

# RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR) CON REDES NEURONALES ESTADO DEL ARTE

## OPTICAL CHARACTER RECOGNITION (OCR) WITH NEURAL NETWORKS STATE OF THE ART

Juan Pablo Ordóñez L.  
Loja  
086244139  
[jpordonez@utpl.edu.ec](mailto:jpordonez@utpl.edu.ec)

### RESUMEN

Los sistemas que, a partir de un texto escrito o impreso en papel o similar, crean un fichero de texto en un soporte de almacenamiento informático, se denominan Sistemas de OCR (Optical Character Recognition), o de Reconocimiento óptico de Caracteres. Un sistema OCR cuenta con las siguientes características: de poder "aprender", mediante una red neuronal, patrones de caracteres que representen las posibles variaciones (tamaño) de la forma de los diferentes caracteres impresos que pueden aparecer en los documentos, para en el futuro y con la misma red, poder "reconocerlos" y realizar la conversión del texto escrito en papel a texto almacenado en un fichero ASCII.

**PALABRAS CLAVE:** Ocr, Redes neuronales, Reconocimiento de manuscritos.

### ABSTRACT

Systems that, from a written or printed on paper or similar, creating a text file on a storage medium for computer systems are called OCR (Optical Character Recognition). An OCR system has the following characteristics: can "learn" through a neural network, patterns of characters representing the possible variations (size) as the different characters that can appear in documents, so in the future and with the same network, able to "recognize" and the

conversion of written text on paper to text stored in an ASCII file.

**WORKS KEY:** Ocr, network neuronal, Recognition of manuscripts.

### HISTORIA DE OCR

En 1929, Gustav Tauschek obtuvo una patente sobre OCR en Alemania, luego, Handel en 1933 obtiene la patente de OCR en EEUU. En 1935, a Tauschek también se le concedió una patente en EEUU por su método.

La máquina de Tauschek era un dispositivo mecánico que utilizaba plantillas. Un foto-detector era colocado de modo que cuando la plantilla y el carácter que se reconocería estuvieran alineados, una luz era dirigida hacia ellos.

En 1950, David Shepard, criptoanalista en la agencia de seguridad de las fuerzas armadas de los Estados Unidos, fue consultado por Rowlett Franco para trabajar con el Dr. Louis Tordella, para recomendar los procedimientos de la automatización de los datos de la agencia. Esto incluía el problema de convertir mensajes impresos en lenguajes para almacenarlos en un computador. Shepard

decide que es posible construir una máquina para realizar ese proceso, y, con la ayuda del cocinero de Harvey, un amigo, construyeron Gismo durante las tardes y fines de semana. Este suceso fue divulgado en los periódicos Washington Daily News y el New York Times en el año 1953, después de que su patente fuera concedida. En este momento, Shepard fundó Intelligent Machines Research Corporation (IMR), comenzando a fabricar el primero de varios sistemas del OCR usados para operaciones comerciales. Mientras que Gismo y los últimos sistemas de IMR, utilizaron análisis de imagen, en comparación con el carácter que emparejaba, pudiendo aceptar una cierta variación de la fuente. Gismo estaba limitado a los registros verticales, mientras que los reconocedores posteriores de la compañía IMR, analizaban caracteres en cualquier parte del campo de exploración, una necesidad práctica en documentos del mundo real.

El primer sistema comercial fue instalado en Readers Digest en 1955, que, muchos años más tarde, fue donado por al museo Smithsonian, donde fue puesto en la exhibición. El segundo sistema fue vendido a los Standard Oil Company de California para leer impresiones en tarjetas de crédito para propósitos de facturación, además se vendieron muchos más sistemas a compañías petroleras. Otros sistemas vendieron por el IMR durante los últimos años 50 incluyeron un escáner de páginas para la fuerza aérea de los Estados Unidos para la lectura y transmisión de mensajes escritos a

máquina. IBM y otras empresas fueron licenciadas más adelante sobre las patentes del OCR de Shepard.

El servicio postal de Estados Unidos ha estado utilizando las máquinas de OCR para clasificar el correo desde que 1965, basados en la tecnología ideada sobre todo por el inventor prolífico Jacob Rabinow. El primer uso del OCR en Europa sucedió en la oficina de Gran Bretaña. En 1965 se comenzó a planear un sistema de actividades bancarias completo, Nacional Giro, usando la tecnología del OCR, ideó un proceso que revolucionó los sistemas del pago de cuentas en el Reino Unido. El correo postal de Canadá ha estado utilizando sistemas OCR desde 1971. Los sistemas OCR leen el nombre y la dirección del destinatario, e imprimen un código de barras en el sobre basados en el código postal del mismo. Después, las cartas necesitan solamente ser clasificadas por los compaginadores, menos costosos que necesitan leer solamente el código de barras. Para evitar interferencia con el campo de dirección escrita a mano, que se puede situar en cualquier parte de la carta, se usa una tinta especial leída bajo una ultravioleta. Esta tinta parece anaranjada en condiciones normales de la iluminación. Los sobres marcados con el código de barras que son leídos por la máquina pueden ser posteriormente procesados. [1]

## **ESTADO ACTUAL DE LA TECNOLOGÍA OCR**

El reconocimiento exacto de la escritura latina, ahora se considera en gran parte un problema solucionado. La exactitud excede el 99%, aunque hay veces en que se exige incluso una exactitud más alta, requiriendo la revisión humana para los errores. Actualmente está en desarrollo el reconocimiento de la mano que escribe, al igual que el reconocimiento del texto impreso en otras lenguas (especialmente en los que tienen un número muy grande de caracteres).

Los sistemas reconocedores el texto impreso a mano han gozado de éxito comercial estos últimos años. Entre éstos están dispositivos de asistencia personales digitales (PDA) tales como los Palm OS. Apple es pionero en esta tecnología. Los algoritmos usados en estos dispositivos toman ventaja del orden, velocidad y dirección de las líneas o segmentos individuales en su entrada son conocidos. También, el usuario puede aprender habilidades nuevas para utilizar solamente formas específicas de la letra (por ejemplo, un triángulo sin su base correspondería a la letra A). Estos métodos no se pueden utilizar en software escanean documentos en papel, por lo que el reconocimiento exacto de documentos impresos a mano sigue siendo en gran parte un problema abierto al desarrollo. Índices de exactitud del 80 al 90% en caracteres impresos a mano, pueden ser alcanzados, pero esta exactitud todavía se traduce en docenas de errores por cada página, haciendo la tecnología útil solamente en contextos muy limitados. Esta variedad de OCR ahora se conoce comúnmente en la industria como ICR, o

el Reconocimiento Inteligente de Caracteres.

El reconocimiento del texto cursivo es un campo de investigación activo, con medidas de reconocimiento incluso más baja que el del reconocimiento de texto impreso a mano. Índices más altos del reconocimiento de la escritura cursiva general no serán probablemente posibles sin el uso de la información del contexto o gramatical. Por ejemplo, el reconocimiento de palabras enteras de un diccionario es más fácil que intentando analizar caracteres individuales de la escritura. La lectura de la línea de la monto de un cheque, es un ejemplo donde usar un diccionario más pequeño especializado en escritura de números, puede aumentar tarifas del reconocimiento enormemente. El conocimiento de la gramática de la lengua que es explorada puede también ayudar a determinar si una palabra es probable ser un verbo o un sustantivo, por ejemplo, permitiendo mayor exactitud.

Para problemas más complejos del reconocimiento, se usan los sistemas de reconocimiento inteligente de caracteres, pues las redes neuronales artificiales que los componen, trabajan indiferentes a las transformaciones lineales y no lineales del proceso de reconocimiento.

Una variante del OCR es el OMR (optical mark recognition) que se utiliza para reconocimiento de marcas. Una aplicación sería la corrección automática de exámenes de tipo test, en los que la respuesta correcta se rodea con un círculo, tipo PSU. [2] [3]






## Software OCR.

Aquí se lista algunas aplicaciones que hacen uso de la tecnología ocr. [4]

Nombre	Licencia	Sistemas Operativos	Notas
ExperVision TypeReader & OpenRTK	Commercial	Windows, Mac OS X, Unix, Linux, OS/2	ExperVision Inc. was founded in 1987, its OCR technology and product won the highest marks in the independent testing performed by UNLV for the consecutive years that ExperVision participated.
ABBYY FineReader OCR	Commercial	Windows	For working with localized interfaces, corresponding language support is required.
OmniPage	Commercial (Nuance EULA)	Windows, Mac OS	Product of Nuance Communications
Readiris	Commercial	Windows, Mac OS	Product of I.R.I.S. Group of Belgium. Asian and Middle Eastern editions.
SmartZone (formerly known as Zonal OCR)	Commercial	Windows	Zonal OCR is the process by which Optical Character Recognition (OCR) applications "read" specifically zoned text from a scanned image.
Computhink's ViewWise	Commercial	Windows	Document Management system
CuneiForm	BSD variant	Windows, Linux, BSD, MacOSX.	Enterprise-class system, multi language, can save text formatting and recognizes complicated tables of any structure
GOOCR	GPL	Many (open source)	Early development
Microsoft Office Document Imaging	Commercial	Windows, Mac OS X	
Microsoft Office OneNote 2007	Commercial	Windows	
NovoDynamics VERUS	Commercial?	?	Specializes in languages of the Middle East
Ocrad	GPL	Unix-like, OS/2	

Brainware	Commercial	Windows	Template-free data extraction and processing of data from documents into any backend system; sample document types include invoices, remittance statements, bills of lading and POs
HOCR	GPL	Linux	Hebrew OCR
OCROPUS	Apache	Linux	Pluggable framework which can use Tesseract
ReadSoft	Commercial	Windows	Scan, capture and classify business documents such forms, invoices and POs.
Alt-N Technologies' RelayFax Network Fax Manager	Commercial	Windows	Multi-language OCR Plug-in is used to convert faxed pages into editable document formats (doc, pdf, etc...) in many different languages.
Scantron Cognition	Commercial	Windows	For working with localized interfaces, corresponding language support is required.
SimpleOCR	Freeware and commercial versions	Windows	
SmartScore	Commercial	Windows, Mac OS	For musical scores
Tesseract	Apache	Windows, Mac OS X, Linux, OS/2	Under development by Google

## Software OCR y los lenguajes que soporta. [4]

Nombre 	Ultima versión 	Año de liberación 	Lenguajes de reconocimiento 	Diccionarios 
ExperVision TypeReader & OpenRTK	7.0		English, French, German, Italian, Spanish, Portuguese, Danish, Dutch, Swedish, Norwegian, Hungarian, Polish, Finnish and Polynesian	
ABBYY FineReader OCR	9.0	2007	Abkhaz, Adyghian, Afrikaans, Agul, Albanian, Altai, Armenian (Eastern, Western, Grabar), Avar, Aymara, Azerbaijani (Cyrillic), Azerbaijani (Latin), Bashkir, Basic,	Armenian (Eastern, Western, Grabar), Bashkir, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch (Netherlands and Belgium), English,

		<p>Basque, Belarusian, Bemba, Blackfoot, Breton, Bugotu, Bulgarian, Buryat, C/C++, COBOL, Catalan, Cebuano, Chamorro, Chechen, Chinese Simplified, Chinese Traditional, Chukchee, Chuvash, Corsican, Crimean Tatar, Croatian, Crow, Czech, Dakota, Danish, Dargwa, Dungan, Dutch (Netherlands and Belgium), English, Eskimo (Cyrillic), Eskimo (Latin), Esperanto, Estonian, Even, Evenki, Faroese, Fijian, Finnish, Fortran, French, Frisian, Friulian, Gagauz, Galician, Ganda, German (Luxemburg), German (new and old spelling), Greek, Guarani, Hani, Hausa, Hawaiian, Hebrew, Hungarian, Icelandic, Ido, Indonesian, Ingush, Interlingua, Irish, Italian, JAVA, Japanese, Jingpo, Kabardian, Kalmyk, Karachay-balkar, Karakalpak, Kasub, Kawa, Kazakh, Khakass, Khanty, Kikuyu, Kirghiz, Kongo, Koryak, Kpelle, Kumyk, Kurdish, Lak, Latin, Latvian, Lezgi, Lithuanian, Luba, Macedonian, Malagasy, Malay, Malinke, Maltese, Mansy, Maori, Mari, Maya, Miao, Minangkabau, Mohawk, Moldavian, Mongol, Mordvin, Nahuatl, Nenets, Nivkh, Nogay, Norwegian (nynorsk and bokmal), Nyanja, Occidental, Ojibway, Ossetian, Papiamento, Pascal, Polish,</p>	<p>Estonian, Finnish, French, German (new and old spelling), Greek, Hebrew, Hungarian, Italian, Latvian, Lithuanian, Norwegian (nynorsk and bokmal), Polish, Portuguese (Portugal and Brazil), Romanian, Russian, Slovak, Slovenian, Spanish, Swedish, Tatar, Thai, Turkish, Ukrainian</p>
--	--	---	--

			<p>Portuguese (Portugal and Brazil), Provencal, Quechua, Rhaeto-romanic, Romanian, Romany, Rundi, Russian, Russian (old spelling), Rwanda, Sami (Lappish), Samoan, Scottish Gaelic, Selkup, Serbian (Cyrillic), Serbian (Latin), Shona, Simple chemical formulas, Slovak, Slovenian, Somali, Sorbian, Sotho, Spanish, Sunda, Swahili, Swazi, Swedish, Tabasaran, Tagalog, Tahitian, Tajik, Tatar, Thai, Tok Pisin, Tongan, Tswana, Tun, Turkish, Turkmen, Tuvinian, Udmurt, Uighur (Cyrillic), Uighur (Latin), Ukrainian, Uzbek (Cyrillic), Uzbek (Latin), Welsh, Wolof, Xhosa, Yakut, Zapotec, Zulu</p>	
OmniPage	16	2007	<p>Afrikaans, Albanian, Aymara, Basque, Bemba, Blackfoot, Breton, Bugotu, Bulgarian, Byelorussian, Catalan, Chamorro, Chechen, Corsican, Croatian, Crow, Czech, Danish, Dutch, English, Esperanto, Estonian, Faroese, Fijian, Finnish, French, Frisian, Friulian, Gaelic (Irish), Gaelic (Scottish), Galician, Ganda/Luganda, German, Greek, Guarani, Hani, Hawaiian, Hungarian, Icelandic, Ido, Indonesian, Interlingua, Italian, Inuit, Kabardian, Kasub, Kawa, Kikuyu, Kongo, Kpelle, Kurdish, Latin, Latvian, Lithuanian, Luba, Luxembourgian,</p>	

			<p>Macedonian, Malagasy, Malay, Malinke, Maltese, Maori, Mayan, Miao, Minankabaw, Mohawk, Moldavian, Nahuatl, Norwegian, Nyanja, Occidental, Ojibway, Papiamento, Pidgin English, Polish, Portuguese (Brazilian), Portuguese, Provençal, Quechua, Rhaetic, Romanian, Romany, Ruanda, Rundi, Russian, Sami Lule, Sami Northern, Sami Southern, Sami, Samoan, Sardinian, Serbian (Cyrillic), Serbian (Latin), Shona, Sioux, Slovak, Slovenian, Somali, Sorbian, Sotho, Spanish, Sundanese, Swahili, Swazi, Swedish, Tagalog, Tahitian, Tinpo, Tongan, Tswana, Tun, Turkish, Ukrainian, Visayan, Welsh, Wolof, Xhosa, Zapotec, Zulu</p>	
Readiris	12 Pro & Corporate	2009	<p>American English, British English, Afrikaans, Albanian, Aymara, Balinese, Basque, Bemba, Bikol, Bislama, Brazilian, Breton, Bulgarian, Byelorussian, Catalan, Cebuano, Chamorro, Corsican, Croatian, Czech, Danish, Dutch, Esperanto, Estonian, Faroese, Fijian, Finnish, French, Frisian, Friulian, Galician, Ganda, German, Greek, Greenlandic, Haitian (Creole), Hani, Hiligaynon, Hungarian, Icelandic, Ido, Ilocano, Indonesian, Interlingua, Irish (Gaelic), Italian, Javanese, Kapampangan, Kicongo, Kinyarwanda, Kurdish,</p>	



			Latin, Latvian, Lithuanian, Luxemburgh, Macedonian, Madurese, Malagasy, Malay, Maltese, Manx (Gaelic), Maori, Mayan, Minangkabau, Nahuatl, Norwegian, Numeric, Nyanja, Nynorsk, Occitan, Pidgin English, Polish, Portuguese, Quechua, Rhaeto-Roman, Romanian, Rundi, Russian, Samoan, Sardinian, Scottish (Gaelic), Serbian, Serbian (Latin), Shona, Slovak, Slovenian, Somali, Sotho, Spanish, Sundanese, Swahili, Swedish, Tagalog, Tahitian, Tok Pisin, Tonga, Tswana, Turkish, Ukrainian, Waray, Wolof, Xhosa, Zapotec, Zulu, Bulgarian - English, Byelorussian - English, Greek - English, Macedonian - English, Russian - English, Serbian - English, Ukrainian - English + Moldovan, Bosnian (Cyrillic and Latin), Tetum, Swiss-German and Kazak	
Readiris	12 Pro & Corporate Middle-East	2009	Arabic, Farsi and Hebrew	
Readiris	12 Pro & Corporate Asian	2009	Simplified Chinese, Traditional Chinese, Japanese and Korean	
SmartZone	v2	2008	English, Danish, Dutch, Finnish, French, German, Italian, Norwegian, Portuguese, Spanish, and Swedish	
Computhink's ViewWise	6.1	2008		
CuneiForm	12	2007	English, German, French, Spanish, Italian, Portuguese, Dutch, Russian, Mixed	

			Russian-English, Ukrainian, Danish, Swedish, Finnish, Serbian, Croatian, Polish and others	
GOCR	0.47	2009		
Microsoft Office Document Imaging	Office 2007	2007	Language availability is tied to the installed proofing tools. For languages not included in your version of MS Office you'd need the corresponding Proofing Tools kit (separate purchase).	
Microsoft Office OneNote 2007				
NovoDynamics VERUS	Middle East Professional	2005	Arabic, Persian (Farsi, Dari), Pashto, Urdu, including embedded English and French. It also recognizes the Hebrew language, including embedded English.	
NovoDynamics VERUS	Asia Professional	2009	Simplified and Traditional Chinese, Korean and Russian languages, including embedded English	
Ocrad				
Brainware				
HOCR	0.10.13	2008	Hebrew	
OCROPUS	0.3.1	2008	All the languages and scripts that Tesseract supports through the Tesseract plugin, and it supports Latin script and English for its native recognizers	
ReadSoft				
Alt-N Technologies' RelayFax Network Fax Manager				
Scantron Cognition				
SimpleOCR	3.5	2008	English and French	
SmartScore				

Tesseract	2.03	2008	Can recognize 6 languages, is fully UTF8 capable, and is fully trainable	
-----------	------	------	--	--

## PLANTEAMIENTO DEL PROBLEMA

Cuando se dispone de información en forma de documento impreso y se desea procesarla mediante un computador, existen dos opciones: una primera consistiría en introducirla a través del teclado, labor larga y tediosa. Otra posibilidad es automatizar esta operación por medio de un sistema de OCR compuesto de un software y hardware adecuado que reduciría considerablemente el tiempo de entrada de datos. En nuestro caso, el hardware utilizado se reduce a un dispositivo de captura de imágenes (escáner) y a un computador personal en el que se ejecuta el software de reconocimiento desarrollado a tal efecto.

El objetivo final del sistema es la generación de un fichero con los códigos ASCII correspondientes a los caracteres de un texto en papel cuya imagen ha sido previamente capturado mediante el escáner. Se trata, en definitiva, de un caso particular de Sistema de Reconocimiento Automático de Formas que, por tanto, consta de una etapa de aislamiento de los objetos a reconocer (en este caso caracteres impresos), otra de extracción de sus características individuales y finalmente la aplicación de una función discriminante que permita establecer la clase a la que pertenece cada objeto (en este caso la clase de carácter, que estará representada por su correspondiente código ASCII).

### ETAPAS DEL SISTEMA OCR DESARROLLADO.

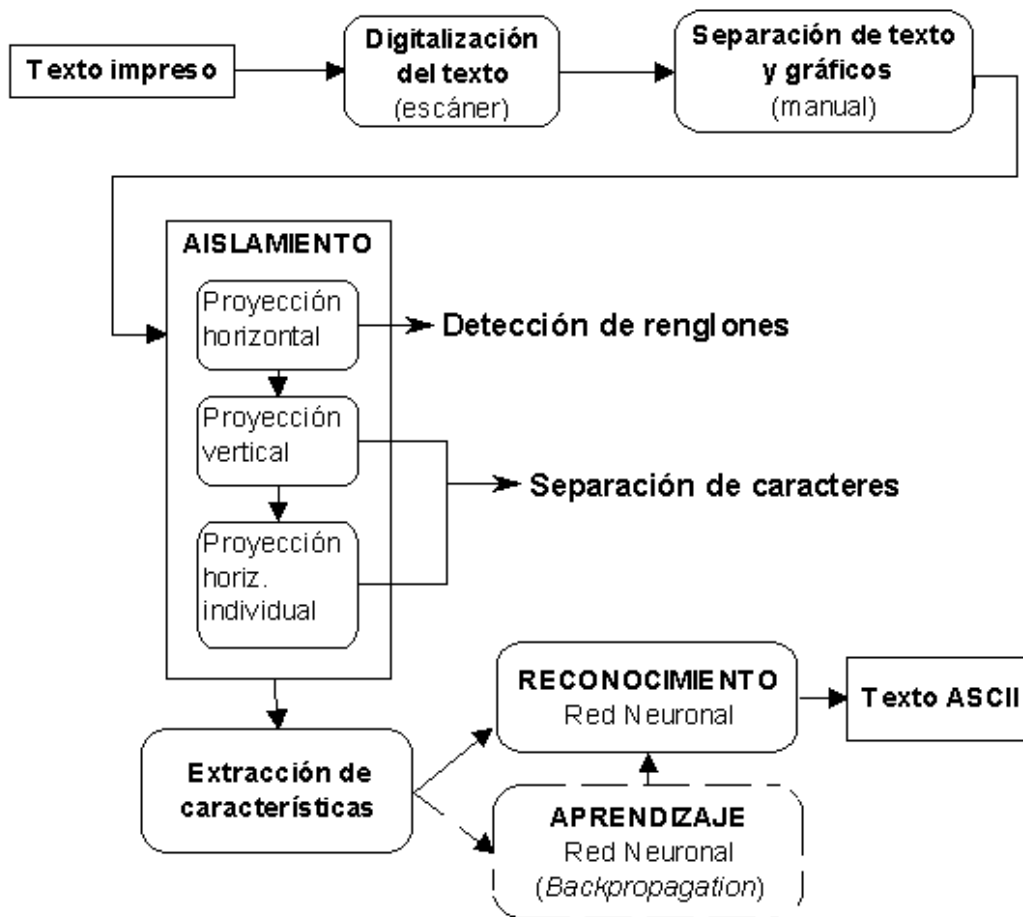


Figura 1

Si la imagen contiene tanto texto como gráficos, se procede a separarlos antes de realizar cualquier otra operación. En nuestro caso, esta separación es manual mediante la selección, con el "ratón" del computador, de las áreas de interés en el documento.

Después de este procesamiento previo de la imagen, se procede al aislamiento de cada uno de los caracteres que forman el texto que se está tratando y posteriormente a la extracción de un conjunto de características capaces de definir cada carácter aislado.

Una vez obtenido el vector de características de cada uno de los caracteres incluidos en el documento, se activa la etapa de reconocimiento que, en nuestro caso, consiste en una red neuronal que capaz de generar como salida el código ASCII de 8 bits del carácter cuyo vector de características recibe como entrada. Para que esto sea posible, previamente se ha sometido a la red neuronal a un proceso de aprendizaje, en el cual se han utilizado documentos impresos conteniendo diferentes juegos de caracteres o abecedarios (todas las letras mayúsculas y minúsculas, los dígitos del "0" al "9" y signos de puntuación) con diferentes tamaños.

Durante este proceso se ajustan los parámetros internos de la red de forma reiterativa hasta que sea capaz de generar como salida el código ASCII del carácter cuya imagen se presenta a su entrada, para todas las formas y tamaños considerados para tal carácter. [5]

### SOLUCION AL PROBLEMA CON REDES NEURONALES.

Las Redes Neuronales son sistemas de computación que permiten la resolución de problemas que no pueden ser descritos fácilmente mediante un enfoque algorítmico tradicional, como, por ejemplo, ocurre en el reconocimiento de formas. Con las redes se expresa la solución de un problema, no como una secuencia de pasos, sino como la evolución de un sistema inspirado en el funcionamiento del cerebro y dotado, por tanto, de cierta "inteligencia". Tal sistema no es sino la combinación de una gran cantidad de elementos simples de proceso (nodos o neuronas) interconectados que, operando de forma masivamente paralela, consiguen resolver el problema.

Las conexiones sirven para transmitir las salidas de unos nodos a las entradas de otros. El funcionamiento de un nodo es similar al de las neuronas biológicas presentes en el cerebro. Suele aceptarse que la información memorizada en el cerebro está relacionada con los valores sinópticos de las conexiones entre las neuronas. De igual forma, se dice que las redes neuronales tienen la capacidad de "aprender" mediante el ajuste de las conexiones entre nodos. Estas conexiones tienen un valor numérico asociado denominado peso, que puede ser positivo (conexiones de excitación) o negativo (conexiones de inhibición).

Existen muchos tipos de redes neuronales, un modelo utilizado en una gran variedad de aplicaciones, entre las que se incluyen las de reconocimiento de formas, es la red multicapa con conexiones unidireccionales hacia delante (feedforward). En la figura 2 se muestra la red de este tipo utilizada en el sistema OCR para el reconocimiento de caracteres.

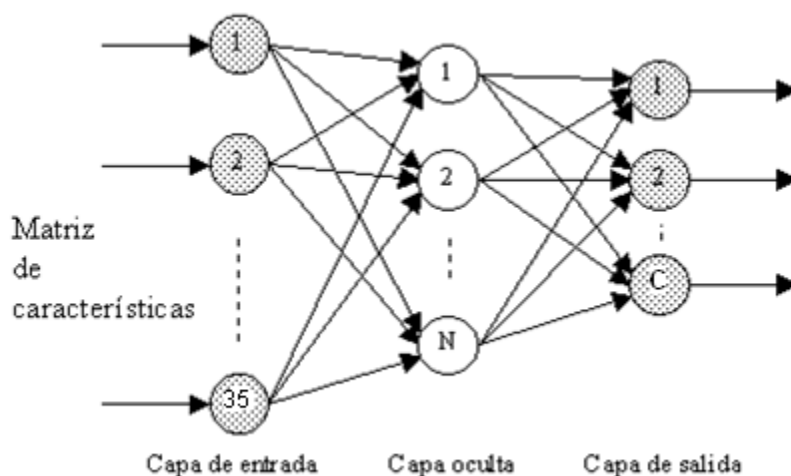


Figura 2.- Estructura de la red neuronal utilizada

El tipo de red que se utilizará para darle solución al problema del reconocimiento óptico de caracteres es una red neuronal tipo multicapa denominada backpropagation.

### BACKPROPAGATION

La propagación hacia atrás de errores o retropropagación es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizar un error (comúnmente cuadrático) por medio de gradiente descendente, por lo que la parte esencial del algoritmo es cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

#### Minimización del Error

Los algoritmos en Aprendizaje Automático pueden ser clasificados en dos categorías: supervisados y no supervisados. Los algoritmos en aprendizaje supervisado son usados para construir "modelos" que generalmente predicen unos ciertos valores deseados. Para ello, los algoritmos supervisados requieren que se especifiquen los valores de salida (output) u objetivo (target) que se asocian a ciertos valores de entrada (input). Ejemplos de objetivos pueden ser valores que indican éxito/fallo, venta/noventa, pérdida/ganancia, o bien ciertos atributos multi-clase como cierta gama de colores o las letras del alfabeto en nuestro caso. El conocer los valores de salida deseados permite determinar la calidad de

la aproximación del modelo obtenido por el algoritmo.

La especificación de los valores entrada/salida se realiza con un conjunto consistente en pares de vectores con entradas reales de la forma  $(\mathbf{x}, \mathbf{t})$ , conocido como conjunto de entrenamiento o conjunto de ejemplos. Los algoritmos de aprendizaje generalmente calculan los parámetros  $\mathbf{W}$  de una función  $N(\mathbf{x}; \mathbf{W})$  que permiten aproximar los valores de salida en el conjunto de entrenamiento.

Si  $(\mathbf{x}^{(q)}, \mathbf{t}^{(q)})$ ,  $q = 1, \dots, p$ , son los elementos del conjunto de entrenamiento, la calidad de la aproximación en el ejemplo  $q$  se puede medir a través del error cuadrático:

$$E(\mathbf{x}^{(q)}; \mathbf{W}) = \frac{1}{2} \|N(\mathbf{x}^{(q)}; \mathbf{W}) - \mathbf{t}^{(q)}\|^2$$

, donde  $\|\cdot\|$  es la norma euclidiana.

El error total es la suma de los errores de los ejemplos:

$$E(\mathbf{W}) = \sum_{q=1}^p E(\mathbf{x}^{(q)}; \mathbf{W})$$

Un método general para minimizar el error es el actualizar los parámetros de manera iterativa. El valor nuevo de los parámetros se calcula al sumar un incremento  $\Delta \mathbf{W}$  al valor actual:

$$\mathbf{W} := \mathbf{W} + \Delta \mathbf{W}$$

El algoritmo se detiene cuando  $\mathbf{W}$  converge o bien cuando el error alcanza un mínimo valor deseado.

Si la función  $N(\mathbf{x}; \mathbf{W})$  usada para aproximar los valores de salida es diferenciable respecto a los parámetros  $\mathbf{W}$ , podemos usar como algoritmo de aprendizaje el método de gradiente descendiente. En este caso, el incremento de los parámetros se expresa como

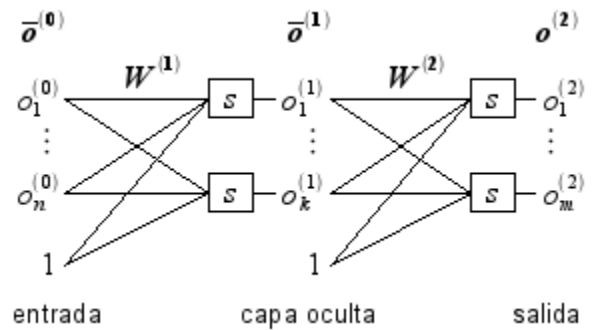
$$\Delta \mathbf{W} = -\gamma \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}},$$

donde  $0 < \gamma < 1$  es un parámetro conocido como factor de aprendizaje.

Antes de continuar introduciremos un poco de notación. Definimos  $\bar{\mathbf{v}} = (v_1, \dots, v_n, 1)^T$  como el vector extendido del vector  $\mathbf{v} = (v_1, \dots, v_n)^T$ . El par  $(\mathbf{x}, \mathbf{t})$  representará a un elemento del conjunto de entrenamiento y una relación de entrada-salida, a menos que se indique otra cosa.

### RED NEURONAL CON UNA CAPA OCULTA

La función la usaremos para aproximar los valores de salida de una red neuronal artificial con una capa oculta. La red está constituida por una capa de entrada (*input layer*), una capa oculta (*hidden layer*) y una capa de salida (*output layer*), tal como se ilustra con la siguiente figura:



Los elementos que constituyen a la red neuronal son los siguientes:

- $s$  es una función de valores reales, conocida como la función de transferencia.
- $\bar{\mathbf{o}}^{(0)}$  es la capa de entrada, considerado como el vector extendido del ejemplo  $\mathbf{o}^{(0)} = \mathbf{x} = (x_1, \dots, x_n)^T$ .
- $\bar{\mathbf{o}}^{(1)}$  es la capa oculta, el vector extendido de  $\mathbf{o}^{(1)} = (o_1^{(1)}, \dots, o_k^{(1)})^T$ .
- $\mathbf{o}^{(2)} = (o_1, \dots, o_m)^T$  es la capa de salida, considerado como el vector que aproxima al valor deseado  $\mathbf{t} = (t_1, \dots, t_m)^T$ .
- $\mathbf{W}^{(1)}$  es una matriz de tamaño  $(n+1) \times k$  cuyos valores  $W_{ij}^{(1)}$  son los pesos de la conexión entre las unidades  $\bar{o}_i^{(0)}$  y  $o_j^{(1)}$ .
- $\mathbf{W}^{(2)}$  es una matriz de tamaño  $(k+1) \times m$  cuyos valores

$W_{ij}^{(2)}$  son los pesos de la conexión entre las unidades  $\bar{o}_i^{(1)}$  y  $o_j^{(2)}$ .

De estos elementos, únicamente las matrices  $\mathbf{W}^{(l)}$  son consideradas como los parámetros de la red, ya que los valores  $\bar{o}^{(l)}$  son el resultado de cálculos que dependen de las matrices de pesos, del valor de entrada  $\bar{\mathbf{x}}$  y de la función de transferencia  $s$ .

La función de transferencia  $s$  que consideraremos en nuestro algoritmo es conocida como función sigmoidea, y esta definida como

$$s(u) = \frac{1}{1 + \exp(-u)}$$

esta función además de ser diferenciable, tiene la particularidad de que su derivada se puede expresar en términos de sí misma:

$$\frac{ds(u)}{du} = s(u)(1 - s(u)).$$

esto nos servirá para simplificar los cálculos en el algoritmo de aprendizaje aquí descrito.

## DESCRIPCIÓN DEL ALGORITMO

A grandes rasgos:

1. Calcular la salida de la red  $\mathbf{o}^{(2)}$  a partir de uno de los conjuntos de valores de prueba  $\mathbf{x}$ .
2. Comparar con la salida correcta  $\mathbf{t}$  y calcular el error según la fórmula:

$$E(\mathbf{x}; \mathbf{W}^{(1)}, \mathbf{W}^{(2)}) = \frac{1}{2} \sum_{i=1}^m (o_i^{(2)} - t_i)^2.$$

3. Calcular las derivadas parciales del error con respecto a los pesos  $\mathbf{W}^{(2)}$  que unen la capa oculta con la de salida.
4. Calcular las derivadas parciales del error con respecto a los pesos  $\mathbf{W}^{(1)}$  que unen la capa de entrada con la oculta.
5. Ajustar los pesos de cada neurona para reducir el error.
6. Repetir el proceso varias veces por cada par de entradas-salidas de prueba. [16]

Cuando se trabaja con redes neuronales como la que se utiliza en este trabajo, se distingue entre una primera etapa de aprendizaje o entrenamiento de la red y una segunda etapa de funcionamiento u operación de la red. En la primera se realiza el ajuste de los pesos en función de lo que se desea que almacene la red. Se suele distinguir entre dos tipos de aprendizaje: supervisado o no supervisado.

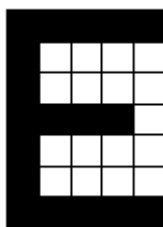
La red utilizada en el sistema de OCR, tiene un aprendizaje supervisado, el cual consiste en el ajuste de los pesos para que "aprenda" a relacionar un conjunto de patrones de entrada con las salidas que se desea que la red genere ante cada uno de dichos patrones. El entrenamiento concluye cuando la diferencia entre las salidas deseadas ante cada patrón y las que se obtienen realmente con los valores calculados de los pesos es suficientemente pequeña.



A partir de este momento los pesos no varían y la red entraría en una fase operacional para cumplir la función que se desea de ella. Por ejemplo, si se va a utilizar para reconocimiento de

caracteres, se supone que ha aprendido a distinguir entre diferentes tipos de imágenes de caracteres de entrada y a asociar a cada una un carácter del alfabeto.

Patrón " X "



[8]

Como se representará en la figura 2, la capa de entrada de la red utilizada consta de 35 neuronas que corresponden a los 35 valores (matriz de 7x5) que se extraen de la parametrización de un carácter [9]. Se eligió este número para que las características extraídas fuesen lo suficientemente relevantes para distinguir unos caracteres de otros. Dado el tamaño de los caracteres en píxeles, una matriz menor podría dar lugar a más confusión, una mayor sería redundante y emplearía un tiempo computacional mayor, aunque el límite final lo tenemos en la memoria disponible en el sistema.

En la capa oculta se pueden tener hasta 35 neuronas, las que en la de entrada. Se precisan más neuronas ocultas si el juego de caracteres que debe aprender la red es grande, para que el proceso de entrenamiento converja. Como se comentará en el apartado de resultados, con más neuronas la red converge antes, además de cometer menos fallos en el reconocimiento, aunque en contrapartida,

el hecho de tener más neuronas supone más de conexiones entre ellas y por tanto, debe hacer mayor número de operaciones empleando más tiempo en cada ciclo de aprendizaje. [5]

**Concretando, un perceptron multicapa tiene tres características:**

$$y_k = \frac{1}{1 + \exp(-u_k)}$$

1. El modelo de cada neurona (figura 3) incluye una función no lineal. En este caso, a diferencia del perceptrón donde es la función escalón, y debido a la necesidad de que sea una función continua y derivable, es la función sigmoide, donde  $u_k$  es la suma total de la actividad interna en la neurona  $k$  (la señal de entrada) e  $y_k$  la salida que se produce en la neurona.

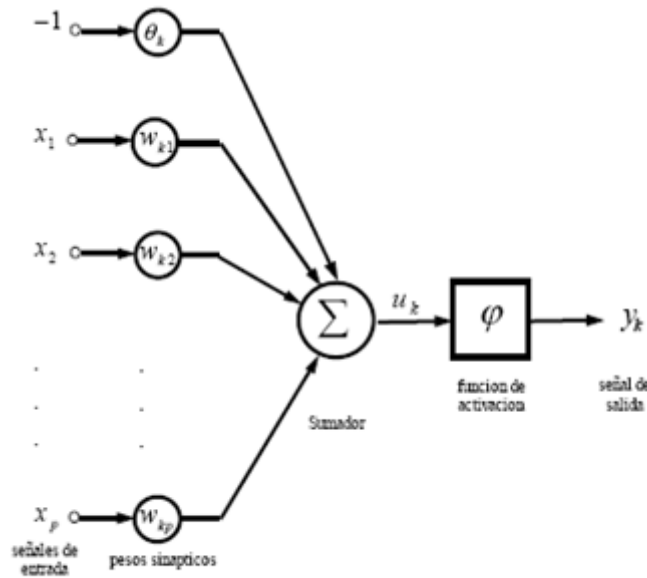


Fig. 3 Modelo de una neurona.

2. La red contiene una o más capas ocultas de neuronas que no forman parte ni de la entrada ni de la salida. Estas neuronas ocultas capacitan a la red para aprender progresivamente cualquier correspondencia entre la entrada y la salida y almacenar internamente esta información.

3. La red posee un gran número de conexiones, estas vienen determinadas por los pesos de la red. Un cambio en la conexión entre las neuronas equivale a un cambio en los pesos. [6]

### IMPLEMENTACION CON MATLAB

Debido al poder de la herramienta de matlab y la existencia de unas librerías (toolbox de redes neuronales) que nos facilitan el trabajo con las redes neuronales, se ha considerado implementar lo que es el reconocimiento de caracteres con la misma para ello vamos a revisar primero la estructura de las librerías, sus funciones y como trabajan.

### TOOLBOX DE REDES NEURONALES (LIBRERIAS MATLAB)

Para trabajar con redes neuronales, seguramente podremos encontrar con una simple búsqueda en Internet un gran número de API's y frameworks que implementen por nosotros la estructura de la mayor parte de los tipos de redes y la funciones necesarias para trabajar con ellas. Uno de estos frameworks es el Toolbox que matlab posee, que nos ofrece una implementación genérica de redes neuronales, así como implementaciones de redes neuronales concretas como las perceptrón, backpropagation, Som, etc.

### Estructura

Matlab utiliza una estructura única que nos dará acceso a todas las propiedades de la red neuronal, independientemente del tipo que esta sea, de manera que utilizando esta propiedad podremos modificar las entradas, capas, conexiones, pesos, etc. De esta manera una vez

configurada la red neuronal según nuestras necesidades invocaremos las funciones de manipulación de redes neuronales disponibles en matlab, (simulación, entrenamiento, inicialización, etc.), pasándole como parámetro la estructura de la red neuronal.

```
net = network;
```

Si ejecutamos el comando anterior y visualizamos el contenido de la variable myNetwork se nos visualizará la estructura mencionada, la cual se puede dividir en cinco secciones:

#### 1. **Arquitectura:**

Define las características básicas de la red neuronal, número de entradas, capas, conexiones de bias, etc.

#### 2. **Subobjetos:**

Contiene referencias a las subestructuras de la red neuronal, que nos permitirán configurar las propiedades de los distintos componentes que forman la red (capas, entradas, salidas, etc.).

#### 3. **Funciones:**

Funciones principales de la red neuronal, utilizadas para ejecutar las operaciones de inicialización, entrenamiento o simulación.

#### 4. **Parámetros:**

Configuración de los parámetros asociados a las funciones seleccionadas en el bloque de funciones.

#### 5. **Valores:**

Aquí se definen las matrices con los valores de los pesos de entrada, conexiones entre capas y bias.

### **Funciones**

Una vez creada la red neuronal, para trabajar con la misma, podremos utilizar las siguientes funciones para realizar las operaciones típicas:

#### 1. **Inicialización (net = init(net)):**

Mediante la función de inicialización, obtenemos una red neuronal con los valores de los pesos y bias actualizados según las funciones de inicialización que le hayamos asociado a la red, mediante su propiedad net.initFcn, o bien net.layers{i}.initFcn y net.biases{i}.initFcn.

#### 2. **Entrenamiento ([net, tr, Y, E, Pf, Af] = train(net, P, T, Pi, Ai, VV, TV)):**

Realiza el entrenamiento de la red neuronal, modificando los pesos asociados a las conexiones entre las diferentes capas y neuronas de la misma. Para esto, debemos indicar unos patrones de entrada a la red (P, matriz de dimensiones MxN siendo M la suma de los tamaños de las capas de entrada de la red neuronal, y N el número de patrones que deseamos aplicar en el entrenamiento). En caso de ser un entrenamiento supervisado también indicaremos los targets (T, matriz de MxN), con estos datos la matriz de patrones se aplica a la red neuronal, y el

toolbox utilizando las funciones de entrenamiento que le hemos indicado en las propiedades “*trainFcn*” se encargará de actualizar los pesos asociados a las conexiones de la red. Los resultados del entrenamiento los obtendremos en la variable de retorno Y y los errores para cada patrón de entrada respecto a la salida esperada en la variable de retorno E.

3. **Simulación ([Y, Pf, Af, E, perf] = sim(net, P, Pi, Ai, T)):**

Función parecida a la anterior pero que no actualizará los pesos de la red neuronal. Una vez que tengamos entrenada la red neuronal y esta ofrezca unos resultados válidos, utilizaremos esta función para analizar nuevos patrones de entrada.

### Redes neuronales conocidas

Normalmente a la hora de trabajar con redes neuronales, queremos trabajar con un tipo de red neuronal concreto, el cual se ajuste mejor a nuestras necesidades. En este caso en vez de utilizar la función “network” para la creación de la estructura base, podemos utilizar funciones específicas para cada tipo de red neuronal, de manera que la estructura base que matlab nos devuelva tenga una configuración de capas de entrada, ocultas, conexiones etc. apropiada para el tipo de red neuronal deseado.

1. Perceptron: newp(P,S)
2. Backpropagation: newff(P, [S1,..., Sn])
3. Radiales: newgrnm(P,T)

Mapas Autoorganizados: newsom(P,S) [13].

### PASOS A SEGUIR PARA LA IMPLEMENTACION EN MATLAB

A la hora de aplicar redes neuronales, y en general para cualquier problema de reconocimiento de patrones, hay una serie de pasos que serán comunes a la mayoría de los problemas.

1. Representación original de los datos
2. Preprocesamiento de los datos originales para codificarlos en un formato apropiado para la red neuronal.
3. Procesamiento de la información. (aplicación de la red neuronal)
4. Interpretación de los resultados. [10]

#### 1. Representación de la información original.

Lo primero que deberemos tener en cuenta, será la representación original de los datos que deberemos procesar, en este caso, el de reconocimiento de caracteres manuscritos, vamos a partir de una imagen, o conjunto de píxeles en blanco y negro que contendrán el carácter a reconocer.

#### 2. Preprocesamiento de los datos originales para codificarlos en un formato apropiado para la red neuronal.

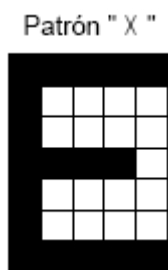
Para aplicar la información con el carácter manuscrito a la red neuronal, deberemos realizar algún tipo de procesamiento para obtener la

información apropiada que le pasaremos como entrada a dicha red.

En este caso, lo que haremos será generar un array de valores enteros 0 o 1, de manera que un 1 significa que hay algo pintado y 0 no, de manera que este array de valores discretos será la información que pasaremos a la entrada de la red neuronal.

Algo a tener en cuenta es el tamaño del vector generado a partir de la imagen, por ejemplo si la imagen original tiene un tamaño de 200×200 píxeles, y creamos un array con un valor para cada píxel de

dicha imagen obtendríamos un array de 40.000 elementos siendo necesario una red neuronal con el mismo número de entradas. Para optimizar mejor esto, vamos a procesar tan solo aquella zona de la imagen donde esté el dígito manuscrito y además, vamos a dividir esta zona en una cuadrícula de por ejemplo 7×5 en nuestro caso, de manera que generaremos un vector de 35 elementos, correspondiéndose cada elemento con una de esas zonas en las que hemos dividido el dígito manuscrito y en caso de que algún píxel de dicha zona esté pintado pondremos un uno y en caso contrario un cero.



[8]

Vector para el patron x =  
(1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,  
0,1,0,0,0,0,1,0, 0,0,0,1,1,1,1,1)

## PROCESAMIENTO DE LA INFORMACIÓN

### Creación de la Red neuronal en matlab

Una vez tenemos creado el vector representando el caracter manuscrito, queremos ser capaces de procesarlo para detectar el caracter real que el usuario a escrito. En este caso sencillo, nos centraremos en predecir los caracteres manuscritos, para lo cual vamos a utilizar una red neuronal de tipo backpropagation [7], y como implementación el toolbox de redes neuronales de matlab.

Como hemos dicho, la entrada de la red neuronal, será un array de 64 elementos conteniendo la información de los píxeles pintados en la imagen, así que necesitaremos una red neuronales de 64 entradas, de manera que en cada entrada se le aplicará un uno o un cero indicando la existencia de un trazo en dicha zona de la imagen.

Como el objetivo es reconocer 27 caracteres (de la **a** a la **z**), o mejor dicho, el objetivo es que la red neuronales aprenda a clasificar los patrones de entrada en 27 clases diferentes, la red neuronal dispondrá de 27 salidas, una para cada carácter, de manera que tan solo se activará la salida correspondiente al carácter reconocido.

### 3.1 Creación de la red neuronal

Para la creación y simulación de la red neuronal, hemos generado primero una serie de patrones de muestra para cada uno de los caracteres que queremos reconocer (35 unidades).

```

patrón(a)
    1;1;1;1;0;1;1;0;1;1;0;1;1;1
;1,...,0
patrón(b)
    1;1;1;1;0;1;1;0;1;1;0;1;1;1
;1,...,0
patrón(c)
    1;1;1;1;0;1;1;0;1;1;0;1;1;1
;1,...,0
patrón(d)
    1;1;1;1;0;1;1;0;1;1;0;1;1;1
;1,...,0
...

patrón(z)
    1;1;1;1;0;1;1;0;1;1;0;1;1;1
;1,...,0

```

En segundo lugar, también tenemos la información de la salida que la red neuronal deberá generar para cada uno de estos patrones (26 caracteres).

```

1;0;0;0;0;0;0;0;0;0,...,0:represent
a 'a'
0;1;0;0;0;0;0;0;0;0,...,0:represent
a 'b'
0;0;1;0;0;0;0;0;0;0,...,0:represent
a 'c'
0;0;0;1;0;0;0;0;0;0,...,0:represent
a 'd'
...

0;0;0;1;0;0;0;0;0;0,...,1:represent
a 'z'

```

Así, pues primero cargaremos dicha información en matlab.

```

load matlab_pattern.txt
load matlab_targets.txt

```

A continuación creamos la red neuronal con la función newff que es del tipo Backpropagation: La red neuronal Backpropagation presenta una gran variedad de opciones de configuración, dependiendo de la necesidad de aprendizaje y de la aplicación que se este desarrollando.

Este tipo de red requiere que le sean especificados los siguientes parámetros

newff: (PR,[S1 S2...SNI],{TF1 TF2...TFNI},BTF,BLF,PF)

PR : Rx2 Matriz de valores máximos y mínimos de cada uno de las R neuronas de entrada.

Si : Número de neuronas para cada una de las capas.

TFi : Función de transferencia a utilizar en cada una de las capas, por defecto utiliza tansig

BTF : Algoritmo de entrenamiento a utilizar, por defecto utiliza trainlm

BLF : Función de actualización de los pesos, por defecto utiliza learnqdm.

PF : Función para evaluar el desempeño de la red, por defecto utiliza mse.[12]

```
>>net=newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');
```

En este caso S1 y S2 representa el número de neuronas de la capa oculta, en este caso se utiliza como función de transferencia para ambas capas ocultas la función **logsig** y la función de entrenaeminto que se utilizara será **traingdx**.

## Selección del número de capas ocultas

Para casi todos los problemas, una capa oculta es suficiente. Dos capas ocultas son necesarias para el modelado de datos con discontinuidades, como un patrón de onda de diente de sierra. Mediante el uso de dos capas ocultas rara vez mejora el modelo, y puede introducir un mayor riesgo de convergencia a un mínimo local. No hay ninguna razón teórica para la utilización de más de dos capas ocultas. [14]

## Decidir el número de neuronas a utilizar en la capa oculta

Una de las características más importantes de una red perceptrón es el número de neuronas en la capa oculta (s). Si se tiene un número insuficiente de neuronas que se utilizaran, la red no estará en condiciones de representar un modelo de datos complejo, y por consiguiente los ajustes de pesos serán pobres.

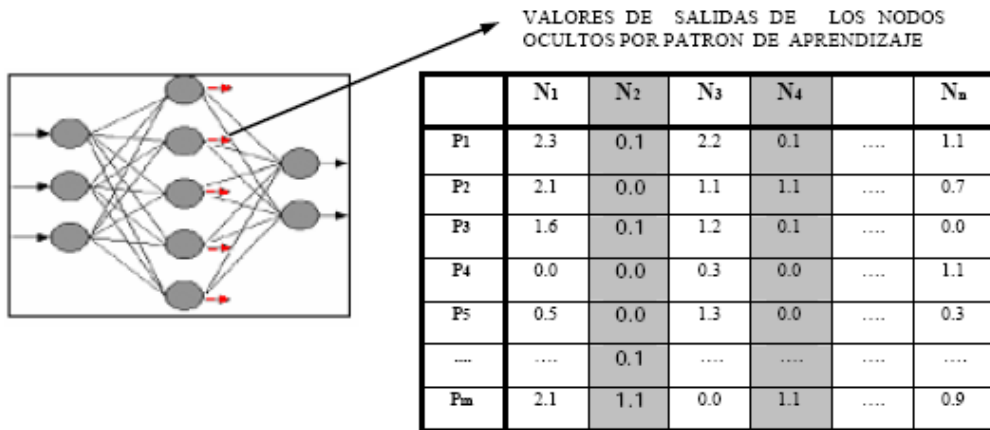
Sólo hay normas generales recogidas en el tiempo y seguidas por la mayoría de los investigadores e ingenieros de aplicaciones de este tipo de arquitectura.

- **Regla Uno:** Como la complejidad de la relación entre los datos de entrada y la salida deseada aumente, el número de elementos de proceso en la capa oculta también debe aumentar.
- **Regla dos:** Si el proceso es separable en varias etapas, entonces, más capas ocultas pueden ser requeridas. Si el proceso no es separable en etapas, entonces, capas adicionales permitirían la memorización del conjunto de entrenamiento, y no una verdadera solución general eficaz con otros datos.

- **Regla Tres:** La cantidad de datos disponibles de formación establece un límite superior para el número de elementos de proceso en la capa oculta. Para calcular este límite superior, utilice el número de casos en el conjunto de datos de entrenamiento y luego dividir esa cifra por la suma del número de nodos en la entrada y salida de las capas de la red. Luego divida ese resultado una vez más por un factor de escala entre cinco y diez. Factores de escala mucho más grandes se utilizan para los datos relativamente menos ruidosos. Si se usa demasiadas neuronas artificiales el conjunto de entrenamiento será memorizado. Si eso ocurre, la generalización de los datos no se producirá, haciendo inútil la red en los nuevos conjuntos de datos. [15]

En un estudio de Optimización de Motores de Inferencia Conexionistas Mediante Algoritmos de Poda y Lógica Difusa se propone hacer lo siguiente para conseguir un número adecuado de nodos ocultos:

Un criterio de optimización es buscar aquellos valores de salida insignificantes de los nodos ocultos al finalizar el proceso de aprendizaje, innecesarios en este proceso. Podemos organizar la información de los valores de salidas de los nodos ocultos en una matriz de N columnas (#nodos ocultos) y M filas (#patrones a aprender), de esta manera podemos analizar los valores de salida asociados a cada neurona y en conjunto decidir si esa neurona es influyente en el aprendizaje. Este proceso se observa en la siguiente figura.



P<sub>i</sub> : i-esimo Patrón de aprendizaje ,  $1 < i < m$  , m es el número de patrones de entrenamiento.

N<sub>j</sub> : j-esima Nodo oculto ,  $1 < j < n$  , n es el número de nodos ocultos.

La matriz describe la participación de cada nodo oculto por cada patrón de aprendizaje, de lo cual podemos distinguir a simple vista aquellos que son útiles y aquellos que no lo son. Esta opción puede ser óptima para redes pequeñas, pero resulta ser muy complicada de utilizar cuando tratamos con redes medianas y peor aun con redes de mayor tamaño, para lo cual necesitamos una representación más explícita de la información la de utilidad de las salidas de los nodos ocultos representados por las columnas de la matriz. [11]

Listo, ya tenemos la red neuronal creada con los parámetros por defecto. Podemos navegar un poco por la estructura del objeto “net” para asegurarnos de que todo está correcto.

```
>>net
```

```
net.inputs{1}
net.numlayers
net.outputs{1}
...
```

### 3.2 Entrenamiento de la red neuronal

Una vez creada la red neuronal, y como ya tenemos los patrones de entrenamiento con sus respectivos targets cargados en matlab, tan solo nos queda invocar a la función train.

```
net = train(net,P,T);
```

Con esto matlab comenzará a entrenar la red neuronal hasta alcanzar el performance deseado, abriéndose una ventana con la información de dicho proceso.

### 3.3 Simulación de la red neuronal

Pues nada, una vez entrenada la red neuronal, ya podemos aplicar un patrón real a la entrada y ver en la salida la clase (o caracter) en la que se ha clasificado.

```
[val,num] = sim(net,
[1;1;1;0;0;1;0;1;1;...;1;1]);
val =0.0000
0.0008
1.0013
```



0.0005  
0.0003  
0.0012  
-0.0004  
0.0004  
-0.0006  
0.0011

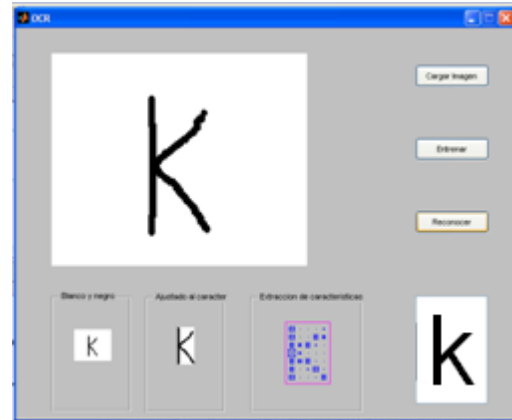
En este caso, de las diez salidas todas tienen un valor muy cercano a cero, excepto la salida correspondiente al dígito dos, que tiene un valor muy cercano a uno, siendo este el dígito reconocido.

#### 4. Interpretación de los resultados

En este caso, los resultados son fáciles de interpretar como hemos visto en el apartado anterior, la salida de la red neuronal que tenga valor 1 (o que más se

parezca al valor 1) será la que tomaremos como salida activa indicando el carácter reconocido.

Para terminar, simplemente mostramos una captura de pantalla de nuestra implementación con matlab.



#### CONCLUSIONES

- ✓ La característica mas importante del algoritmo backpropagation es su capacidad para la generalización es decir su facilidad para salidas satisfactorias a entradas que el sistema nunca vio en la fase de entrenamiento esta característica le da versatilidad para que este puede ser implementado en aplicaciones muy complejas.
- ✓ En el preprocesamiento de las imágenes que contienen los caracteres se debe considerar el tamaño de la matriz que posee información de las características de cada carácter, pues no debe ser

ni muy redundante ni tampoco información no relevante.

- ✓ Matlab posee sus librerías de redes neuronales que nos proporcionan funciones con las cuáles nos permiten manejar las mismas de una manera muy genérica adaptándolas a cualquier aplicación que se desee construir.
- ✓ En el preprocesamiento de la imagen se debe considerar que los límites de la imagen estén justo en los límites del carácter para que se ajuste solamente el carácter a la matriz en la que se almacenan las características, además para tratar la imagen como espacios en blanco o llenos se debe pasar la imagen a binario (blanco o negro) (0 o 1).

- ✓ El número de nodos de la red neuronal esta muy ligado al número de patrones que tenga el caso en estudio en nuestro caso como se trata de reconocer las letras minúsculas se tiene entonces 27 patrones, por lo tanto se parte también con 27 neuronas.
- ✓ Si se quiere un numero de neuronas que optimicen el tiempo que le toma a la red realizar la clasificación se puede recurrir a una matriz de resultado obtenidos después del enteramiento aquí analizaríamos las neuronas y su poder de decisión frente a los patrones y dependiendo si la neurona tiene una influencia grande al momento de la clasificación la podríamos descartar o no.

## REFERENCIAS

[1] Jonatan –blog Estudiante de Ingeniería Civil en Informática, UdeC. Actualmente Presidente de Centro de Alumnos de la carrera. Ayudante docente de Computación y Programación 2-2008, Proyectos de Sistemas Informáticos Historia de OCR, <http://psig5.blogspot.com/2007/08/historia-ocr.html>

[2] Idpalma, Informatica en C7, Reconocimiento Optico de Carateres (OCR), Estado actual de la tecnología OCR, editado el 26 Mayo 2008, [http://blogs.clarin.com/el-](http://blogs.clarin.com/el-comercial-siete/2008/5/26/reconocimiento-optico-carateres-ocr-)

[comercial-siete/2008/5/26/reconocimiento-optico-carateres-ocr-](http://blogs.clarin.com/el-comercial-siete/2008/5/26/reconocimiento-optico-carateres-ocr-)

[3] Jonatan –blog Estudiante de Ingeniería Civil en Informática, UdeC. Actualmente Presidente de Centro de Alumnos de la carrera. Ayudante docente de Computación y Programación 2-2008, Proyectos de Sistemas Informáticos Historia de OCR, <http://psig5.blogspot.com/2007/08/estado-actual-de-la-tecnologa-ocr.html>

[4] Wikipedia, Optical character recognition, editado en octubre del 2008, [http://en.wikipedia.org/wiki/Optical\\_character\\_recognition](http://en.wikipedia.org/wiki/Optical_character_recognition)

[5] José R. Hilera González, Juan P. Romero Villaverde, Jose A. Gutiérrez de Mesa, Departamento de Ciencias de la Computación, Universidad de Alcalá de Henares, [http://www.cc.uah.es/hilera/docs/1996/c\\_jiacse1/c\\_jiacse1.htm](http://www.cc.uah.es/hilera/docs/1996/c_jiacse1/c_jiacse1.htm)

[6] Emiliano Aldabas-Rubira, UPC-Campus Terrassa-DEE-EUETIT Colom, Introducción al reconocimiento de patrones mediante redes neuronales, <http://www.jcee.upc.es/JCEE2002/Aldabas.pdf>

[7] Daniel Saucedo Aranda, Departamento de Ciencias de la Computación e Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, <http://decsai.ugr.es/~castro/MCII/Transparencias/RN3.pdf>

[8] Drob, Redes neuronales, Reconocimiento de patrones, editado el 06/28/2008

<http://www.scribd.com/doc/3684177/Redes-Neuronales>

[9] Ricardo Conejo Muñoz, Departamento de Lenguajes y Ciencias de la Computación

Universidad de Málaga, Complejo Tecnológico, Campus de Teatinos, <http://www.lcc.uma.es/~jmortiz/archivos/Tema5.pdf>

[10] Poncos, Matlab: ejemplo backpropagation, editado el 15 Enero del 2009,

<http://poncos.wordpress.com/2009/01/15/matlab-ejemplo-backpropagation/>

[11] José Talavera Herrera, Omar Vivas-Arapa, Ernesto Cuadros-Vargas, Optimización de Motores de Inferencia Conexionistas Mediante Algoritmos de Poda y Lógica Difusa,

<http://www.usp.edu.pe/~ecuadros/papers/Paper263CLEI2002.pdf>

[12] Maria Isabel Acosta, Halord Salazar, Camilo Zuluaga, Tutorial de redes

neuronales, DESCRIPCIÓN DE LAS FUNCIONES UTILIZADAS EN MATLAB, universidad tecnológica de Pereira, facultad de ingeniería eléctrica, <http://ohm.utp.edu.co/neuronales/Anexos/AnexoA.htm>

[13] Poncos, Matlab: toolbox de redes neuronales, Introduccion, editado el 14 Abril 2008,

<http://poncos.wordpress.com/2008/04/14/matlab-toolbox-de-redes-neuronales/>

[14] DTREG, Software For Predictive Modeling and Forecasting - Multilayer Perceptron Neural Networks,

<http://www.dtreg.com/mlfn.htm>

[15] Julian Simon, a professor in the University of Maryland's Business School, Classification - Neural Networks Prediction,

[http://www.resample.com/xlmine/help/NNC/NNClass\\_intro.htm](http://www.resample.com/xlmine/help/NNC/NNClass_intro.htm)

[16] Wikipedia, Propagación hacia atrás - La propagación hacia atrás de errores o retropropagación, editado 18 mar 2009,

[http://es.wikipedia.org/wiki/Back\\_propagation](http://es.wikipedia.org/wiki/Back_propagation)