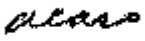
4.1 Generación de los Archivos de Patrones.

Para generar los archivos de patrones utilizados en esta investigación, primero se analizaron las muestras que se tenían anteriormente, existen dos grupos de imágenes de palabras [Linares & Spínola 00], las que están tal y como fueron extraída desde el documento, y el otro grupo son las mismas palabras, solo que fueron pre-procesados (se modifico la inclinación) como se muestra en la figura 4.1.



a) Normal (acaso.gif)

b) Retocada (acaso_s.gif)

Figura 4.1 Muestra de imágenes a) original. b) retocada

Este paso tuvo que hacerse para obtener una mejor segmentación y posteriormente un mejor reconocimiento. Realizamos un conteo de cada letra que compone a la palabra y este fue el resultado.

	a	b	c	d	e	f	g	h	i	J	k	l	m	N	o	p	q	r	s	t	u	v	w	x	y	Z
Total	55	4	26	32	65	2	9	6	37	5	0	24	13	36	41	9	4	30	32	18	26	4	0	1	0	2

Tabla 4.1. Imágenes sin pre-procesamiento con un total de 481 de letras.

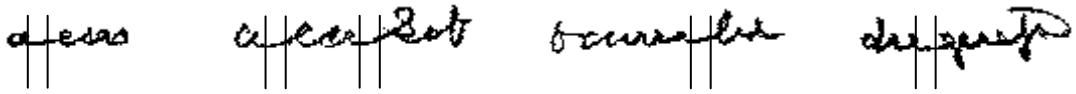
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	Z
Total	52	4	26	30	61	2	8	5	34	4	0	23	12	34	40	8	4	29	32	17	25	4	0	1	0	1

Tabla 4.2. Imágenes con pre-procesamiento con un total de 456 letras.

Cabe señalar que varias imágenes no fueron pre-procesadas es por eso que no corresponden los totales en cada grupo. Puede observarse que no se tienen un buen conjunto de letras e incluso no existe muestra de varias (k, w, y), en otros casos si se tienen un buen conjunto de muestras. Al final en el anexo C se muestra la lista de palabras de muestra así como el conteo de cada letra.

Posteriormente se realizó la segmentación manual de cada imagen y se observó lo siguiente.

Para la imagen de aesas_s.gif (figura 4.2 a) se tuvo que segmentar de la siguiente forma. Como se observa en la figura 4.2 a) está unida la a con la palabra “esas”, por lo tanto se segmentó ese espacio que debe corresponder al espacio en blanco. Lo mismo paso para la imagen aesegob_s.gif (figura 4.2 b), ocurraud_s.gif (figura 4.2 c), suqueja_s.gif (figura 4.2 d) en esta ultima imagen también adorna la letra a.



a) b) c) d)

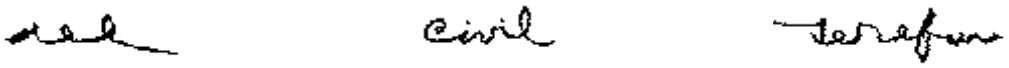
Figura 4.2 Imágenes con palabras unidas.

También existen imágenes donde el escritor dejó mucho espacio entre letras en una palabra, un ejemplo es de_s.gif (figura 4.3 a) y con_s.gif (figura 4.3 b).

a) b)

Figura 4.3 Imágenes con mucha separación entre letras.

Otro problema también presentaron las imágenes con cola tanto al inicio como al final como se observa en la figura 4.4.



a) b) c)

Figura 4.4 Imágenes con cola.

Cabe señalar que en la figura 4.4 c), la palabra “refiere” esta cortada, solo esta la parte de “refie”. Esto también sucedió en las imágenes respecto_s.gif, indígenas_s.gif, corresponde_s.gif, selecciones_s.gif.

Además que en imágenes como intrigado_s.gif, libertad_s.gif, luis_s.gif, por_s.gif, se observó que una letra ocupa espacio de la siguiente letra como se puede ver en la figura 4.5. Se nota que en la figura 4.5 a) la letra “t” ocupa espacio de la letra “r”.

a) b) c) d)

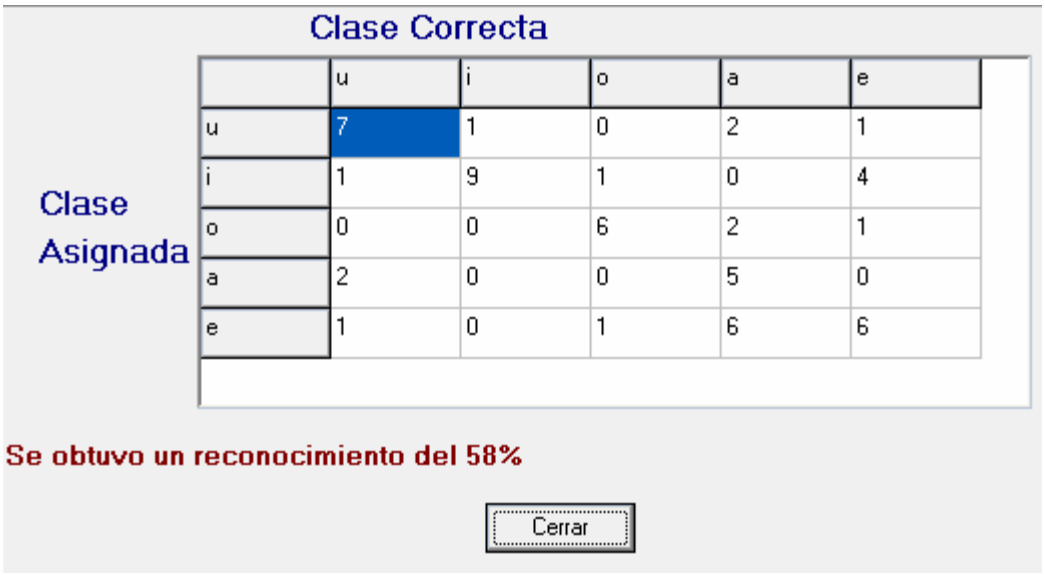
Figura 4.5 Sobre posición de letras.

Una vez realizado este pre - procesamiento, se crearon los archivos para calcular los centroides por medio de K-Means, y el de entrenamiento de la red de *Kohonen*. Cabe mencionar que el caracter normalizado tiene una dimensión de 12 renglones por 18 columnas, que da un total de 216 puntos. Que serán lo nodos de entrada para la red, y el número de coordenadas para el K-Means y Vecino Más Cercano.

4.2 Resultados usando Vecino más Cercano

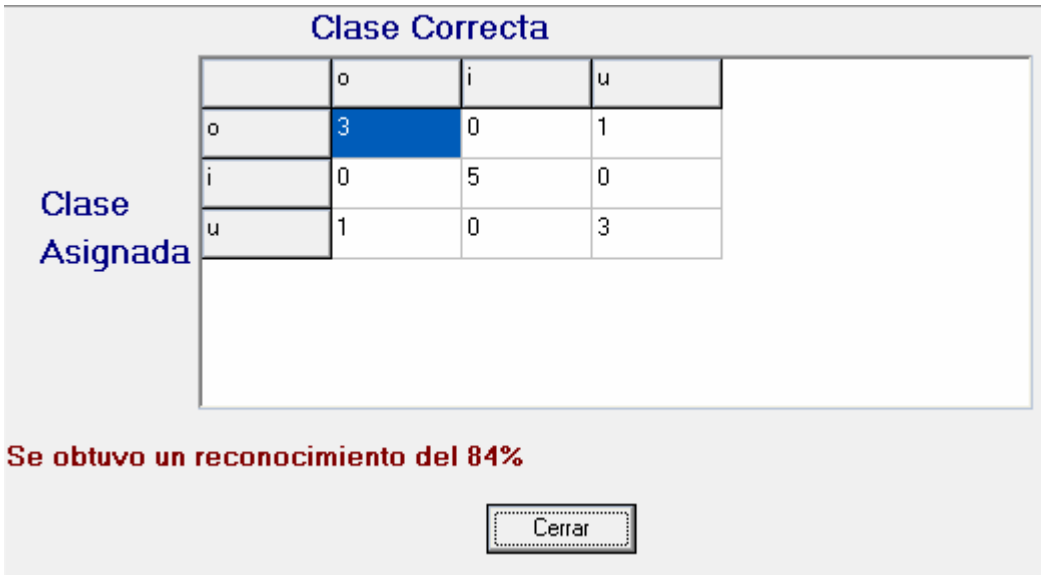
En este caso se probó con un conjunto de 56 letras (sólo vocales), el cual dió como resultado un 58% (ver figura 4.6 matriz de confusión), se presenta una matriz donde se observa que la letra “a” la confundió con la letra “e” 6 de 15 veces. De igual manera la letra “e” la confundió con la “i” 4 de 12 veces. Estas dos letras “a” y “e” fueron las que tuvieron más problemas. Se hizo otra prueba con 13 letras (solo las vocales “i”, “o” y “u”) (ver

figura 4.7), con lo que se obtuvo un resultado de 84%, sólo tuvo 2 confusiones con la letra "u" y la "o".



Clase Asignada \ Clase Correcta	u	i	o	a	e
u	7	1	0	2	1
i	1	9	1	0	4
o	0	0	6	2	1
a	2	0	0	5	0
e	1	0	1	6	6

Figura 4.6 Matriz de Confusión obtenida con el Vecino Más Cercano.



Clase Asignada \ Clase Correcta	o	i	u
o	3	0	1
i	0	5	0
u	1	0	3

Figura 4.7 Matriz de Confusión obtenida con el Vecino Más Cercano con las vocales "i", "o", "u".

Se realizó una prueba con un conjunto de patrones compuesto por 86 letras (a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, q, r, s, t, u, v), dando un resultado muy bajo del solo 6% de reconocimiento.

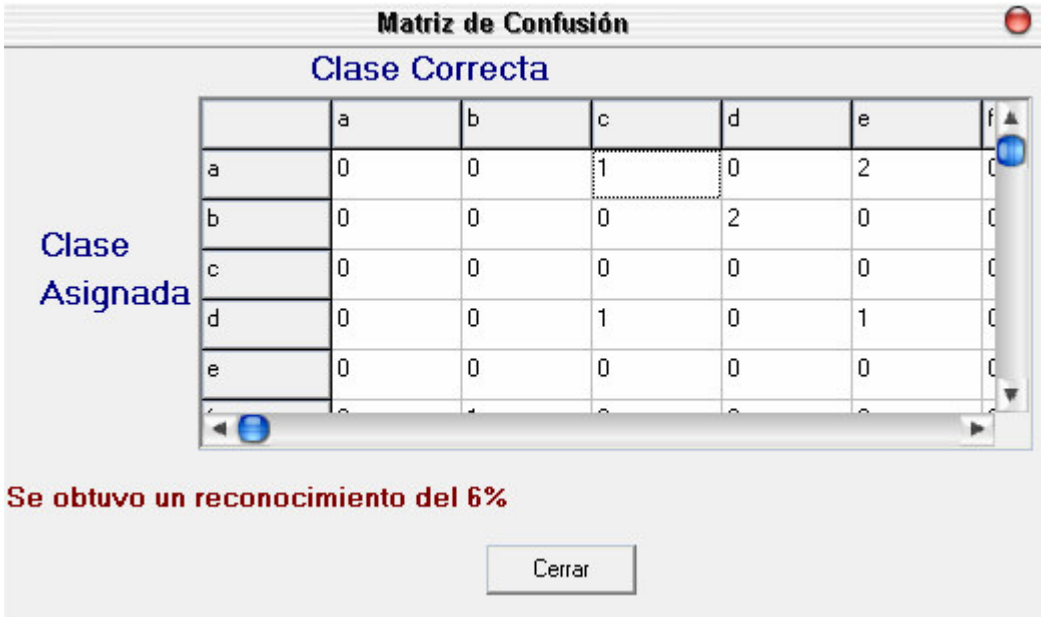


Figura 4.8 Matriz de Confusión obtenida con el Vecino Más Cercano.

4.3 Resultados usando la Red de *Kohonen*

Para todas las arquitecturas de la redes la entrada es de 216 neuronas.

Se usaron los mismos archivos de patrones para el entrenamiento de la red y se probaron con estos mismo dando como resultado:

Para el conjunto de 56 letras vocales, y una red de *Kohonen* de 5 X 2 neuronas de salida, se muestra su matriz de confusión en la siguiente figura (ver figura 4.9).

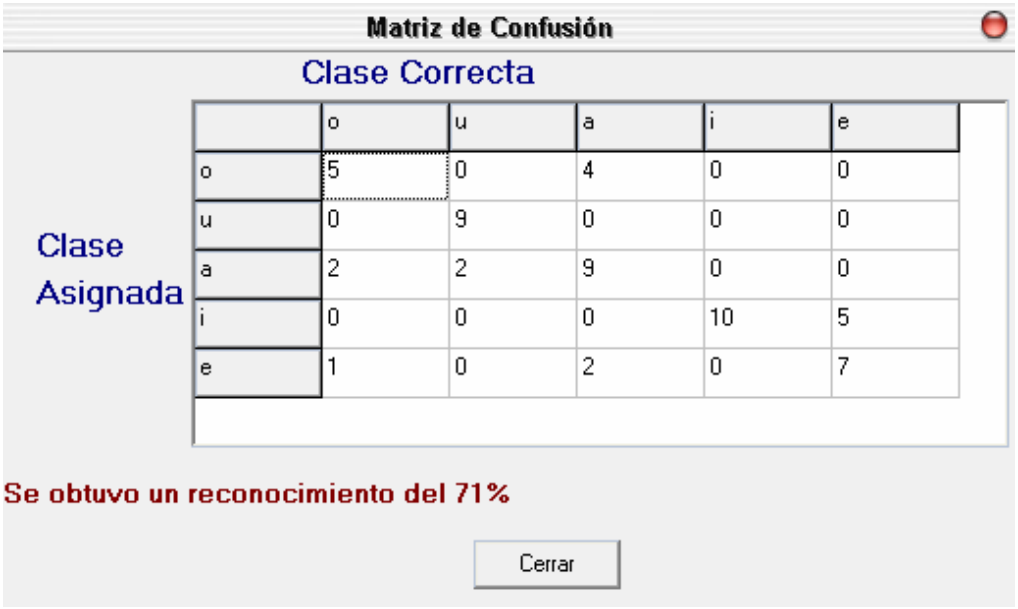


Figura 4.9 Matriz de Confusión obtenida con la Red de *Kohonen*.

Se obtuvo un reconocimiento de 71% que fue mayor que el vecino más cercano con un 58%.Se probó con una red con 5 X 5 neuronas de salida, dando un resultado del 73% de reconocimiento como se muestra en la figura 4.10.

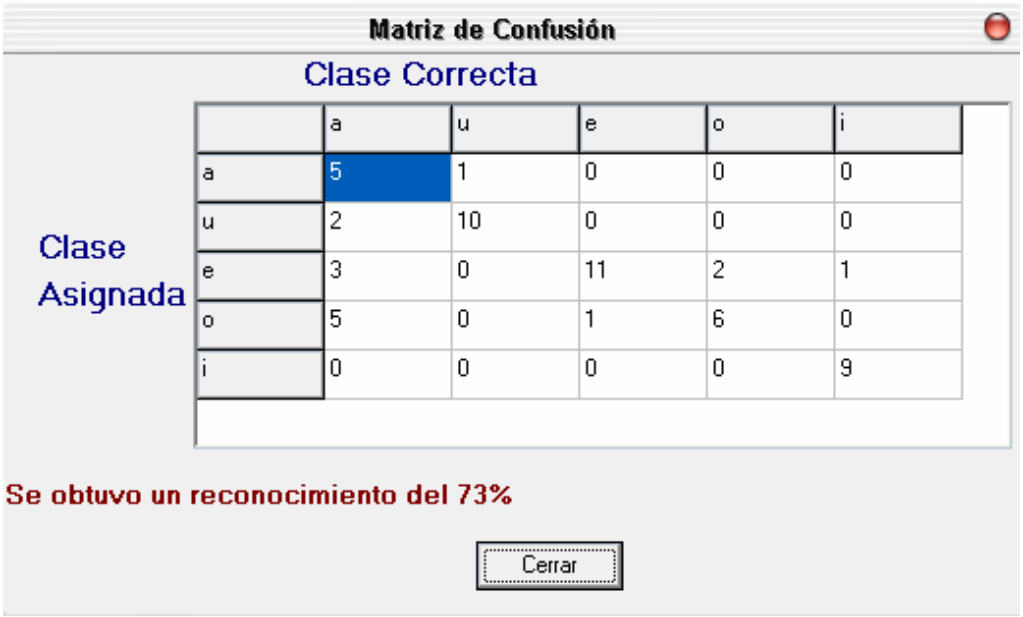
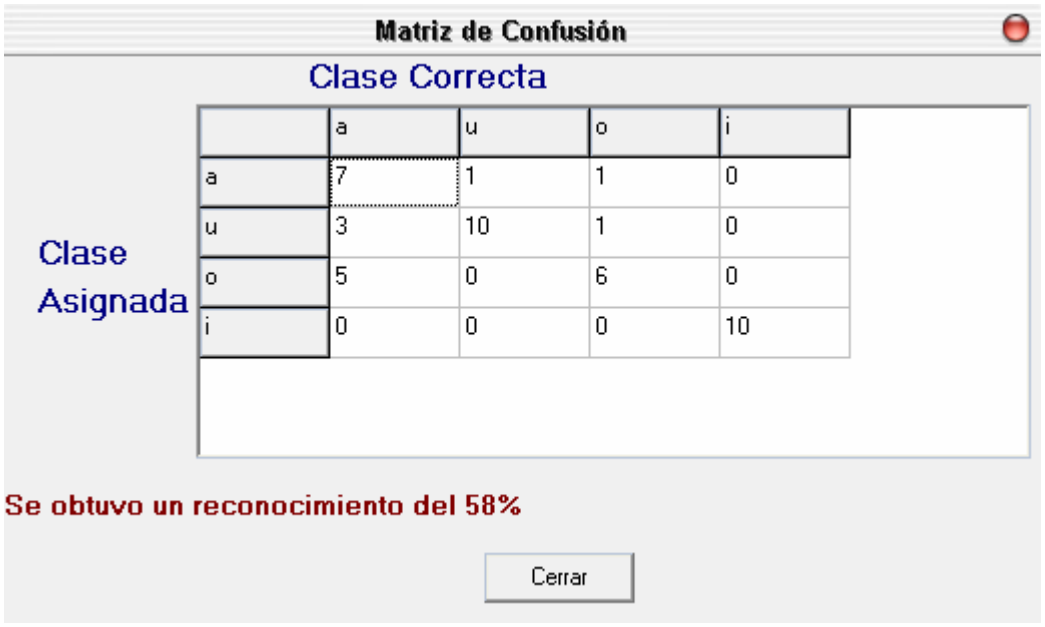


Figura 4.10 Matriz de Confusión de la Red de *Kohonen* con 5 X 5 neuronas de salida.

El mapa de caracteres generado por la red es el siguiente:

Figura 4.11 Mapa de Caracteres generado por la Red de *Kohonen* con 5 X 5 neuronas de salida.

Después se probó con una red con 5 X 1 neuronas de salida (ver figura 4.12), esta arrojó un resultado del 58% que es casi similar al del obtenido con el Vecino Más Cercano. Este porcentaje se debe a que no hay una neurona que represente a la vocal "e", como se muestra en la figura 4.13 que es el mapa autoorganizado de la Red de *Kohonen* obtenido después del entrenamiento.



	a	u	o	i
a	7	1	1	0
u	3	10	1	0
o	5	0	6	0
i	0	0	0	10

Figura 4.12 Matriz de Confusión de la Red de *Kohonen* con 5 X 1 neuronas de salida.

0000000000000000000	0000000000001110000	0000000000000000000	0000000001111000000	0000000000000000000
0000111111000000000	0000100000001110000	0000001010000001000	0000000011111000000	0000000000100000000
0000111100011100000	0011111000011111100	0000011100000011000	0000000111111111000	0000000000000000000
0111110000000100000	0111110000011111100	0000011100000011000	0000000111111111100	0000000000000000000
0111110000011111000	0111110000000111100	0000011100000011000	0000001111111111100	0000000000000000000
0111000000111100000	0111011000000111100	0001111100000011110	0000011111111111111	0000000000000000000
0111000000111100000	1111111000111111110	0001111100000111000	0000111111111101111	0000000010000000000
1111000000111100000	1100110000111111100	0011111100111111110	1111111111001010000	0000000111100000000
0111110101111111000	1100111111111101110	0111111111111111111	0001101111001111000	0000111111111000000
0111111111111011100	0000011111111100011	1111111111111100111	0000001111111111110	0111111011111011111
0111111111000001111	0000000111000000000	1111011111111000000	0000000011110000000	1111111000111111111
0000111100000000000	0000000010000000000	1000001111111000000	0000000011110000000	1110000000000111100

Figura 4.13 Mapa Autoorganizado de la Red de *Kohonen*, con 5 X 1 neuronas de salida.

Se hizo la misma prueba con 13 letras (solo las vocales “i”, “o” y “u”) como en el Vecino Más Cercano. En este caso dio un porcentaje del 92% de reconocimiento (ver figura 4.14). Aquí se usaron 3 X 3 neuronas de salida.

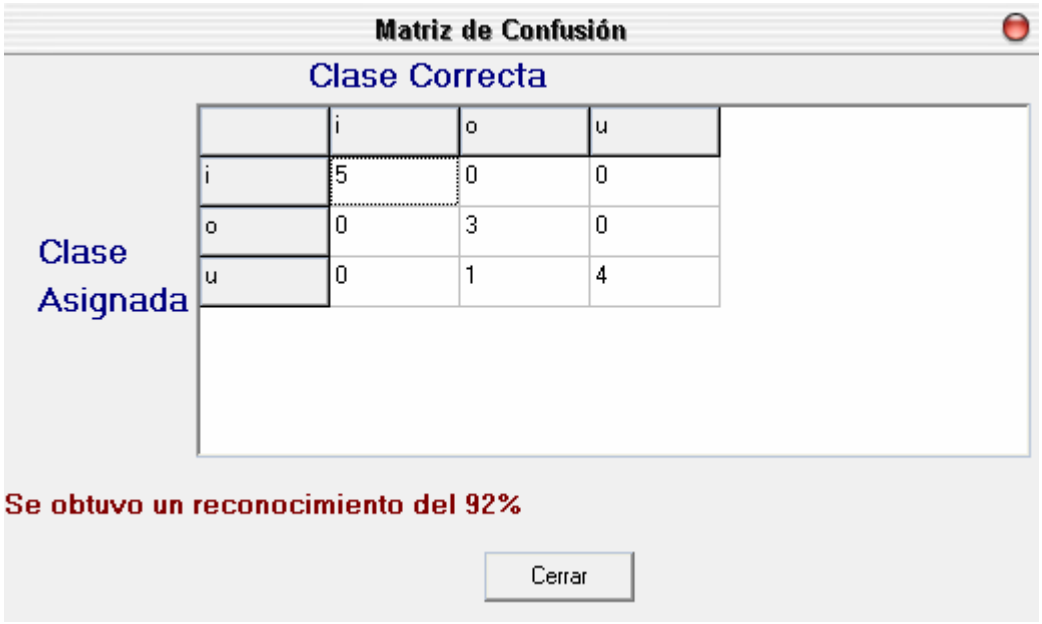


Figura 4.14 Matriz de Confusión obtenida con la Red de *Kohonen* con las vocales “i”, “o”, “u”.

A continuación se muestra (figura 4.15) las letras (vocal “a”) usadas para estas pruebas:

000000000000100001	000011110000000000	000000000000011100	000000001111111100	000001110000000000
000000111100111001	000111111100000000	000000000000011100	000000001111111100	000001111000000000
000111111100111001	000111101110000000	000111111111111100	000000111111111111	000011111000000000
001110011001100001	011110000011110000	001111100011111111	000011110000111111	000011100100000000
001111011001111001	011100000001110000	001110000011111100	000000110000111111	011110001100011100
011110001111100000	011100000011110000	111110000011111100	000000110011111111	011100000001111100
011110001111011001	011100000011110000	111110000011111100	000011111111111111	011100000001111100
111100111110011111	011100000011110000	111110000011111100	000011111111001111	111100000011110000
111100111000000110	110000000001110000	111111111111111111	111111111111001111	111000000111100000
111111100000000000	011100000111111001	001111111111100111	111111000000001111	011100001111110000
011111100000000000	001100001100111111	000111110000000000	110000000000000011	011101111100011101
011000000000000001	001111111100001100	000111110000000000	110000000000000011	001111111000001111
000000011000000000	000011111111100000	000000011111100000	000001011110000000	000000000001111110
000000011100000000	000011111111110000	000001111111111000	000011111111100000	000011111111111110
000000111111100000	001111111001111000	000111111000111100	001111100000110000	000111111111111110
000001111111111000	001111100000111100	001111100001111110	011110000000111000	011111110000001110
000001111101111000	011111000000111000	001111100000011110	111100000001111100	111111100000001111
000111111101111100	011110000001111000	011111000000011110	011000000001111100	111111000000111111
011111111101111100	111100000001110000	111111000001111110	011000000000111000	111100000001111110
011111111001111100	111110000001110000	011111001111111110	110000000111111000	111100000011111000
011111111000111110	011100000001111000	000111111111000111	011000011111111111	111100011111111110
011111111000111111	011100000111111100	000000000000000001	001101111111101111	111000011110001111
111001111110111111	001110011110000110	000000000000000111	000111111110000001	111111111110000011
000000111111111100	000011111100000001	000000000000000001	000001110000000000	011111111100000000
000000111111110001	000000001100000000	000000000011000000	000000000111000000	000000000011110000
000000111111111000	000001111110000000	000000011111100000	000000001111111000	000000011111111000
000000111111111000	000011111111100000	000000011111110000	000000001111111000	000000011111111000
000011111111111000	000011111111100000	000000011111110000	000000001111111000	000011111111111000
000011110111111111	000011100011110000	000001111111110000	000000011111111110	000011111000111110
000011101111111111	000011100011110000	000001110011110000	000000111111111111	011111111000111110
000011101111111111	000011000111110000	011101110011100000	000000111111111111	011111111000111110
000011111111001111	000111000111110000	111111110111110000	000011111101111111	111111111111111111
111111111110001101	011111010111111000	111101111111111111	000111111111110011	111011111111000011
111111111100000001	011111010111111000	111101111111111111	111111110111100011	000000111110000000
111111111100000001	111111111110011111	001100111110011111	111111110111100011	000000111110000000
000000011000000001	111011111100000111	000000011000000110	000001111100000000	000000011000000000

Figura 4.15 Letras usadas para la construcción del archivo de entrenamiento y prueba.

Como se puede observar la misma vocal esta representada de muchas formas, esto hace que ambos reconocedores tengan poco porcentaje de reconocimiento.

Se realizó una prueba con un conjunto de patrones compuesto por 86 letras (a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, q, r, s, t, u, v), dando un resultado de 63% de reconocimiento (ver figura 4.16). La letra “c” no las pudo reconocer, confundiendo con la “l” y la “v”. La arquitectura de la red fue de 5 Renglones por 12 Columnas neuronas de salida.

Modificando la arquitectura de la red ahora con 2 Renglones por 30 Columnas reconoció un 70%, la letra con más confusión fue la “n” (ver figura 4.17).

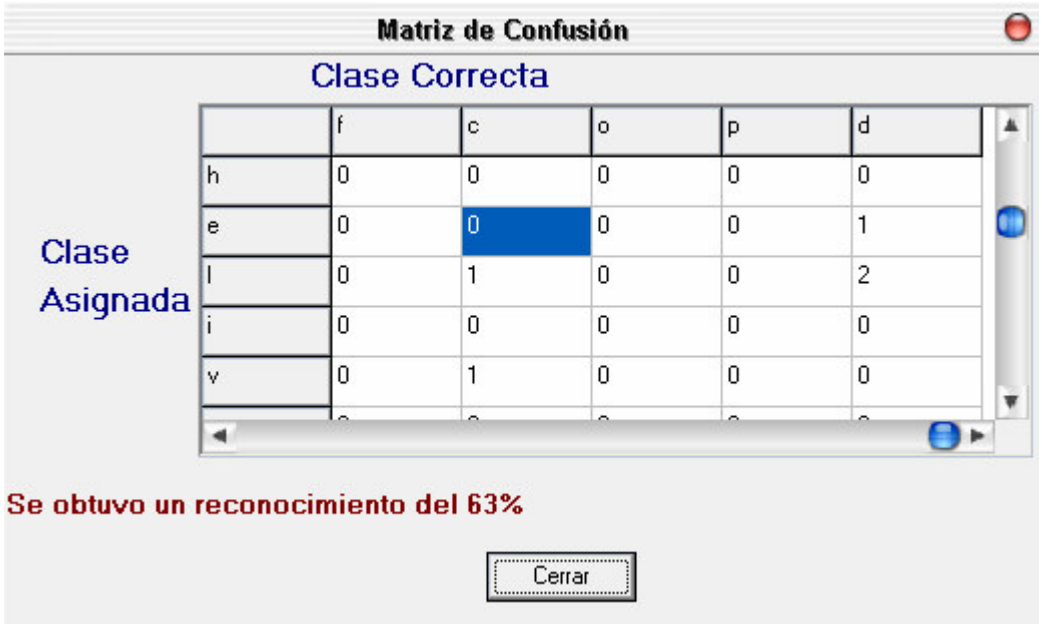


Figura 4.16 Matriz de Confusión obtenida con la Red de *Kohonen*, con 5 renglones X 12 Columnas neuronas de salida.

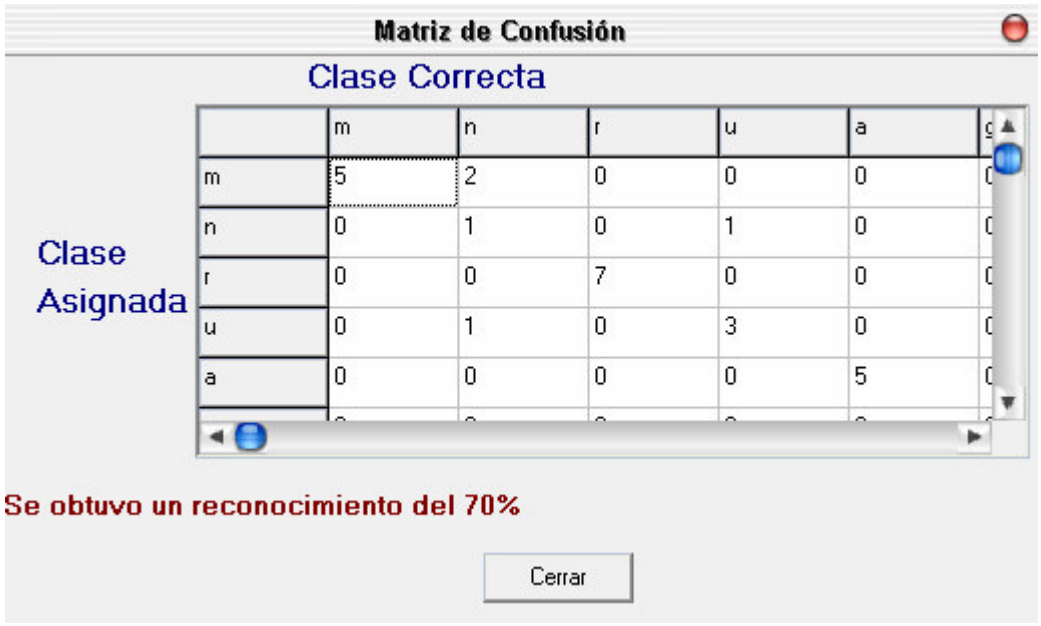


Figura 4.17 Matriz de Confusión obtenida con la Red de *Kohonen*, con 2 renglones X 30 Columnas neuronas de salida.

A continuación se muestra los Mapas de Caracteres generados por la red de *Kohonen* con diferentes topologías.

Figura 4.18 Mapa de Caracteres generado por la Red de *Kohonen* con 5 renglones X 12 columnas neuronas de salida.

Figura 4.19 Mapa de Caracteres generado por la Red de *Kohonen* con 2 renglones X 30 columnas neuronas de salida.

Se puede observar que la red esta realizando los clusters, esto es, que las letras parecidas están relativamente cerca una de otra.

4.4 Conclusiones.

Se realizó el estudio y la implementación de dos algoritmos de reconocimiento de patrones (Vecino Más Cercano, Red de *Kohonen*), los cuales dieron resultados regulares alrededor del 71% de reconocimiento la red de *kohonen* y el 50% el Vecino Más Cercano; esto se debe principalmente a que las letras tienen muchas inconsistencias de una palabra a otra palabra, e incluso dentro de la misma palabra.

El Vecino Más Cercano es un algoritmo supervisado, esto es, que tiene que saber de antemano cual es el prototipo que representa a cada clase. Para esto se implementó el algoritmo de *K-Means*, que se encarga de calcular los centroides, que son los representantes de cada clase, entrada del clasificador Vecino Más Cercano.

En las pruebas se notó que las vocales "a" y "e", confundían bastante al reconocedor, como se muestra en los resultados parciales. Cuando se eliminaron estas vocales, se incrementó el porcentaje de reconocimiento considerablemente, una posible solución sería crear subgrupos de cada letra, ya que existen muchas variantes de la misma letra.

El reconocedor de la Red de *Kohonen*, obtuvo mejores resultados que el Vecino Más Cercano. Se comporto mejor cuando se hicieron las pruebas con las mayorías de las letras dando un aceptable porcentaje de reconocimiento del 70%. Cabe hacer notar que observamos que los conjuntos de entrenamiento y pruebas tuvieron pocos ejemplos. Esto se debe a que en el conjunto de imágenes no es lo suficiente extensa, como puede observarse en la tabla "**Lista de Archivos de Imágenes Retocada"** en el Anexo C. En este se encuentra el conteo para cada letra, y cabe mencionar que no se cuenta con muestras de todas ellas y que algunas tienen muy pocos patrones.

Consideramos que la red de *Kohonen* ofrece mejor reconocimiento que el Vecino Más Cercano porque la red de *Kohonen* puede almacenar más información (según el número de nodos de salida) y en el Vecino Más Cercano solo se tiene un representante para cada clase y existen clases muy parecidas.

Cabe mencionar que esta es una investigación abierta todavía, y falta mucho por experimentar tanto con las técnicas desarrolladas como con otras más (por ejemplo aplicar técnicas de procesamiento de imágenes).

4.5 Trabajos Futuros.

Cabe mencionar que todavía falta mucho por realizar en el sistema, empezando desde la unión de la herramienta realizada por [Navarrete 02] hasta los aquí implementados. En el proyecto completo falta de resolver el problema de la extracción de palabras, líneas del documento original, y la eliminación del ruido en las imágenes, ya que esto se realiza manualmente.

También falta realizar más pruebas en la red de *Kohonen*, por ejemplo cambiando el tamaño de la matriz de cada letra, ya que actualmente es de 18 renglones por 12 columnas, a uno más grande por ejemplo 60 renglones por 40 columnas, se podrían probar con otras formas de organizar los nodos de salida (renglones por columnas) en la arquitectura de la red de *Kohonen*.

Puede agregarse la comparación de la salida del reconocedor de caracteres contra un diccionario, esto serviría para incrementar a un más el porcentaje de reconocimiento o en su caso implementar una gramática para la verificación de la sintaxis.