ISSN 1870-4069

Comparación de dos métodos para reconocimiento de dígitos manuscritos fuera de línea

María Cristina Guevara Neri[1], Osslan Osiris Vergara Villegas[1], Vianey Guadalupe Cruz Sánchez[1], Juan Humberto Sossa Azuela[2, 3]

[1] Universidad Autónoma de Ciudad Juárez, Chihuahua, México

[2] Instituto Politécnico Nacional (CIC), México

[3] Instituto Tecnológico de Estudios Superiores Monterrey (Campus Guadalajara), México

mc_guevara_neri@hotmail.com, {overgara, vianey.cruz}@uacj.mx, hsossa@cic.ipn.mx

Resumen. En el presente artículo, se muestra el resultado de la comparación del desempeño de dos métodos para el reconocimiento de dígitos manuscritos fuera de línea. El primer método, es una red de perceptrones con la cual se clasificaron las imágenes tras realizar una comparación por pares de clases; el segundo, es un método novedoso que realiza una comparación pixel por pixel entre la imagen por clasificar, y las imágenes de referencia. Para las pruebas, se utilizó un subconjunto de 450 imágenes de la base de datos MNIST. Cada método fue evaluado en dos partes: primero, con un conjunto de 100 imágenes de entrenamiento, y segundo, con un conjunto de 350 imágenes de prueba. Con el primer clasificador se obtuvo una exactitud del 93.86%, y con el segundo se consiguió una del 95.14%. Después del análisis de los resultados obtenidos se demuestra que el segundo método se desempeñó mejor que el primero. La fortaleza del método novedoso radica principalmente en su robustez y tiempo de ejecución.

Palabras clave: Reconocimiento de dígitos manuscritos, red de perceptrones, comparación vector con vector, MNIST.

Comparison of Two Off-Line Handwritten Digits Recognition Methods

Abstract. In this paper, the results of the comparison between two off-line handwritten digits recognition methods are presented. The first method is a network of perceptrons with which the images were classified after making a comparison by pairs of classes; the second, is a new method that performs a pixel by pixel comparison between the image to be classified, and the reference images. For the tests, a subset of 450 images from the MNIST database was used.

Each method was evaluated in two parts: first, with a set of 100 training images, and second, with a set of 350 test images. With the first classifier, an accuracy of 93.86% was obtained, and with the second, an accuracy of 95.14%. After the analysis of the results, it is shown that the second method outperformed the first. The strength of the new method lies mainly in its robustness and execution time.

Keywords: Handwritten digits recognition, perceptron network, comparison between vectors, MNIST.

1. Introducción

En la última década (2008-2018), se ha observado un incremento en la digitalización de documentos, por lo que, el reconocimiento de dígitos manuscritos, y caracteres en general, es un área de interés para científicos, académicos e industriales [1]. El reconocimiento de caracteres manuscritos ha ganado popularidad en el área de investigación, y ha generado el reto entre los investigadores de lograr emular el sistema de procesamiento visual humano, mediante el desarrollo de algoritmos computacionales.

El reconocimiento de dígitos manuscritos es una rama correspondiente a la tecnología de reconocimiento óptico de caracteres, en la que el reto consiste en cómo utilizar un dispositivo procesador para reconocer automáticamente los dígitos [2]. La lectura de caracteres manuscritos representa una tarea complicada para una máquina, ya que no cuenta, como lo hace un ser humano, con la posibilidad de tomar decisiones fuera de un esquema establecido.

Las técnicas de reconocimiento de dígitos manuscritos se clasifican en dos grandes áreas que son en línea y fuera de línea. En la primera técnica, el flujo de dígitos es reconocido en el momento en el que es escrito, mientras que, en la segunda, que suele ser más compleja, los dígitos son reconocidos mediante la captura o escaneo de imágenes. Usualmente, el reconocimiento de dígitos manuscritos consiste en las siguientes etapas: 1) adquisición, 2) pre-procesamiento, 3) segmentación, 4) extracción de características, y 5) clasificación [3].

La etapa de adquisición o digitalización consiste en adquirir la imagen de los dígitos manuscritos a través de una fotografía o escaneo. En el pre-procesamiento se aplican diversos algoritmos de mejoramiento con la finalidad de mejorar la calidad de la información. Por ejemplo, en las imágenes de dígitos manuscritos se puede presentar ruido debido al color del papel, la textura, imágenes de fondo, emborronamiento, distorsiones por la perspectiva de la imagen, variaciones en la iluminación, el cual debe ser eliminado en la medida de lo posible [4].

La segmentación de los objetos de la imagen es la fase donde se tiene como objetivo separar el área correspondiente al dígito, y el área del fondo [5]. En la extracción de características se transforma la información de entrada en descriptores esenciales para distinguir un dígito de otro [6]. Finalmente, la clasificación representa la predicción de la clase (o etiqueta) para un objeto, basado en la similitud del dígito contra los dígitos de referencia [7]. Uno de los algoritmos más simples de clasificación es el vecino más

cercano (k-NN, del inglés k-Nearest Neighbor), el cual se basa en el uso de distancias (Euclidiana, Manhattan, Hamming, etc.) [8]. Por otro lado, las Redes Neuronales Artificiales (RNA) tienen como objetivo procesar información de la misma manera que el cerebro humano lo hace, por lo que cuentan con una fase de entrenamiento y aprendizaje y han sido muy utilizadas en la etapa de clasificación de dígitos manuscritos [1, 2, 7]. La técnica de Máquina con Vectores Soporte (SVM, del inglés Support Vector Machine), se basa en el descubrimiento de un hiperplano que se usa para separar los datos de las clases existentes [9]. En [10] se puede consultar un estudio de las diferentes técnicas de clasificación de dígitos manuscritos.

Aun cuando en la literatura se han presentado diversos trabajos sobre el reconocimiento de dígitos manuscritos, la tarea sigue representando un reto en el área de visión por computadora, debido a la diversidad de estilos de escritura asociados a diferentes personas, las herramientas utilizadas para escribir (pluma, lápiz), y la falta de cuidado al realizar los trazos. Por lo tanto, en el presente trabajo se realiza una comparación entre dos métodos de clasificación para el reconocimiento de dígitos manuscritos fuera de línea: un método fundamentado en el uso de una RNA de perceptrones, y un nuevo método basado en comparaciones similar al k-NN.

2. Materiales y métodos

La metodología para la clasificación de dígitos manuscritos utilizada en el presente artículo es similar a la explicada en la sección 1 y se puede observar en la Figura 1.

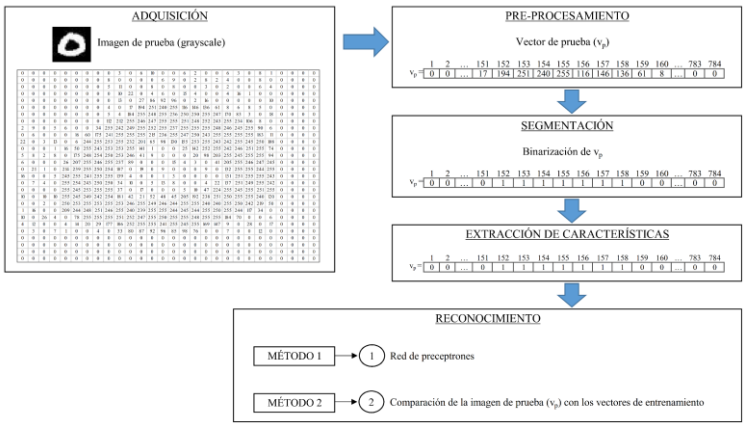

Fig. 1. Etapas propuestas para el reconocimiento de dígitos manuscritos

En la literatura sobre reconocimiento de dígitos manuscritos, existen diversas bases de datos tales como: The Street View House Numbers (SVHN) [11], The National Institute of Standards and Technology (NIST) Special Database 19 [12], y The Modified National Institute of Standards and Technology (MNIST) database [13, 14], que se han utilizado como estándar para probar la robustez de los métodos propuestos. Sin embargo, MNIST creada por Lecun [14], es la que más se utiliza en la literatura,

Fig. 2. Ejemplo de algunos dígitos del subconjunto seleccionado de MNIST.

por lo que fue seleccionada en el presente artículo. La base de datos MNIST se forma por un conjunto de 60000 imágenes de entrenamiento y 10000 de prueba de dígitos manuscritos del 0 al 9, las cuales están normalizadas, es decir, cada dígito tiene un tamaño de 20 x 20 pixeles, y se encuentra sobre un fondo de 28 x 28 pixeles (centrado mediante el centro de gravedad). Para probar los dos métodos de clasificación del presente artículo, se seleccionó al azar un subconjunto de 450 imágenes, 100 para la fase de entrenamiento, 10 por cada dígito desde el 0 al 9, y 350 para la fase de pruebas, 35 por cada dígito desde el 0 al 9. En la Figura 2, se muestra un ejemplo de algunos de los dígitos seleccionados.

Cada una de las 450 imágenes fue preprocesada por medio de una transformación (reacomodo) que genera una representación vectorial de 1 x 784 píxeles. En seguida, cada vector (que se encuentra en escala de grises, con valores entre 0 y 255) es segmentado por medio de un proceso de binarización. Para binarizar las imágenes se aplicó la regla siguiente: para cada pixel de la imagen de entrada (Im), si el valor del pixel es menor que 100, que es el valor del umbral[1], se le asigna un valor de 0, y si el valor del pixel es mayor o igual que 100, entonces se le asigna un valor de 1, como se puede observar en las Ecuaciones 1 y 2.

$$Si\ Im(i,j) < 100, \quad Im(i,j) = 0\ con\ i,j\ de\ 1\ a\ 28, \tag{1}$$

$$Si\ Im(i,j) \geq 100, \quad Im(i,j) = 1\ con\ i,j\ de\ 1\ a\ 28. \tag{2}$$

Los 784 valores obtenidos del proceso de segmentación son utilizados como el vector de características para describir cada uno de los dígitos, cabe mencionar que en el presente trabajo no se realizó una etapa de selección de características. Finalmente, se realiza el reconocimiento de los dígitos por medio de dos métodos diferentes.

Antes de describir los métodos utilizados para clasificación es importante mencionar que: a) El primer método, utiliza un concepto conocido: el perceptrón, b) El segundo método, es un método nuevo propuesto en el presente trabajo, el cual consiste en la aplicación de una regla de comparación entre imágenes, c) Ambos métodos se aplicaron a los mismos conjuntos de imágenes, d) Los experimentos para ambos métodos se realizaron bajo las mismas condiciones.

[1] Después de un análisis exhaustivo donde se analizó el valor de cada pixel de la imagen, se obtuvo que el umbral 100 permite que todos los dígitos queden segmentados de manera adecuada.

2.1. Método 1: red de perceptrones

El primer método utilizó una red de perceptrones para la clasificación de las imágenes. Un perceptrón es un algoritmo de aprendizaje binario que permite determinar si el dígito a clasificar pertenece o no a determinada clase [15].

Para crear la red, se utilizó la función *perceptron* incluida en la librería estándar de MATLAB, la cual emplea como base para la clasificación la función de transferencia *hardlimit*. La función *perceptron* requiere una fase de entrenamiento (*train*), y, utiliza el parámetro *epoch*, que es la aplicación de la regla de aprendizaje de la red a cada ejemplo en el conjunto de datos (1 *epoch* = 1 aplicación de la regla). El método utilizó, en promedio, alrededor de 6 *epochs* por perceptrón.

Como el perceptrón realiza la clasificación entre dos tipos de clases, se construyó una red de 90 perceptrones para poder realizar todas las comparaciones entre la imagen de prueba y las 10 clases posibles como se muestra en la Figura 3.

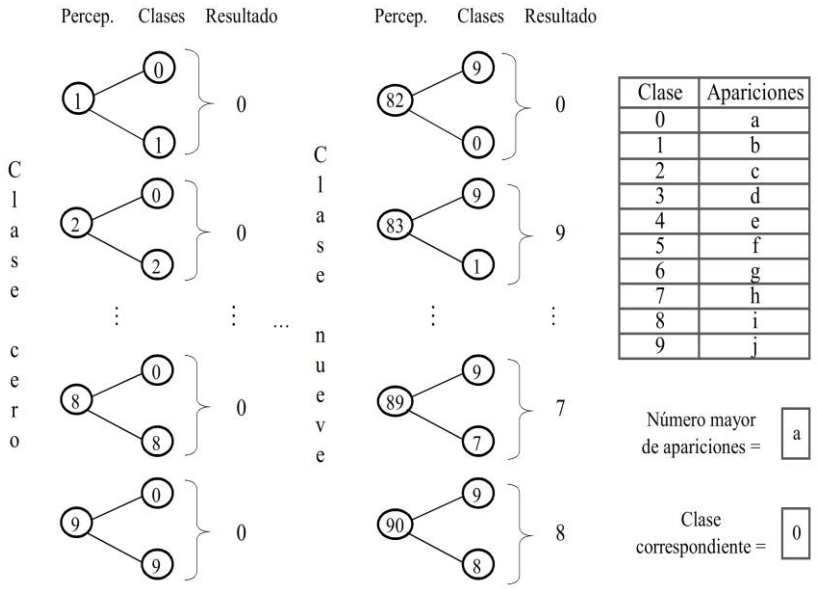

Fig. 3. Red de perceptrones utilizada en el método 1.

Las comparaciones para decidir cuál era la clase a la que más se parecía la imagen de prueba se realizaron por pares, y finalmente, se realizó la clasificación con la clase más repetida en todas las comparaciones hechas. La Tabla 1 muestra un ejemplo con una imagen de prueba correspondiente a la clase cero.

Cada renglón de la Tabla 1 corresponde a dos perceptrones, donde las columnas referentes a las clases indican cuáles fueron las clases analizadas (por pares), y la columna de clasificación muestra cuál fue el resultado obtenido. En la Tabla 2, se muestra el número de apariciones de cada una de las clases predichas por el clasificador para la imagen de la Tabla 1.

Tabla 1. Clasificación de una imagen de prueba correspondiente a un cero.

Clase 1	Clase 2	Clasificación	Clase 1	Clase 2	Clasificación
cero	uno	cero	cinco	cero	cero
cero	dos	cero	cinco	uno	cinco
cero	tres	cero	cinco	dos	cinco
cero	cuatro	cero	cinco	tres	tres
cero	cinco	cero	cinco	cuatro	cinco
cero	seis	cero	cinco	seis	cinco
cero	siete	cero	cinco	siete	cinco
cero	ocho	cero	cinco	ocho	cinco
cero	nueve	cero	cinco	nueve	cinco
uno	cero	cero	seis	cero	cero
uno	dos	dos	seis	uno	seis
uno	tres	tres	seis	dos	seis
uno	cuatro	cuatro	seis	tres	tres
uno	cinco	cinco	seis	cuatro	cuatro
uno	seis	seis	seis	cinco	cinco
uno	siete	siete	seis	siete	siete
uno	ocho	ocho	seis	ocho	ocho
uno	nueve	nueve	seis	nueve	seis
dos	cero	cero	siete	cero	cero
dos	uno	dos	siete	uno	siete
dos	tres	tres	siete	dos	dos
dos	cuatro	dos	siete	tres	tres
dos	cinco	cinco	siete	cuatro	cuatro
dos	seis	seis	siete	cinco	siete
dos	siete	dos	siete	seis	siete
dos	ocho	dos	siete	ocho	ocho
dos	nueve	dos	siete	nueve	siete
tres	cero	cero	ocho	cero	cero
tres	uno	tres	ocho	uno	ocho
tres	dos	tres	ocho	dos	dos
tres	cuatro	tres	ocho	tres	tres
tres	cinco	cinco	ocho	cuatro	ocho
tres	seis	tres	ocho	cinco	cinco
tres	siete	tres	ocho	seis	ocho
tres	ocho	tres	ocho	siete	ocho
tres	nueve	tres	ocho	nueve	ocho
cuatro	cero	cero	nueve	cero	cero
cuatro	uno	cuatro	nueve	uno	nueve
cuatro	dos	dos	nueve	dos	dos
cuatro	tres	tres	nueve	tres	tres
cuatro	cinco	cinco	nueve	cuatro	cuatro
cuatro	seis	cuatro	nueve	cinco	cinco
cuatro	siete	cuatro	nueve	seis	nueve
cuatro	ocho	cuatro	nueve	siete	siete
cuatro	nueve	cuatro	nueve	ocho	ocho

Tabla 2. Resumen del número total de aparición de clases para la imagen de la Tabla 1.

Clase	Apariciones	Clase	Apariciones
cero	18	cinco	14
uno	0	seis	5
dos	10	siete	7
tres	15	ocho	9
cuatro	9	nueve	3

Como se puede observar en la Tabla 2, la clase que más se predijo fue la clase cero, con 18 apariciones (respecto al total de los 90 resultados), por lo cual la imagen fue clasificada como un cero. Cabe mencionar que en ninguna de las pruebas se presentó algún caso de empate entre el número mayor de apariciones de las clases, por lo que no se generó un criterio de desempate.

2.2. Método 2: comparación vector contra vector

El segundo método consiste en la aplicación de un nuevo algoritmo de comparación creado en MATLAB. El método se basa, como su nombre lo indica, en una comparación directa entre el vector de prueba (v_p) y los vectores de entrenamiento (v_e). Los vectores de entrenamiento se componen por 10 vectores renglón por dígito, cada uno de ellos correspondiente a una imagen, conformando así 100 vectores de entrenamiento en total. El vector de prueba corresponde al vector renglón que representa la imagen que se quiere clasificar. Es importante aclarar que los vectores de entrenamiento hacen referencia a los vectores que se están tomando como punto de referencia para la comparación que realiza el método, por lo tanto, el método propuesto al igual que un k-NN **no tiene** una fase de aprendizaje, además el método propuesto no necesita una medida de distancia.

El vector de prueba es comparado contra cada uno de los vectores de entrenamiento, es decir, cada una de sus componentes se contrasta contra cada una de las componentes de los 100 vectores, para calcular a cuál se parece más. Por lo que, el número de comparaciones hechas por cada imagen de prueba que se compara, contra cada clase, es de 78400 (784 componentes del vector de prueba por 100 vectores de entrenamiento). En cada comparación, se asigna un valor específico que se obtiene de la siguiente manera: por cada componente correspondiente de los vectores que sea exactamente igual, a la comparación se le fija un valor 0, y por cada que sea distinta se le fija un valor 1, como se muestra en las Ecuaciones 3 y 4.

$$Si\ v_p(1,i) = v_e(1,i), \qquad a = 0\ con\ i\ de\ 1\ a\ 784, \tag{3}$$

$$Si\ v_p(1,i) \neq v_e(1,i), \qquad a = 1\ con\ i\ de\ 1\ a\ 784. \tag{4}$$

Finalmente, todos los valores de cada comparación entre componentes se suman y generan como resultado un valor de comparación final, el cual se almacena en un vector (z), como se muestra en la Ecuación 5, de manera que cuando se realizan las 100 comparaciones, se selecciona aquella con el valor más pequeño (ver Ecuación 6), y se le determina una clase del cero al nueve, de acuerdo con la posición del valor mínimo.

Cabe mencionar que en las pruebas realizadas no se presentó algún caso de empate entre los valores más pequeños, por lo que no se generó un criterio de desempate.

$$z(j,1) = \sum_{i=1}^{784} a_i \ , con\ j\ de\ 1\ a\ 100, \tag{5}$$

$$Clase \leftarrow \min(z). \tag{6}$$

En la Figura 4 se muestra de manera visual un ejemplo del funcionamiento del nuevo método propuesto.

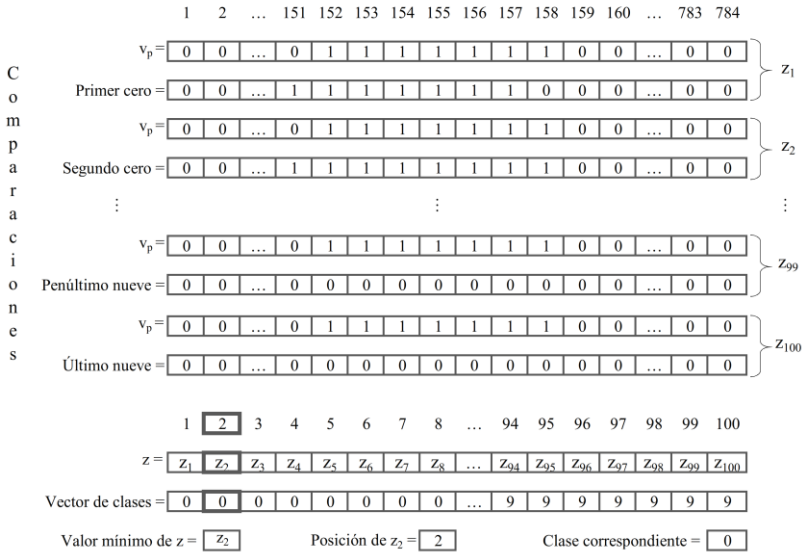

Fig. 4. Funcionamiento del método de comparación de vector contra vector.

3 Experimentación y resultados

Para cada uno de los métodos se realizaron experimentos con los mismos dos conjuntos de imágenes: 1) entrenamiento (100), y 2) prueba (350). En ambos casos se presentan los resultados del entrenamiento y de la prueba. Todas las pruebas se realizaron en una laptop HP con procesador Intel Core i5 con memoria RAM de 8Gb y las implementaciones de los dos métodos se realizaron con MATLAB.

La Tabla 3 muestra la notación, fórmula y descripción de las variables medidas. Es importante aclarar que el comportamiento de ambos métodos a lo largo de los experimentos siempre fue el mismo, por lo que los datos presentados en las siguientes subsecciones corresponden a una única corrida por imagen.

3.1. Resultados obtenidos con el método 1

Tabla 3. Variables de medición obtenidas de [16].

Variable (notación)	Fórmula	Descripción
Verdaderos Positivos (VP)	-	Casos que fueron predichos SÍ, y realmente fueron SÍ
Verdaderos Negativos (VN)	-	Casos que fueron predichos NO, y realmente fueron NO
Falsos Negativos (FN)	-	Casos que fueron predichos NO, y realmente fueron SÍ
Falsos Positivos (FP)	-	Casos que fueron predichos SÍ, y realmente fueron NO
Exactitud	(VP+VN) / total	En general, ¿qué tan frecuente el clasificador acierta?
Tasa de error	(FP+FN) / total	En general, ¿qué tan frecuente el clasificador NO acierta?
Sensibilidad	VP / SÍ real	Cuando es un SÍ real, ¿qué tan frecuente el clasificador predice un SÍ?
Especificidad	VN / NO real	Cuando es un NO real, ¿qué tan frecuente el clasificador predice un NO?
Precisión	VP / SÍ predicho	Cuando el clasificador predice SÍ, ¿qué tan frecuente está en lo correcto?

Tabla 4. Método 1: resultados obtenidos con 100 imágenes de entrenamiento.

	CLASE										
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
VP	9	10	10	10	10	10	10	10	10	10	99
VN	90	90	90	90	90	89	90	90	90	90	-
FP	0	0	0	0	0	1	0	0	0	0	-
FN	1	0	0	0	0	0	0	0	0	0	1
SÍ reales	10	10	10	10	10	10	10	10	10	10	100
NO reales	90	90	90	90	90	90	90	90	90	90	-
SÍ predichos	9	10	10	10	10	11	10	10	10	10	-
NO predichos	91	90	90	90	90	89	90	90	90	90	-
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Media**
Exactitud	0.9900	1.0000	1.0000	1.0000	1.0000	0.9900	1.0000	1.0000	1.0000	1.0000	0.9980
Tasa de error	0.0100	0.0000	0.0000	0.0000	0.0000	0.0100	0.0000	0.0000	0.0000	0.0000	0.0020
Sensibilidad	0.9000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9900
Especificidad	1.0000	1.0000	1.0000	1.0000	1.0000	0.9889	1.0000	1.0000	1.0000	1.0000	0.9989
Precisión	1.0000	1.0000	1.0000	1.0000	1.0000	0.9091	1.0000	1.0000	1.0000	1.0000	0.9909

Primero, se midió la capacidad para aprender del método 1. Los resultados obtenidos con las 100 imágenes de entrenamiento se muestran en la Tabla 4.

Los resultados obtenidos al aplicar el método 1 a las 350 imágenes de prueba se muestran en la Tabla 5.

Tabla 5. Método 1: resultados obtenidos con 350 imágenes de prueba.

	CLASE										
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
VP	31	31	18	27	27	21	23	23	22	19	242
VN	308	314	303	308	305	289	302	307	298	309	-
FP	7	1	12	7	10	26	13	8	17	6	-
FN	4	4	17	8	8	14	12	12	13	16	108
SÍ reales	35	35	35	35	35	35	35	35	35	35	350
NO reales	315	315	315	315	315	315	315	315	315	315	-
SÍ predichos	38	32	30	34	37	47	36	31	39	25	-
NO predichos	312	318	320	316	313	303	314	319	311	325	-
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Media**
Exactitud	0.9686	0.9857	0.9171	0.9571	0.9486	0.8857	0.9286	0.9429	0.9143	0.9371	0.9386
Tasa de error	0.0314	0.0143	0.0829	0.0429	0.0514	0.1143	0.0714	0.0571	0.0857	0.0629	0.0614
Sensibilidad	0.8857	0.8857	0.5143	0.7714	0.7714	0.6000	0.6571	0.6571	0.6286	0.5429	0.6914
Especificidad	0.9778	0.9968	0.9619	0.9778	0.9683	0.9175	0.9587	0.9746	0.9460	0.9810	0.9660
Precisión	0.8158	0.9688	0.6000	0.7941	0.7297	0.4468	0.6389	0.7419	0.5641	0.7600	0.7060

Tabla 6. Método 2: resultados obtenidos con 100 imágenes de entrenamiento.

	CLASE										
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
VP	10	10	10	10	10	10	10	10	10	10	100
VN	90	90	90	90	90	90	90	90	90	90	-
FP	0	0	0	0	0	0	0	0	0	0	-
FN	0	0	0	0	0	0	0	0	0	0	0
SÍ reales	10	10	10	10	10	10	10	10	10	10	100
NO reales	90	90	90	90	90	90	90	90	90	90	-
SÍ predichos	10	10	10	10	10	10	10	10	10	10	-
NO predichos	90	90	90	90	90	90	90	90	90	90	-
Variables	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Media**
Exactitud	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Tasa de error	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Sensibilidad	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Especificidad	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Precisión	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

3.2. Resultados obtenidos con el método 2

Los resultados obtenidos al aplicar el método 2 a las 100 imágenes de entrenamiento se muestran en la Tabla 6.

Los resultados obtenidos al aplicar el método 2 a las 350 imágenes de prueba se muestran en la Tabla 7.

Tabla 7. Método 2: resultados obtenidos con 350 imágenes de prueba.

	CLASE										
VARIABLES	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**TOTAL**
VP	33	34	22	23	28	23	28	27	24	23	265
VN	305	301	312	310	308	304	310	303	312	300	-
FP	10	14	3	5	7	11	5	12	3	15	-
FN	2	1	13	12	7	12	7	8	11	12	85
SÍ reales	35	35	35	35	35	35	35	35	35	35	350
NO reales	315	315	315	315	315	315	315	315	315	315	-
SÍ predichos	43	48	25	28	35	34	33	39	27	38	-
NO predichos	307	302	325	322	315	316	317	311	323	312	-
VARIABLES	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**MEDIA**
Exactitud	0.9657	0.9571	0.9543	0.9514	0.9600	0.9343	0.9657	0.9429	0.9600	0.9229	0.9514
Tasa de error	0.0343	0.0429	0.0457	0.0486	0.0400	0.0657	0.0343	0.0571	0.0400	0.0771	0.0486
Sensibilidad	0.9429	0.9714	0.6286	0.6571	0.8000	0.6571	0.8000	0.7714	0.6857	0.6571	0.7571
Especificidad	0.9683	0.9556	0.9905	0.9841	0.9778	0.9651	0.9841	0.9619	0.9905	0.9524	0.9730
Precisión	0.7674	0.7083	0.8800	0.8214	0.8000	0.6765	0.8485	0.6923	0.8889	0.6053	0.7689

3.3. Discusión

De acuerdo con los resultados obtenidos con el primer conjunto de imágenes (de entrenamiento) mostrados en las Tablas 4 y 6, se observa que el método 2 presentó un mejor desempeño, ya que clasificó correctamente todos los casos, mientras que el método 1 clasificó correctamente 99 de las 100 imágenes. Aunque los resultados arrojados por el método 1 son (relativamente) buenos, no son los esperados, ya que se esperaba que pudieran ser reconocidas todas las imágenes del conjunto, tal como lo hizo el método 2.

Con los resultados obtenidos en los experimentos hechos sobre el conjunto de imágenes de prueba mostrados en las Tablas 5 y 7, se observa que el método 2 es mejor que el método 1. La comparación de las variables obtenidas mediante la aplicación de ambos métodos se muestra en la Tabla 8.

Tabla 8. Comparación de los métodos.

Variables	Método 1	Método 2
Exactitud	0.9386	0.9514
Tasa de error	0.0614	0.0486
Sensibilidad	0.6914	0.7571
Especificidad	0.9660	0.9730
Precisión	0.7060	0.7689

Como se puede observar en la Tabla 8, con el método 1 se obtuvo una sensibilidad de 69%, es decir, clasificó de manera correcta 242 de las 350 imágenes, mientras que con el método 2 se obtuvo una sensibilidad de 75%, es decir, de las 350 imágenes, clasificó correctamente 265. De igual forma, el método 2 presentó mejores resultados en exactitud, pues el clasificador fue mejor al momento de clasificar de manera general

Fig. 5. Ejemplos de resultados para las imágenes de prueba.

las imágenes, es decir, acertó de manera más frecuente sobre cuándo una imagen pertenece a una clase, y cuándo no pertenece a una clase.

La Figura 5 ilustra algunos ejemplos tomados del conjunto de imágenes de prueba, con su respectiva clasificación (se debe observar que existen dígitos que a simple vista son difíciles de reconocer). El número a la izquierda (en negritas) representa la clase correcta a la que pertenece la imagen, los siguientes dos números refieren a la clase dada por ambos clasificadores, (método 1/método 2).

Aunque los resultados entre los dos métodos se encuentran relativamente cerca, la principal diferencia radica en el tiempo de ejecución. El método 2 fue significativamente más rápido para la prueba y entrenamiento, en comparación con el método 1. Para crear una idea sobre qué tan significativa fue la diferencia respecto a los tiempos de ejecución, considere lo siguiente: el método 2 tardó en clasificar las 450 imágenes en un tiempo menor a 10 s, mientras que el método 1 tardó en clasificar las mismas imágenes, y bajo las mismas condiciones, alrededor de 90 min. Por lo que se si se utilizaran, por ejemplo, 42000 imágenes de MNIST, el método 2 terminaría de clasificarlas todas, en el mismo tiempo que el método 1 solamente clasificaría 70.

Con respecto a la bondad del método presentado contra los trabajos de la literatura se observa que es difícil realizar una comparación real. Aun cuando existen trabajos que utilizan MNIST, normalmente, no se utiliza el mismo subconjunto de imágenes y se tendrían que programar los otros métodos. Sin embargo, para el caso de la clasificación nuestro trabajo obtuvo una exactitud de 0.9514 con el nuevo método propuesto, mientras que en el trabajo de [2], se utilizó una RNA de pico de descenso aproximado normalizado y se obtuvo una exactitud de 0.9817; en el trabajo de [5], se utilizó una SVM y se obtuvo una exactitud de 0.9691, en la investigación de [7], se utilizó una red de creencias profundas con aprendizaje Q y se obtuvo una exactitud de 0.9918, además se reporta un tiempo de 21.46 s para la clasificación de 100 imágenes de MNIST. Finalmente, en el trabajo presentado en [9], se utilizó un híbrido de una SVM y una red neuronal convolucional en la que se reporta una exactitud de 0.9981. Como se puede observar, el método 2 presentado en este artículo obtuvo resultados de exactitud competitivos contra los presentados en el estado del arte y además buen tiempo de ejecución, con la característica adicional que nuestro método es computacionalmente menos complejo.

4. Conclusiones

Se presentó una comparación entre dos métodos para el reconocimiento de dígitos manuscritos fuera de línea. Para poder realizar una comparación justa para ambos métodos se utilizó un subconjunto de 450 imágenes de MNIST. En el método 1 se implementó una red de perceptrones, mientras que en el método 2 se presentó un nuevo

algoritmo que consiste en la comparación entre las características de imágenes de prueba y las características de imágenes de referencia similar al k-NN con la diferencia de que no se calculan distancias.

De acuerdo con los resultados obtenidos se concluye que el nuevo método propuesto (2) tiene un mejor desempeño comparado con el método 1. La conclusión se obtiene a partir de la eficiencia del clasificador al realizar la clasificación con los diversos conjuntos de imágenes, así como el tiempo de ejecución que utiliza cada método para realizar la tarea asignada. Además de la eficiencia en la clasificación es importante destacar que, el método propuesto en el presente trabajo está basado en el uso de operaciones de comparación, en donde se conoce qué está siendo comparado, y cómo se realiza la comparación, mientras que en el método 1 no ocurre lo mismo, puesto que al utilizar el perceptrón se conoce qué se está clasificando, pero no cómo está siendo clasificado.

Como trabajo futuro se considera: aumentar el tamaño del conjunto de imágenes de entrenamiento y de prueba, y modificar el método 1 para reducir el tiempo de ejecución, a través de la reducción del número de perceptrones utilizados, mediante el descarte temprano de algunas clases. Además, será importante realizar pruebas con otros tipos de clasificadores.

Agradecimientos. H. Sossa agradece al Instituto Politécnico Nacional y al CONACYT, a través de los apoyos económicos en el marco de fondos SIP 20180730 y 65 (Fronteras de la Ciencia). María Cristina Guevara Neri agradece al CONACYT por la beca otorgada para la realización de sus estudios de doctorado.

Referencias

1. Kulkarni, S., Rajendran, B.: Spiking neural networks for handwritten digit recognition-supervised learning and network optimization. Neural Networks, 103, pp.118–127 (2018)
2. Sueiras, J., Ruiz, V., Sanchez, A., Velez, J.: Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing, 289, pp. 119–128 (2018)
3. Kumar, M., Jindal, M., Sharma, R., Rani, S.: Character and numeral recognition for non-Indic and Indic scripts: A survey. Artificial Intelligence Review, pp. 1–27 (2018)
4. Mandal, S., Mahadeva, S., Sundaram, S.: GMM posterior features for improving online handwriting recognition. Expert Systems with Applications, 97, pp. 421–433 (2018)
5. Gattal, A., Chibani, Y., Hadjadji, B.: Segmentation and recognition system for unknown-length handwritten digit strings. Pattern Analysis and Applications. 20(2), pp. 307–323 (2017)
6. Surinta, O., Karaaba, M., Schomaker, L., Wiering, M.: Recognition of handwritten characters using local gradient feature descriptors. Engineering Applications of Artificial Intelligence, 45, pp. 405–414 (2015)
7. Qiao, J., Wang, G., Li, W., Chen, M.: An adaptive deep Q-learning strategy for handwritten digit recognition. Neural Networks. Article in Press, pp. 1–18 (2018)
8. Bhattacharya, G., Ghosh, K., Chowdhury, A.: An affinity-based new local distance function and similarity measure for kNN algorithm. Pattern Recognition Letters. 33(3), pp. 356–363 (2012)
9. Niu, X., Suen, C.: A novel hybrid CNN-SVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), pp. 1318–1325 (2012)

10. Al-Helali, B., Mahmoud, S.: Arabic online handwriting recognition (AOHR): A survey. ACM Computing Surveys, 50(3), pp. 1–35 (2017)
11. The Street View House Numbers (SVHN) Dataset: http://ufldl.stanford.edu/housenumbers/, last accessed (2018)
12. NIST: National Institute of Standards and Technology. NIST Special Database 19 (2018)
13. The MNIST database: http://yann.lecun.com/exdb/mnist/ (2018)
14. LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, A., Sackinger, E., Simard, P., Vapnik, V.: Learning algorithms for classification: A comparison on handwritten digit recognition. Neural Networks: The Statistical Mechanics Perspective, pp. 261–276 (1995)
15. Kurkova, V., Sanguineti, M.: Probabilistic lower bounds for approximation by shallow perceptron networks. Neural Networks, 91, pp. 31–41 (2017)
16. Markham, K.: Simple guide to confusion matrix terminology, http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/ (2018)