INTRODUCCIÓN AL DESARROLLO DE REDES NEURONALES PERCEPTRÓN MULTICAPA APLICADAS EN TECNOLOGÍA ANDROID

Sergio Andrés Ñustes[1], Jorge Luis Hurtado[2], Giomar Alexandra Bedoya[3], Luis Gabriel Marin[4]

Recibido: 2 de abril del 2013. Aceptado: 2 de mayo de 2013

Resumen

Las redes neuronales son una rama computacional de la Inteligencia Artificial basada en el modelo neuronal de los seres vivos. Actualmente, las redes neuronales son aplicadas a campos como la ingeniería de control, ingeniería del software, telecomunicaciones entre otras. Los resultados de dicha aplicación hoy día han penetrado en los hogares por medio de electrodomésticos como televisores, lavadores, refrigedadores que llevan impresa esta tecnología. Por otro lado los SmartPhone o teléfonos inteligentes son una tecnología en ascenso y parte de las investigaciones actuales en ingeniería se enfocan en cómo implementar la cibernética, el software y las telecomunicaciones en dichas terminales. Este artículo presenta los resultados de la implementación de una red neuronal perceptrón multicapa en un teléfono con tecnología Android, la cual puede reconocer los números del 0 al 9 dibujados en la pantalla táctil.

Abstract

Las redes neuronales son una rama computacional de la Inteligencia Artificial basada en el modelo neuronal de los seres vivos. Actualmente, las redes neuronales son aplicadas a campos como la ingeniería de control, ingeniería del software, telecomunicaciones entre otras. Los resultados de dicha aplicación hoy día han penetrado en los hogares por medio de electrodomésticos como televisores, lavadores, refrigedadores que llevan

[1] Sergio Andrés Ñustes. Estudiante Ingeniería de Sistemas Universidad la Amazonia E-mail: infinito84@gmail.com

[2] Jorge Luis Hurtado. Estudiante Ingeniería de Sistemas Universidad de la Amazonia E-mail: jorgehurtado1991@hotmail.com

[3] Giomar Alexandra Bedoya Estudiante ingeniería de sistemas Universidad de la Amazonia E-mail: gioalex6@gmail.com

[4] Luis Gabriel Marín, Ingeniero electrónico, Especialista en evaluación pedagógica, Candidato a magister en ciencias de la información y las comunicaciones. Universidad de la Amazonia E-mail: lgmarin@gmail.com

impresa esta tecnología. Por otro lado los SmartPhone o teléfonos inteligentes son una tecnología en ascenso y parte de las investigaciones actuales en ingeniería se enfocan en cómo implementar la cibernética, el software y las telecomunicaciones en dichas terminales. Este artículo presenta los resultados de la implementación de una red neuronal perceptrón multicapa en un teléfono con tecnología Android, la cual puede reconocer los números del 0 al 9 dibujados en la pantalla táctil.

Palabras claves: inteligencia artificial, tecnología móvil, reconocimiento de patrones, java, neuroph

Keywords: artificial intelligence, mobile technology, paterns recognition, java, neuroph

Introducción

"Una red neuronal artificial es un procesador distribuido en paralelo de forma masiva que tiene una tendencia natural para almacenar conocimiento de forma experimental y lo hace disponible para su uso" (Aldabas, 2012). Las redes neuronales actualmente son una tecnología en desarrollo, aunque aún no se ha alcanzado su máximo desarrollo han proporcionado una alternativa a la computación clásica. Las redes neuronales no ejecutan una secuencia de operaciones, sino que responden en paralelo a las entradas que se les presenta. (Castro, 2006)

Las redes neuronales se caracterizan por aprender de la experiencia, generalizar casos nuevos a partir de casos anteriores, pueden procesar datos incompletos o distorsionados, y capaces de seguir funcionando a pesar de lesiones. (Castro, 2006).

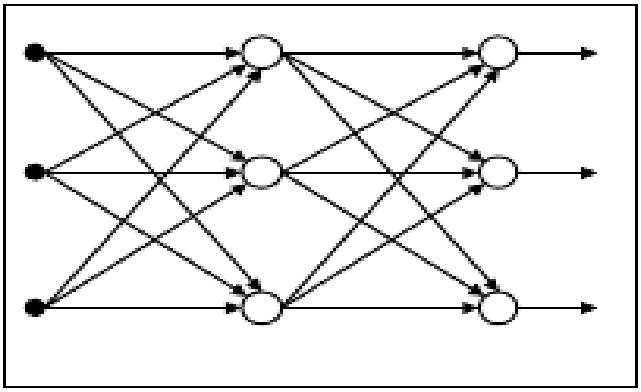


Figura 1: Esquema de una red perceptrón multicapa

En la Figura 1 (Romero y Calonge, 2012), se observa el esquema de una red neuronal perceptrón multicapa, su característica principal es la necesidad de hablar capas de entrada y salida de neuronas, y de capas ocultas que le proveen a la red la capacidad de manejar datos complejos. Adicionalmente, presenta un esquema altamente conexionista debido a que todas las salidas de las neuronas de la capa anterior son las entradas de la capa subsecuente.

El estudio de las redes neuronales implica el tener que hablar de patrones. Romero y Calonge (2012) citando a S. Watanabe[5] definen a un patrón como una entidad a la que se le puede dar un nombre y que está representada por un conjunto de propiedades medidas y relacionadas entre ellas.

Aldabas (2012) implementó en la herramienta computacional Matlab un perceptrón multicapa para reconocimiento de números almacenados en una matrix 7x5. El sistema era capaz de reconocer los números perfectamente y capaz de dar resultados frente a entradas con ruido.

Muñoz et al (2006) realizaron una comparación de un sistema de máquina de vectores con un 97,03% de efectividad frente a una red perceptrón multicapa con un porcentaje de acierto de 94,96%, los cuales realizaban reconocimiento de caracteres alfanuméricos escritos a mano.

[5] S. Watanabe – Pattern Recognition: Human and Mechanical. Wiley, New York 1985

García (2011) define al smartphone como un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono celular común. Algunas tecnologías que incorporan estos teléfonos inteligentes son sensores de compás, orientación, acelerómetro, luz, magnéticos, GPS, así cómo otras tecnologías como Wifi, cámaras, micrófonos, salidas de audio, etres otras. (García, 2011). Es importante mencionar que las tecnologías móviles inteligentes se caracterizan por tener teclado táctil lo cual a popularizado su uso.

La compañía Nielsen es una multinacional dedicada el estudio y estadísticas de consumidores, en agosto-octubre de 2011 realizó un estudio denominado *The mobile media report* donde consigna estadísticas actualizadas acerca del uso de smartphone en dicho año en los países con más auge de esta tecnología.. Entre las conclusiones de dicho reporte se destaca que en el 2009 había una penetración del 18% de la tecnología smarphone respecto a toda la tecnología móvil en estados unidos y esta pasó en el 2011 al 56% con una tasa relativamente mayor del 2% del uso de mujeres respecto a los hombres. Otro dato importante en las estadísticas es la presencia de los sistemas operativos en dichas tecnologías con un dominio del 44.2% de Android respecto a iPhone con 28.6% que años anteriores dominaba el mercado. En este punto, según la información del reporte diversas empresas proveen terminales con tecnología Android, principalmente, HTC, Samsung, Motorola, entre otros, frente a iPhone que sólo permite que móvile Apple tengan su sistema operativo, con la cual se sustenta la penetración de mercados en Android. En dicho estudio se observa que Android destaca en todos los rangos de edad como líder dejando atrás plataformas como iPhone, Blackberry, Windows Phone, entre otros. En el estudio de Nielsen a nivel latinoamericano se destaca el grado de penetración en Argentina con 60% respecto a la tecnología móvil y en Brazil en un crecimiento del 165% de las ventas de esta tecnología (The Nielsen Company, 2011).

En este artículo se presenta la aplicación de una red neuronal perceptrón multicapa para el reconocimiento de los números del 0 al 9 en un teléfono inteligente con tecnología Android.

Materiales y métodos

Para la implementación del sistema de reconocimiento de patrones en el teléfono Android se dividió el problema en varias fases. La primera fase de programación consiste en generar una aplicación Android en el lenguaje Java utilizando el plugin Android para el IDE Eclipse que fuese capaz de dibujar a través de una entrada táctil caracteres en la pantalla del dispositivo. El resultado en esta fase es una aplicación en formato "apk" que se puede utilizar en cualquier dispositivo. Una segunda fase utilizada en esta investigación fue la utilización de una herramienta software que fuese capaz de entrenar nuestra red perceptrón multicapa bajo las entradas generadas por el smartphone, para esto se utilizó Neuroph Studio. Finalmente, se incorporan los pesos generados por Neuroph Studio en nuestra aplicación para su validación. Todas las herramientas utilizadas en esta investigación están bajo licencias libres de uso y en algunos casos *open source* y fueron ejecutadas en el sistema operativo Kubuntu 11,04 con kernel Linux.

Para la implementación del sistema de reconocimietno de números se utilizó una Tablet QBEX con sistema operativo Android 2.3.1 Gingerbread, con una resolución de 800x600 pixeles. A continuación se describe cada una de las etapas utilizadas.

Aplicación Java en Android

El desarrollo de una aplicación Android requiere de varias herramientas para su desarrollo. Android SDK es un paquete de utilidades para desarrollar en Android que trae desde emuladores, jar-clases, documentación y ejemplos para desarrolalr en esta tecnología. Una de las herramientas importantes de este SDK es el "adb" la cual permite hacer una conexión USB o TCP directametne con una terminal física Android, es nuestras pruebas utilizamos conexión USB debido a que la TCP necesita permisos de super usuario y los telefonos móviles por seguridad no traen habilitado el super usuario sin embargo, la opción TCP es válida en PC's con tecnología Android simproblemas. El editor utilizado para la programación en el lenguaje Java fue Eclipse ya que este se integra perfectamente con el SDK de Android a través del plugin que Android provee para él, en la investigación se utilizó la versión Indigo de Eclipse. El uso de estas tecnologías facilitó bastante los tiempos de desarrollo y

compilación pues prácticamente se hacían ajustes en la aplicación, se compilaba y con una demora aproximada de 2 segundos ya se visualizaba en el dispositivo Android.

Para el desarrollo de la aplicación como base se utilizó la clase RNAActivity que hereda de Activity que es el equivalente a JFrame en programación de escritorio en Java. Esta clase se encarga de invocar a una clase Lienzo que hereda de SurfaceView que es el equivalente a la clase Canvas en programación de escritorio en Java. La clase SurfaceView es la que permite que se dibujen primitivas como líneas, óvalos, rectángulos, imágenes entre otras en la pantalla. Además en esta clase es donde se escuchan los eventos producidos por el TouchScreen por medio del método onTouchEvent que recibe como entrada el objeto MotionEvent producido por la acción de pulsar o tocar la pantalla. Cada vez que se hace un evento Touch en la pantalla el sistema toma la posición X y Y de este y lo almacena en una pila de objetos Punto. La clase Punto se encarga de guardar cada posición detectada por el TouchScreen del dispositivo.

En la clase Lienzo se implementa un hilo que constantemente refresca el contenido de la pantalla, dado que el usuario está interactuando con la pantalla, es una buena práctica utilizar este hilo para no producir bloqueos en la aplicación pues el proceso principal interactúa con el usuario y el secundario dibujar haciendo más eficiente a la aplicación. El hilo en Android se implementó de forma tradicional como en una aplicación de escritorio hecha en Java, en la clase Lienzo se implementó la interfaz Runnable y se creó un método “run” para procesarlo, básicamente se realizan dos procesos en el hilo, actualizar contenido mediante el método “update” y mostrar en pantalla con el método “repaint” es necesario aclarar que estos métodos a diferencia de la clase Canvas no vienen por defecfo y en la investigación se hizo una implementación propia a partir de las opciones que proporciona la clase SurfaceView.

Finalmente existen algunas variables de control, las cuales se encargan de alternar el contenido visto en la pantalla inicialmente se muestra lo que se dibuja con el TouchScreen pero dados dos eventos TouchScreen de soltado, es decir que se deja de tocar la pantalla, el sistema los detecta y muestra la matriz asociada a lo que se dibujó junto con la descripción del número que se recibió. En las figuras 2 y 3 se pueden ver

los dos vistas generadas respectivamente.

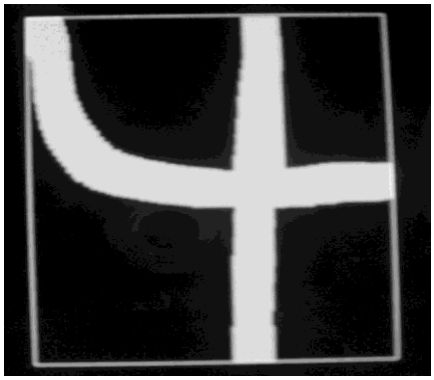
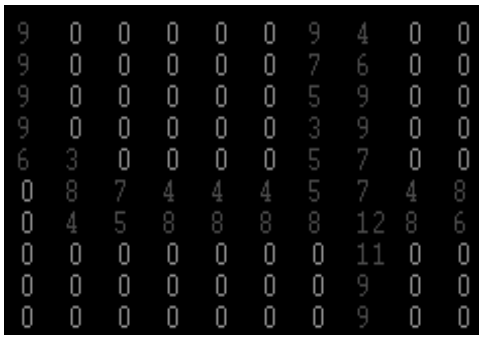


Figura2: Imagen de un patrón dibujado tomado tomado desde el dispositivo.



9	0	0	0	0	0	9	4	0	0
9	0	0	0	0	0	7	6	0	0
9	0	0	0	0	0	5	9	0	0
9	0	0	0	0	0	3	9	0	0
6	3	0	0	0	0	5	7	0	0
0	8	7	4	4	4	5	7	4	8
0	4	5	8	8	8	8	12	8	6
0	0	0	0	0	0	0	11	0	0
0	0	0	0	0	0	0	9	0	0
0	0	0	0	0	0	0	9	0	0

Figura 3: Matriz generada en el smartphone

Para la adquisición de los datos y generar la matriz es necesario antes realizar un proceso que se llama segementación. Con la segementación se busca separar el patrón del ruido y entregarle al clasificador la información necesaria para que identifique el patrón. (Muñoz, 2006). En el caso de la aplicación debido a que la pantalla permite tanto que se han caracteres grandes o pequeños, alrededor de esta como se ve en la figura 2, se va creando un recuadro y es bajo este recuadro que se crea la matriz 10x10 correspondiente que se entrega como entrada de la red para que esta nos genere la salida. La aplicación Android generada está de manera horizontal y por lo tanto los patrones deben dibujarse sólo en sentido vertical.

Diseño y tratamiento de la red

Para el diseño y tratamiento de la red se utilizó una herramienta computacional libre llamada Neuroph Studio basada en el IDE Netbeans la cual de manera visual permite crear la red, capturar los datos de entrenamiento vía formato “txt” y entrenarla según

los parámetros utilizados. La red utilizada utilizó una capa de entrada con 100 neuronas, dos ocultas con 50 y 20 respectivamente y una de salida con 10. Se utilizaron 10 de salida para identificar el grado de pertenencia con el cual clasifica cada uno de los 10 patrones o números a clasificar. (Ver figura 4)

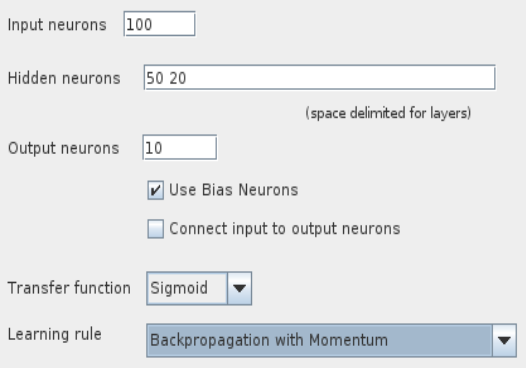


Figura 4: Red creada en Neuroph Studio

Para el entrenamiento Neuroph Studio se tuvo la posibilidad de cargar archivos "txt". Inicialmente debido a que la matriz crear datos con rangos desde 0 a 19 se normalizaron de 0 a 1,sin embargo no daban resultados satisfactores, en una segunda prueba se binarizaron dando un resultado incorrecto. Finalmente, se tomó la tarea de escribir cada matriz como una secuencia de valores en el archivo "txt" se utilizó la herramienta "kate" de Linux para esta tarea. Cada línea del archivo representa una matriz generada en el smartphone. En total el archivo contiene 100 líneas que contienen 10 muestras respectivas de cada patrón del 0 al 9.

En la tabla 1 se denotan los resultados de los respectivos entrenamientos.

No	Neuronas entrada	Neuronas ocultas	Regla aprendizaje	Función de activación	Maximo error	Velocida aprendizaje	Total Error cuadratico	Iteraciones
1	100	50 20	Backpropagation	Sigmoid	0.01	0.2	0,14	94
2	100	50 20	Backpropagation	Tanh	0.01	0,2	0,148	600000
3	100	100 80 20	Backpropagation	Tanh	0.01	0,2	0,152	500000
4	100	50 20	Backpropagation Momentun	Sigmoid	0.01	0,2	0,00095	32
5	100	50 20	Backpropagation Momentun	Tanh	0.01	0.2	-	-
6	100	50 20	Dynamic Backpropagation	Sigmoid	0,01	0,5	0,0015	1339
7	100	50 20	Dynamic Backpropagation	Tanh	0,01	0,5	1,001	400000

Tabla 1: Estadísticos de los entrenamientos realizados en Neuroph Studio

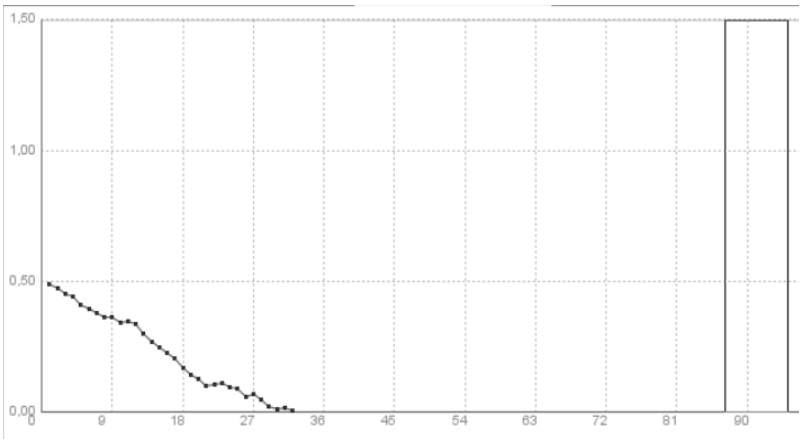


Figura 5: Gráfica de error vs iteraciones

En los datos presentados en la Tabla 1, se concluye que el algoritmo Backpropation con Momentun es el único capaz de converger a una salida deseada según la aplicación planteada. Y también es necesaria la utilización de la función de activación tipo Sigmoide. En la figura 5 se va la correlación entre el error total de la red y las iteraciones en el entranamiento.

Implementación de los pesos en la aplicación

Finalmente, para aplicar el clasificador en la aplicación Android se creó una clase MLPerceptron con arreglos para cada capa oculta y salida y con matrices de pesos para cada oculta y salida, se utilizó una matriz porque cada peso de la capa anterior (arreglo) es la entrada para cada neurona de la capa siguiente (arreglo) por lo tanto se genera dicha matriz. Al crear la clase está carga desde el directorio “assets” del proyecto los archivos “pesos1”, “pesos2” y “pesos_salida” en donde están los respectivos pesos para las capas ocultos y de salida más los bías. El primero archivo tiene 101x50 líneas, el segundo 51x20 líneas y el último 21x10 líneas. Ver figura 6

```
float capa1[]=new float[51];
float capa2[]=new float[21];
float salida[]=new float[10];

float pesos1[][]=new float[50][101];
float pesos2[][]=new float[20][51];
float pesos_salida[][]=new float[10][21];

Context app;

public MLPerceptron(Context contexto){
	app=contexto;
	Matrix("pesos1",pesos1);
	Matrix("pesos2",pesos2);
	Matrix("pesos_salida",pesos_salida);
}

public void Matrix(String file,float matrix[][]){
	InputStream entrada = null;
	try {
		entrada = app.getAssets().open(file);
	} catch (IOException e) {
		e.printStackTrace();
	}
	Scanner sc = new Scanner(entrada);
	for(int i=0;i<matrix.length;i++){
		for(int j=0;j<matrix[i].length;j++){
			matrix[i][j]=Float.parseFloat(sc.nextLine());
		}
	}
}
```

Figura 6: Parte 1 de la clase MLPerceptron.java

Finalmente, después de haber creado la matriz que representa el dibujo en el TouchScreen esta se envía a un función “Test” de la clase MLPerceptron que se encarga de llenar el arreglo “salida” de la misma clase el cual es luego visualizado en la pantalla del teléfono y nos dice bajo la matriz (ahora arreglo) de entrada, que número reconoció. Ver figura 7.

```
public void Sigmoide(float matrix[]){
	for(int i=0;i<matrix.length;i++){
		matrix[i]=(float) (1/(1+Math.exp(-matrix[i])));
	}
}

public void Test(float entrada[]){
	for(int i=0;i<pesos1.length;i++){
		capa1[i]=0;
		for(int j=0;j<pesos1[i].length;j++){
			capa1[i]+=pesos1[i][j]*entrada[j];
		}
	}
	Sigmoide(capa1);
	capa1[50]=1;
	for(int i=0;i<pesos2.length;i++){
		capa2[i]=0;
		for(int j=0;j<pesos2[i].length;j++){
			capa2[i]+=pesos2[i][j]*capa1[j];
		}
	}
	Sigmoide(capa2);
	capa2[20]=1;
	int res=-1;
	float per=0;
	String text="";
	for(int i=0;i<pesos_salida.length;i++){
		salida[i]=0;
		for(int j=0;j<pesos_salida[i].length;j++){
			salida[i]+=pesos_salida[i][j]*capa2[j];
		}
		if(salida[i]>per){
			per=salida[i];
			text+=salida[i]+", ";
			res=i;
		}
	}
	Sigmoide(salida);
}
```

Figura 7: Parte 2 de la clase MLPerceptron.java

Resultados y discusión

Después de implementar el sistema se ejecuta en Eclipse y este nos genera un archivo "Rna.apk" en el directorio "bin" del proyecto e instala la aplicación vía "adb" en la terminal Android, al realizar de nuevo 10 intentos por cada número la aplicación genera resultados de la tabla 2. Finalmente, al validar la aplicación se obtiene que la aplicación tiene una efectividad del 76.53% con un resultado bajo debido a los datos de entrenamiento que se dieron y teniendo en cuenta que los datos de validación se realizaron de manera rápida para esforzar la clasificación de la red. Con un poco más de cuidado e intentando escribir correctamente los números la red clasifica correctamente.

Numero	Intento 1	Intento 2	Intento 3	Intento 4	Intento 5	Intento 6	Intento 7	Intento 8	Intento 9	Intento 10	Total Error	Efectividad
0	0,93	0,95	0,94	0,71	0,83	0,93	0,95	0,95	0,96	0,93	0,092	0,908
1	0,71	0,88	0,58	0,88	0,15	0,4	0,89	0,94	0,32	0,02	0,423	0,577
2	0,78	0,08	0,39	0,14	0,29	0,9	0,95	0,88	0,05	0,95	0,459	0,541
3	0,91	0,07	0,79	0,88	0,79	0,31	0,88	0,89	0,21	0,52	0,375	0,625
4	0,96	0,96	0,98	0,96	0,98	0,97	0,91	0,97	0,67	0,9	0,074	0,926
5	0,39	0,96	0,97	0,98	0,18	0,84	0,32	0,96	0,81	0,98	0,261	0,739
6	0,95	0,93	0,95	0,95	0,88	0,85	0,94	0,88	0,88	0,18	0,161	0,839
7	0,78	0,89	0,21	0,95	0,98	0,97	0,96	0,14	0,65	0,67	0,28	0,72
8	0,93	0,55	0,55	0,92	0,93	0,81	0,94	0,94	0,87	0,94	0,162	0,838
9	0,94	0,91	0,97	0,96	0,97	0,98	0,97	0,97	0,98	0,75	0,06	0,94
											Total	Total
											0,2347	0,7653

Tabla 2: Resultados de implementación

Conclusiones

- El algoritmo Backpropation con Momentun es el único capaz de converger a una salida deseada según la aplicación de red creada mediante la herramienta computacional libre llamada Neuroph Studio basada en el IDE Netbeans. Para estos efectos es también necesaria la utilización de la función de activación tipo Sigmoide.
- La identificación de los números por parte de la aplicación deben escribirse cuidadosamente para que la red la clasifique correctamente.

Bibliografía

Aldaba E. (2012). *Introducción al reconocimiento de patrones mediante redes neuronales*. UPC – Campus Terrassa. Barcelona.

Castro, J. (2006). *Fundamentos para la implementación de red neuronal perceptrón multicapa mediante software. Escuela de Ingeniería Mecáncia y Eléctrica*. Guatemala. Universidad de San Carlos de Guatemala.

Romero, L. y Calonge, T. (2012). *AIRENES*, Capitulo 1: Redes neuronales y reconocimiento de patrones. Universidad de Salamanca y Universidad de Valladolid. España.

Muñoz, P.; Ibargüen, F. y Cardona, J. (2006). *Sistema para el reconocimiento fuera de línea de caracteres manuscritos*. Revista de Investigaciones. Vol 17. Grupo GAMA 5. Universidad del Quindío

García, J. (2011). *El mundo en tus manos*. Smartphones y Tablets. Jornadas TIC para el personal técnico del PAS.

The Nielsen Company. (2011). *The mobile media report. State of the media. Q3 2011*. Estados Unidos.