Reconocimiento de dígitos en placas vehiculares usando una red neuronal convolucional (CNN)

Yves Maillard*, Lorena Nuñez† *Posgrado en Ingeniería eléctrica, Facultad de Ingeniría, UNAM, CDMX, México
†Posgrado en Ingeniería eléctrica, Facultad de Ingeniería, UNAM, CDMX, México

***Abstract*—En este trabajo se implementa la arquitectura de una red neuronal convolucional (CNN) que permite reconocer dígitos manuscritos y se prueba para el reconocimiento de dígitos presentes en placas vehiculares. Para el desarrollo e implementación de la CNN se utilizó el lenguaje de programación Python y la biblioteca PyTorch para aprendizaje profundo. Para la etapa de entrenamiento de la CNN se utilizaron imágenes presentes en el conjunto de entrenamiento de la base de datos MNIST y para la etapa de prueba se tomaron imágenes presentes en el conjunto de prueba de MNIST y otras obtenidas a partir de imágenes de placas vehiculares. Para la obtención y preparación de las imágenes que se ingresaron a la red se implementaron diversos algoritmos de Procesamiento Digital de imágenes.**

***Index Terms*—Procesamiento digital de imágenes, Red convolucional, segmentación, crecimiento de regiones, lectura automática.**

I. Objetivo

Implementar un reconocedor de dígitos en imágenes de placas vehiculares empleando para ello una red neuronal convolucional (CNN) desarrollada en un lenguaje de programación interpretado y entrenada inicialmente para reconocer dígitos manuscritos.

II. Introducción

CON el drástico aumento de vehículos motorizados presentes en las ciudades, gestionarlos de forma segura y eficiente se convirtió en un gran problema. Para resolverlo, la tecnología de reconocimiento de placas se ha convertido en una tendencia, ya que su uso se encuentra en aplicaciones diversas como el control de acceso a estacionamientos, cobro automático en autopistas, detección de vehículos robados, entre otros.

Las secciones precedentes describen la metodología utilizada y los resultados obtenidos para un clasificador de dígitos utilizando una CNN. Inicialmente la CNN se entrena para reconocer imágenes con dígitos manuscritos de la base de datos MNIST, pero posteiormente se prueba con imágenes de dígitos y letras provenientes de imágenes de placas vehiculares.

El reconocedor de dígitos implementado en este trabajo implicó diversas técnicas y algoritmos de Procesamiento Digital de Señales. Entre ellas la segmentación y el uso de una CNN. El algoritmo de segmentación utilizado, una descripción de las CNN y el Dataset empleado para entrenar y probar la CNN utilizada se presentan a continuación:

A. Segmentación de imágenes

La segmentación consiste en encontrar y diferenciar diversas partes, objetos o secciones de interés dentro de una imagen [1]. Existen diversos métodos y algoritmos para realizar la segmentación de una imagen; uno de ellos y el utilizado en este trabajo es el método de *segmentación por crecimiento de regiones*.

La *segmentación por crecimiento de regiones* consiste en establecer uno o un conjunto de pixeles denominados "semilla" e ir anexando pixeles vecinos a la semilla para ir creciendo una región hasta que se convierte en la región de interés que se desea segmentar. En un principio se toma la imagen de entrada $f(x, y)$ y se establece un conjunto de puntos semilla con un determinado valor de intensidad, usualmente binario. Se crea una imagen que contiene a los puntos semilla $S(x, y)$; en las localizaciones de los puntos semilla se tendrá el valor asignado a ellos y en el resto de la imagen se tendrá un valor diferente. También se establecen un conjunto de condiciones C que se deben cumplir para anexar los pixeles vecinos a los puntos semilla, a partir del cumplimiento de estas condiciones es como irá creciendo la región de pixeles dentro de la segmentación. Posteriormente se aplica el algoritmo descrito por los pasos siguientes:

1) Tomar un pixel peteneciente al conjunto de semillas y los pixeles vecinos conectados en 4 o en 8.
2) Crecer la región con los pixeles de la vecindad que cumplan con las condiciones C establecidas. Esto es, asignar el valor de intensidad de la semilla a los pixeles vecinos a la misma que cumplen con las condiciones C, dentro de la imagen $S(x, y)$. Usualemente estas condiciones se basan en umbrales de decisión establecidos a partir de un cierto valor de intensidad thr. Una condición amplimente utilizada y presente en el algoritmo implementado es:

$$C = \begin{cases} 1 & |I_{seed} - I(x_i, y_j)| \leq thr \\ 0 & otro\ caso \end{cases} \quad (1)$$

   Donde I_{seed} es el valor de intensidad asignado a la semilla e $I(x_i, y_j)$ es el valor de intensidad de alguno de los vecinos conectados a la semilla.
   Los pixeles vecinos a una semilla que cumplen con C también se marcan como pixeles semilla.
3) Repetir 1) y 2) para todos los pixeles semilla.

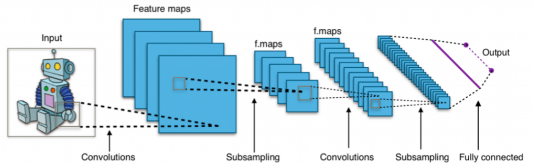

Fig. 1. Red Convolucional típica [2]

B. Redes convolucionales

Las redes convolucionales se han vuelto de uso frecuente en tareas de visión computacional en los últimos años. Esto se debe a que manejan de una forma mucho más eficiente entradas de dos dimensiones como son las imágenes.

Las redes convolucionales suelen usarse para tareas de clasificación, en este caso la idea general consiste en una etapa de extracción de características que suele estar formada por capas convolucionales, de submuestreo y de activación en forma alternada. Estas capas permiten formar mapas de características los cuales son la respuesta a la aplicación de distintos tipos de filtros. Por otro lado las capas de submuestreo reducen la dimensionalidad de los mapas sin que exista pérdida de información esencial, esto con la finalidad de que el costo computacional del procesamiento disminuya y así a su vez se pueda aumentar el tamaño del sistema en profundidad. Posteriormente se agregan capas completamente conectadas más una últma capa de activación las cuales son las que permiten realizar la clasificación. Un ejemplo de esta aqrquitectura se puede observar en la figura 1.

Las redes convolucionales pertenecen al grupo de técnicas de aprendizaje supervisado, esto quiere decir que todos los pesos de los filtros no son calculados por el diseñador, sino que son calculados por la red al ser entrenada. Por lo que una de las desventajas típicas de esta técnica es la necesidad de contar con un grupo grande de ejemplos para el entrenamiento.

C. Datasets

Para entrenar y probar la red convolucional se utilizó MNIST [3], el cual es un dataset de imágenes de dígitos manuscritos muy utilizado. Está formado por 60.000 imágenes para entrenamiento y 10.000 para prueba. Las imágenes tienen un tamaño de 28x28 pixeles, centradas y en escala de grises (Int 32 bits). La ventaja de esta base de datos es que todas las imágenes estan etiquetadas y sus clases están equilibradas. Una muestra de varios ejemplos de este dataset se puede observar en la figura 2.

Para probar el sistema se utilizó un grupo de 12 imágenes de placas vehiculares tomadas de internet, la idea consistió en seleccionar placas con diseños diferentes y que no necesariamente se encontaran en buen estado. El grupo de placas se muestra en la figura 3.

III. DESARROLLO

La implementación del reconocedor de dígitos consta de dos procesos principales y varios subprocesos dentro de los principales. Estos son:

Fig. 2. Muestras del MINST dataset [4]

1) Preprocesamiento de las imágenes para su ingreso a la red.
   a) Obtención de imágenes provenientes de las placas.
   b) Preparación de las imágenes para su ingreso a la red.
2) Clasificación de imágenes mediante la CNN.
   a) Definición y establecimiento de la arquitectura.
   b) Entrenamiento de la red.
   c) Validación.
   d) Prueba.
      i) Con imagénes de la base de datos MNIST.
      ii) Con imágenes provenientes de las placas.

A. Preprocesamiento de las imágenes para su ingreso a la red

Obtención de imágenes provenientes de las placas

Se tomó un grupo de 12 imágenes de placas vehiculares tomadas de internet. El grupo de placas se muestra en la figura 3. De cada una de las placas se extrajeron los dígitos y las letras que sirven para identificación, es decir, los de mayor tamaño. Los siguientes pasos muestran la forma en que se realizó esto.

1) *Conversión a escala de grises*: Se convirtieron las imágenes RGB de las placas a imágenes en escala de grises.
2) *Segmentación por crecimiento de regiones*: Se utilizó el algoritmo de segmentación por crecimiento de regiones descrito en la *subsección II-A* para segmentar el "fondo" de las imágenes de las placas, es decir, la sección que ocupa el área mayor dentro de las placas. Los puntos semilla iniciales utilzados para la segmentación se obtuvieron de forma automática, para realizar esto último se siguió el algoritmo descrito en [5] tomando como intervalo representativo del fondo el primero o el último de los intervalos representativos. La figura 4 muestra una placa muestra proveniente del conjunto de imágenes con las placas antes de realizar la segmentación. La figura 5 muestra el resultado obtenido después de realizar la segmentación. El resultado obtenido era una imagen binaria con valores 0 y 1.
3) *Detección y seguimiento de contornos*: Se obtuvieron y marcaron los contornos que delimitaban los núumeros, letras y figuras geométricas presentes en la imagen obtenida en la segmentación, figura 5. Esta obtención de contornos se llevó a cabo utilizando el algoritmo descrito

Fig. 3. Imágenes de placas de entrada al sistema

Fig. 4. Imagen de placa vehicular tomada como muestra

por *Suzuki, Satoshi et al.* presente en [6].La imagen de la figura 6 muestra una imagen representativa del resultado que se obtenía después de este paso.

4) *Aproximación poligonal y rectángulo delimitador*: Una vez encontrados los bordes en el paso anterior se aplicó el algoritmo de *Ramer-Douglas-Peucker* [7], [8] para simplificar las curvas obtenidas mediante la obtención de contornos y aproximarlos a figuras poligonales. Posteriormente a partir de los polígonos generados se delimitó el área dentro de ellos mediante un rectángulo que rodeaba al perímetro del polígono. Los resultados obtenidos en este paso se muestran en la figura 7.Las áreas delimitadas por los rectángulos encerraban a cada uno de los números y letras de identificación en la placa y otras figuras geométricas presentes.
5) *Recorte de imágenes y ajuste de tamaño*: Para cada una de las áreas delimitadas mediente los rectángulos obtenidos en el paso anterior se generó una imagen. Esta imagen estaba formada por los pixeles de la imagen proveniente de la segmentación, figura 5 comprendidos por cada rectángulo delimitador. Es decir, se copiaba o recortaba cada conjunto de pixeles delimitado por cada uno de los rectángulos. Con esto se consiguieron recortar los números y dígitos de identificación que interesaban y otro conjunto de imágenes que surgieron como consecuencia del paso 2 y 3. A cada imagen recortada se le aplicó zero-padding, se modificaron sus dimensiones a 28x28 px y se ajustarón los valores binarios de 0 y 1 a 0 y 255 respectivamente; obteniendo imágenes con características similares a las de la base de datos MNIST. Una imagen resultante de este paso y representativa del conjunto obtenido se muestra en la figura 8

Una vez que las imágenes de las placas han sido preprocesadas se obtuvo un conjunto muy grande de elementos recortados, por lo que se seleccionó manualmente aquellos elementos que fueran letras o dígitos de interés y se introdujeron en orden

Fig. 5. Placa Segmentada

Fig. 6. Imagen resultante al aplicar el algoritmo de seguimiento de bordes

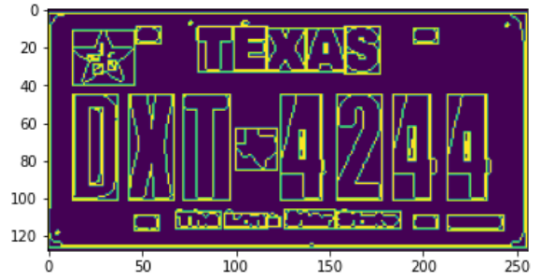

Fig. 7. Imagen resultante al aplicar el algoritmo de delimitación de áreas

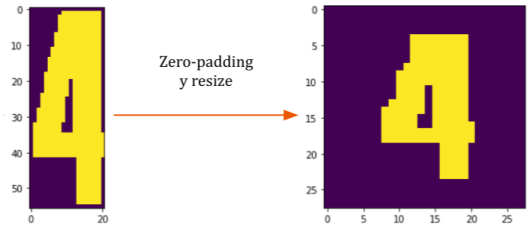

Fig. 8. Dígito recortado con base en el área delimitada y dígito con zero padding y cambio de tamaño.

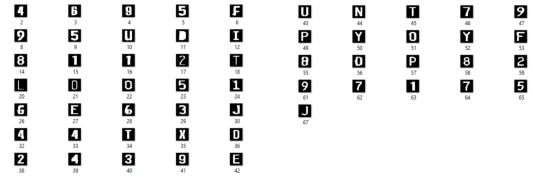

Fig. 9. Dígitos segmentados seleccionados

a la red convolucional. El set de imágenes que se introdujo está formado por 67 imágenes y se muestra en la figura 9.

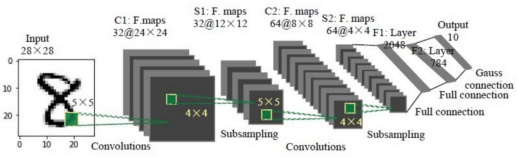

Fig. 10. Red Convolucional tomada de [9]

Preparación de las imágenes para su ingreso a la red. Tanto las 67 imágenes obtenidas de las placas como las provenientes de la base de datos MNIST se convirtieron a imagenes con formato de punto flotante contenidas en un arreglo que se convertía a un tensor para poder ser ingresado a la red neuronal. Para las 67 imágenes obtenidas de las placas se generaron sus respectivas etiquetas de forma manual.

B. Clasificación de imágenes mediante la CNN

La arquitectura de la red convolucional utilizada fue tomada del trabajo de *D. Ge, et. al.* [9] y se muestra en la figura 10. Se puede notar que posee dos bloques convolucionales formado cada uno por una capa convolucional, una capa de submuestreo y una capa de activación tipo ReLu, seguido todo esto de dos capas completamente conectadas más la última capa de activación que debe ser de tipo Softmax. Esta arquitectura fue implementada en Python usando librerias de Pytorch.

Los parámetros de la red se calculan de la siguiente manera, para las capas convolucionales:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1, \tag{2}$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1, \tag{3}$$

donde W_i y H_i son el ancho y alto de los mapas de características, F es el tamaño de los filtros, P el padding y S el stride o salto.

Se evaluaron estas ecuaciones para poder determinar los parámetros que cumplen con los valores definidos en 10 de allí se obtiene que $F = 5$, $P = 0$ y $S = 1$.

Para las capas de submuestreo:

$$W_3 = \frac{W_2 - F}{S} + 1, \tag{4}$$

$$H_3 = \frac{H_2 - F}{S} + 1, \tag{5}$$

Nuevamente se evaluaron para determinar los parámetros que se van a colocar en la red y se obtiene que para cumplir con la reducción del 50 por ciento del tamaño del mapa $F = 2$ y $S = 2$.

Como optimizador se utilizó la opción de descenso por gradiente estocástico con un paso de 0.001 y como función de costo la entropía cruzada.

El dataset se dividió en entrenamiento y validación, el tamaño del lote fue de 128 imágenes y se entrenó la red en 10 épocas, lo cual en promedio tardó 20 minutos en la plataforma Google Colab.

TABLE I
EXACTITUD DE LOS RESULTADOS

Dígitos manuscritos	**Dígitos placas**
0.969	0.493

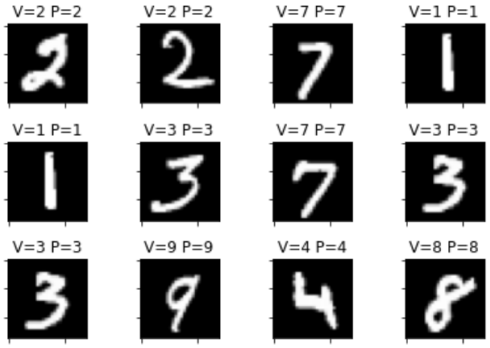

Fig. 11. Inferencia sobre algunas imágenes del conjunto de prueba de MINST

Al hacer inferencia sobre un conjunto de datos de los dígitos manuscritos presentes en la sección de prueba del Dataset MNIST, el resultado obtenido es el mostrado en la figura 11 y su respectiva matriz de confusión se puede observar en la figura 12. Además el valor de exactitud obtenido fue de 0.949. De estos resultados se puede decir que la red fue entrenada satisfactoriamente.

Al hacer inferencia sobre las 67 imágenes de los dígitos y letras recortados provenientes de las placas se obtuvo la matriz de confusión de la figura 14 y una exactitud de los resultados de 0.493. Es notable que en este caso la matriz de confusión es bastante dispersa lo cual es consecuente con su bajo valor de exactitud. También se debe notar que la red fue entrenada solomente con dígitos.

En la figura 13 se muestra un ejemplo de la imagen de una de las placas, sus tres últimos dígitos segmentados y el resultado de la clasificación. Allí se puede ver que clasificó exitosamente los dos primeros dígitos, sin embargo el tercero que correspondía a un 7 lo confundió con un 2, esto probablemente debido al punto al final de la placa.

IV. CONCLUSIONES

La red neuronal implementada utilizando PyTorch y entrenada con MNIST, permite clasificar con mayor exactitud los dígitos manuscritos provenientes del dataset que aquellos fuera de este último. El preprocesamiento de imágenes y los datos introducidos a una red neuronal utilizada en clasificación permiten incrementar la exactitud de la misma. A pesar de que tanto el entrenamiento de la red como el preprocesamiento se consideran exitosos, los resultados obtenidos de la clasificación de los dígitos y letras provenientes de las placas muestran una exactitud menor al 50%, esto se debe en parte a que en el set de elementos de prueba no sólo había dígitos sino también letras y dígitos similares. Se recomienda extender el

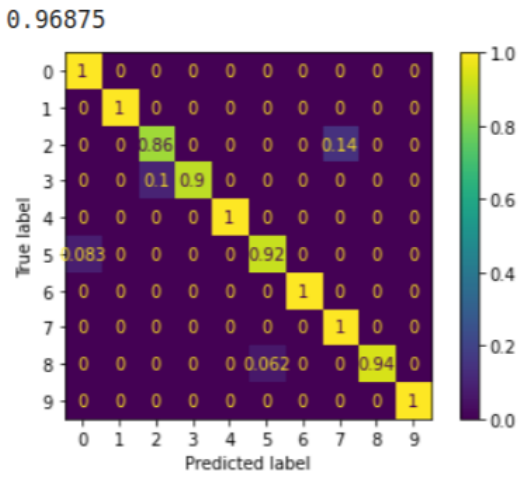

Fig. 12. Matriz de confusión usando como entrada al sistema imágenes de dígitos manuscritos

Fig. 13. Ejemplo de lectura de una de las placas

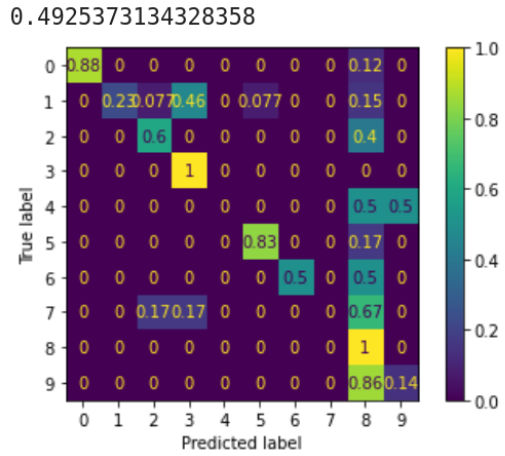

Fig. 14. Matriz de confusión usando como entrada al sistema imágenes de dígitos segmentados de las placas vehiculares

entrenamiento de la red utilizando un dataset más amplio que contenga tanto dígitos como letras.

REFERENCES

[1] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital image processing using MATLAB*. Pearson Education India, 2004.

[2] Pocho Costa, "Redes neuronales convolucionales explicadas." https://pochocosta.com/podcast/redes-neuronales-convolucionales-explicadas/, 2013. Online; accessed 28 January 2021.

[3] Yann Lecun, " THE MNIST DATABASE of handwritten digits." http://yann.lecun.com/exdb/mnist/, 2013. Online; accessed 28 January 2021.

[4] Julien Simon, " Training MXNet — part 1: MNIST." https://chatbotslife.com/training-mxnet-part-1-mnist-6f0dc4210c62, 2013. Online; accessed 28 January 2021.

[5] O. Gómez, J. A. González, and E. F. Morales, "Image segmentation using automatic seeded region growing and instance-based learning," in *Iberoamerican Congress on Pattern Recognition*, pp. 192–201, Springer, 2007.

[6] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[7] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves"," *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244 – 256, 1972.

[8] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, 1973.

[9] Ge, Dong-yuan and Yao, Xi-fan and Xiang, Wen-jiang and Wen, Xue-jun and Liu, En-chen, "Design of high accuracy detector for mnist handwritten digit recognition based on convolutional neural network," in *2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, pp. 658–662, IEEE, 2019.