* del archivo *mnistdeepauto.m* equivale a 10.

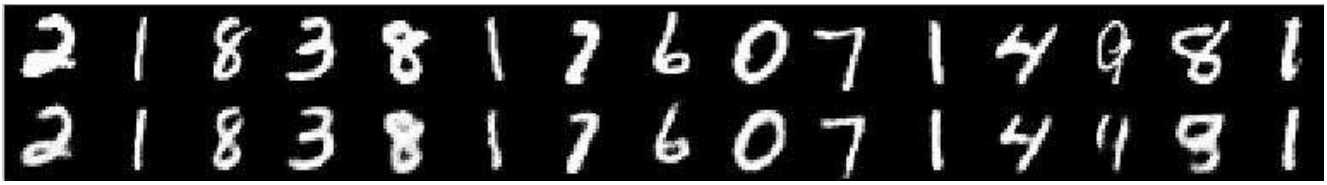

Figura B16. Imagen obtenida en la época 1 de *backprop.m* cuando *maxepoch* vale 10

Figura B17. Imagen obtenida en la época 5 de *backprop.m* cuando *maxepoch* vale 10

Se observa en las Figuras B18 y B19 las imágenes obtenidas de la época 1 y 5 de *backprop.m*, cuando el valor de *maxepoch* del archivo *mnistdeepauto.m* es igual a 20.

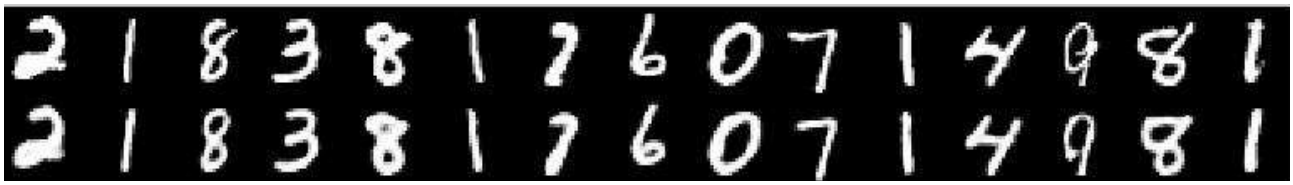

Figura B18. Imagen obtenida en la época 1 de *backprop.m* cuando *maxepoch* vale 20

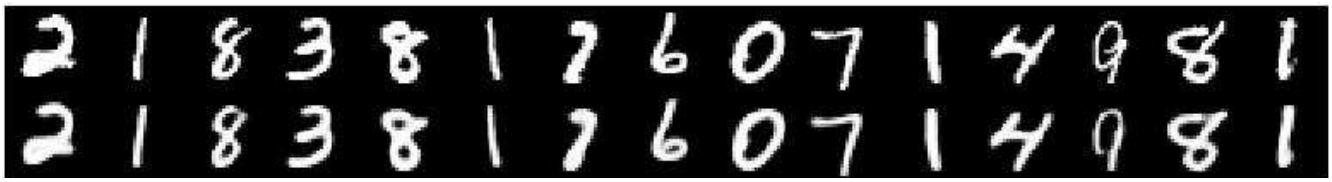

Figura B19. Imagen obtenida en la época 5 de *backprop.m* cuando *maxepoch* vale 20