RECONOCIMIENTO DE ROSTROS UTILIZANDO SECUENCIAS DE HISTOGRAMAS COMO TRAMAS ESPACIO-TEMPORALES

Juan Moreno[1], Francisco J. Gómez[2], Miguel A. Fernández[2], A. Fernández Caballero[2]
Departamento de Informática, Escuela Politécnica Superior,
Universidad de Castilla-La Mancha, 02071 – Albacete, España
e_mail: [1] jmoreno@sancho.info-ab.uclm.es
e_mail: [2]{fgomez, miki, caballer}@info-ab.uclm.es

RESUMEN DEL TRABAJO. Este artículo describe un modelo de reconocimiento de rostros humanos, utilizando redes espacio-temporales (STN). Esta red neuronal incorpora una capa de entrada que se encarga de transformar la imagen de entrada al sistema en una secuencia de vectores normalizados que serán aplicados a una red espacio-temporal. Finalmente, la red tiene una capa de salida que utiliza unidades outstar de Grossberg[1].

1. INTRODUCCION

El reconocimiento de rostros es un problema ampliamente abordado utilizando diferentes tipos de características. El principal problema que aparece es que las características varían con pequeñas modificaciones de la posición o el gesto del rostro. Así pues, el problema debe ser abordado con herramientas de amplia tolerancia a fallos, de las que son un buen ejemplo las redes neuronales. Dentro de este campo, aparecen diferentes soluciones como las redes de retropropagación, perceptrón multicapa, etc. Nosotros hemos preferido abordar el problema desde otra perspectiva.

Este trabajo propone una solución al reconocimiento de rostros, que convierte la imagen bidimensional en una secuencia espacio-temporal de vectores. Se trata de un modelo que, a partir de la imagen, calcula el histograma de cada una de las filas que la componen. Una imagen de NF filas y NC columnas con NG niveles de gris, dará lugar a NF histogramas, cada uno correspondiendo a una fila, de NG componentes cada uno. La secuencia ordenada de estos NF histogramas puede utilizarse para identificar una imagen. Aunque en teoría es posible que dos imágenes diferentes tengan la misma secuencia ordenada de histogramas, la posibilidad de que esta situación se produzca en un proceso de identificación es casi nula. En una secuencia de histogramas normalizados, cada histograma corresponde a una fila de la imagen. En ella, se ofrece una información bastante robusta acerca de la luminosidad de la misma. Nuestra solución propone analizar el rostro como una secuencia del nivel de luminosidad desde la parte superior a la parte inferior del mismo. Por sus características, las redes espacio-temporales [3] son una buena herramienta para reconocer estas secuencias ordenadas con una gran

tolerancia a fallos, por lo que la consideramos adecuada para la naturaleza de la señal de entrada (secuencia de vectores de histogramas normalizados).

La solución que presentamos se describe como sigue. La imagen es la entrada a la primera capa (*capa de entrada),* que calcula los histogramas correspondientes a cada una de las filas de la imagen, y procede a continuación a normalizar dichos histogramas. Una capa interna recibe la salida de la capa de entrada. Dicha capa interna es una red espacio-temporal. Finalmente, una capa de salida (*outstar*), tiene como función la de elegir el rostro ganador de entre todas los rostros aprendidos. Este proceso se muestra en la figura 1.

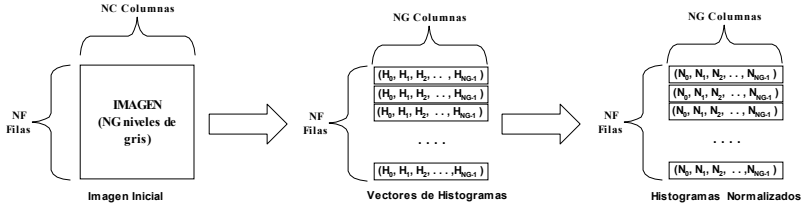


Figura 1. Función de la capa de entrada

Cada imagen de entrada se corresponde con una secuencia de histogramas (vectores) normalizados; la secuencia de histogramas en el tiempo identificará la imagen. Sea $\{Q_{11}, Q_{12}, ..., Q_{1n}\}$ la secuencia de histogramas normalizados que entra en nuestra red; la avalancha [2] para el reconocimiento de un rostro viene representada en la figura 2.

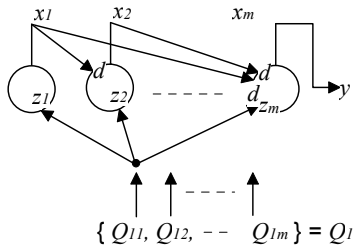


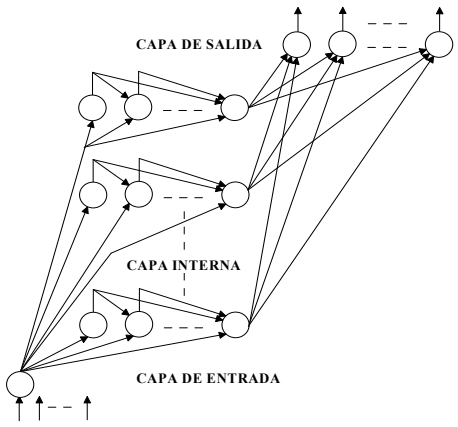
Figura 2. Estructura de la red

Cada uno de los Q_{1i} es un vector normalizado (función que realiza la *capa de entrada*), y se aplica a las entradas de todas y cada una de las unidades de la *STN*; se les permite permanecer allí durante un tiempo *t*; después se aplica el vector

siguiente Q_{1i+1}, y así sucesivamente. Durante ese tiempo *t*, cada unidad ajusta dinámicamente su activación y sus valores de salida de acuerdo con la regla que veremos más adelante. Obsérvese que la salida de cada unidad está conectada a cada una de las unidades sucesivas, de izquierda a derecha, con una intensidad de conexión *d*, donde *0<d<1*. Después de procesar todos los dígitos por las capas de entrada y *STN*, la *outstar* elige un rostro entre los aprendidos.

2. DESCRIPCION

Hemos diseñado una arquitectura de tres capas, como se muestra en la figura 3. Una *capa de entrada* con un único elemento de proceso transforma la imagen de entrada. Una *capa interna*, que responde a una red espacio-temporal, aprende una secuencia de vectores normalizados. Y la *capa de salida* asocia a la subred ganadora con un rostro de entre los aprendidos.



La imagen se pasa a la única unidad de la capa de entrada. Entonces dicha capa propaga su salida (secuencia de histogramas normalizados) simultáneamente a todas las unidades de la capa oculta. Una vez procesada toda la imagen por las capas de entrada y oculta, la capa de salida asocia el grupo ganador de la capa oculta con un rostro que muestra como salida.

Figura 3. Arquitectura de nuestra red

2.1 La capa de entrada

La capa de entrada realiza la misión de transformar una imagen de entrada en una secuencia de vectores de histogramas normalizados. La imagen consta de NF filas y NC columnas, con NG niveles de gris. Esta capa calcula el histograma de cada una de las NF filas, dando como resultado una secuencia de NF histogramas de NG

componentes cada uno. Por último, y dado que las *STN* [3][4][5] requieren que su entrada esté normalizada, se normaliza cada uno de estos NF histogramas según la ecuación:

$$X = \left(\frac{x_1}{\sqrt[2]{x_1^2 + \ldots + x_{NG}^2}}, \frac{x_2}{\sqrt[2]{x_1^2 + \ldots + x_{NG}^2}}, \ldots, \frac{x_{NG}}{\sqrt[2]{x_1^2 + \ldots + x_{NG}^2}} \right) \quad (2.1.1)$$

2.2 La capa interna (STN)

En nuestra arquitectura, esta capa corresponde a una red espacio-temporal, compuesta de NI*NF unidades, donde NI es el número de imágenes (rostros) a aprender, y NF es el número de elementos (histogramas) de que consta una secuencia. Estas redes se rigen por una serie de ecuaciones que pueden consultarse en [3][6].

La capa interna está dividida en grupos de NF unidades, cada una de las cuales se encarga de aprender y reconocer los NF histogramas correspondientes a una imagen. La salida de la última unidad de cada grupo será el valor de salida del grupo. La unidad de salida de cada grupo que tenga mayor respuesta será la única que tenga salida no nula. Es decir, convertimos a 1 la mayor de las salidas de cada grupo, y a 0 el resto. Dichas salidas convertidas son las que la capa de salida recibirá como entrada.

Con respecto a los parámetros, a δt le hemos asignado el valor 1, con el fin de dejar tal cual la salida del valor de actualización. Al parámetro *c*, que representa el valor de descarga de la intensidad de salida de cada unidad, le hemos asignado el valor 1/NF con el fin de que una vez cargada una unidad, ésta no se descargue rápidamente. La *entrada total* se filtra con un *valor umbral(Γ)*. Para nuestra aplicación hemos asignado a Γ el valor 0, puesto que un valor umbral mayor no permitiría que las primeras unidades de la red se activasen, ya que el vector normalizado tiene componentes con valor muy bajo, debido a que estamos trabajando con 256 componentes (niveles de gris). A los parámetros *a* y *b*, que indican la importancia de la salida anterior y la *entrada total* respectivamente, les hemos asignado a ambos el valor 1, puesto que queremos que pese de igual forma la activación anterior de la unidad y la entrada a la unidad en un instante dado. El parámetro *d* representa la influencia de las conexiones de las unidades anteriores a la *entrada total* de la unidad. Hemos decidido que tome el valor 1/NF para no perder la carga de una secuencia de histogramas.

2.3 La capa de salida

Finalmente, la capa de salida consta de NF elementos de proceso llamados *outstar* [3][1]. La *outstar* alcanza rápidamente un valor de equilibrio igual al valor del peso que haya en la conexión procedente de la unidad ganadora de entre las salidas de los grupos. Una forma sencilla de visualizar este procesamiento consiste en darse cuenta de que la salida de equilibrio de la *outstar* es igual a la entrada de la *outstar*,

$$y_k^{'eq} = \sum_j w_{kj} z_j$$

donde z_j es la entrada que se recibe del correspondiente grupo de la *STN*. Dado que $z_j = 0$ a no ser que $j=i$, entonces:

$$y_k^{'eq} = w_{ki} z_i = w_{ki} \qquad (2.3.1)$$

Este sencillo algoritmo utiliza valores de equilibrio de las actividades y salidas de los nodos. De esta manera, se evita tener que resolver numéricamente todas las ecuaciones diferenciales correspondientes.

2.4 Aprendizaje

En este apartado vamos a describir el aprendizaje de las distintas capas de nuestra arquitectura. La *capa de entrada* no necesita una fase de aprendizaje, pues para la obtención de los NF histogramas normalizados no es necesario ningún aprendizaje. El aprendizaje de cada unidad de la *STN* se realiza a partir del vector de pesos inicial *w*, que va evolucionando según la ecuación diferencial $\dot{w} = -cw + dIy$, en donde y es la salida, y $c,d>0$. La misión de una unidad de la *STN* es aprenderse un vector de entrada y normalizado, proporcionando una mayor intensidad de salida cuanto más se parezca el vector de entrada al vector aprendido. Por otra parte, los pesos de cada unidad *outstar* de la *capa de salida* evolucionan durante su proceso de aprendizaje, según la ecuación diferencial

$$\dot{y}'_i = -ay'_i + by_i + c * net_i$$

donde $a, b, c>0$ y $net_i = I*W$.

3. IMPLEMENTACION

Para probar la solución propuesta se ha implementado un simulador utilizando el lenguaje de programación *Visual C++ 5.0*. Dicho simulador acepta como entrada una imagen y genera como resultado otra imagen. La imagen de entrada corresponde al rostro que queremos reconocer y la de salida al rostro seleccionado de entre los aprendidos. Aparte de las clases que genera automáticamente *Visual C++ 5.0*, hemos implementado las clases correspondientes a las distintas capas y de todo el conjunto de nuestra red.

La implementación de la capa de entrada es muy simple; basta con implementar la ecuación (2.1.1) que normaliza como un método de la clase *CEntrada*. La *STN* está formada por un conjunto de unidades. Para implementar una unidad se ha realizado la clase *CUnit*. Esta clase está formada por un conjunto de atributos y métodos que realizan las funciones descritas en el apartado 2.2. Además, está parametrizada con un entero que indica el número de entradas procedentes de las unidades anteriores de su grupo. La clase *CRedSTN* (red *espacio-temporal*) está formada por una lista de objetos *CUnit*. Esta clase se encarga de que cada unidad realice su aprendizaje y propagación, y finalmente, se realiza la competición entre las salidas de la última unidad de cada grupo. Esta competición no es otra cosa que cambiar a 1 la salida de la unidad con mayor intensidad, y a 0 el resto. La capa de salida *outstar* consta de un conjunto de objetos de la clase *COutstar*. Esta clase utiliza la fórmula (2.3.1) para calcular su salida. Con respecto al aprendizaje, a cada $outstar_i$ de la *capa de salida*, nuestra clase asigna a cada peso w_i el valor correspondiente.

La arquitectura completa utiliza las clases ya descritas *CEntrada*, *CRedSTN* y *COutstar*, formando una clase general, *COurArchitecture*, con constructor parametrizado con dos parámetros que indican el número de grupos y el número de elementos de la secuencia de entrada, es decir, la clase *COurArchitecture* puede tener un número variable de grupos a aprender y de elementos de dichos grupos. Así, este diseño nos permite implementar arquitecturas flexibles.

4. RESULTADOS

Para la verificación de resultados hemos preparado una batería de pruebas, formada por 16 series de 6 imágenes cada una. Las imágenes tienen una dimensión de 112 filas por 92 columnas, con 256 niveles de gris. Dichas series se muestran en la figura 4. Damos las gracias por la cortesía de *The ORL Database of Faces* (www.cam-orl.co.uk/facedatabase.html) por el uso público de estas imágenes.

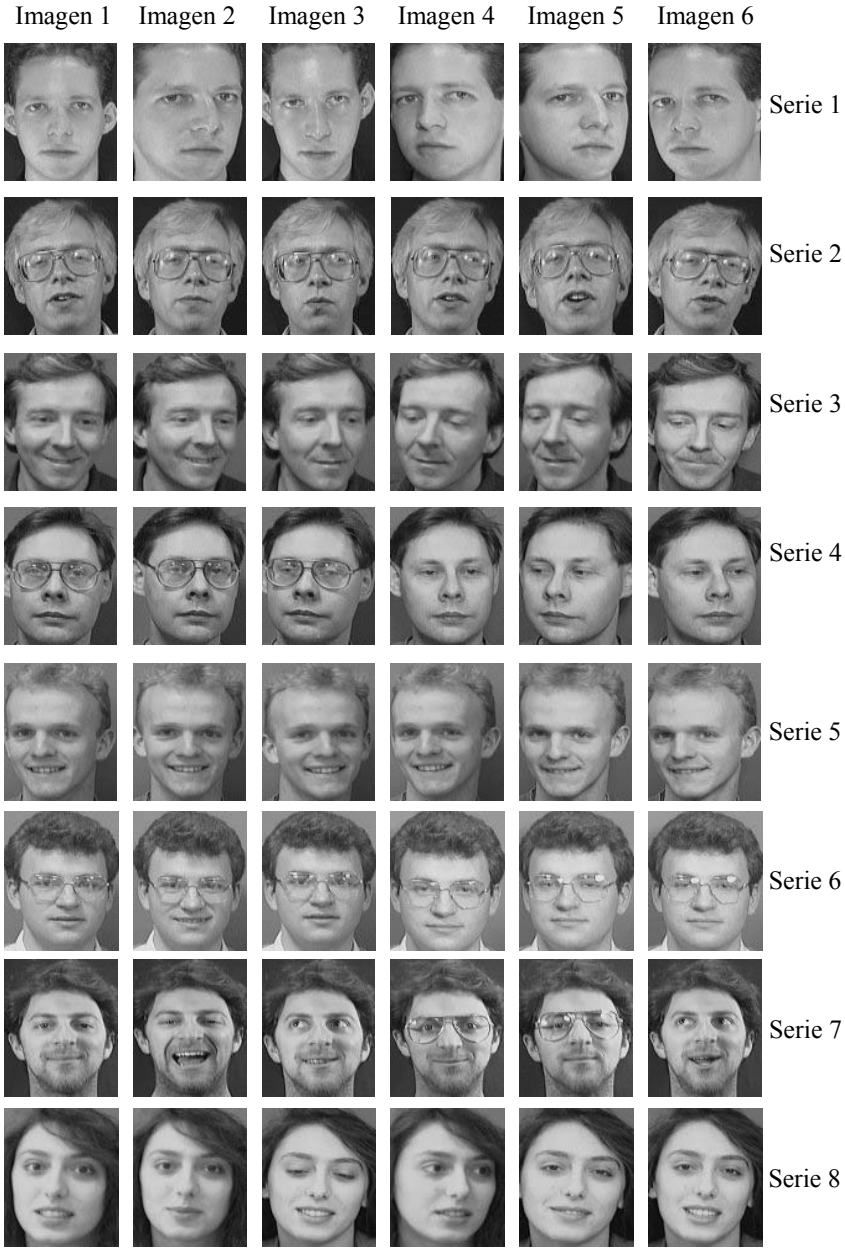


Imagen 1
Imagen 2
Imagen 3
Imagen 4
Imagen 5
Imagen 6
Serie 1
Serie 2
Serie 3
Serie 4
Serie 5
Serie 6
Serie 7
Serie 8

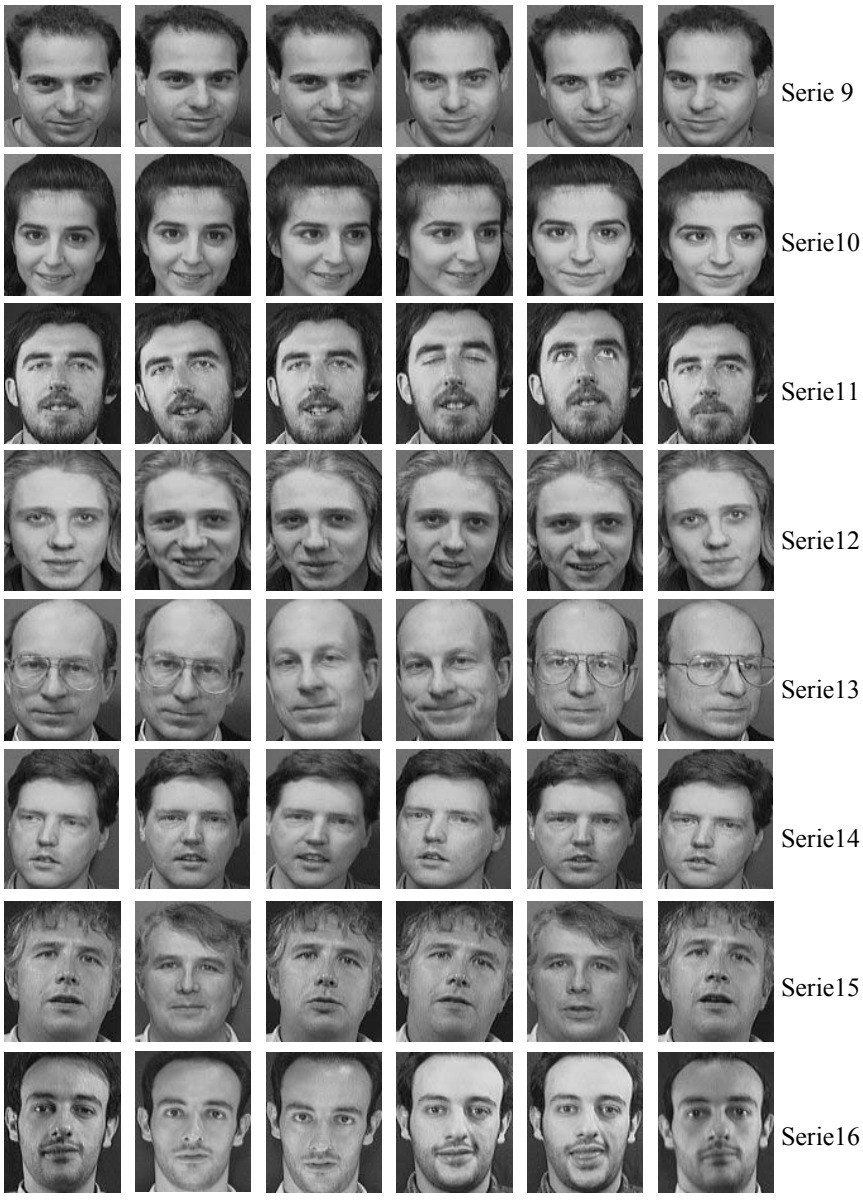


Figura 4. Series de imágenes

El aprendizaje se ha realizado mostrando al sistema únicamente la primera imagen de cada serie. Los resultados obtenidos se indican en la tabla 1. Un asterisco indica que la imagen de la serie correspondiente no ha sido reconocida.

		Series															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Imagen	1																
	2																
	3			*									*	*	*	*	*
	4			*	*				*	*			*	*		*	*
	5			*	*			*	*	*			*		*	*	*
	6			*	*				*	*						*	*

Tabla 1. Resultados

Como se puede observar, 6 de las 16 series son reconocidas totalmente. Todas ellas tienen en común que la luminosidad se mantiene constante en toda la serie. Algunas veces, en series en las que se aprecia un cambio importante de luminosidad, se producen fallos en la identificación; véase como ejemplo la serie 16 en las imágenes 4 y 5. Otra situación que provoca fallos es que en una misma serie el individuo pueda aparecer con o sin gafas. Véanse como ejemplo, las series 4 y 13, aunque hay casos en que no representa un problema (véase serie 7, imagen 4).

Este método presenta una gran tolerancia a fallos, obteniéndose muy buenos resultados cuando se produce un giro o movimiento de la cabeza que no afecta a la luminosidad de la imagen. Véanse como ejemplo las series 1, 5 y 11. Las series elegidas son representativas de un gran abanico de rostros posibles (mujeres, hombres, con o sin barba, calvos o con pelo, con gafas o sin ellas, rubios o morenos, etc.), obteniendo un resultado global más que aceptable. Así, de 96 imágenes, sólo se han presentado 29 fallos en la identificación, lo cual nos da un 70% de aciertos. Hay que tener en cuenta que no se ha realizado ningún tipo de preprocesado a las imágenes para intentar corregir o mejorar ciertas características, como por ejemplo la luminosidad. Además, casi todas las imágenes contienen una zona de fondo que no es significativa y que por tanto se puede eliminar, lo cual reduciría con seguridad la tasa de fallos.

5. CONCLUSIONES

La principal novedad aportada por este trabajo es el hecho de abordar el reconocimiento de rostros considerando éstos como una secuencia de histogramas,

es decir, luminosidad. Para nuestro sistema, la secuencia de histogramas por filas, desde la parte más alta del rostro hasta la parte inferior, configura una trama espacio-temporal que identifica dicho rostro. Como puede verse, la capacidad de la red espacio-temporal para absorber variaciones en las secuencias de entradas se adecúa muy satisfactoriamente a las variaciones que aparecen al variar la posición o el gesto del rostro, siendo la información de la secuencia de luminosidad almacenada en la STN suficiente para diferenciarlo del resto de rostros que han configurado la secuencia de entrenamiento.

Con respecto a los resultados de simulación, destacar que el índice de aciertos es del 70%. Además, se debe recordar que no se ha hecho ningún tipo de preprocesado a las imágenes de entrada. Comentar también que este método es bueno en los casos en que el rostro aparece girado o desplazado. Conviene hacer notar que uno de los puntos clave del diseño del sistema es el ajuste de los parámetros de funcionamiento, ya que un mal ajuste de éstos puede llevar a que la STN funcione inadecuadamente.

A la vista de los resultados obtenidos, consideramos que la solución propuesta puede ser adecuada para reconocimiento de rostros en aplicaciones donde el número de rostros a reconocer no sea excesivo, ya que esto haría a la STN computacionalmente inabordable a sus dimensiones.

6. REFERENCIAS

[1] S. Grossberg. Studies of Mind and Brain. Vol. 70 de Boston Studies in the Philosophy of Science. D.Reidel Publishing Company, Boston, 1982.

[2] S. Grossberg. Learning by Neural Networks. Editado por Stephen Grossberg en Studies of Mind and Brain. D. Reidel Publishing, Boston, MA, 65-156, 1982.

[3] J. A. Freeman y D. M. Skapura. Redes Neuronales. Algoritmos, aplicaciones y técnicas de programación. Addison-Wesley/Diaz de Santos, USA, 1993.

[4] R. Hecht-Nielsen. Neurocomputing. Addison-Wesley, Reading, MA, 1990.

[5] R. Hecht-Nielsen. Nearest matched filter classification of espatiotemporal patterns. Informe técnico, Hecht-Nielsen Neurocomputer Corporation, San Diego, CA, Junio de 1986.

[6] J. Moreno, G. Sebastián, Miguel A. Fernández y A. Fernández Caballero . A Neural Architecture for the Identification of Number Sequences. Proceedings of the Fifth Brazilian Symposium of Neural Networks SBRN´98, Belo Horizonte, (Brasil), Diciembre 9-11, 1998.