RECONOCIMIENTO DE CARACTERES ALFANUMÉRICOS HACIENDO USO DE MEMORIA ASOCIATIVAS ALFA-BETA

Luis E. Alanís Carranza

Instituto Politécnico Nacional/Centro de Investigación e Innovación Tecnológica

lalanisc1600@alumno.ipn.mx

Moisés V. Márquez Olivera

Instituto Politécnico Nacional/Centro de Investigación e Innovación Tecnológica

mvmarquez@ipn.mx

Octavio Sánchez García

Instituto Politécnico Nacional/Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada

osanchez0112@gmail.com@ipn.mx

Viridiana G. Hernández Herrera

Instituto Politécnico Nacional/Centro de Investigación e Innovación Tecnológica

vhernandezhe@ipn.mx

Resumen

En la literatura se han utilizado muchos algoritmos de Inteligencia Artificial para el reconocimiento de textos en imágenes, algunos de estos métodos más utilizados son redes neuronales, las máquinas de vectores de soporte y el más común el reconocimiento de caracteres óptico por sus siglas en inglés (OCR). En este trabajo se presenta la utilización de un algoritmo mexicano para el reconocimiento de caracteres alfanuméricos llamado Memorias Asociativas Alfa-Beta programadas en el lenguaje de programación C#. Al entrenar el algoritmo con el método de validación K-Fold Cross Validation se obtuvo un índice de asertividad del 93% utilizando una base de datos de 10 patrones o imágenes por cada clase de números y letras con resolución de 100 x 200 pixeles. El método

propuesto muestra una alta competitividad contra otros sistemas de reconocimiento de caracteres.

Palabra(s) Clave: Caracteres, Reconocimiento, Memorias Asociativas.

Abstract

In the literature, many models of Artificial Intelligent (AI) have been used to text recognition in images. Some models more use are Artificial Neuronal Networks (ANN), support vector machine (SVM) and the most common Optical Character Recognition (OCR). This Work shows the use of a Mexican algorithm to alphanumeric character recognition called Memorias Asociativas Alfa-Beta, programming them in language C#. The algorithm was trained with K-Fold Cross Validation getting a 93% success rate. Our data base has 10 patters per every number and letter class with a resolution 100 x 200 pixels. The proposed method shows a high competitiveness against other character recognition systems.

Keywords**:** *Character, Memorias Asociativas, Recognition*

1. Introducción

Los crecientes avances científicos y tecnológicos en la línea de Inteligencia Artificial (IA) han permitido que en los últimos años sea posible implementar esta clase de algoritmos en la solución de diversos problemas, uno de ellos es el reconocimiento de caracteres alfanuméricos, cuya importancia radica en las múltiples aplicaciones que tendría un sistema que de forma automática fuera capaz de identificar y reconocer bajo diferentes condiciones de ángulo, iluminación y ruido diversos tipos de caracteres en letreros, placas de automóviles, texto escrito en letra de molde y cursiva [1] [2] [13], entre otras aplicaciones. No obstante, el problema no es trivial ya que para brindar una solución eficiente a este tipo de problemas es necesario no solo hacer uso de los modelos de IA ya que al trabajar con imágenes es necesario integrar y en su caso fusionar estos modelos con los enmarcados en la línea de Visión Artificial (VA).

La comunidad científica interesada en brindar una solución al reconocimiento automático de caracteres ha tenido en los últimos años resultados interesantes.

Arica et al. [1], se enfocan en el reconocimiento de letra cursiva poniendo un poderoso algoritmo de segmentación que separa los caracteres de una imagen, divide cada carácter en 3 partes y extrae las características más representativas de los caracteres, posteriormente emplean Modelos Ocultos de Markow (HMM por sus siglas en inglés) para el reconocimiento de los mismos. Chen et al. [3], proponen un método para reconocimiento de placas vehiculares en imágenes reales, el cual consiste en extraer las características de los caracteres haciendo uso de la variación del modelo Scale-invariant feature transform por sus siglas (SIFT). En el et al. [4], proponen un algoritmo hibrido para el reconocimiento de caracteres en placas vehiculares en donde se emplea K-Near-Neighbor (KNN por sus siglas en inglés) como primer clasificador y posteriormente para caracteres similares se hace uso de Máquinas de Soporte Vectorial (SVM por sus siglas en inglés). Kocer et al. [5], Desarrollo un sistema de reconocimiento de placas vehiculares, en la etapa de extracción de características de los caracteres utilizó Average Absolute Deviation y posteriormente para la parte de reconocimiento uso dos redes neuronales, con la finalidad de tener una menor probabilidad de confusión y mejorar el índice asertividad. Téofilo et al. [6], Proponen atacar la problemática que tienen los sistemas de reconocimiento de caracteres óptico en imágenes reales de diferentes países, para ello realizaron pruebas con distintos modelos de reconocimiento y el que mejores resultados dio al utilizar 15 patrones por clase fue Multiple Kernel learning (MKL). Ryan et al. [7], Realizaron un sistema de reconocimiento de caracteres para ID cards, para realizar la parte del reconocimiento utilizaron el modelo pixel-by-pixel ya que las plantillas que se obtienen por este modelo son menos ruidosas**.** Al-Yousefi [8]**,** Propusieron un sistema de reconocimiento de caracteres en Arabia en el cual utilizaron clasificadores Bayesianos ya que de acuerdo a este trabajo son de baja sensibilidad en las variaciones de los patrones por lo cual para patrones muy parecidos es muy útil.

En este trabajo se utilizó un algoritmo mexicano llamado memorias asociativas Alfa-Beta, las cuales creo el Doctor Cornelio Yáñez Márquez en el Centro de Investigación en Computo del Instituto Politécnico Nacional [14]. Actualmente se

han realizada muchos trabajos con respecto a este tema y han obtenido resultados relevantes. Israel Román [9] utilizo el algoritmo de memorias asociativas Alfa-Beta para resolver problemas de Bioinformática. Sánchez López y Torres Casanova [10] uso las memorias asociativas para el reconocimiento de dígitos escritos a mano y finalmente otro trabajo relevante fue el de Moisés Vicente [11] el cual consistía en el reconocimiento de expresiones faciales en video.

2. Métodos

En la figura 1 se muestran los pasos que conforman el método para el reconocimiento de caracteres:

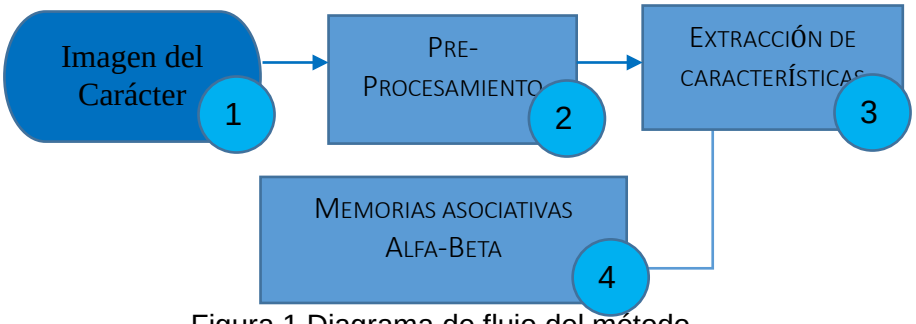

Figura 1 Diagrama de flujo del método.

Imagen de entrada

Las imágenes de entrada tienen una resolución de 100 x 200 pixeles a color, con diferente iluminación, ruido y ángulo. En la figura 2 se muestran algunos ejemplos de imágenes de caracteres que a simple vista parecen ser a escala de grises, sin embargo esto se debe a que son extraídos de textos en blanco y negro de una imagen RGB.

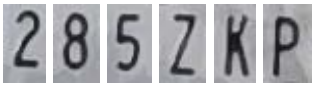

Figura 2 Muestras de imágenes de caracteres.

Pre-Procesamiento

En esta etapa las imágenes a color pasan a escala de grises, con la finalidad poder realizar una binarización. Es necesaria la binarización debido a que las memorias asociativas Alfa-Beta trabajan con operaciones binarias, es decir con valores de 0 y 1.

Escala de grises

Se utilizó el modelo YUV utilizado en [12] para realizar la conversión de una imagen en modo RGB a escala de grises. Este modelo utiliza los valores presentados en la tabla 1.

Tabla 1 Valores del modelo YUV.

Canal del color	Porcentaje %
Verde	59
Azul	30
Rojo	11

Lo que pretende este modelo es pasar una imagen de los tres canales de color rojo, verde y azul a un solo canal de grises. La función de este modelo consiste en sumar los valores que se obtienen al multiplicar el valor de los pixeles con su respectivo porcentaje y dividirlo entre el número de canales de RGB. En la figura 3 se muestra un ejemplo de los valores en cada canal en una imagen de color.

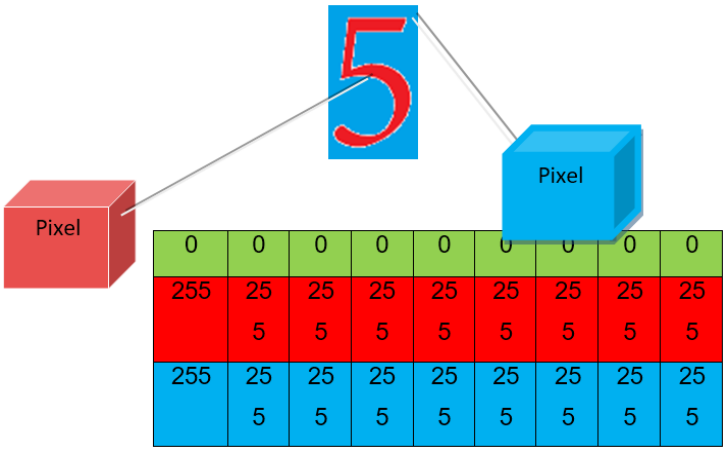

Figura 3 Valores de los pixeles de una imagen RGB.

Por lo tanto al utilizar el modelo YUV con los valores de la figura 3 se obtiene:

$$(0 * 0.59) + (255 * 0.30) + (255 * 0.11) = 0 + 76.5 + 28.05$$

$$\frac{104.55}{3} = 34.85$$

Finalmente al realizar este proceso en cada uno de los pixeles en la figura 4 se muestra el resultado al convertir la imagen de color a escala de grises después de realizar el cálculo con el modelo YUV.

34.85	34.85	34.85	34.85	34.85	34.85	34.85	34.85	34.85

Figura 4 Resultado en la imagen.

Binarización

Para realizar la binarización se toma en cuenta un umbral o un rango llamado U_{xy} y la imagen binarizada es llamada como $bi(x,y)$. Este método está dado por la ecuación 1.

$$bi(x.y) = \begin{cases} 1 & Si\ S(x,y) > U_{xy} \\ 0 & Si\ S(x,y) < U_{xy} \end{cases} \quad (1)$$

Donde $S(x,y)$ son las coordenadas del pixel de la imagen en escala de grises y al aplicar el algoritmo en esa imagen, figura 5.

Figura 5 Imagen binarizada.

Extracción de características

Cuando se realiza la binarización como se muestra en la figura 5, se procede a concatenar, esto significa convertir esta imagen en un vector o patrón que

permitan a las memorias asociativas Alfa-Beta aprender ese carácter. Para cada uno de los caracteres se utilizó 10 conjuntos de números, 23 de letras y estos conjuntos cuentan con 10 patrones, se muestra en la figura 6.

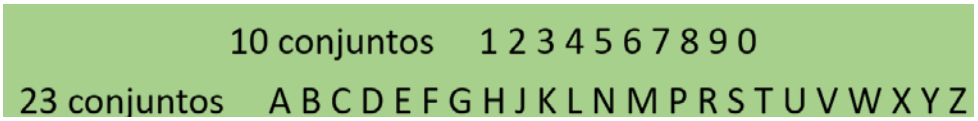

Figura 6 Caracteres de la base de datos.

La razón de por qué no se colocó la O, Q o la I es debido a que tienen un gran parecido con algunos otros conjuntos y es más probable que se confundan.

En la figura 7 se muestra como de una imagen binarizada es convertida en un patrón, el cual se utilizara para las memorias asociativas Alfa-Beta. Se realiza este proceso para cada uno de los patrones.

Para entender un poco mejor este pasó, en la figura 8 se muestra un ejemplo mejor explicado.

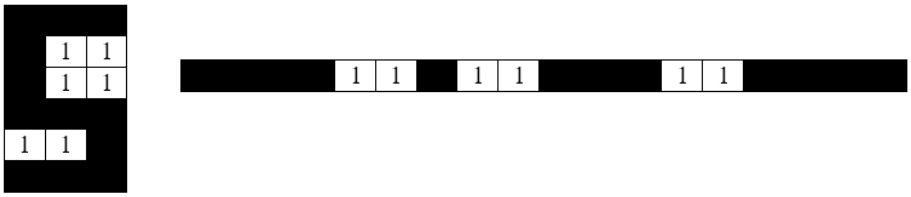

Figura 7 Patrón de entrada.

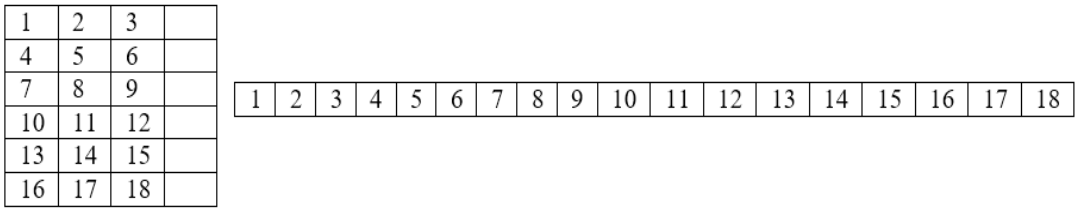

Figura 8 Explicación de la concatenación.

En la figura 8 se muestra como se forma ese patrón de entrada con base a la imagen que esta binarizada y el orden que debe de generarse. En pocas palabras una matriz bidimensional de m x n pasa a una matriz de una sola dimensión.

Memorias asociativas Alfa-Beta

Las memorias asociativas Alfa-Beta es un algoritmo de Inteligencia Artificial, la cual genera una memoria o matriz M conformada por información de patrones de entrada, figura 9.

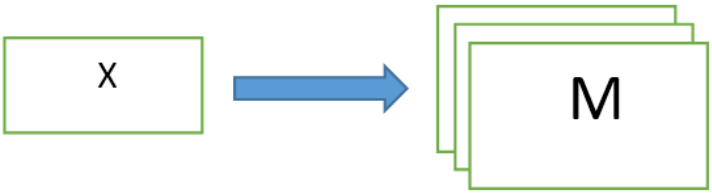

Figura 9 Representación de una memoria asociativa.

La memoria asociativa tiene como objetivo asociar un patrón de entrada X con uno de los patrones previamente aprendidos y dar como salida Y al conjunto o clase al que lo asocio. Este algoritmo es capaz de recuperar todos los patrones y por lo tanto tener un factor de olvido del 0%, figura 10.

Figura 10 Memoria asociativa recuperando patrón.

Fases de las memorias asociativas Alfa-Beta

Las memorias asociativas están conformadas de tres fases: La fase de aprendizaje, recuperación y prueba:

- **Fase de aprendizaje**: En esta etapa tiene como objetivo en crear la memoria asociativa con base a información de patrones de entrada. Esta matriz tiene los operadores necesarios para relacionar o asociar los patrones de entrada y salida.
- **Fase de recuperación:** Esta fase consiste en poner a prueba lo que aprendió de los patrones de entrada. Cuando se genera la matriz M

obtenida, se le muestra un patrón de los anteriormente aprendidos y se verifica si lo asocio al conjunto correspondiente.

- **Fase de prueba**: En esta fase se le muestra un patrón no aprendido a la matriz de aprendizaje y se verifica si acertó al conjunto correcto. Los patrones que entran a la matriz M suelen tener ruido aditivo por lo cual la probabilidad de que no asocie correctamente es mayor que en la de recuperación.

Operadores Alfa-Beta

Este modelo de Inteligencia Artificial utiliza dos operadores binarios llamados Alfa y Beta. En el caso del operador α es utilizado para la fase de aprendizaje y el operador β en la fase de recuperación y prueba. Cada uno de estos dos operadores tiene un conjunto y se denominaran como A y B. A continuación se especifican los números que se utilizan en cada conjunto.

$A = 0,1$ y $B = 0, 1,2$

Entonces para la realización de la operación α está dada por la tabla 2. Y el del operador β está dado por la tabla 3.

Tabla 2 Tabla del operador α donde $\alpha = AxA$.

X	Y	$\alpha(x,y)$
0	0	1
0	1	0
1	0	2
1	1	1

Tabla 3 Tabla del operador β donde $\beta = BxA$.

X	Y	$\beta(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Hasta el momento se han hablado sobre los conjuntos A y B, y de los operadores α y β. Sin embargo dentro de las memorias asociativas también es necesario la utilización de otros operadores llamados max $\wedge$ y min $\wedge$ que más adelante se explica su utilidad.

Fase de aprendizaje

En esta parte del artículo se hablara a detalle de cómo construir la matriz de aprendizaje M. Para ello los patrones binarios de entrada están denotados por $x_i^n|\forall_n \in \{1,2,\dots,p\}, \forall_i \in \{1,2,\dots,l\}$ donde p = número de patrones y l = tamaño del patrón. El primer paso es la obtención de la transpuesta de ese patrón que está dado por $y_i^m|\forall_m \in \{1,2,\dots,c\}, \forall_i \in \{1,2,\dots,l\}$ donde c = número de clases o conjuntos y l = número de patrones. Seguido se aplica el operador α de la tabla 2, ecuación 2 y 3.

$$x_i^n = \begin{pmatrix} x_1^n \\ x_2^n \\ . \\ . \\ . \\ x_l^n \end{pmatrix} \qquad y_i^m = \begin{pmatrix} y_1^m \\ y_2^m \\ . \\ . \\ . \\ y_l^m \end{pmatrix} \tag{2}$$

$$[M']^n = [x^n \, \alpha \, (y^m)^t] = \begin{bmatrix} x_1^n \\ x_2^n \\ \vdots \\ . \\ x_l^n \end{bmatrix} \alpha [y_1^m \;\; y_2^m \;\; \dots \;\; y_l^m] \tag{3}$$

Por lo tanto se tiene como resultado una matriz llamada $[M']^n |\forall n \in \{1,2,\dots,p\}, \forall i,j \in \{1,2,\dots,l\}$. Esto quiere decir que se debe realizar este proceso para cada uno de los patrones de entrada.

Se tendrán n cantidad de matrices dependiendo del número de patrones de entrada. Cuando se tienen las matrices se procede a utilizar el operador max $\vee$, ecuación 4.

$$v_{ij} = \bigvee_{n=1}^{p} [\alpha(x_i^n, (y_i^m)^t)]^n \tag{4}$$

El objetivo de este operador es en buscar el máximo o el valor más grande de cada una de las matrices en la posición α ($\mathbf{x}_i^n, \mathbf{y}_i^m$). Dando como resultado una sola matriz M la cual corresponde a la matriz de aprendizaje, ecuación 5.

$$\boldsymbol{M} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1l} \\ v_{21} & v_{22} & \cdots & v_{2l} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ v_{l1} & v_{2l} & \cdots & v_{ll} \end{bmatrix} \tag{5}$$

Fase de recuperación

Para obtener el patrón de salida, se utiliza el operador β utilizando la tabla 2 con la matriz M obtenida en la fase de aprendizaje y con el patrón de entrada que se desea clasificar y se obtiene como resultado la ecuación 6.

$$\boldsymbol{y_i^m} = \begin{bmatrix} \boldsymbol{\beta}(v_{11}, x_1^n) & \boldsymbol{\beta}(v_{12}, x_1^n) & \cdots & \boldsymbol{\beta}(v_{1l}, x_1^n) \\ \boldsymbol{\beta}(v_{21}, x_2^n) & \boldsymbol{\beta}(v_{22}, x_2^n) & \cdots & \boldsymbol{\beta}(v_{2l}, x_2^n) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \boldsymbol{\beta}(v_{l1}, x_i^n) & \boldsymbol{\beta}(v_{2l}, x_i^n) & \cdots & \boldsymbol{\beta}(v_{ll}, x_i^n) \end{bmatrix} \tag{6}$$

Cuando se realiza esta operación se procede a buscar los mínimos de la matriz utilizando el operador min $\wedge$.

$$\boldsymbol{y_i^m} = \begin{bmatrix} \bigwedge_{i=1}^{i} \boldsymbol{\beta}(v_{1i}, x_i^n) \\ \bigwedge_{i=1}^{l} \boldsymbol{\beta}(v_{2i}, x_i^n) \\ \cdot \\ \cdot \\ \cdot \\ \bigwedge_{i=1}^{l} \boldsymbol{\beta}(v_{li}, x_i^n) \end{bmatrix} \tag{7}$$

Se obtendrá como salida un patrón con las mismas dimensiones que el de entrada. Finalmente se comparara patrón de salida con los de la base de datos y lo clasificara a un conjunto según el parecido.

K-Fold Cross Validation

Para poder entrenar el algoritmo de memorias asociativas Alfa-Beta con la base de datos de los caracteres, se requiere de un método de validación que

pueda dar un porcentaje de cuentas veces no clasifico bien el patrón de entrada. Un método de validación consiste en dividir la base de datos en dos conjuntos, el de prueba y el fundamental:

- Conjunto fundamental: Se utiliza en la fase entrenamiento y de recuperación de las memorias asociativas para obtener un factor de olvido, es decir cuánto olvido el algoritmo.
- Conjunto prueba: Se utiliza en la fase de prueba de las memorias asociativas Alfa-Beta para obtener un índice de asertividad.

Como se había mencionado anteriormente este método consiste en tener que dividir la base en dos conjuntos. Este método realiza un total de 10 rondas para dividir la base de datos ya que toma el 10% de la base y la coloca en el conjunto prueba y el 90% restante va para el conjunto fundamental, figura 11. Para la segunda ronda se presenta en la figura 12.

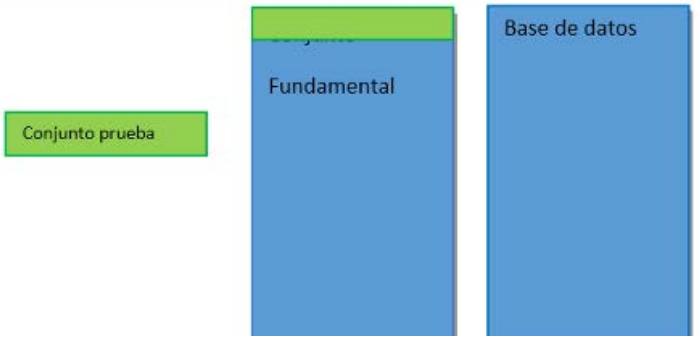

Figura 11 Toma del 10%

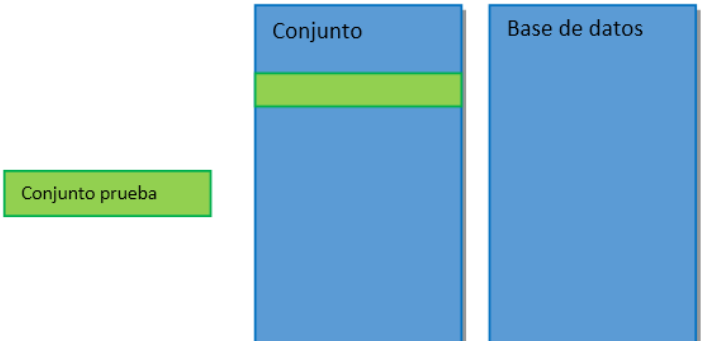

Figura 12 Toma del segundo 10%.

El método termina hasta que se terminen las 10 rondas y por cada ronda obtiene un índice de asertividad el cual indica que tanto el algoritmo no clasifico bien. Finalmente se tienen 10 índices de asertividad y se promedian para conocer el resultado final. Este proceso se utiliza en la fase de recuperación y de prueba.

Clasificación

El algoritmo de memorias asociativas obtiene un patrón de entrada como el de la figura 7 y aplica el operador β para obtener el patrón de salida. Finalmente ese patrón se compara con los demás patrones de la base de datos. Para clasificar ese carácter con algún conjunto se cuanta el número de veces que coinciden los ceros y los unos, figura 13.

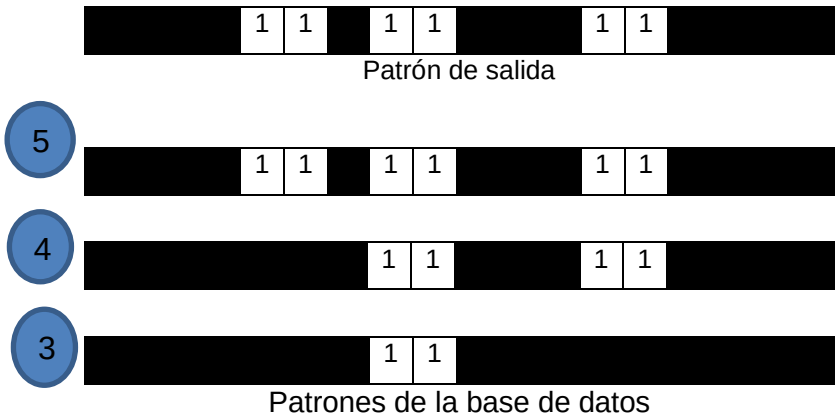

Figura 13 Ejemplo de cómo se clasifica.

En el caso de la figura 13 el patrón de salida coincide más veces con el del conjunto 5 y por lo tanto el algoritmo lo clasifica como un cinco.

3. Resultados

Fase de recuperación

Como se mencionó en el capítulo 2 se utilizó el algoritmo de K-Fold Cross Validation para obtener el factor de olvido. Este factor mientras más pequeño sea el porcentaje mejor y determinara cuanto el algoritmo no pudo recuperar los

patrones previamente aprendidos. Para ello se mostraran algunos ejemplos de los caracteres utilizados para conformar la base de datos, figura 14.

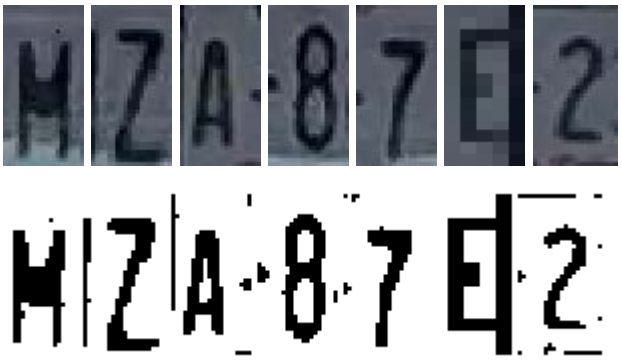

Figura 14 Pre-Procesamiento de la imagen.

Al utilizar el algoritmo de K-Fold Cross Validación con las memorias asociativas para obtener el factor de olvido se obtuvo un porcentaje del **0%.** Esto quiere decir que el algoritmo pudo clasificar correctamente todos los patrones previamente aprendidos a comparación de otros algoritmos.

Fase de prueba

Después de que el algoritmo fue entrenado se procede a ponerlo a prueba mostrándole patrones que nunca aprendió y así poder conocer que tan asertivo es el algoritmo al clasificar un patrón no aprendido. Esto es a lo que se le conoce como índice de asertividad. Se realizó la prueba con 10 patrones de cada conjunto y al utilizar el método de validación K-Fold Cross Validation se obtuvo un índice de asertividad del **93%**.

4. Discusión

Al utilizar las memorias asociativas Alfa-Beta en un sistema de reconocimiento de caracteres muestra que puede ser competitivo con respecto a otros métodos, sin embargo al utilizar una resolución de 100 x 200 pixeles los costos computacionales son mucho mayores y por lo tanto el tiempo de respuesta es elevado. Para ello se puede utilizar una resolución mucho más pequeña para reducir el tiempo de respuesta y utilizar una mayor cantidad de patrones por clase

para aumentar el índice de asertividad. El método propuesto puede solventar algunos de los problemas que tienen los sistemas de reconocimiento de placas, en el reconocimiento de textos en imágenes y en la identificación automática de ID cards. Por lo tanto el método puede utilizarse en varias aplicaciones y es eficaz en el reconocimiento de caracteres pero con un alto costo en tiempo de respuesta.

5. Bibliografía y Referencias

[1] N. Arica y F. T. Yarman-Vural, Optical character recognition for cursive handwriting, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, nº 6, pp. 801-813, 2002.

[2] S.-W. Lee, D.-J. Lee y H.-S. Park, A new methodology for gray-scale character segmentation and recognition, IEEE transactions on pattern analysis and machine intelligence, vol. 18, nº 10, pp. 1045-1050, 1996.

[3] H. Chen, B. Hu, X. Yang, M. Yu y J. Chen, Chinese character recognition for LPR application, Optik-International Journal for Light and Electron Optics, vol. 125, nº 18, pp. 5295-5302, 2014.

[4] S. S. Tabrizi y N. Cavus, A hybrid KNN-SVM model for Iranian license plate recognition, Procedia Computer Science, vol. 102, pp. 588-594, 2016

[5] H. E. Kocer y K. K. Cevik, Artificial neural networks based vehicle license plate recognition, Procedia Computer Science, vol. 3, pp. 1033-1037, 2011.

[6] T. de Campos, B. R. Babu y M. Varma, Character recognition in natural images, 2009.

[7] M. Ryan y N. Hanafiah, An Examination of Character Recognition on ID card using Template Matching Approach, Procedia Computer Science, vol. 59, pp. 520-529, 2015.

[8] H. Al-Yousefi y S. Udpa, Recognition of Arabic characters, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, nº 8, pp. 853-857, 1992.

[9] Román Godínez, Aplicación de los modelos asociativos Alfa-Beta a la Bioinformática, 2007.

[10] S. Lopez, E. Bernabe y E. E. Torres Casanova, Aplicacion de los modelos asociativos alfa-beta en el reconocimiento de digitos escritos a mano, 2009.

[11] M. V. Márquez Olivera, Reconocimiento de expresiones faciales empleando memorias asociativas Alfa-Beta, 2016.

[12] M. R. Asif, Q. Chun, S. Hussain, M. S. Fareed y S. Khan, Multinational vehicle license plate detection in complex backgrounds, Journal of Visual Communication and Image Representation, vol. 46, pp. 176-186, 2017.

[13] J. Mao, P. Sinha y K. Mohiuddin, A system for cursive handwritten address recognition, de Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on, 1998.

[14] C. Yánez-Márquez, Associative memories based on order relations and binary operators (in spanish). 28, 2000.