Desempeño de una Red Neuronal Convolucional para Clasificación de Señales de Tránsito Vehicular

R. Vizcaya Cárdenas[1], J. M. Flores Albino, V. M. Landassuri Moreno y S. Lazcano Salas
Maestría en Ciencias de la Computación, Universidad Autónoma del Estado de México. Centro Universitario UAEM Valle de México., Km. 11.5 Carretera Atizapán de Zaragoza-Nicolás Romero S/N, México.
rvizcayac@uaemex.mx, jmfloresa@uaemex.mx, vmlandassurim@uaemex.mx, slazcanos@uaemex.mx

***Área de participación**: Sistemas Computacionales*

Resumen

El paradigma del **Deep Learnig**, o **Aprendizaje Profundo**, se ha beneficiado del incremento de información de la actualidad, así como del notable avance de las **Redes Neuronales Convolucionales** o **CNN's**. En los últimos cinco años, las CNN's han estado al frente en aplicaciones de reconocimientos de patrones usando imágenes o video, debido a las ventajas que tienen en comparación con otras técnicas; incluso, en algunos casos, llegando a superar la capacidad humana, como se muestra en el trabajo de Graham [2015]. En el presente trabajo se emplean señales de tránsito empleadas en México, para investigar el tiempo de entrenamiento y error de clasificación (desempeño) de una CNN de dos capas de convolución, el entrenamiento y prueba se llevada cabo en un CPU.

Palabras clave: *Redes N*euronales Convolucionales (CNN), Deep Learning, Reconocimiento de Patrones, Señales de Tránsito, Clasificación de Imágenes.

Abstract

The Deep Learning paradigm has benefited from the data increase of today. Convolutional Neural Networks or CNN's, are among the most widely used algorithms of this paradigm. In the last five years, CNN's have been leading in pattern recognition applications using images or video because of the advantages they have compared to other techniques; even beating human capacity, Graham [2015]. *Traffic signals employed in Mexico are utilized in this article to investigate the training time and classification error (performance) of a CNN. Where the architecture of the network has been fixed with two layers of convolution, and the training and testing are carried out on a CPU.*

Keywords: *Convolutional Neural Networks (CNN), Deep Learning, Pattern Recognition, Traffic Signs, Image Classification.*

Introducción

Desde los primeros trabajos sobre **Redes Neuronales Convolucionales** (**CNN**) tales como: LeCun, Bottou, Bengio, & Haffner [1998], se proyectaba que las mismas impactarían de manera notable sobre el desarrollo de la Inteligencia Artificial, concretamente en aplicaciones como el reconocimiento de patrones en imágenes; la primera CNN se diseñó para la tarea de clasificar dígitos escritos a mano (MNIST). Sin embargo, este desarrollo se ve limitado por la capacidad de cómputo disponible en aquellos años, además de que otras técnicas como las Máquinas de Soporte Vectorial en trabajos como Schölkopf, Burges, & Smola [1999], Vapnik [1995] o las llamadas redes neuronales superficiales Neal & Zhang [2006], presentaban resultados de buena calidad, razón por la que las CNN no se estudiaron de manera muy intensa. Sin embargo, no se abandonó la investigación en este y otros sub-campos con el **aprendizaje profundo** o **Deep Learning**, y en particular con las **Redes Neuronales Convolucionales** Simard, Steinkraus, & Platt [2003]. Para 2006, las **CNN's** ya presentaban mejores resultados en cuanto a la precisión y la tasa de error en las tareas de clasificación de dígitos escritos a mano, en comparación con otras técnicas, Ranzato, Poultney, Chopra, & LeCun [2007]. Las desventajas que presentaban fueron tanto en el tiempo de entrenamiento requerido, como en la implementación de la CNN para aplicaciones prácticas, esto debido a la cantidad de parámetros y cálculos a manejar. En ese mismo año se comenzó a utilizar los procesadores gráficos (GPU) para esos propósitos, mostrando un incremento de 4 veces en la velocidad, en comparación con un CPU convencional, Chellapilla, Puri, & Simard, [2006]. A partir de esos resultados, la implementación de los diferentes algoritmos de Deep Learning basados en GPU se volvió una constante Ranzato, Huang, Boureau, & LeCun [2007], Fernandez, Graves, & Schmidhuber [2007], Raina, Madhavan, & Ng, [2009], Ciresan D. C., Meier, Gambardella, & & Schmidhuber [2010], Ciresan D. C., Meier, Masci, Gambardella, & Schmidhuber [2011].

[1] Becario CONACYT con No. CVU 712316

En el caso del reconocimiento de patrones en general, sobre imágenes o video, las CNN dominan el estado del arte desde el 2012, siendo el parteaguas el trabajo realizado en Krizhevsky, Sutskever, & E. Hinton [2012]. En este último trabajo, se entrenó una CNN para clasificar 1.2 millones de imágenes de 227x227 pixeles, entre 1000 clases diferentes. La arquitectura usada para tal propósito utiliza 60 millones de parámetros y 650,000 neuronas, los autores reportan que el tiempo de entrenamiento, fue de poco más de cinco días empleando dos GPUs (tarjetas NVIDIA GTX 580 de 3GB en RAM). Esta red se sometió por primera vez (empleando CNN) en el ImageNet Challenge LSVRC-2010, superando los mejores resultados obtenidos hasta ese entonces con métodos tradicionales. En la **Tabla 1** se muestra parte de dichos resultados.

Tabla 1. Comparación de resultados en el ImageNet Challenge LSVRC-2010, Krizhevsky et al. [2012]

Modelo	Tasa de error Top-1	Tasa de error Top-5
Sparse coding	47.1%	28.2%
SIFT + FVs	45.7%	25.7%
CNN	37.5%	17.0%

En general, esa tendencia de mejora continua con CNN's se ha conservado hasta la actualidad. Las categorías en las que se compite anualmente en el ImageNet Challenge, son: clasificación, detección y localización de objetos en imágenes y video. Los últimos resultados reportados, competencia de 2017, Russakovsky, y otros [2017], indican al equipo ganador con un porcentaje de error en la localización de 6.19% y en la tarea de clasificación de 2.711%. Existen otras competencias similares, como MINIST, CIFAR-10, CIFAR-100, STL-10 y SVHN por mencionar algunas. En el caso de Reconocimiento de Objetos en Imágenes CIFAR-10, compitieron 231 equipos en el último concurso y el equipo ganador implementó una CNN que alcanzó una exactitud de 96.53%, que supera al ser humano estimado en 94% aproximadamente para esta tarea. En dicho concurso, se trata de clasificar 60,000 imágenes a color de 32x32 pixeles en 10 clases; Benenson [2017] muestra los resultados de estas competencias, donde los equipos ganadores utilizan CNN's como principal algoritmo. Otras tareas en las que el estado del arte es dominado por CNN's son: descripción de escenas, clasificación de video, seguimiento de objetos en video, entre otras. Del 2012 a la fecha se ha incrementado la investigación en cuanto a CNN se refiere. En la **Tabla 2** se muestra una comparación de las redes más populares publicadas y sus características.

Tabla 2. Comparación entre diferentes Redes Neuronales Convolucionales.

	# Capas de Convolución	MACCs [x 10^6]	Parámetros [x 10^6]	Activaciones [x 10^6]	ImageNet top-5 error %
AlexNet (2012)	5	1140	62.4	2.4	19.7
Network-in-Network (2013)	12	1100	7.6	4.0	19.0
VGG-16 (2014)	16	15470	138.3	29.0	8.1
GoogLeNet (2015)	22	1600	7.0	10.4	9.2
ResNet-50 (2015)	50	3870	25.6	46.9	7.0
Inception v3 (2016)	48	5710	23.8	32.6	5.6
Inception-ResNet-v2 (2016)	96	9210	31.6	74.5	4.9
SqueezeNet (2016)	18	860	1.2	12.7	19.7

Se observa de la **Tabla 2** que, con los años, se fue incrementando la cantidad de "capas de convolución" empleadas, esto trajo como resultado un mejor desempeño en cuanto al error, pero obviamente un incremento en la cantidad de operaciones multiplicación-acumulación (MACC) necesarias. Aquí solo se muestra la cantidad de capas de convolución que contienen las redes, no así las demás capas, donde la columna de activaciones indica la cantidad de funciones de salida de las neuronas, y en la última columna se muestra el porcentaje de error en la clasificación.

Está claro que las CNN's han tenido un notable éxito en los últimos cinco años, y las podemos encontrar en casi cualquier aplicación de reconocimiento de patrones que involucre imágenes o visión artificial. Una característica importante de estas, es que requieren de varias capas de convolución (entre otras capas), una gran cantidad parámetros y de operaciones a implementar, de ahí la necesidad de usar GPU's u otros dispositivos capaces de procesar en paralelo el cómputo requerido, para poder ser implementas.

El principal objetivo del presente documento es evaluar el desempeño, en cuanto al tiempo de entrenamiento y al error en la clasificación de señales de tránsito vehicular, usadas en México, de una CNN que cuenta con sólo dos capas de convolución y dos de agrupamiento. Lo que se pretende con esto es, determinar si es posible

emplear Redes Neuronales Convolucionales con menor cantidad de capas, que las reportadas en el estado del arte, y aun así obtener resultados satisfactorios al entrenarlas y probarlas en un CPU. El entrenamiento y prueba de la red, se hace en un CPU Quad-Core Intel Xeon E5 a 3.7 GHz y 12 GB de memoria RAM, en lugar de algún otro dispositivo con el cuál acelerar el proceso. El banco de imágenes de señales de tránsito se generó a partir de imágenes obtenidas de la página de la Secretaría de Comunicaciones y Transportes, Secretaria [2017]. A este conjunto inicial, se les hizo un tratamiento para obtener variantes de las mismas y con ello conseguir un conjunto de 8,000 imágenes en total. Estas imágenes se dividieron en ocho clases. La separación de los conjuntos de entrenamiento y prueba se hizo de forma aleatoria, se consideró un criterio de 80% de muestras para entrenamiento y 20% para prueba.

Redes Neuronales Convolucionales

Las redes neuronales convolucionales son una extensión del perceptrón multicapa, con la diferencia de que las CNN's realizan operaciones de "convolución" entre los parámetros y los datos de la red, en lugar de productos punto. Son apropiadas para aplicaciones en las que los datos se encuentran en forma de una rejilla, como matrices, por ejemplo. A diferencia de una Multi Layer Perceptron (MLP), las CNN procesan las imágenes por secciones y han tenido un notable éxito, ver, Lin, Chen, & Yan [2013], Chatfield, Simonyan, Vedaldi, & Zisserman [2014], Szegedy y otros [2015], He, Zhang, Ren, & Sun [2015], Szegedy, Vanhoucke, Ioffe, & Shlens [2015], Szegedy, Ioffe, & Vanhoucke, [2016]. Debido a su arquitectura, que opta por expandir una arquitectura bidimensional de una MLP, por una tridimensional de las CNN's, es que son utilizadas en aplicaciones que involucran visión artificial. El principio es manejar los datos que representan los pixeles en forma de volúmenes, y no como vectores. Las capas básicas para que funcione cualquier CNN son tres: capas de convolución, junto con su función de activación, capas de agrupación y las capas completamente conectadas. En la **Figura 1** se muestra la arquitectura básica de una red neuronal convolucional, en esta, la entrada es una imagen a color de 254x254 pixeles. Le siguen las capas que integran la CNN, se muestran las capas de convolución, agrupamiento, las capas completamente conectadas y la capa de salida.

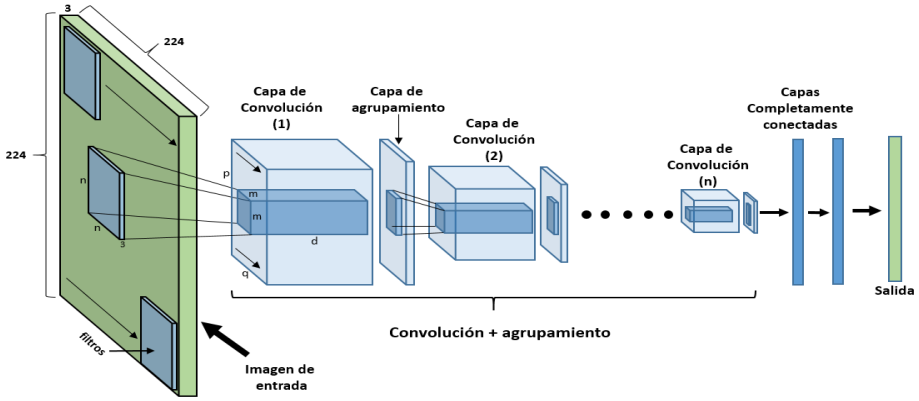

Figura 1. Arquitectura básica de una red neuronal convolucional.

Dependiendo de la aplicación, las últimas capas pueden variar. Para el caso de la tarea de clasificación de imágenes, en la última capa se tendría la misma cantidad de nodos que la cantidad de clases, un nodo por cada clase, los cuáles se activarían con diferente intensidad, indicando el grado de pertenencia de la instancia a la clase. Estos nodos están completamente conectados a cada nodo de la capa anterior.

Capas de convolución

En este concepto de basa el fundamento de las CNN. La operación de convolución, para nuestro caso, es digital y de dos dimensiones, queda definida como:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \qquad (1)$$

Donde I representa la imagen y K un filtro o kernel de tamaño *m x n*. Por lo regular *m = n*, de ahí que el tamaño del filtro representado en la **Figura 1** es de *n x n*. Los subíndices *i, j* representan la posición de los pixeles en la imagen sobre la que se hace la operación de convolución. Los filtros consisten en conjuntos de parámetros, típicamente de tamaños: 3x3, 5x5, 7x7 u 11x11, además de que cada filtro debe tener la profundidad de los datos de entrada, por ejemplo, si la entrada es una imagen a color, la profundidad de cada filtro sería tres, que

corresponden a los canales (**RGB**). Estos filtros se desplazan barriendo toda la imagen de entrada, desde la esquina superior izquierda, hasta la esquina inferior derecha, como se observa en la **Figura 1**. Conforme se hace el desplazamiento, se van haciendo las operaciones de producto punto entre los valores de los pixeles de la imagen y los que corresponden a los parámetros de los filtros. De la operación de convolución, resulta otro grupo de datos en forma de un volumen, de tamaño (*p* ***x*** *q* ***x*** *d*). Donde *d*, corresponde a la cantidad de filtros usados en la capa anterior. A estos datos se le puede repetir el mismo procedimiento, por varias capas, con nuevos conjuntos de filtros (*m* ***x*** *m* ***x*** *c*), donde *c* corresponde a la cantidad de filtros de la capa anterior. En el transcurso de estas capas, la dimensión de los cubos se va reduciendo en cuanto al ancho y altura, pero la profundidad se conserva igual a la cantidad de filtros usados en la capa anterior.

El propósito de los filtros es producir mapas de activación, o de características, los cuáles se activarán al pasar los filtros por regiones de la imagen que contengan las características buscadas, como bordes, líneas, contornos, etc. El fundamento al entrenar este tipo de redes consiste en actualizar los parámetros de esos filtros, de forma que cada filtro extraiga las correspondientes características buscadas.

En cuanto a las dimensiones del volumen de los datos de salida para cada capa, depende de cuatro factores: **la *cantidad de filtros*** (K) usados en esta capa, ***el paso*** del filtro (S), el ***relleno de ceros*** (P) y el ***campo visual*** (F) del filtro. La *profundidad* de los datos en la salida corresponde a la cantidad de filtros que se está empleando. El *paso* del filtro se refiere a la cantidad de pixeles que se desplazará el filtro al moverse por la imagen. En cuanto al relleno de ceros, en ocasiones se debe rellenar los bordes de una imagen con ceros, de forma que el tamaño de la imagen se ajuste al tamaño de los filtros y se pueda hacer el barrido completo. Si los datos de entrada a una capa de convolución son: $W_1xH_1xD_1$, que corresponden al ancho, altura y profundidad de los datos respectivamente, el volumen de datos en la salida queda determinado por las ecuaciones 2 a 4, para el ancho, altura y profundidad, respectivamente.

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1 \qquad (2)$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1 \qquad (3)$$

$$D_2 = K \qquad (4)$$

Capas de agrupamiento

El propósito de estas capas es reducir el tamaño espacial de los datos progresivamente, con el fin de reducir la cantidad de parámetros a tratar y consecuentemente la cantidad de cálculos, ayudando a prevenir el overfitting (demasiados parámetros). Estas capas, operan de forma independiente a los datos de entrada de la capa anterior, sólo reducen el tamaño espacial de los datos usando una operación *MAX*, que consiste en dividir los datos en secciones, para extraer los valores máximos de cada sección, discriminando los demás. O bien extrayendo el promedio de este grupo de datos. Estas capas reciben un volumen de entrada de $W_1xH_1xD_1$, que corresponden al ancho, altura y profundidad de los datos de entrada. Producen un volumen de salida $W_2xH_2XD_2$, que corresponden al ancho, altura y profundidad del volumen de salida, estos quedan determinados por las ecuaciones (5) a (7), respectivamente.

$$W_2 = \frac{(W_1 - F)}{S} + 1 \qquad (5)$$

$$H_2 = \frac{(H_1 - F)}{S} + 1 \qquad (6)$$

$$D_2 = D_1 \qquad (7)$$

La **Figura 2** muestra la forma de llevar a cabo esta operación. Para el ejemplo de la figura, a la izquierda se muestra un volumen de datos de entrada de *24x24x6*, y como resultado se obtiene un volumen de *12x12x6*. A la derecha se muestra un ejemplo con la operación *MAX*, como se observa en la figura, $W_1 = 4$, $H_1 = 4$, $F = 2$ y $S = 2$ (algo muy común en la práctica). Según las ecuaciones 5 y 6, W_2 y H_2, que corresponden al ancho y altura de salida respectivamente, es 2. Como se observa, el tamaño espacial de los datos permanece igual en cuanto a la profundidad de los mismos se refiere, pero se reducen en cuanto al ancho y altura.

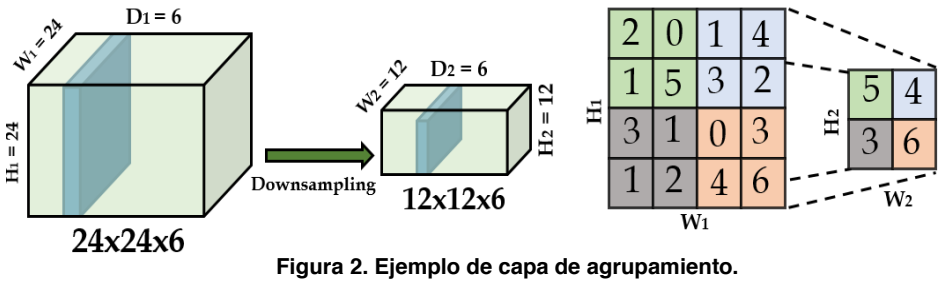

Figura 2. Ejemplo de capa de agrupamiento.

Capas completamente conectadas

Estas funcionan igual que en un MLP, es decir, cada nodo se encuentra completamente conectado a las salidas de la capa anterior. Cada nodo hace una suma ponderada de sus parámetros por el valor de entrada, además de la entrada independiente o bias. Con ello, el que tenga la mayor puntuación, será el que tenga una mayor probabilidad. En sí, hacen una clasificación indicando la probabilidad de cada clase.

Entrenamiento y prueba de la CNN

Para nuestro caso de estudio, el banco de imágenes de señales de tránsito se tomó de la página de la Secretaría de Comunicaciones y Transportes. Estas imágenes se encuentran en un formato PNG a color, con un tamaño de 1249x1249 pixeles. Se acondicionaron las imágenes para adecuarlas a la CNN. El acondicionamiento consistió en pasarlas a escala de grises y escalarlas a 28x28 pixeles. Además, para tener una base de datos lo suficientemente grande y con diversidad, se generaron más imágenes, a partir de las que se tenían. Estas nuevas imágenes se formaron agregando ruido de Poisson y de sal y pimienta al 10% a las originales, rotarlas a 90, 180 y 270 grados, y aplicando un proceso de dilatación. El resultado final es una base de datos con 8,000 imágenes y de ocho clases diferentes. Dado que las imágenes de entrada a la CNN se encuentran en escala de grises, el volumen de entrada en nuestro caso es de 28x28x1. Se emplean seis filtros (K=6) de 5x5, no habrá relleno de ceros, por lo que P = 0 en las ecuaciones 2 y 3. El paso del filtro (S) es 1. Con esto, el volumen de salida será de 24x24x6, según las ecuaciones 2 a 4. Las capas de agrupamiento, o “downsampling”, que resultan de las de convolución serán de 12x12x6, según las ecuaciones 5 y 6 (F= S = 2). El proceso se repite con las capas resultantes para obtener nuevos volúmenes de 8x8x12 y 4x4x12, respectivamente. En la **Figura 3** se muestra el proceso y la arquitectura de la CNN implementada. Tanto para la generación de imágenes, como para el entrenamiento y prueba de la CNN, se empleó el software Matlab R2017a® como lenguaje de programación, dada la facilidad de trabajar con imágenes, operaciones con matrices y el toolbox de Deep Learning, Berg Palm [2017].

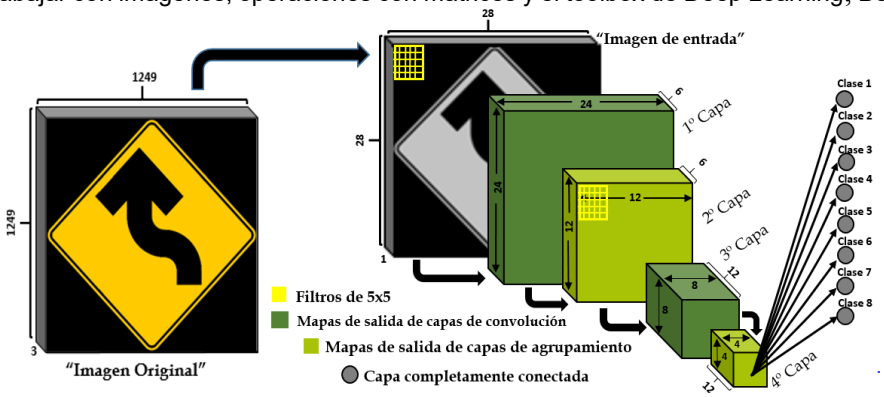

Figura 3. Arquitectura implementada.

Resultados

Para ver el desempeño de la CNN se hicieron varios experimentos. Estos consisten en entrenar la red durante diferentes cantidades de épocas, esto con el propósito de medir el tiempo de entrenamiento y la exactitud para cada caso. El entrenamiento se hace por lotes, para ver el efecto de variar el tamaño del lote que se le presentan a la red durante el entrenamiento, los experimentos anteriores se hicieron para lotes de 32, 100 y 200 muestras, dejando la tasa de aprendizaje fija en todos los casos. En la tabla 3 se muestran los resultados obtenidos hasta 2200 épocas de entrenamiento y para cada tamaño de lote. En el caso de los experimentos con 32 en el tamaño del lote, significa que cada 200 lotes se la muestra el conjunto de 6400 muestras de entrenamiento a la CNN, con

lo que se tendría una época de entrenamiento. Para 100 en el tamaño del lote, una época de entrenamiento será cada 64 lotes, y en el caso de 200 cada 32 lotes.

Tabla 3. Resultados obtenidos hasta 2200 épocas.

# de Épocas	Exactitud (%)			Tiempo de entrenamiento (Segundos)			Tiempo Ent./Época (Segundos)		
	L32	L100	L200	L32	L100	L200	L32	L100	L200
200	82.94	73.50	64.25	1623.1	1281.2	**1200.98**	8.12	6.41	**6.00**
600	89.69	84.06	77.56	4870.2	3854.7	**3620.9**	8.12	6.42	**6.03**
1000	88.50	85.75	83.81	8116.3	6420.8	**6032.5**	8.12	6.42	**6.03**
1400	**91.13**	88.75	86.19	**11361.5**	8993.9	**8454.8**	8.12	6.42	**6.04**
1800	89.56	88.50	86.81	14610.4	11554.4	**10887.5**	8.12	6.42	**6.05**
2200	90.19	**89.62**	**88.50**	17842.2	14110.0	**13229.4**	8.11	6.41	**6.01**

En la **Figura 4** se muestra el comportamiento del Error Cuadrático Medio obtenido durante el entrenamiento, para el experimento de 2200 épocas y 100 en el tamaño del lote. Ese mismo comportamiento se observa en los demás casos de experimentación. En la **figura 5** se muestra la tendencia del error en la clasificación al probar la CNN para cada experimento realizado.

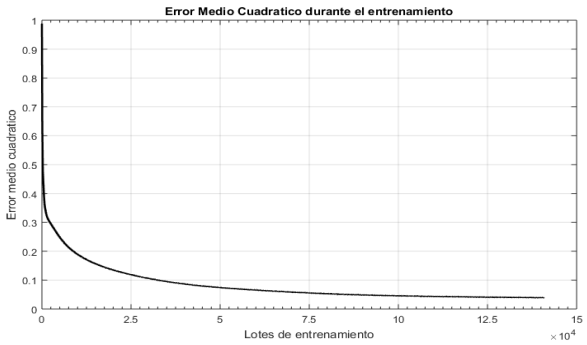

Figura 4. Error Cuadrático Medio obtenido durante el entrenamiento, para el experimento de 2200 épocas y lote de 100 muestras.

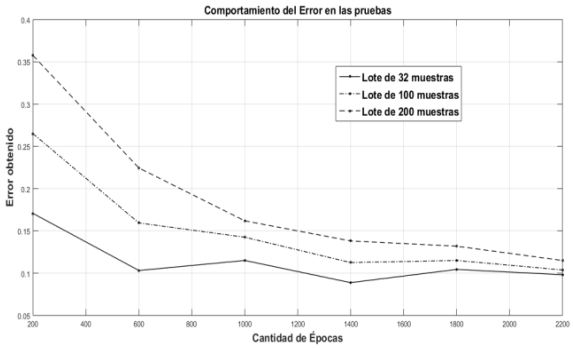

Figura 5. Tendencia de error en la clasificación para cada experimento.

Con el propósito de ver el desempeño de la red para una mayor cantidad de épocas, se hizo un experimento más, el tamaño de los lotes fue de 200 y la cantidad de épocas de 3800. La tendencia del Error Cuadrático Medio fue la misma que la mostrada en la **figura 4**, la exactitud que se alcanzó en este último experimento fue de 90.19%, en un tiempo de 22,947.17 segundos (6.37 horas), un promedio de 6.04 segundos por época.

Los tamaños de los lotes se eligieron tomando en cuenta la cantidad de imágenes de entrenamiento (6400), de la **tabla 3** se observa que con forme se reduce el tamaño del lote, se mejora la exactitud en la clasificación, pero se incrementa el tiempo de entrenamiento. Esto sugiere que para tamaños de 16, 8 o 4 muestras en los lotes, se obtendría un mejor desempeño en cuanto a la exactitud, pero tiempos de entrenamiento más grandes. En cuanto al comportamiento del Error Cuadrático Medio, es el esperado al entrenar cualquier red neuronal.

Conclusiones

El presente trabajo se llevó a cabo con el propósito de evaluar si es posible implementar Redes Neuronales Convolucionales, que tengan pocas capas de convolución y aun así obtener resultados adecuados en la tarea que realizan. Esto tiene la ventaja de poderlas implementar con dispositivos que no requieran de un alto consumo de energía, como es el caso de los GPU's, además de un tamaño reducido. Lo que sería ideal para cualquier aplicación embebida. El error en la clasificación más bajo que se obtuvo fue de 8.87%, cuando el tamaño del lote fue de 32 y el experimento de 1400 épocas (en 3.16 horas de entrenamiento), lo que significa que de cada 100 imágenes de tránsito vehicular que se le presenten a la CNN, esperamos que aproximadamente 91 de ellas las clasifique correctamente. El menor tiempo de entrenamiento se obtiene en los experimentos de 200 en el tamaño del lote, alcanzando 90.19% de exactitud en el experimento de 3800 épocas. Aunque este tipo de problemas

están prácticamente resueltos con CNN's (empleando GPUs), según los resultados que se muestran en Benenson [2017], no se puede hacer una comparación directa contra el presente trabajo, debido, por un lado, al tipo y cantidad de imágenes que se tratan en cada caso y, sobre todo, la cantidad de capas de convolución (y otras más) que se manejan. Una referencia directa que si pudiera ser un punto de comparación es con Berg Palm [2017], donde se maneja la misma arquitectura, sólo que ahí la tarea es clasificar imágenes de dígitos escritos a mano (MNIST), el autor reporta una exactitud de 98.8%. La diferencia que se observa puede ser explicada por la mayor diversidad que se tiene, en cuanto a las imágenes que pertenecen a una misma clase. Por ejemplo, para este trabajo, imágenes de vuelta a la izquierda, vuelta a la derecha, un entronque y otras, pertenecen a la misma clase. O los límites de velocidad de 70, 80, 90, 100 km/hr, pertenecen a la misma clase que una señal de stop, o de no rebasar, por ejemplo. Otro factor importante por investigar, es la cantidad de imágenes de entrenamiento en cada caso (6400 contra 60,000), en la literatura se menciona que el desempeño de una CNN mejora con forme se incrementa la cantidad de información. En cuanto a los tiempos de entrenamiento que se obtuvo en los experimentos, son congruentes con la plataforma que se empleó para este propósito y la configuración de la red obtenida pudiera implementarse en un dispositivo con poca capacidad, en cuanto a los recursos computacionales.

Trabajo futuro

Dada la discusión presentada aquí, existen diversos trabajos pendientes, como el incrementar la cantidad de capas de convolución de la CNN y ver la diferencia en el desempeño, así como incrementar la cantidad y variabilidad de las imágenes. Otro punto, es probar la red con imágenes tomadas en avenidas o autopistas que contengan señales de tránsito, así como probar la configuración obtenida en un sistema embebido.

Referencias

1. Benenson, R. (06 de 03 de 2017). *What is the class of this image ? Discover the current state of the art in objects classification.* Obtenido de Classification datasets results: http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#494c5356524332303132207461736b2031
2. Berg Palm, R. (20 de Mayo de 2017). *GitHub Repository.* Recuperado el 12 de Mayo de 2017, de GitHub Repository: https://github.com/rasmusbergpalm/DeepLearnToolbox
3. Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (4 de Septiembre de 2014). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. *arXiv:1409.1556v6*, 1-14. Obtenido de https://arxiv.org/abs/1409.1556v6
4. Chellapilla, K., Puri, S., & Simard, P. (2006). High performance convolutional neural networks for document processing. En *Tenth International Workshop on Frontiers in Handwriting Recognition.* La Baule, France: Guy Lorette. Obtenido de https://hal.inria.fr/inria-00112631/file/p1038112283956.pdf
5. Ciresan, D. C., Meier, U., Gambardella, L. M., & & Schmidhuber, J. (2010). Deep big simple neural nets for handwritten digit recogntion. *Neural Computation,, 22*(12), 3207-3220.
6. Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (Julio de 2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 1237-1242. doi:10.5591/978-1-57735-516-8/IJCAI11-210
7. Fernandez, S., Graves, A., & Schmidhuber, J. (2007). Sequence Labelling in Structured Domains with Hierarchical Recurrent Neural Networks. En *Proceedings of the 20th International Joint Conference on Artificial Intelligence.*
8. Graham, B. (2015). Fractional Max-Pooling. *arXiv:1412.6071v4 [cs.CV]*, 1-10.
9. He, K., Zhang, X., Ren, S., & Sun, J. (10 de Diciembre de 2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385v1*, 1-12. Obtenido de https://arxiv.org/abs/1512.03385
10. Krizhevsky, A., Sutskever, I., & E. Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25*, 1097-1105.
11. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 2278--2324.
12. Lin, M., Chen, Q., & Yan, S. (16 de Diciembre de 2013). Network In Network. *arXiv:1312.4400v3*, 1-10. Obtenido de https://arxiv.org/abs/1312.4400
13. Neal, R. M. (2006). Classification with Bayesian neural networks. *Lecture notes in computer science, 3944*, 28-32.

14. Neal, R. M., & Zhang, J. (2006). High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. En *Feature Extraction. Studies in Fuzziness and Soft Computing* (Vol. 207, págs. 265-296). Berlin, Heidelberg: Springer. doi:https://doi.org/10.1007/978-3-540-35488-8_11
15. Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale Deep Unsupervised Learning using Graphics Processors. *Proceedings of the 26th annual International Conference on Machine Learning*, 873-880.
16. Ranzato, M. A., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *In Proc. computer vision and pattern recognition conference* , 1-8.
17. Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems 19*, 1137-1144. Obtenido de http://papers.nips.cc/paper/3112-efficient-learning-of-sparse-representations-with-an-energy-based-model.pdf
18. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., . . . Fei-Fe, L. (27 de Mayo de 2017). *Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)*. Obtenido de http://image-net.org/challenges/LSVRC/2017/results
19. Schölkopf, B., Burges, C. J., & Smola, A. J. (1999). *Advances in kernel methods: support vector learning.* Cambridge, MA, USA: MIT Press.
20. Secretaria, d. C. (12 de Mayo de 2017). *SCT*. Obtenido de Banco Digital de Señalizaciòn Vial: http://www.sct.gob.mx/bancodigital/
21. Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Seventh international conference on document analysis and recognition*, (págs. 958-963).
22. Szegedy, C., Ioffe, S., & Vanhoucke, V. (23 de Febrero de 2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261v2*, 1-12. Obtenido de https://arxiv.org/abs/1602.07261
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going Deeper with Convolutions. *IEEE Xplore "Open Access version, Computer Vision Fundation"*, 1-9.
24. Szegedy, C., Vanhoucke, V., Ioffe, S., & Shlens, J. (2 de Diciembre de 2015). Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567*, 1-10. Obtenido de https://arxiv.org/abs/1512.00567
25. Vapnik, V. N. (1995). *The nature of statistical learning theory.* New York: Springer.