Reconocimiento de Comandos Gestuales utilizando GesRN

Diego Yanivello[1]
diegoy@sol.info.unlp.edu.ar

LIFIA (Laboratorio de Investigación y Formación en Informática Avanzada)

Lic. Laura Lanzarini[2]
laural@lidi.info.unlp.edu.ar

III LIDI (Instituto de Investigación en Informática LIDI)

Facultad de Informática. Universidad Nacional de La Plata.
La Plata, Argentina, 1900

Resumen

La redes neuronales artificiales han demostrado tener un muy buen desempeño en la resolución de problemas pertenecientes al área de reconocimiento de patrones debido a su alta tolerancia al ruido existente en la información de entrada.

Este artículo presenta una nueva arquitectura neuronal, denominada **GesRN**, formada por varias redes neuronales, que permite realizar la interpretación de comandos gestuales utilizados en el acceso a las diferentes funcionalidades de una aplicación.

Un comando gestual es una figura realizada con un dispositivo digital manual, como el mouse o el lápiz óptico, que permite ejecutar un conjunto de acciones asociadas. La correcta interpretación de este tipo de comandos resulta una tarea compleja debido a la gran variedad de patrones que pueden generarse para un mismo gesto.

Las mediciones realizadas han demostrado que **GesRN** posee una alta efectividad en la resolución del problema planteado.

Según la arquitectura propuesta, la incorporación de nuevos gestos no obliga a realizar un reentrenamiento masivo de toda la estructura reduciendo de esta forma el tiempo de aprendizaje.

Finalmente se presentan las conclusiones y se plantean algunas líneas de trabajo futuras.

Palabras clave: Redes Neuronales. Aprendizaje. Comandos gestuales.

[1]Becario LIFIA - Fac. Informática - UNLP
[2]Profesor Titular Dedicación Exclusiva. Fac. de Informática. Universidad Nacional de La Plata.

1 Introducción

Las aplicaciones proveen diferentes formas de acceder a su funcionalidad, ya sea por medio de menúes desplegables, menúes contextuales o teclas de acceso rápido. Algunas aplicaciones más modernas poseen una forma diferente de acceder utilizando como disparador un *comando gestual* [11].

Un comando gestual es una *figura* realizada con un dispositivo digital manual como el mouse o el lápiz óptico que, en caso de ser reconocido por el sistema como válido, ejecuta una o mas acciones asociadas a dicho gesto.

El trabajo de reconocimiento de gestos es un problema complejo y requiere tener en cuenta los siguientes factores:

- La captura de los puntos que representan el recorrido realizado con el dispositivo manual al hacer el gesto, varía según la velocidad con la que éste se dibuja. Esto provoca que la entrada del algoritmo de reconocimiento cambie, aún para gestos que han sido dibujados de misma forma pero a distinta velocidad.
- El algoritmo debe tener una cierta tolerancia a distorsiones del gesto. Puesto que los gestos son comúnmente utilizados como *short-cuts*, estos se dibujan con un movimiento rápido, lo que hace que no se representen siempre de la misma forma.
- El mismo gesto realizado en dos instantes de tiempo puede variar en tamaño.
- Los gestos que el algoritmo deberá reconocer no están prefijados, sino que se van agregando al sistema a medida que el usuario lo requiera.

En la sección 2 se describe la forma de representación adoptada para cada gesto que es independiente de la escala y la velocidad de dibujo utilizadas. También se propone la manera de modificarla para obtener una representación independiente de la rotación. La sección 3 describe detalladamente la arquitectura utilizada. En la sección 4 se presentan los resultados obtenidos en la resolución del problema planteado y finalmente, en la sección 5 se analizan las conclusiones y se proponen algunas líneas de trabajo futuras.

2 Representación de Gestos basada en ángulos relativos

Existen diferentes alternativas para representar un gesto. En este trabajo se adoptó una representación basada en los ángulos relativos que se forman entre diferentes segmentos dentro del recorrido. Otras opciones basadas directamente en la gráfica han sido descartadas por su distorsión a los cambios de tamaño o a pequeñas variaciones dentro de un mismo gesto.

Para poder obtener los ángulos es preciso realizar una interpolación de los puntos que componen la figura del gesto. Luego se particiona la curva en un número fijo de secciones y finalmente se miden los ángulos entre secciones consecutivas. De esta forma se obtiene una secuencia de ángulos que indican las variaciones relativas a puntos anteriores en la traza del gesto.

Esta representación resuelve varios de los problemas existentes en la forma en que el gesto es captado. Por un lado, la interpolación asegura la independencia de la velocidad con la cual se

registra el movimiento. Por otro lado, la representación a través de ángulos es independiente de la escala del gesto. También es interesante considerar que sólo el primer ángulo de la secuencia es quien marca la rotación absoluta del gesto; es decir que podría obtenerse una representación independiente de la rotación con solo ignorar el primer ángulo de la secuencia. La figura 1

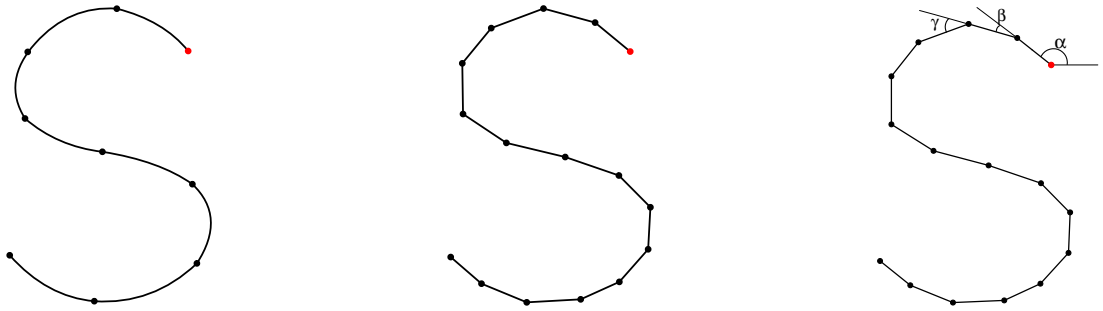

Figura 1: (a) Representación del gesto, (b) Gesto dividido en secciones, (c) Ángulos del gesto

muestra un ejemplo de un gesto en forma de `S` cuya especificación consta de 9 puntos (Fig.1 a). Luego de la interpolación, el gesto es dividido en 15 segmentos (Fig.1 b). La representación a través de los ángulos está formada por, en primer lugar, el ángulo absoluto del primer segmento y a continuación los ángulos relativos entre cada par de segmentos adyacentes.

Los ángulos correspondientes a los segmentos de la figura 1.c) son los siguientes:

```
#(142°, 22°, 35°, 31°, 38°, 58°, 18°, -5°,-28°,-47°,-41°,-23°,-23°,-26°,-19°)
```

3 Arquitectura de GesRN

GesRN es una estructura formada por tres capas: una capa de entrada, una capa oculta y una capa de salida.

La capa de entrada recibe los ángulos del gesto a reconocer utilizando la representación indicada en la sección 2.

Cada elemento de la capa oculta, a diferencia de las otras capas de esta estructura, es una subred neuronal cuya arquitectura se describe en 3.1 y posee la capacidad de interpretar un gesto específico. Básicamente, estas subredes forman una capa competitiva del tipo "el ganador se lo lleva todo" donde cada uno de sus miembros, ante el estímulo de un gesto de entrada, calcula su grado de interpretación. Se considerará ganadora a la subred que mejor reconozca el gesto presentado. Todas las demás subredes tendrán salida nula.

La última capa puede verse como una *outstar* de Grossberg ya que sólo recibe excitación de la subred ganadora de la capa oculta. La cantidad de neuronas aquí empleadas dependerá de la representación que se desee utilizar para identificar cada gesto. Por ejemplo, podría asociarse a cada uno de ellos un número de orden y obtener como salida la representación binaria de dicho número. Nótese que los pesos que unen a cada subred con esta capa no requieren entrenamiento.

La Fig.2 ejemplifica la arquitectura **GesRN** anteriormente descripta suponiendo que la i-ésima subred es la ganadora de la competencia. Nótese que el vector de salida final coincidirá con

el vector de pesos que une la subred ganadora con cada una de las neuronas de la capa de salida permitiendo de esta forma elegir la representación que se considere más adecuada para identificar el gesto ganador.

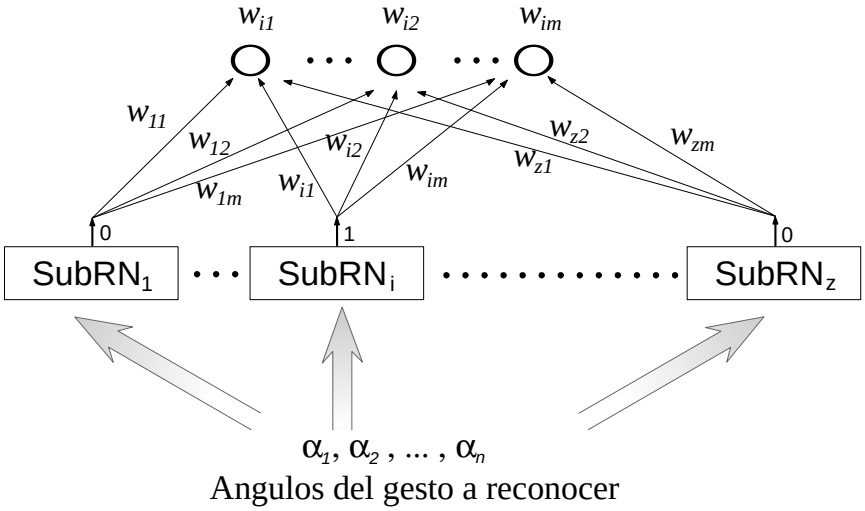

Figura 2: **GesRN**

3.1 Arquitectura correspondiente a cada subred de la capa intermedia

Cada subred de la capa intermedia de **GesRN** está formada por una arquitectura *feedforward*[1][2] de cuatro capas: una de entrada, dos ocultas y una de salida [4].

A continuación se describe el funcionamiento de las capas de cada subred junto con las funciones de activación utilizadas.

3.1.1 Capa de Entrada

Esta capa está formada por las neuronas $(e_1, e_2, \ldots, e_n)$ donde n es la cantidad de ángulos que caracterizan al gesto. Cada una de estas neuronas posee memoria propia en la cual se almacena el valor que le corresponde al ángulo del gesto patrón. De esta forma la primera neurona guardará el ángulo de la rotación absoluta del gesto, la segunda el primer ángulo relativo y así sucesivamente.

La entrada de esta capa estará dada por la secuencia de ángulos del gesto a evaluar y su salida dependerá de la diferencia entre los ángulos del patrón original y los del patrón actual.

La función utilizada para calcular la salida de cada neurona de esta capa debe poseer las siguientes propiedades:

- Su valor debe ser siempre positivo y representar el error entre el ángulo esperado y el ángulo recibido como entrada. Esto evita que los errores se cancelen al ser sumados a la entrada de la siguiente capa.
- La función debe separar claramente tres secciones:

- Zona Cercana al valor deseado (menor a 10 grados de error) donde la diferencia entre el ángulo real y el ángulo patrón es cercana a 0.
- Zona crítica (entre 10 y 50 grados de error) que representa los errores más significativos y por lo tanto, es el rango en el que se busca una distinción mayor.
- Zona de exclusión (más de 50 grados de error) que representa los casos que no deberían ser aceptados.

y es la indicada en (1)

$$f_i^e(p_i, o_i) = \frac{2}{e^{-(p_i - o_i)^2/700} + 1} - 1 \tag{1}$$

3.1.2 Capas ocultas y capa de salida

Cada subred consta de dos capas ocultas $H_1 = (h_{1_1}, h_{1_2}, \ldots h_{1_r})$ y $H_2 = (h_{2_1}, h_{2_2}, \ldots h_{2_t})$ y una capa de salida $S = (s_1)$; esta última formada por una única neurona.

Todas utilizan funciones de salida similares en cuanto a su expresión pero difieren en su sensibilidad a los valores de entrada. La forma general es la indicada en la ecuación 2.

$$f(x) = \frac{2}{e^{-\frac{x^2}{2*d}} + 1} - 1 \tag{2}$$

donde d es un factor que permite manipular la sensibilidad de cada capa. En particular se han utilizado los valores 60, 1 y 0.1 para determinar las funciones de salida de las capas 2, 3 y 4 respectivamente.

Cada neurona de cada capa, siguiendo el modelo *backpropagation*, se encuentra totalmente interconectada con las neuronas de la capa anterior y recibe como entrada el estímulo acumulado de la manera habitual, como puede verse en la Fig.3

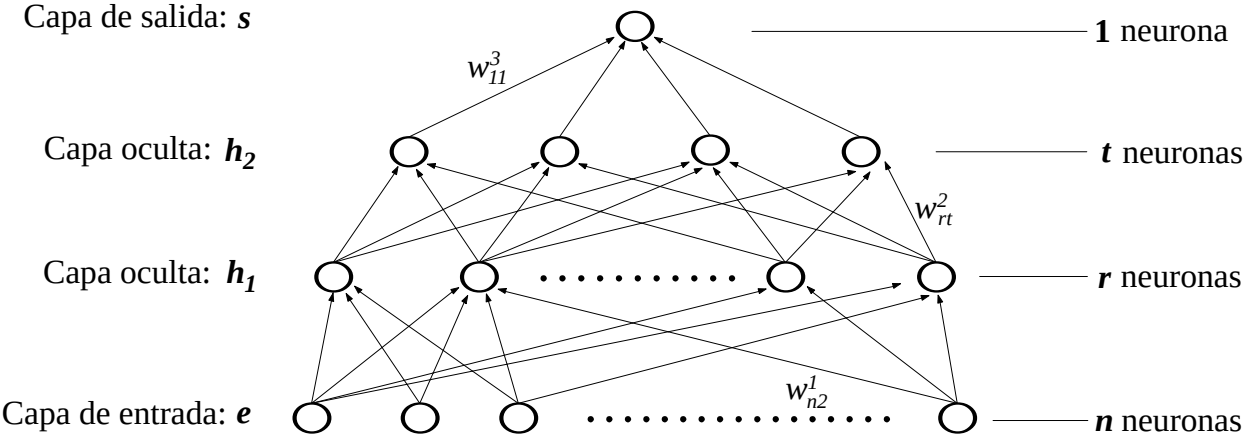

Figura 3: Forma de las subredes de **GesRN**

A continuación se describen los pasos a seguir para realizar la propagación hacia delante dentro de una subred, suponiendo que ya se ha efectuado el entrenamiento correspondiente.

Se utilizará la notación w_{ij}^N para el arco que va desde de la i-ésima neurona de la capa $(N-1)$ y llega a la neurona j-ésima de la capa N, para $N = 2, 3, 4$.

3.2 Correspondencia hacia adelante

(a) Ingresar a la subred el vector de entrada $P = (p_1, p_2, \ldots, p_n)$, correspondiente a la secuencia de ángulos del patrón actual a reconocer.

(b) Calcular la salida de cada neurona e_i de la capa de entrada de la siguiente forma:

$$f_i^e(p_i, o_i) = \frac{2}{e^{-\frac{(p_i - o_i)^2}{700}} + 1} - 1 \tag{3}$$

donde $O = (o_1, o_2, \ldots, o_n)$ es la secuencia de ángulos obtenidos del patrón original almacenada en la memoria de las neuronas de la capa de entrada, según lo expresado en 3.1.1. Por lo tanto, $(p_i - o_i)$ es un valor perteneciente al intervalo $[-180 \ldots 180]$.

(c) Calcular la entrada neta y la salida de cada neurona de la primer capa oculta de la siguiente forma:

$$neta_j^{h_1} = \sum_{i=1}^{n} f_i^e(p_i - o_i) * w_{ij}^{h_1} \qquad \text{con } j = 1 \ldots r \tag{4}$$

$$f_j^{h_1}(neta_j^{h_1}) = \frac{2}{e^{-\frac{(neta_j^{h_1})^2}{120}} + 1} - 1 \tag{5}$$

(d) Calcular la entrada neta y la salida de cada neurona de la segunda capa oculta de la siguiente forma:

$$neta_k^{h_2} = \sum_{j=1}^{r} f_j^{h_1}(neta_j^{h_1}) * w_{jk}^{h_2} \qquad \text{con } j = 1 \ldots t \tag{6}$$

$$f_k^{h_2}(neta_k^{h_2}) = \frac{2}{e^{-\frac{(neta_k^{h_2})^2}{2}} + 1} - 1 \tag{7}$$

(e) Calcular la entrada neta y la salida de la única neurona que compone la capa de salida, que dará como resultado el valor de aceptación del gesto presentado a la red:

$$neta_1^o = \sum_{k=1}^{t} f_k^{h_2}(neta_k^{h_2}) * w_k^s \tag{8}$$

$$f^s(neta_1^o) = \frac{2}{e^{-\frac{(neta_{s1}^o)^2}{0.2}} + 1} - 1 \tag{9}$$

Como puede observarse, todas las funciones de activación retornan valores cercanos a cero cuando reciben una entrada neta baja; y a medida que el estímulo total recibido aumenta, la salida se va aproximando a 1. En otras palabras, cada neurona busca representar la presencia de error.

3.3 Entrenamiento de cada subred

Para el entrenamiento de cada subred se crean patrones similares al gesto dado por el usuario, obtenidos mediante la aplicación de un *factor de distorsión* a cada ángulo de la representación, determinando así dos grupos de patrones:

- Patrones con distorsión baja, los cuales deberán ser aceptados por la red
- Patrones con distorsión alta, los cuales deberán ser rechazados

El algoritmo utilizado es el de backpropagation [10] aplicando las funciones descriptas en 3.2. La expresión del error también ha sido adaptada para este problema [6].

La ecuación 10 muestra la expresión utilizada:

$$f(x) = \frac{(ValorEsperado - x) * (0.0175 + (x * (1 - x)))}{3} \tag{10}$$

donde *ValorEsperado* es cero para los patrones con distorsión baja y 1 para los que poseen distorsión alta.

El algoritmo de entrenamiento de la i-ésima subred de la estructura **GesRN** es el siguiente:

Inicializar los pesos de $SubRN_i$ de manera aleatoria.
CantIteraciones := 0
repetir
    CantInteraciones := CantIteraciones + 1
    Para cada patrón (i)
        Propagar el patrón hacia delante según los pasos de 3.2
        Calcular el error de la capa de salida. (ii)
        Calcular el error de cada capa oculta. (iii)
        Modificar los vectores de pesos adecuadamente.
hasta *obtener un error aceptable* (iv)
    o *(CantIteraciones > CantMax)*

En (i) se utilizan tanto los patrones que deben ser aceptados por la red como los que deben ser rechazados.
El valor de salida esperado en (ii) será cero para los patrones que deben ser aceptados y 1 en caso contrario.
El cálculo del error de (ii) y (iii) se realiza según el algoritmo backpropagation [1][5][8].
(iv) Todas las subredes utilizan un mismo umbral que representa el máximo error aceptable. Si el patrón de entrada debe ser aceptado, es decir si se trata del patrón original mas una distorsión baja, el valor de salida deberá estar por debajo de este umbral; y si debe ser rechazado, deberá superarlo.

3.4 Detalles de implementación

Para la generación de los patrones de entrenamiento de cada gesto se utilizaron los siguientes factores de distorsión:

- Factor de aceptación: 0,05

  En este caso la modificación del ángulo original se realiza mediante un valor aleatorio en el rango [-180*0,05 ... 180*0,05]. Es decir que se utilizan valores en [-9 ... 9].

- Factor de rechazo: 0,12

  En este caso la modificación del ángulo original es mediante una valor aleatorio en el rango [-21,6 ... 21,6].

En lo que se refiere a la cantidad de neuronas utilizadas en cada capa de una subred [9], se tuvieron en cuenta diferentes aspectos:

- La cantidad de neuronas de la capa de entrada se corresponde con la cantidad de segmentos en que se divide un gesto. Si bien, una mayor cantidad de neuronas permitirá una mejor especificación del gesto de entrada, el número utilizado es directamente proporcional al incremento del tiempo del proceso de entrenamiento. El número de neuronas utilizados en esta capa fue de 15.

- La capa de salida consta de una única neurona que permite obtener un valor acorde al reconocimiento por parte de la subred del gesto presentado.

- Las capas ocultas buscan reconocer características generales del gesto[7]. Nuevamente, si se utiliza un número bajo de neuronas, la red no podrá lograr la generalización adecuada y por el contrario, si son demasiadas se producirá un ajuste excesivo inadecuado. Las pruebas realizadas demostraron que para una entrada de 15 neuronas, se obtienen resultados adecuados con 15 neuronas en la primer capa oculta y 10 en la segunda capa oculta.

[5][8] Considerando que cada subred posee aproximadamente 45 neuronas y que se utilizan 20 patrones de entrada por cada gesto, un entrenamiento de 150 iteraciones sobre cada arquitectura implica evaluar 135000 funciones de activación por cada gesto que se desee reconocer.

La implementación de este trabajo está realizada en *Smalltalk* sobre VisualWorks$^{(r)}$ 7. Dado que en *Smalltalk* la evaluación de expresiones aritméticas no es óptima, fue necesario analizar diferentes estrategias para reducir el tiempo de evaluación de cada función de activación. A continuación se detallan las soluciones propuestas:

- Tener una tabla de valores precalculados de la función de activación (en el rango en que será utilizada), con una precisión determinada (por ejemplo, millonésimos).

  Cuando se requiere el cálculo de la función de activación para un valor x, este se aproxima al valor más cercano en la tabla (simplemente perdiendo precisión) y se accede al resultado directamente usando una cuenta simple sobre x para determinar el índice dentro de la tabla.

- Utilizar una librería de linking dinámico *DLL* escrita en $\mathbf{C}^{++}$ con la implementación de las funciones de error. Esta librería es cargada desde *Smalltalk* en la primera invocación de la función de activación.
- Utilizar una combinación de los métodos anteriores, implementando una tabla de valores precalculados dentro de la misma librería *DLL*.

A continuación se presenta una tabla de comparación entre las distintas estrategias de implementación de la función de activación. La tabla indica para cada función, el tiempo mínimo, máximo y medio de ejecución[3].

	Benchmark	**Minimum**	**Maximum**	**Median**
1	Signed Sigmoid (ecuación expresada en (2))	21.793	22.504	22.056
2	Precalculated Signed Sigmoid	13.294	15.192	15.144
3	Signed Sigmoid from DLL	7.674	8.227	7.744
4	Precalculated Signed Sigmoid from DLL	7.482	8.227	8.106

4 Resultados obtenidos

Se ha desarrollado un ambiente que permite cubrir los tres aspectos de este trabajo: la definición de gestos, el entrenamiento en base a distorsiones de los patrones originales especificados y el reconocimiento de una entrada dada permitiendo de esta forma su asociación con una acción determinada.

Los gestos a ser utilizados en la etapa de entrenamiento son distorsiones de un patrón original el cual es especificado a partir de ciertos puntos de referencia o *knots*. Para llevar a cabo esta tarea se ha desarrollado un editor que permite capturar la información indicada mediante un dispositivo externo y señalar allí los puntos de interés y la función de interpolación a utilizar. Si bien actualmente el entorno sólo permite interpolación spline lineal y cúbica [3], esto puede extenderse fácilmente en caso de ser requerido.

Luego de definir un gesto es necesario asociarle la acción a realizar. Si se trata de un gesto global al sistema, la acción asociada estará implementada como un bloque de código *Smalltalk*, que recibe como parámetro la ventana sobre la cual se realizó el gesto. Si se trata de un gesto local a una ventana, se enviará directamente a la aplicación un mensaje configurado por el usuario. Es importante considerar que una acción puede ser cualquier expresión *Smalltalk*; por ejemplo abrir, cerrar o minimizar una ventana, escribir un texto, inspeccionar el texto seleccionado, abrir un browser de clases, etc.

Los gestos para los cuales se ha definido una forma y una acción, son incorporados en el sistema. Posteriormente se entrenan las redes asociadas a cada uno de ellos dando lugar así a la estructura **GesRN**. Recuérdese que la capa de salida sólo depende de la representación final deseada para cada gesto asignando a los vectores de pesos correspondientes los valores esperados.

[3]Si bien estos tiempos de ejecución están ligados a las características del equipo en donde fueron medidos, sirven como punto de comparación para las estrategias de evaluación descriptas.

El proceso de reconocimiento utilizando **GesRN** con varios gestos simultáneamente, ha dado resultados satisfactorios. La arquitectura propuesta ha demostrado ser efectiva en un 97% de los casos, incluso cuando se trata de reconocer gestos muy similares (por ejemplo, un gesto con forma de `C` y otro de `G`). En la figura 4 se muestran algunos gestos que fueron utilizados como prueba, y cuyo resultado fue satisfactorio.

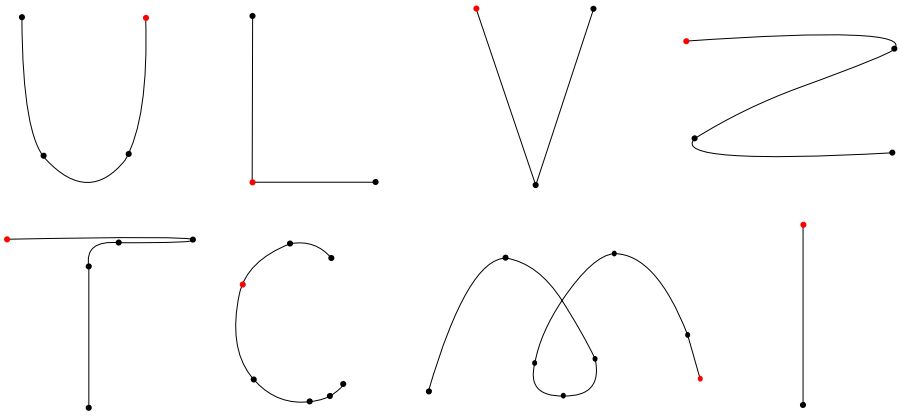

Figura 4: Gestos de ejemplo

5 Conclusiones y Líneas de Trabajo Futuras

Se ha presentado una arquitectura formada por varias redes neuronales que permite realizar el reconocimiento de comandos gestuales con un muy buen desempeño. Según el modelo propuesto, la incorporación de nuevos gestos no obliga a realizar un reentrenamiento masivo de toda la estructura reduciendo de esta forma el tiempo de aprendizaje.

Además, se ha desarrollado un ambiente de programación que permite cubrir todos los aspectos de este trabajo incluyendo la representación automática de los gestos.

Pese a los buenos resultados obtenidos, se han detectado algunos problemas basados en la representación utilizada que lleva a continuar la investigación en esta dirección. Esto se debe a que gestos muy distintos ocasionalmente pueden tener asociadas representaciones similares diferenciándose únicamente en el ángulo inicial. Esta característica puede resultar negativa, ya que la responsabilidad de reconocer un gesto está concentrada sólo en una neurona, y no distribuida a lo largo de la red.

Como alternativas para resolver este problema se están analizando las siguientes modificaciones a la arquitectura:

- Utilizar más de una neurona que reciba como entrada el primer ángulo del gesto. Esta *repetición* de neuronas permitirá que pequeñas diferencias en el primer ángulo, tengan mas participación en el error final que produce la red.
- Cambiar la función de activación de la primer neurona en la capa de entrada, para que la diferencia entre el ángulo esperado y el dado, produzca un error más significativo.

Pese a estas observaciones referidas a la representación, las pruebas realizadas demuestran una efectividad del 97%, permitiendo afirmar que se trata de una arquitectura adecuada para la resolución de este tipo de problema donde se presentan una gran variedad de patrones de entrada asociados a un mismo gesto.

Referencias

[1] J.A.Freeman and D.M.Skapura.
Neural Networks, Algorithms, Applications and Programnig Techniques.
Addison-Wesley, 1993

[2] S. Abid, F. Fnaiech, and M. Najim.
A Fast Feedforward Training Algorithm Using a Modified Form of the Standard Backpropagation Algorithm.
IEEE Trans on Neural Networks, Vol. 12, Nº 2, 2001.

[3] Evgeny Demidov.
An Interactive Introduction to Splines. 2003.
`http://www.ibiblio.org/e-notes/Splines/Intro.htm.`

[4] P. Poirazi, C. Neocleous, C.S. Pattichis and C.N. Schizas.
Classification Capacity of a Modular Neural Network Implementing Neurally Inspired Architecture and Training Rules.
IEEE Trans on Neural Networks, Vol. 15, Nº 3, 2004.

[5] Fahlman, S. E. 1988.
An empirical study of learning speed in back-propagation networks.
Tech. Rep. CMU-CS-88-162, Carnegie Mellon.

[6] R. Ghosh and B.Verma.
Finding optimal architecture and weights using evolutionary least square based learning.
Proceedings of the 9th International Conference on Neural Infomation Processing (ICONIPÓZ), Vol. 1, 2002.

[7] A. Weigend.
On overfitting and the effective number of hidden units.
In Proceedings of the 1993 Connectionist Models Summer School.

[8] Dilip Sarkar.
Methods to Speed Up Error Back-Propagation Learning Algorithm.
ACM Computing Surveys, Vol 27, No. 4, 1995.

[9] Teck-Sun Tan and Guang-Bin Huang.
Time constrain optimal method to find the minimum architectures for feed-forward neural networks.
Proceedings of the 9th International Conference on Neural Infomation Processing (ICONIPÓZ), Vol. 1, 2002.

[10] Le Cun, Y. 1988.
A theoretical framework for backpropagation.
In Proceedings of the 1988 Connectionist Models Summer School.

[11] F. Flórez, J.M. García, J. García and A. Hernández.
Hand gesture recognition following the dynamics of a topology-preserving network.
U.S.I. Informática Industrial y Redes de Computadores Universidad de Alicante. 2002.