Algoritmos de aprendizaje neurocomputacionales para su implementación hardware

Tesis Doctoral

D. Francisco Ortega Zamorano

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Junio de 2015

Algoritmos de aprendizaje neurocomputacionales para su implementación hardware

Memoria que presenta para optar al título de Doctor por la Universidad de Málaga

D. Francisco Ortega Zamorano

Dirigida por los Doctores

Dr. Leonardo Franco y Dr. José Manuel Jerez Aragonés

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Junio de 2015

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniera Informática
Universidad de Málaga

El Dr. D. Leonardo Franco, Profesor Titular de Universidad, y el Dr. D. José Manuel Jerez Aragonés, Profesor Titular de Universidad, ambos pertecientes al área de Ciencias de la Computación e Inteligencia Artificial de la E.T.S. Ingeniera Informática de la Universidad de Málaga,

Certifican que,

D. Francisco Ortega Zamorano, Ingeniero en Telecomunicaciones, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

Algoritmos de aprendizaje neurocomputacionales para su implementación hardware

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizamos la presentación de este trabajo en la Universidad de Málaga.

Málaga, Junio de 2015

Fdo.: Dr. Leonardo Franco

Fdo.: Dr. José Manuel Jerez Aragonés

A toda mi familia por confiar en mí.
Sin ellos nada de esto hubiera sido posible.

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a los directores de mi Tesis Doctoral, Leonardo Franco y José Manuel Jerez Aragonés, por su inestimable colaboración tanto en mi tesis doctoral como en mi carrera investigadora, gracias a su ayuda he podido afrontar todos los retos que han ido surgiendo y he podido iniciar una etapa profesional en la que espero seguir contando con su apoyo y amistad.

Quiero hacer una mención especial a Ignacio Molina por confiar en mí y darme la oportunidad de conocer el mundo de la investigación; a Marcelo Montemurro por su acogida durante mi estancia de investigación en Manchester, ofreciéndome la oportunidad de disfrutar de una gran experiencia que espero repercutan en futuros lazos de colaboración y a Paula Monasterio por ofrecerme de forma totalmente desinteresada su ayuda y sus conocimientos en los inicios de mi etapa doctoral.

Me gustaría también expresar mi más profunda gratitud a mis compañeros de laboratorio de Inteligencia Computacional en Biomedicina (ICB): Subi, Dani, Rafa, Esteban, Héctor y Julio; por echarme una mano cada vez que lo he necesitado, por acogerme con la mejor de las sonrisas y por su amistad que espero perdure en el tiempo.

No podía faltar en estos agradecimientos una mención especial para todos mis amigos que han soportado estoicamente mis comentarios e historias sobre mi tesis, Nio, Cristi (ella sabe lo duro que es una tesis), Pablo Tabo, Moni, Pipe, Rake, Nico, Arturo,... y a los que me dejo seguro en el tintero.

Para terminar, agradecer de una forma muy especial a Sandra el apoyo incondicional, con ella todos los agobios y problemas son más fáciles de llevar. A mis padres por su cariño y porque siempre han demostrado que puedo contar con ellos. También a mi Hermana que siempre está en los malos momentos y espero que siempre este en los buenos; y a mis niñas porque su alegría siempre me contagia. Gracias a todos.

Resumen

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso central de los animales. Han tenido una gran evolución desde que en 1943 McCulloch y Walter Pitts introdujeran el concepto de neurona artificial, gracias en gran medida a modelos y algoritmos más complejos publicados con posterioridad como el modelo de Hopfield y el algoritmo Backpropagation.

Hoy en día los sistemas neurocomputacionales se emplean en toda una variedad de aplicaciones en sectores tan importantes como el financiero, medico, energético, industrial, de la robótica o el científico. En la mayoría de estos campos la utilización de algoritmos neurocomputacionales se han ido extendiendo y ampliando en su uso, y en todos ellos van apareciendo nuevas aplicaciones donde la utilización de la programación tradicional sobre ordenadores no puede dar una solución de manera eficiente a un problema dado, ya sea por el elevado tiempo de cómputo en problemas complejos o por su consumo y dimensiones en sistemas empotrados.

Los sistemas en tiempo real y las redes de sensores son dos de las tecnologías más extendidas donde la utilización de modelos neurocomputacionales requiere un desarrollo en dispositivos y el empleo de técnicas de programación diferente a las convencionales. En este tipo de aplicaciones otros dispositivos hardware como las FPGAs (Field Programmable Gate Array) o microcontroladores son más adecuados a la hora de la implementación de redes neuronales artificiales.

Una FPGA es un circuito integrado semiconductor basado en una matriz de bloques de lógica configurables conectados entre sí y, a su vez, con celdas de entrada y salida. Dichas interconexiones (también programables) forman una matriz de enrutado modificable según la funcionalidad necesaria por parte del usuario. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica hasta sistemas combinacionales complejos, siendo utilizadas sobre todo en aplicaciones de sistemas en tiempo real que requieran un alto grado de paralelismo.

Por otro lado, un microcontrolador es un circuito integrado programable capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto por varios bloques funcionales que cumplen una tarea específica, como la unidad de procesamiento, memoria, los puertos de entrada/salida, etc. Por su versatilidad son dispositivo que se encuentran en todo tipo de aplicaciones, desde instrumentos de la vida cotidiana a la más avanzada tecnología aeroespacial.

En ciencias de la computación, los sistemas en tiempo real son aquellos sistemas hardware y software que están sujetos a unas limitaciones temporales dadas por la naturaleza del propio sistema. Controlan o actúan sobre un entorno mediante la recepción de información, el procesamiento de la misma y la devolución de una respuesta con la suficiente rapidez (en un rango de tiempo determinado) como para actuar sobre

el entorno. Las respuestas en tiempo real a menudo son del orden de milisegundos y en ocasiones microsegundos. Por el contrario, un sistema sin restricciones temporales no puede garantizar una respuesta dentro de un período de tiempo prefijado.

La estructura y operatividad de las FPGAs ofrecen la posibilidad de realizar diseños eficientes de sistemas en tiempo real debido a que se pueden implementar funciones complejas para que sean ejecutadas de forma simultánea aprovechando su paralelismo, superando en potencia de cómputo a los procesadores digitales convencionales con paradigma de ejecución secuencial. De esta forma se pueden controlar las señales de entradas y salidas del dispositivo a nivel hardware, consiguiendo unos tiempos de respuesta muy acotados que coinciden con los requerimientos de una aplicación en tiempo real.

Existen multitud de aplicaciones en tiempo real y en muchas de ellas se emplean modelos neurocomputacionales. La FPGAs son dispositivos muy adecuados para implementaciones de algoritmos de redes neuronales ya que el tratamiento de información en este tipo de algoritmos se realiza de forma paralela motivando su utilización, más aún cuando se imponen restricciones temporales que las conviertan en un sistema neurocomputacional en tiempo real.

Existen diferentes estrategias de implementaciones neurocomputacionales en función de si se desarrolla o no el proceso de aprendizaje dentro de la FPGA, llamándose "on-chip" y "off-chip". En aplicaciones donde no se incluyan el proceso de aprendizaje en el propio dispositivo ("off-chip") todo el proceso se realiza generalmente en un ordenador personal, transmitiendo la red resultante a una FPGA que actúa a modo de acelerador hardware en la fase de explotación del modelo. Este tipo de implementación es más fácil de desarrollar ya que se puede reutilizar el software ya diseñado para trabajos previos, por el contrario ofrece una estructura muy rígida que no permite cambiar ni la arquitectura de la red ni modificar los pesos sinápticos de forma ágil y eficiente. Por otro lado, si el proceso de aprendizaje está incluido dentro de la propia FPGA ("on-chip") los recursos hardware utilizados son superiores, pero se obtienen estructuras flexibles que posibilitan hacer modificaciones de forma sencilla. La elección del algoritmo utilizado es determínate en este último tipo de sistemas ya que influirá en el tiempo de aprendizaje y la complejidad de la arquitectura a utilizar.

Por este motivo, una de las finalidades principales de esta tesis es el análisis y desarrollo de implementaciones hardware para sistemas neurocomputacionales en tiempo real con estructura de aprendizaje "on-chip". En esta línea de investigación existen dos posibles alternativas para este tipo de sistemas. Una es realizar implementaciones específicas de algoritmos ya existentes y muy conocidos en los que se realicen modificaciones para simplificar el proceso con el fin de conseguir modelos adaptados a los dispositivos hardware. Si bien los cambios realizados deben ser suficientes para simplificar el proceso, no deben alterar la estructura del algoritmo ya que la ventaja de esta alternativa es el uso de modelos muy estudiados que han demostrado su rendimiento. La otra opción es diseñar implementaciones específicas y eficientes de nuevos algoritmos que se adapten mejor a este tipo de dispositivos. Aunque a priori esta opción pueda parecer la más eficaz, puede resultar poco beneficiosa si el algoritmo presentado no genera un modelo neurocomputacional válido a pesar de obtener una implementación hardware eficiente. Una de las motivaciones principales para la realización de esta tesis es conocer las limitaciones y fortalezas de ambas alternativas de desarrollo y así determinar la opción más viable y productiva en este tipo de implementaciones.

Una de las tecnologías que más ha avanzado en los últimos años ha sido las redes de sensores inalámbricas, debido en mayor medida a los avances tecnológicos registrados

en la última década sobre la capacidad de los microcontroladores usados en este tipo de aplicaciones, así como la reducción en su coste y consumo energético lo que permite la utilización a gran escala de redes de este tipo de dispositivos. En la actualidad muchas aplicaciones precisan dotar de un actuador en los nodos sensores que convierta una señal eléctrica en un movimiento físico para modificar el entorno según la información recibida.

La programación tradicional de sensores/actuadores puede conducir a decisiones incorrectas cuando las condiciones ambientales evolucionan con el tiempo, por lo que es necesario cambiar o adaptar el proceso de toma de decisiones a las nuevas circunstancias. Una motivación de esta tesis es de dotar de inteligencia a las redes de sensores en este tipo de escenarios. Esta inteligencia consistirá en proporcionar cierta información adicional junto a las variables medidas para dar un soporte automatizado a la toma de decisiones y al procesamiento distribuido, aportando una respuesta variable en función del cambio producido.

A pesar del gran avance tecnológico producido en los últimos años en el campo de la microelectrónica, que permite disponer actualmente de microcontroladores con una capacidad de cómputo y memoria muy elevados a precios asequibles, estos recursos siguen siendo una importante limitación para la implementación de algoritmos de aprendizaje sofisticados sobre este tipo de dispositivos. Por ello es necesario diseñar implementaciones de modelos neurocomputacionales eficientes para desarrollar redes de sensores inteligentes.

Por lo tanto esta tesis doctoral tiene como objetivo avanzar en el conocimiento científico y tecnológico necesario para el diseño y desarrollo de modelos neurocomputacionales en dispositivos hardware específicos (microcontroladores y FPGAs), con el fin de permitir su utilización en aplicaciones de sistemas en tiempo real y redes de sensores. Para ello se investigan y desarrollan estrategias de diseño para este tipo de dispositivos que maximicen la eficiencia en la utilización de recursos.

De forma más detallada, esta tesis busca alcanzar los siguientes objetivos parciales: (i) analizar el conocido algoritmo de red neuronal Backpropagation con el fin de implementarlo en los dos dispositivos hardware a considerar, desarrollando y evaluando diferentes técnicas de optimización del algoritmo para cada dispositivo, comparando dichos resultados con una programación tradicional ejecutada en un PC y probar así la mejoría de las técnicas propuestas; (ii) evaluar una alternativa al algoritmo Backpropagation para poder realizar una implementación hardware maximizando el uso de recursos. Se estudia y compara el algoritmo de red neuronal constructivo, C-Mantec, como alternativa aprovechando sus singularidades a la hora de generar arquitecturas de red reducidas; (iii) realizar un estudio exhaustivo de una implementación eficiente hardware del algoritmo C-Mantec para evaluar su posible utilización en aplicaciones de sistemas en tiempo real, siendo los tiempos empleados en el aprendizaje y la explotación las variables críticas para determinar su utilización; y (iv) evaluar una implementación del algoritmo constructivo C-Mantec sobre microcontroladores para su utilización en redes de sensores con aplicación a predicción microclimáticas, alarma de emergencia y sensor de caídas.

Implementación del algoritmo Backpropagation en dispositivos hardware: FPGA y microcontrolador

En el capítulo 2 se expone un estudio detallado de las implementaciones del algoritmo de aprendizaje neurocomputacional Backpropagation para dos dispositivos hardware diferentes como son una FPGA y un microcontrolador, centrándose en la obtención de implementaciones eficientes en términos de uso de recursos y velocidad de cálculo. El algoritmo Backpropagation ha sido ampliamente estudiado en numerosos trabajos, con lo que en primer lugar se ha analizado su diagrama de flujo para determinar los procedimientos a diseñar. En este primer paso se identifica a la función sigmoidea como función de transferencia para la salida de dicho modelo.

El algoritmo se ha implementado en ambos casos utilizando la estrategia entrenamiento/validación/testeo con el fin de evitar el sobreajuste provocado por excesivas iteraciones. Para ello se divide el conjunto de datos a estudiar en tres subconjuntos, uno para realizar el entrenamiento con el que la red modifica sus valores, otro para comprobar la tendencia del error generado por la red neuronal y un último para verificar la capacidad de aprendizaje del modelo resultante. Cada vez que la red realiza una iteración (entrenar el modelo con todos los datos del subconjunto de entrenamiento) se comprueba el error cuadrático medio del subconjunto de validación; si éste es menor al mínimo almacenado por el sistema, se almacena la red que ha obtenido dicho error y el error mínimo se cambia por el obtenido. Por último la red resultante (aquella con el error cuadrático mínimo) se ejecuta con el subconjunto de testeo para comprobar su precisión en generalización.

Para el caso específico de la implementación hardware en la FPGA se ha introducido un nuevo diseño consistente en una neurona (Primera Capa) que combina los elementos de la entrada con las funcionalidades de la primera capa oculta, permitiendo reducir drásticamente el número de recursos utilizados para su implementación. Esto es debido a que los recursos empleados en los elementos de la entrada son absorbidos, con un pequeño incremento de uso de celdas lógicas, por las neuronas de la "Primera Capa" evitando el uso de la memoria y bloques específicos para realizar las operaciones inherentes a los elementos de entrada.

Además, debido a las características intrínsecas del algoritmo se ha introducido un esquema de división en el tiempo para la realización de las multiplicaciones derivadas de la ejecución del algoritmo. Este esquema de procesamiento es viable puesto que en el modelo propuesto las multiplicaciones se hacen de manera secuencial, con lo que sólo es necesario utilizar un bloque multiplicador en cada neurona. Dichos bloques pueden implementarse de dos modos diferentes, mediante combinación de celdas lógicas o mediante el uso de bloques específicos, siendo esta última la más eficiente al usar un DSP (digital signal processor) por cada multiplicador.

Otra estrategia diseñada a la hora de realizar la implementación hardware ha sido emplear una tabla de búsqueda junto a una interpolación para aproximar la función de transferencia. Para ello se utiliza una tabla donde se registran una serie de valores equiespaciados de la función sigmoidea y con el fin de reducir el error producido por el redondeo, se calcula una interpolación lineal de los valores tabulados obteniendo un error absoluto por debajo de 0,001. La interpolación es posible realizarla sin un incremento significativo de recursos ya que se dispone de un bloque multiplicador para realizar las operaciones en cada neurona, el cual está disponible en el momento del cálculo gracias a la multiplexación por división en el tiempo previamente descrita. Esta

estrategia se ha modificado para poder ser utilizada también en la implementación sobre microcontrolador, comprobando que se obtiene una mejora sustancial en la velocidad de operación del algoritmo en dicho dispositivo.

Para ambas implementaciones se ha redefinido la representación del tipo de datos utilizado, pasando de la clásica representación en punto flotante usada en este tipo de modelos a la representación de punto fijo, con la consiguiente reducción en la memoria utilizada para el manejo de las variables y aumento en la velocidad de procesamiento. La reducción de memoria se debe a que con representación entera o de punto fijo (si es necesaria una parte fraccionaria) se puede seleccionar el número de bits a utilizar, mientras que la mayoría de las representaciones de punto flotante se realizan con doble representación, es decir con 64 bits. La reducción de velocidad es intrínseca al tipo de datos, puesto que aunque en la actualidad existen unidades aritmético-lógicas muy potentes para las operaciones en punto flotantes, éstas siguen siendo inferiores a las de números enteros.

En una implementación hardware real la limitación en cuanto al tamaño de las arquitecturas de las redes neuronales que pueden ser simuladas viene determinada por el dispositivo FPGA utilizado (una placa Virtex V -XC5VLX110T- en este trabajo). Estudiando las características de la placa y los recursos necesario para cada neurona se observa que la principal limitación proviene de la cantidad de bloques DSP disponibles. La FPGA mencionada incluye 64 bloques DSP, por lo que el número total de neuronas que se pueden implementar de manera simultánea es 63 ya que es necesario un bloque DSP adicional para realizar el proceso de validación.

Se han realizado pruebas de diseño para una arquitectura determinada con el fin de demostrar la correcta implementación del modelo en una FPGA y un microcontrolador, para ello se ha realizado un estudio de la evolución de los errores cuadráticos medios en el proceso de aprendizaje, validación y testeo de las dos implementaciones junto a una tercera realizada en un PC, observando que los errores son similares para validación y testeo en las tres implementaciones. El uso de una representación con menor precisión afecta al error de aprendizaje (que es menor para la aplicación PC), pero no es relevante respecto a la exactitud de la predicción.

Implementación FPGA del algoritmo constructivo C-Mantec

En el capítulo 3 se analiza la implementación hardware de todo el proceso de aprendizaje ("on-chip") de un nuevo algoritmo de red neuronal constructivo que, a diferencia de los modelos basados en el algoritmo Backpropagation, generan la arquitectura de la red de forma automática mientras se ejecuta el proceso de entrenamiento. Este nuevo algoritmo es el C-Mantec, el cual genera arquitecturas muy compactas y posee una muy buena capacidad de generalización sin necesidad de una capa de salida. Además sustituye la función sigmoidea como función de transferencia de salida de la red por la función mayoría, calculando la activación de la red en función del estado de la mayoría de las neuronas.

Cada neurona dispone de una temperatura específica (T_{fac}) que va descendiendo conforme los datos del conjunto de entrada necesiten ser aprendidos (es decir, la decisión de la red sea incorrecta). Cada vez que un dato de entrada precise ser aprendido la neurona con mayor T_{fac} de las que hayan tomado la decisión incorrecta modifica su peso sináptico con el objetivo de aprender dicho dato de entrada. Cuando la neurona encargada de realizar el proceso tenga un T_{fac} tan pequeño que el cambio de su peso

sináptico sea mínimo entonces la arquitectura añade una neurona y se resetean todas las temperaturas. Una de las principales ventajas de utilizar este nuevo algoritmo es el hecho de que evita el problema de seleccionar la arquitectura adecuada, ya que este proceso se realiza automáticamente de acuerdo con la complejidad de la los datos de entrada, siendo una ventaja adicional su robustez respecto al ajustes de los parámetros.

La implementación de este algoritmo se ha diseñado buscando la reducción en el uso de recursos y simplificando los procesos complejos, prestando especial atención a los procedimientos del cálculo de la función mayoría, del valor de la temperatura específica en cada neurona y de las multiplicaciones del algoritmo.

El cálculo de la función mayoría se realiza comparando la suma de la salida de todas las neuronas con salida activas con respecto a la mitad de las neuronas disponibles en ese momento en la arquitectura. Este proceso puede realizarse en sólo un ciclo de reloj puesto que la división entre dos del número de neuronas activas se puede implementar con un registro de desplazamiento.

El cálculo de la temperatura específica (T_{fac}) involucra dos procedimientos; el primero el cálculo del propio valor del T_{fac}, que a su vez requiere de una función exponencial que se implementa mediante la interpolación de los valores tabulados, procedimiento similar al descrito para la función sigmoidea; el segundo procedimiento es especificar qué neurona con posibilidades de aprender tiene el mayor valor de T_{fac}, para lo que se realizan una serie de comparaciones en grupos de 16 neuronas con el fin de maximizar las comparaciones sin extra dimensionar el comparador.

Las multiplicaciones necesarias en el desarrollo del algoritmo se realizan con un solo bloque multiplicador con estructura de división en el tiempo para cada neurona. Se puede implementar con un solo bloque específico o mediante lógica combinacional. En este caso se optó por la lógica combinacional debido a la posibilidad de exportar dicha implementación a otras placas de diferentes familias y marcas en detrimento del rendimiento en cuanto a velocidad de procesamiento.

Se realizaron diversas pruebas sobre diferentes conjuntos de datos de referencia estudiados en múltiples trabajos, demostrando que la generalización obtenida en la implementación hardware es muy similar a la que se obtiene con una implementación realizada en el lenguaje de programación C sobre PC. Además se observa un claro incremento en la velocidad de cálculo en comparación con las implementaciones estándares realizadas mediante el PC. Para funciones complejas, que requieren una mayor arquitectura de red para su aprendizaje, los tiempos de cómputo en un PC tienen una tasa de crecimiento muy superior a los de la implementación en FPGA, lo que las hace más idóneas para implementaciones de problemas complejos.

Un estudio pormenorizado en la velocidad de cómputo revela que en la implementación en PC el tiempo empleado en añadir otra nueva neurona crece exponencialmente conforme el algoritmo va añadiendo neuronas a la capa oculta , mientras que en una FPGA el tiempo en añadir una nueva neurona crece linealmente; esto es debido principalmente al carácter paralelo de cómputo de la FPGA, haciéndola más eficiente en términos de velocidad cuanto mayor es la complejidad de la arquitectura a simular (más neuronas en la capa oculta).

Redes de sensores inteligentes basados en modelos neurocomputacionales

En el capítulo 4 se introduce un nuevo concepto en la reprogramación de nodos sensores. Las técnicas desarrolladas hasta la actualidad como método para la actualización

de los nodos en una red de sensores requieren generalmente del uso de reprogramación. Si los sensores se encuentran en entornos cambiantes dicha reprogramación puede llegar a ser habitual, incrementando de forma sustancial los costes en términos de tiempo y energía.

Se presenta una alternativa al enfoque tradicional para la reprogramación de nodos sensores/actuadores en entornos cambiantes, basada en un esquema de aprendizaje en el propio sensor ("on-chip") para que de forma automática adapte el comportamiento a las condiciones del entorno. El modelo de aprendizaje propuesto se basa en el C-Mantec, ya mencionado anteriormente, que al generar arquitecturas de tamaño muy compacto lo hacen especialmente adecuado para las implementaciones del microcontrolador al reducir significativamente el uso de memoria.

La placa microcontroladora utilizada ha sido el Arduino UNO, se trata de una placa muy popular de código abierto, económico y eficiente. Las ventajas de utilizar esta familia de placas es diverso pero cabe destacar que al ser de código abierto dispone de documentación y tutoriales muy completos que la hacen fácil de programar, además de presentar una comunidad de usuarios muy extensa que facilita el acceso a nuevos prototipos.

Con el objetivo de realizar la implementación íntegra del algoritmo C-Mantec en la placa Arduino se ha cambiado el paradigma de representación de datos del tradicional punto flotante usado en este tipo de algoritmos a la representación en punto fijo. Una vez realizada la implementación, su correcta ejecución se verifica con una comparación de los resultados obtenidos y los valores teóricos en generalización de una serie de conjuntos de datos de entrada.

El sistema neurocomputacional implementado se ha validado en una red de sensor/actuador en tres casos de estudios con el fin de demostrar la eficiencia y la versatilidad de la aplicación resultante. Los tres casos seleccionados son problemas definidos en entornos cambiantes y por lo tanto la toma de decisiones se debe adaptar acorde a los cambios observados, requiriendo una adaptación del modelo de red neuronal que controla el proceso de decisión.

El primer caso de estudio es una alarma de incendios. Generalmente las alarmas incorporan un proceso de toma de decisiones en función de los valores medidos, si dichos valores pasan un determinado umbral la alarma se activa. Para generar las reglas de decisión se realiza un estudio en un ambiente genérico, pero si las condiciones en la estancia a controlar no son los estándares la alarma sonará. Para no realizar un estudio de cada estancia a medir se ha diseñado un sensor con una red neuronal para tomar la decisión de activación, teniendo en cuenta que si toma una decisión incorrecta el propio sistema modifica su red neuronal para adecuarla a la estancia a sensar.

El segundo caso de estudio es la predicción meteorológica para decidir si activar el sistema de riego de un campo de regadío. Existen multitud de estudios relacionado con las variables climáticas y muchos de ellos utilizan modelos computacionales. El problema surge cuando se quiere realizar una acción sin tener en cuenta las variables microclimáticas de la zona. En este caso los estudios generalistas pueden no resultar adecuados y los estudios específicos pueden resultar muy costosos, por lo que se ha decidido la implementación de un modelo de red neuronal en el propio sensor para que capte las variables climáticas con el fin de modificar la red conforme se tengan datos de la zona sensada. Con el fin de maximizar la memoria disponible se ha discretizado las variables sensadas debido a que almacenar los datos reales es muy costoso en cuanto a memoria utilizada.

Por último se ha estudiado un sistema de detección de caídas en personas mayores. El sensor viene equipado con un acelerómetro que indica la posición en el eje x, y, z del individuo. El conjunto de datos del problema viene definido por medidas de una serie de caídas simuladas y otras medidas de acciones habituales que no son caídas. La dificultad surge cuando el sistema de caídas se coloca en diferentes tipos de personas con diferente morfología, comportamiento y hábitos de vida, en ese momento el sistema puede conducir a decisiones incorrectas. Si los sensores van equipados con un modelo neurocomputacional adaptarán su toma de decisiones en función de las variables anteriormente mencionadas dando lugar a un sensor específico para cada usuario.

Conclusiones y líneas de trabajos futuros

En conclusión, se puede decir que el algoritmo Backpropagation se implementó correctamente en dos dispositivos hardware, una FPGA y un microcontrolador, realizándose dichas implementaciones bajo un paradigma de aprendizaje que incluye un sistema de validación para evitar el sobreajuste.

La implementación FPGA precisó de diferentes actuaciones sobre la estructura del algoritmo para minimizar los recursos utilizados. Las modificaciones diseñadas, que se centran en Introducir el paradigma de una nueva neurona de la "Primera Capa", incorporar la multiplexación por división en el tiempo del bloque multiplicador, así como tabular e interpolar los valores de la función sigmoidea, han permitido un reducción de recursos, en media, de un 25,8 % de celdas lógicas y un 50'3 % de bloques específicos en una serie de arquitecturas de diferente tamaño en comparación con implementaciones hardware tradicionales sin mejoras.

Al analizar las limitaciones impuestas por la cantidad de recursos disponibles, se observa que la limitación principal es el número de bloques específicos DSP de la placa FPGA, limitando el número de neuronas disponibles puesto que cada neurona precisara de un bloque específico DSP. Al comparar los resultados con las publicaciones de trabajos previos se observa una reducción en términos de recursos de bloques específicos y celdas lógicas, lo que permite un incremento sustancial en el número máximo de neuronas que dispondría una arquitectura de red neuronal. Sin embargo, el aumento de dicho tamaño, para casos particulares, dependerá de la combinación de los valores de la arquitectura de red neural y recursos de la placa FPGA a utilizar.

En el caso específico del microcontrolador la modificación del tipo de representación de los datos permite un incremento en la velocidad de cómputo de entre 8 a 18 veces más rápido, además de una reducción importante en la cantidad de memoria utilizada. Estos resultados avalan el empleo de sistemas neurocompuacionales en el propio microcontrolador con estructura de aprendizaje "on-chip".

El estudio del algoritmo Backpropagation muestra que las implementaciones específicas de dispositivos hardware diferentes al PC deben diseñarse con modificaciones que las hagan más adecuadas a cada dispositivo en los que se desarrolla. De esta forma se consigue maximizar los recursos disponibles y reducir drásticamente los tiempos de cómputo en los dispositivos FPGAs y microcontroladores.

La implementación íntegra (con estructura de aprendizaje "on-chip") del algoritmo C-Mantec ha sido realizada de forma específica para su implantación en una placa FPGA y la eficacia del diseño se ha demostrado mediante una comparación frente a los valores teóricos de los resultados obtenidos de la generalización de una serie de conjunto de datos conocidos.

Además, haciendo un análisis del tiempo de cómputo de cada conjunto de datos se observa una disminución del tiempo empleado en las implementaciones sobre FPGA en relación a las realizadas sobre PC, señalando que dicho aumento de velocidad es proporcional al tamaño de la arquitectura de la red. Esto es debido a que el tiempo de cómputo en un ordenador crece de forma exponencial conforme se añaden neuronas a la capa oculta y de manera lineal en una FPGA, dando lugar a implementaciones hasta 47 veces más rápidas en arquitecturas más complejas.

Las pruebas anteriormente mencionadas muestran que la representación de datos utilizada (representación en punto fijo) es suficiente para lograr resultados comparables en cuanto generalización y tamaño de la arquitectura a los ejecutados en un PC con representación en punto flotante, siendo este algoritmo muy robusto en cuanto al tamaño de palabra utilizado en la representación de los datos.

La implementación hardware del algoritmo C-Mantec es un 15 % más eficientes en recursos hardware utilizados en comparación a la del algoritmo Backpropagation. El análisis de los resultados confirma que las arquitecturas implementadas en una misma placa FPGA del algoritmo C-Mantec permiten mayor número de neuronas que las realizadas para el algoritmo Backpropagation. Además, las ejecuciones del algoritmo C-Mantec precisan de menor tamaño de representación de datos para que sean precisas, por lo que se constata que dicha implementación es menos sensible a la longitud de palabra utilizada, lo que permite un ahorro de recursos al reducir la complejidad de la implementación.

Para finalizar es importante mencionar que el tiempo de aprendizaje de las implementaciones FPGAs en ambos algoritmos es notablemente menor que el tiempo empleado por un PC, siendo la del C-Mantec sustancialmente inferior al del Backpropagation. Además el tiempo de ejecución del modelo es considerablemente inferior para el C-Mantec, lo que supone una ventaja en la fase de explotación del modelo. A la luz de los resultados observados se puede concluir que el presente análisis demuestra la idoneidad del algoritmo C-Mantec para su aplicación en problemas industriales del mundo real en las que se precise de modelos neurocomputacionales en tiempo real.

El algoritmo C-Mantec se ha implementado en la placa Arduino, para lo que se ha modificado el paradigma de representación de datos reduciendo considerablemente la memoria utilizada para el almacenamiento de variables y aumentando el número de neuronas máximas disponible por la arquitectura. Además, un estudio de la velocidad de procesamiento revela que las implementaciones del mismo algoritmo en punto fijo (8 bits en parte decimal) son hasta cinco veces más rápidas que las de punto flotante para las arquitecturas más grandes, debido principalmente a que el tamaño de la arquitectura repercute negativamente de forma severa en la velocidad de cómputo para las representaciones de punto flotante ya que el manejo de memoria y el uso de una unidad aritmético lógica en este tipo de representación son más complejos.

La correcta implementación del algoritmo ha sido verificada con la comparación de los resultados obtenidos para una serie de conjuntos de datos. Se ha comparado la generalización y tamaño de las arquitecturas con los valores teóricos para dichos conjuntos, obteniendo como resultado que el tamaño de las arquitecturas en el microcontrolador es ligeramente superior a las teóricas conforme crece el número de entradas del conjunto de datos a entrenar, observándose también una pequeña reducción de la precisión. Si bien se aprecia una degradación del modelo debido principalmente a los efectos producidos por el redondeo de la representación del tipo de datos, esto no condiciona la eficiencia del sistema resultante ya que la generalización se mantiene en unos valores

cercanos a los óptimos.

El algoritmo implementado se ha empleado como una red de sensor/actuador en tres casos de estudios con el fin de demostrar la eficiencia y la versatilidad de la aplicación resultante. Los tres casos de estudios seleccionados son problemas definidos en entornos cambiantes, y por lo tanto la toma de decisiones del sensor/actuador ha de adaptarse en consecuencia a los cambios observados, por lo que requieren una reconversión del modelo de red neuronal que controla el proceso de decisión. Los tiempos de reprogramación observados son significativamente bajos en los tres casos de estudio, siendo en consecuencia el consumo de energía del dispositivo también bastante pequeño. Incluso sin una comparación exhaustiva con el caso tradicional en el que el nuevo código tiene que ser transmitido desde una unidad de control central, los resultados hacen evidente una reducción en el gasto energético, cualidad muy importante en este tipo de tecnología debida a la corta duración de las baterías que lo alimentan.

Como resultado, se ha demostrado la idoneidad del algoritmo C-Mantec para su aplicación en una tarea compleja utilizando un microcontrolador Arduino UNO. Hoy en día, dada la existencia de dispositivos con mucho más poder de cómputo y recursos que la placa considerada, el presente estudio permite confirmar la potencial aplicación del algoritmo propuesto en tareas reales que necesitan sensores/actuadores.

Finalmente, como conclusión global se puede decir que esta tesis doctoral profundiza en el estudio de implementaciones neurocompuacionales para dispositivos diferentes a los tradicionales (FPGA y microcontroladores) en tecnologías y aplicaciones que así lo requieran. Además demuestra las ventajas potenciales de las FPGAs como dispositivos usados a modo de aceleradores hardware para aplicaciones de neurocomputación dada sus capacidades intrínsecamente paralelas, mientras que en relación con el uso de las redes neuronales en microcontroladores destacamos la estructura de aprendizaje "on-chip" que permiten su uso en sensores remotos utilizando un modo de funcionamiento autónomo. Después de un análisis exhaustivo se puede determinar que las implementaciones de estos dispositivos deben diseñarse de acuerdo a sus particularidades de composición y configuración, consiguiendo unos resultados en términos de eficiencia de recursos hardware y velocidad de cómputo muy superiores a los modelos neurocomputacionales desarrollados en un PC. Además, se ratifica que existen otros modelos de red neuronal más adecuados a los tradicionales, con los que se consiguen arquitecturas superiores en tamaño y tiempo de respuesta inferiores.

Las futuras líneas de investigación que puede seguir el estudio realizado en esta tesis es muy variado, alguno de los iniciados son evolucionar la implementación hardware del algoritmo Backpropagation para disponer de arquitecturas más grandes gracias a la reutilización de una sola capa que permite simular diferentes capas ocultas de la arquitectura de red; o analizar la posibilidad de emplear las FPGAs como aceleradoras hardware para simulaciones de sistemas complejos con una necesidad de cómputo elevado como el modelo Ising, usado para estudiar el comportamiento de los materiales ferromagnéticos. Además existen otras posibilidades de futuros trabajos que necesitan ser explorados como son estudiar otros modelos computacionales con otras reglas de decisión para su uso en redes de sensores inteligentes con el fin de ahorrar recursos al no necesitar guardar una cantidad ingente de datos; o analizar la posibilidad de aplicar los sistemas neurocomputacionales en tiempo real en tareas complejas como un sistema de estabilización de un cuadricóptero.

Abstract

Artificial neural networks are a paradigm of learning inspired by the automatic processing and functioning of the central nervous system of animals. They have had a great evolution since 1943 when Warren McCulloch and Walter Pitts introduced the concept of artificial neuron, thanks largely to the introduction of more complex models and algorithms such as the Hopfield model and the Backpropagation algorithm in the 80s. These two models produced a neural network renaissance, and nowadays neurocomputational systems are used in a variety of applications in key sectors such as finance, medical, energy, industrial, robotics or science. In most of these fields the use of neurocomputational algorithms have been extended and expanded, and in almost all of them new applications are appearing where the use of traditional programming on computers cannot provide an efficient solution, partly due to the high computation needs for complex problems or factors like energy consumption and dimensions for the case of the implementation of models in embedded systems.

Real-time systems and sensor networks are two of the most widespread technologies in which neurocomputational applications may require the use of alternative devices with different programming techniques. In these kinds of applications hardware devices such as FPGA (Field Programmable Gate Array) or microcontrollers might be more adequate for the implementation of artificial neural networks.

FPGAs are semiconductor integrated circuits based on an array of configurable logic blocks interconnected between them and with input/output signals. The programmable interconnections generate a routing matrix, modifiable by the user according to the needed functionality. They can be programmed to generate a broad range of processes from very simple function, like those carried out by a logic gate to very complex combinational systems like functions needed for the operation of real time systems.

On the other hand, a microcontroller is an integrated programmable circuit able to execute orders etched into its memory. It consists of several functional blocks that execute a specific task as the processing unit, memory, input/output, etc. Due to its versatility, microcontrollers are found in several applications, ranging from everyday life devices to the most advanced aerospace technology.

In computer science, real-time systems are hardware and software systems subject to tight time restrictions determined by the nature of the application. They control or act according to environmental conditions by receiving information, processing it and returning a response quickly enough. Real-time responses are often on the order of milliseconds, even microseconds sometimes. By contrast, a non-real time system is not constrained to respond within a predetermined time period.

The structure and effectiveness of FPGAs offer the possibility for the efficient design of the real-time systems because they can implement complex functions that can be executed simultaneously by taking advantage of their intrinsic parallelism, having also

a computing power similar or higher than conventional sequential operated digital processors. FPGAs permit inputs and outputs to be controlled at hardware level, and can provide faster response times and specialized functionality that meet the requirements of real-time applications.

FPGAs are devices well suited for neurocomputational algorithms due to the parallel processing of information in such algorithms, motivating their use especially when time constraints are imposed for the case of real-time systems.

A classification of neurocomputational implementations can be performed depending on whether the learning process is developed within the FPGA ("on-chip") or not ("off-chip"). In applications where the learning process is not included in the device ("off-chip") this process is generally performed in a personal computer, to then transmit the resultant neural network to the FPGA which acts as a hardware accelerator in the execution phase of the model. This kind of implementation is easier to develop since only the final architecture is codified into the FPGA as training is done externally, usually in a PC with the possibility of reusing existing software. On the contrary, this scheme offers a very rigid structure that does not allow for the modification of the architecture and/or synaptic weights in a quick and efficient manner. The other situation in which the learning process is included within the FPGA ("on-chip" learning) larger hardware resource usage is needed, but this scheme offers more flexibility as several algorithm features can be easily modified on-line. Furthermore, for this last scheme, the choice of training algorithm is a very relevant feature as it strongly influences the FPGA code implementation, learning times and final model architecture, all critical factors in terms of the time and effort required for the problem implementation and the final execution model.

For the previously mentioned reasons, one of the main purposes of this thesis is the analysis and development of hardware implementations for real-time neurocomputational systems with an "on-chip" learning scheme. In this line of research two main alternatives are possible: a) The first one is to develop specific implementations of well-known existent algorithms introducing modifications to simplify the process in order to get the models adapted to the hardware devices; the changes should be sufficient to simplify the process although should not alter the structure of the algorithm since the advantage of this alternative is the use of well-studied models that have demonstrated their performance. b) The second option is to design implementations of new algorithms better suited to the devices in order to achieve efficient implementations. This option seems the most effective if the proposed algorithm does generate a valid neurocomputational model. One of the motivations of this thesis is to analyze the limitations and strengths of both alternatives to determine the most viable and productive option.

Apart from FPGAs, wireless sensor networks are one of the technologies that has advanced more in recent years due to the technological advances on the microcontrollers used in these kind of applications, together with a drastic reduction both in cost and power consumption, that allow for the use of large-scale networks. The traditional programming way of sensor/actuator systems can lead to wrong decisions when conditions change over time, so in several situations it is necessary to change or adapt the decision making process to new circumstances. Another motivation of this thesis is to include intelligence into sensor networks in changing environments, in order to use some additional information along with the measured variables to give an automated support decision-making and distributed processing, providing a smart environmental

dependent response according to the sensed information.

In spite of the great technological advances in recent years in the field of microelectronics that permitted to construct microcontrollers with high computing and memory capacities at very affordable prices, microcontrollers resources still remain a major constraint to implementations of sophisticated learning algorithms. Therefore it is necessary to design efficient neurocomputational implementations models to develop smart sensor networks.

As an overall goal, this thesis aims to advance in the scientific and technological knowledge necessary for the design and development of neurocomputational models for specific hardware devices (microcontrollers and FPGAs) to allow their use in real-time and sensor network applications. In more detail, this thesis seeks to achieve the following partial objectives: (i) analyzing the well-known neural network algorithm, Backpropagation, for its implementation in specific hardware devices, developing and evaluating different techniques of optimization in order to compare its performance with its standard PC version, and thus prove the improvement of the proposed techniques; (ii) evaluating alternatives to the Backpropagation algorithm to perform a hardware implementation in which device resources are maximized. The constructive neural network algorithm, C-Mantec is analyzed as an alternative since it produces very compact architectures; (iii) developing a comprehensive study of an efficient hardware implementation of the algorithm C-Mantec to assess their possible use in systems applications in real time, being the computational time in the learning and the execution processes the critical variables to determine the use; (iv) evaluating an implementation of C-Mantec constructive algorithm in microcontrollers to use as neural model in smart sensor networks by checking in real applications as microclimate predictions, emergency alarms and elderly people fall detection.

Implementation of the Backpropagation algorithm in hardware devices: FPGA and microcontroller

In Chapter 2 a detailed study of implementations of neural computational learning algorithm Backpropagation is carried out in two different hardware devices such as FPGA and a microcontroller, focusing on obtaining efficient implementations in terms of resource utilization and computing speed. The Backpropagation algorithm has been extensively studied in several works, so the first step is to analyze the flowchart to determine procedures which should be designed. In this step the sigmoid function is identified as the transfer function for the output of the model.

The algorithm has been implemented in the two cases using the training/validation/testing strategy to avoid overfitting caused by excessive training iterations; in order to obtain this scheme the dataset is divided into three subsets, a first to perform the training to modify the synaptic weight values of the model, another to check the evolution of the error, and the last one to verify the predictive capacity of the resulting network. Whenever the network carry out an iteration (training the model with all training data set) the mean squared error of the validation data set is checked; if it is less than the minimum stored by the system, the network which generates this error is saved and the stored minimum error is changing by the obtained one. Finally, the obtained network (the one with the minimum mean squared error) runs with the test data set to check the quality of generalization.

For the specific case of hardware implementation in FPGA a new design of a neuron

(First Layer) has been introduced that it combines the elements of the inputs with the functionalities of the first hidden layer, allowing a drastic reduction on the resources used for the implementation. In this new scheme, the resources used for the inputs are incorporated in the neurons of the "First Layer" with a small increase in use of logic cells, without needing the use of memory and specific blocks to perform the inherent operations for input elements.

Furthermore, due to the intrinsic characteristics of the algorithm a scheme of time-division multiplexing to perform the multiplications involved in the process is designed This processing scheme is feasible since the proposed model calculates sequentially the multiplications, so it is just necessary to use a single multiplier block for every neuron. These blocks can be implemented in two different ways, by combining logic cells or using specific blocks, the latter being the more efficient when using a DSP (digital signal processor) in each multiplier.

Another strategy designed when the hardware implementation is performed has been to develop a lookup table plus a linear interpolation to approximate the transfer function. For this purpose a table where several equidistant values are stored of the sigmoid function and in order to reduce the error produced by rounding of the values a linear interpolation of the tabulated values is calculated, obtaining an absolute error below 0.001. The interpolation is possible to carry out without a significant increase in resources since a multiplier block for operations in each neuron is available at the time of calculation due to described time-division multiplexing. This strategy has been modified to be also used in the implementation of the microcontroller, checking that a substantial improvement of the computing speed is obtained in this device.

For both implementations, the data type representation has been modified from the classical floating point representation used in these type of models to fixed-point representation, in order to reduce the used memory for the management of variables and for increasing the computation speed. Memory reduction is achieved when using an integer representation (also called fixed point representation when a fractional part is used) because the number of bits used for representing the values can be optimized by the programmer while in general for the floating point cases the double float representation is employed, that needs 64 bits. Speed reduction is intrinsic to data type, and even if powerful arithmetic-logic units for operations in floating point representation have been developed, their improvements are still inferior to using integer numbers.

In a real hardware implementation the limitation in the size of the neural network architectures which can be simulated is determined by the used FPGA board. In our case, a Virtex V (XC5VLX110T) board has been employed, and by analyzing the characteristics of this device and the resources necessary for implementing a single neuron, it is observed that the main constraint for the current implementation is the number of available DSP blocks used to compute products related to a neuron output. The employed FPGA includes 64 DSP blocks so the total number of neurons that can be simultaneously implemented is 63 because one DSP block is required to perform the validation process.

In order to demonstrate the correct implementation of the Backpropagation model both in the FPGA board and Arduino microcontroller several tests have been carried out, in particular by analyzing the evolution of the mean squared error for training, validation and test pattern sets. A comparison to the standard implementation of the algorithm on a PC is also carried out, checking that the level of error is similar for all considered cases. The use of a fixed point representation for the implemented models in

the FPGA and microcontrollers may produce rounding effects larger than in the case of the PC application, but the results demonstrate that the representation used is long enough as the accuracy of the implemented models is not much affected.

FPGA Implementation of the constructive algorithm C-Mantec

In chapter 3 the FPGA implementation of a new constructive neural network algorithm named C-Mantec is analyzed. Unlike models based on the Backpropagation algorithm, C-Mantec generates the network architecture automatically while the training process is executed. Furthermore, this new algorithm generates very compact architectures with a single hidden layer of threshold neurons that permit to obtain very good predictive capabilities. Another difference with the Backpropagation based architectures is that C-Mantec replaces the sigmoid function as output transfer function of the network by a majority function.

Every neuron has a specific temperature (T_{fac}) which will decrease as the input data set is learned. Whenever an input pattern is misclassified and requires to be learned the neuron with the largest T_{fac} among those wrongly computing its output, modifies its synaptic weights in order to learn the data. When the neuron in charge of carrying out the learning process has a T_{fac} too small so that the change of the synaptic weights of the architecture is almost negligible then the system adds one neuron to the network and all specific temperatures are reset. One of the main advantages of using this new algorithm is the fact that it avoids the problem of selecting the right architecture, as this process is performed automatically according to the complexity of the input data, with a further advantage regarding its robustness the parameter settings.

The implementation has been designed aiming at reducing the use of resources and simplifying the whole process, with particular attention to the procedures of calculating the majority function, set of values of the specific temperature of every neuron and the multiplications necessary for the execution of the algorithm.

The computation of the majority function is performed by comparing the number of active neurons in the hidden layer with the total number of neurons present in this layer divided by two, noting that this process can be carried out in only one clock cycle since the division by two of the total number of neurons can be implemented using a shift register.

The calculation of the internal temperature of every neuron (T_{fac}) involves two procedures; the first for computing the value of T_{fac} itself that in turn involves an exponential function which is implemented by using tabulated values plus a linear interpolation scheme, a similar method as described previously for the sigmoid function. The second procedure is dedicated to the selection of the neuron with the largest T_{fac} value (the candidate neuron that will try to learn the current input pattern), and for this task a series of comparisons is implemented in groups of 16 neurons, in order to optimize resource usage.

The implementation of the C-Mantec algorithm involves several multiplications that are performed in a single block using a time-division multiplexing scheme for every neuron. Each of these products can be implemented with one specific block or by using combinational logic. This last option has been selected in this case due to the possibility of exporting the implementations to other different boards.

Several tests on different benchmark data sets previously analyzed in several works have been performed, demonstrating that the obtained generalization using the new

C-Mantec hardware implementation is quite similar to the one obtained from a implementation performed with traditional programming by language C in an PC, noting that also a significant increment of computing speed is obtained in comparison to the traditional PC implementation. In particular, the computation of complex functions requires large network architectures for which longer computational times are required, and thus FPGA implementations are suitable for these cases.

A detailed study on the computation speed reveals that the rate at which a new neuron is added to the network grows exponentially in the PC implementation, while for the FPGA this time grows linearly, confirming the advantage of the parallel nature of the FPGA, being more efficient in terms of speed when the architecture is more complex (more neurons are present in the hidden layer).

Smart sensor networks based on neurocomputational models

Chapter 4 introduces a new concept in relationship to sensor nodes reprogramming. Standard techniques for updating nodes in a sensor network require the reprogramming of the nodes software, and thus if sensors are located in changing environments such reprogramming might be done frequently, increasing substantially the involved costs in terms of time and energy.

This chapter presents an alternative to the traditional approach for sensor/actuator node reprogramming in changing environments based on a scheme of training the sensor node itself to adapt automatically its behavior to the environment conditions. The proposed learning model is based on neural network constructed using the C-Mantec algorithm described in the previous chapter, which generates very compact architectures that makes the approach particularly suitable for microcontroller implementations as memory resources are limited in these type of devices. The microcontroller board chosen has been the Arduino UNO that is a very popular open source computer hardware, economic and efficient. Among the advantages of using this family of boards, the fact that it is developed under an open source format means that very complete documentation and tutorials are easily available as there is a large user community that facilitates the access to new prototypes.

In order to develop the whole implementation of the C-Mantec algorithm into the Arduino board the traditional data type representation has been changed from the standard floating point one used in this type of algorithms to a fixed-point representation that helps to optimize resource utilization. In order to check the correct implementation of the algorithm several checks have been carried out verifying that the results obtained for the generalization ability on benchmark data sets are the same as those obtained with the PC version of the algorithm.

The implemented neurocomputational system has been used as a smart sensors/actuators network in three case studies to demonstrate the efficiency and versatility of the resultant application. The chosen cases correspond to three problems set in changing environments, and therefore a decision-making approach should be adopted to deal with the observed changes, requiring a re-training of the neural network model that controls the decision process.

The first case study is a fire alarm. Usually alarm systems incorporate a decision process that depends on the measured values such that if these values exceed a set threshold value then the alarm is activated. An analysis can be carried out in a generic environment to generate decision rules, although the alarm could be activated when

conditions in the space to control are not the standards. In order to avoid analyzing every possible case, the system modifies its neural based decision model when a wrong choice has been taken, adapting it to the sensed conditions.

The second case study is the problem of weather forecasting for deciding whether to activate or not the irrigation system of a given field. Several studies about climatic prediction have been performed based on neural network models, noting that many complications arise from microclimatic conditions when they are not properly considered. In the present case general studies may not be appropriate, and specific studies can be very expensive. A neural network model within the sensor itself has been implemented in order to sense the climatic variables and modify the network according to the measured data of a specific area. As storing the data set in real values format is very costly, the measurements have been discretized into categories in order to reduce the amount of information to be saved.

Finally, a fall detection system for elderly or disabled people has been analyzed. In this case, the sensor is equipped with an accelerometer which indicates the position of a person using spatial x, y, z coordinates. Several measurements are obtained for simulated falls and for normal situations, to obtain a training/test data set for the problem. As the system might be used by different kinds of people in different environments where their behavior and lifestyles of the user are also different, the system can lead to wrong decisions, so to avoid possible malfunctioning, the sensor will train the neural network based decision making system according to the aforementioned variables in order to obtain a specific adjustment for each case.

Conclusions and future lines of research

In conclusion, it can be said the Backpropagation algorithm has been implemented successfully into two different hardware devices, a FPGA and a microcontroller; these implementations have been developed from a learning scheme with validation process to avoid overfitting problems.

The FPGA implementation requires several changes in the structure of the algorithm in order to minimize the employed resources. Introducing a new paradigm for neurons " First Layer ", incorporating time-division multiplexing of the multiplier block as well as tabular and interpolate the values of the sigmoid function. These changes have allowed an average reduction of resources 25.8 % of logic cells and 50.3 % of specific blocks in a series of architectures of different size in comparison to previous hardware implementations without upgrades.

In the moment to analyze the constraints imposed by the available resources, it can be observed that the main limitation is the number of specific DSP blocks of the FPGA board, therefore DSP blocks restrict the number of available neurons for the architecture as a DSP block is required in each neuron and another for the validation process. Comparing these results with previous published works, a reduction in terms of resources specific blocks and logic cells is observed, allowing a substantial increase in the maximum number of neurons that the architecture of a neural network can manage. However the increase of the size of the architecture, in particular cases, depends on the combination of the neural network values and FPGA board resources used.

In the specific case of the microcontroller, introducing change in data type representation allows an increase in the computing speed of between 8 and 18 times faster, and a reduction in the amount of used memory. These results permit carry out neu-

rocomputational models in the own microcontroller with scheme of "on-chip" learning also in this device.

The study of Backpropagation algorithm shows that the specific implementations in different device to PC should be performed with changes to adapt it to each device in which they are carried out. These methods maximise available resources and dramatically reduce computation times, being shown for FPGAs and microcontrollers.

The full implementation (with "on-chip" learning scheme) of the C-Mantec algorithm has been specifically performed for developing in an FPGA board, the precise design has been demonstrated by a comparison against theoretical values of the generalization of a series of well-known data sets.

In addition a reduction of spent time to learning process in FPGA in relation to PC is observed when a detailed analysis of computation time of each data set is carried out; noting that the speed increase is proportional to the size of the network architecture. This is due to the fact that computation time in a computer grows exponentially when neurons are added to the hidden layer and linearly in an FPGA, resulting implementations up to 47 times faster for more complex architectures.

The aforementioned tests show that the used data type representation (fixed-point representation) is sufficient to obtain similar results in terms of generalization and size of the architecture in comparison to PC implementations with floating point representation, being this algorithm very robust as regards the word size used in the data type representation.

The hardware implementation of the C-Mantec algorithm is 15 % more efficient in terms of the used hardware resources compared to the Backpropagation algorithm. By analyzing these results it can be confirmed that the generated architectures in a FPGA board of the C-Mantec algorithm allow greater number of neurons than for the Backpropagation algorithm.

In addition, the C-Mantec algorithm executions require less word length in data representation for being precise, so it can be verified that such implementation is less sensitive to the used word length, allowing resource savings since the complexity is reduced in the implementation.

To complete the comparison it can be demonstrated that the learning time is substantially smaller in both implementations compared to the process in a PC, noting that this time is notably smaller in the C-Mantec implementation than the Backpropagation. In addition, runtime of the C-Mantec model is considerably lower which is an advantage in the execution phase of the neural network. In light of the observed results it can be concluded that this analysis demonstrates the suitability of the C-Mantec algorithm for application in real-world industrial problems in which real-time neurocomputational models is required.

The C-Mantec algorithm has been implemented in the Arduino, for this the paradigm of data type representation has been changed, greatly reducing the used memory for storing variables as a consequence the maximum number of neurons available for the architecture of the network increases. In addition a study of computing speed reveals that for the same data set, the fixed point implementations (8-bit decimal part) are up to five times faster than floating point for the larger architecture, mainly due to the size of the architecture affects the computational speed in a negative way for floating point representation since memory management and the use of an arithmetic logic unit in this kind of representation are more complex.

Successful implementation of the algorithm is verified by comparing the obtained results with those from previous studies using several benchmark data sets. The generalization accuracy and size of the obtained architectures have been analyzed, observing that the microcontroller architectures are slightly larger than the theoretical values, also noting a small reduction in training accuracy. These two observed changes can be justified as a consequence of rounding effects generated by the data type representation used, noting that the differences are always small and generalization values are quite close to the optimal ones previously obtained using a real data type representation.

The implemented algorithm has been used as a sensors / actuators network in three case studies to demonstrate the efficiency and versatility of the resulting application. The three selected case studies are problems defined in changing environments where the decisions sensor / actuator should be tuned to the sensed conditions, requiring in this way an almost continuous modification of the underlying neural network models used. The observed reprogramming times were significantly low in the three cases and as a consequence the power consumption of the device is also quite low.

Even if a thorough comparison with the traditional case where a new code has to be transmitted from a central control unit were not carried out, the results indicate that a clear energy consumption reduction can be obtained, constituting a very important aspect of this technology. As a result, the suitability of the C-Mantec algorithm for its use in a complex task has been demonstrated using a microcontroller Arduino UNO. Further, given that nowadays there are also newer devices with more computing power and resources than the considered board, the present study confirms the potential of the proposed algorithm for real applications in tasks where sensors/actuators are needed.

Finally, as an overall conclusion we can say that the present thesis extends the study and application of neurocomputational models to FPGA and microcontrollers, devices that constitutes an alternative to standard computers for applications where a dynamic adaptation is needed to the actual conditions. It also demonstrates the potential benefits of FPGAs used as hardware accelerators in neurocomputational applications given their inherently parallel capabilities, while in relation to the use of neural networks in microcontrollers highlight the "on-chip" learning scheme utilized which allows for the use of remote sensors in stand-alone operation. From the carried out analysis, it can be concluded that implementations for these types of devices should be designed according to their specific configuration, as this might permit to achieve important improvements in terms of efficiency of hardware resource usage and computation speed, well above neurocomputational models developed on a PC computer. Further, the results indicate that alternative neurocomputational models might be more suited than the traditional ones for the implementation of larger architectures and in order to obtain faster response times.

Future lines of research that the made study in this thesis can follow is varied. Ones of which have been initiated are evolving the hardware implementation of Backpropagation the algorithm to obtain larger architectures by a layer multiplexing to simulate different hidden layers of the architecture; or analyzing the possibility of using FPGAs as hardware accelerators for simulations of complex systems with a high level of computational needs as the Ising model used to study the behavior of ferromagnetic materials. There are other possibilities for future work which will be studied as the use of the other neurocomputational models with different learning method for being developed in smart sensor networks in order to save resources by not storing a huge amount of data; or analyze the possibility of applying the neurocomputational systems

in real time on complex tasks such as a stabilization system of a quadcopter.

Índice

"La estupidez real siempre vence a la inteligencia artificial"

Terry Pratchett

Capítulo 1

Introducción

1.1. Conceptos generales previos

1.1.1. Redes Neuronales Artificiales

Las redes de neuronas artificiales (ANN: Artificial Neural Networks) son un paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso central de los animales (el cerebro en particular). Las ANN se presentan generalmente como un sistema de "interconexionado de neuronas" que calculan una respuesta a partir de un conjunto de estímulos de entradas. Un sistema de estas características se puede entender como procesadores paralelos masivamente distribuidos cuya función es almacenar conocimiento y representarlo para que esté disponible. Se basan en los siguientes principios:

1. El procesamiento de la información se ejecuta a través de múltiples elementos simples llamados unidades de proceso (neuronas).

2. Las neuronas están conectadas a través de conexiones sinápticas, las cuales transmiten las señales entre ellas.

3. Cada conexión sináptica entre dos neuronas tiene asociado un peso (peso sináptico) que tiene un efecto multiplicador sobre la señal transmitida.

4. Cada neurona dispone de una función de activación (o transferencia) para determinar su señal de salida.

Historia

- *1943:*Warren McCulloch y Walter Pitts publican un modelo de neurona que puede computar funciones aritméticas y lógicas [McCulloch y Pitts (1943)].

- *1949:* Donald Hebb plasma sus ideas sobre el aprendizaje neuronal en la conocida "regla de Hebb".

- *1958:* Frank Rosenblatt empieza el trabajo que conduce a la creación del concepto de perceptrón, siendo la red neuronal más antigua capaz de reconocer patrones [Rosenblatt (1962)].

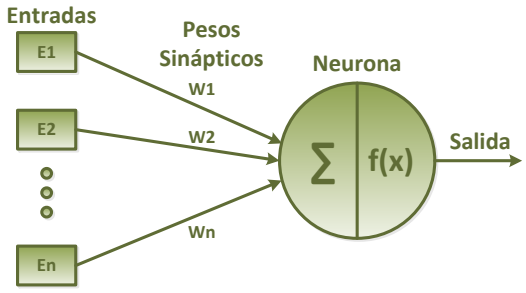

Figura 1.1: Representación de la arquitectura más básica de redes neuronales, un perceptrón simple con n entradas.

- *1960:* Widrow y Hoff desarrollan el "ADALINE", que fue la primera aplicación industrial real.
- *1967:* Marvin Minsky y Seymour Papert frenan esta primera etapa en el desarrollo de redes neuronales por la publicación sobre las limitaciones del perceptrón [Minsky y Papert (1969)].
- *1982:* Jhon Hopfield, con la publicación de su modelo [Hopfield (1982)] junto con la invención del algoritmo "BackPropagation" [Rumelhart et al. (1986)], consigue devolver el interés en el campo de la computación neuronal.
- *1987:* Se celebra en San Diego la primera conferencia abierta sobre redes neuronales (IEEE International Conference on Neural Networks), con más de 1700 participantes, y se constituye la International Neural Network Society (INNS).
- *1988:* Nace la revista Neural Networks; le siguen la revista Neural Computation en 1989 y la IEEE Transaction on Neural Networks en 1990. Posteriormente han ido apareciendo otras muchas y se han creado Institutos de Investigación y programas de formación en Neurocomputación.

Estructura

Una ANN está compuesta de elementos básicos de procesamiento (neuronas) que integran información desde múltiples entradas. Esta información es normalmente procesada mediante un sumador cuyo resultado actúa como entrada de una función de transferencia que calcula la respuesta de dicha neurona. La salida de una neurona se puede conectar a la entrada de otras mediante conexiones ponderadas (pesos sinápticos) que representan la eficacia de la sinapsis de las conexiones neuronales. En la Fig 1.1 se puede observar una representación del esquema más básico de una neurona artificial (perceptrón simple) . Las funciones de transferencias más utilizadas son la función escalón para salidas binarias y la función sigmoidea o la función tangente hiperbólica para salidas reales.

Una red neuronal consiste en un conjunto de neuronas interconectadas de una forma específica. El procesamiento en las ANN no reside solamente en el modelo de las neuronas sino en la manera en la que estas se conectan. Esta manera o estructura se conoce como arquitectura de la red y generalmente se organiza en grupos de neuronas llamados niveles o capas. Una de las arquitecturas más usadas es la de conexionado hacia delante

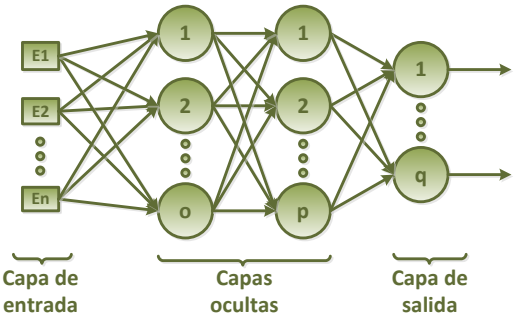

Figura 1.2: Representación gráfica de la estructura genérica de una arquitectura de red Feed-forward con n entradas 2 capas ocultas con o y p número de neuronas en cada capa y con q salidas.

o "feed-forward" que consiste en una capa de entrada, que proporciona información a la red, una serie de capas ocultas, que mapean la información de las entradas y una capa de salida que proporciona la respuesta de la red. La Fig 1.2 representa un modelo genérico de ANN con una arquitectura de red Feed-forward.

Propiedades

El poder de cómputo de una red neuronal deriva principalmente de su estructura masivamente paralela y de su capacidad de aprendizaje y generalización. Además, las redes neuronales disponen de las siguientes propiedades y capacidades que las hacen especialmente atractivas [Haykin (1998)]:

- *Sistemas distribuidos no lineales:* Una neurona puede ser un elemento de proceso lineal o no lineal, con lo que una red neuronal también puede simular sistemas no lineales y caóticos.
- *Relación entrada-salida:* Las ANN son capaces de vincular una salida en función de unos parámetros de entrada bajo la premisa de aprendizaje supervisado.
- *Adaptabilidad:* Una red tiene capacidad de adaptación al entorno a través de la modificación del conjunto de pesos sinápticos que conforman la red.
- *Autoorganización:* Las ANN usan su capacidad de aprendizaje adaptativo para organizar la información que reciben durante el aprendizaje y/o la operación.
- *Tolerancia a fallos:* Una red neuronal es un sistema computacionalmente robusto con alta tolerancia a fallos. Su rendimiento se degrada lentamente bajo un funcionamiento adverso (pérdidas de neuronas).
- *Implementación VLSI:* Gracias al paralelismo intrínseco, las ANN pueden conseguir gran rapidez de cómputo en la realización de determinadas tareas, lo que permite que puedan ser aplicados a sistemas en tiempo real, permitiendo simular sistemas biológicos mediante circuitos de silicio.

Reglas de aprendizaje

Una de las características más importantes de las redes neuronales es su capacidad de aprender interactuando con su entorno o con alguna fuente de información. El aprendizaje de la red es un proceso adaptativo mediante el cual se van modificando los pesos sinápticos de la red para mejorar el comportamiento de la misma. Se distinguen tres paradigmas de aprendizaje:

- *Aprendizaje supervisado:* Se dispone de un conjunto de patrones de entrenamiento para los que se conocen la salida deseada de la red. Un objetivo de este aprendizaje podría ser minimizar el error cuadrático medio cometido entre la salida de la red y la deseada.

- *Aprendizaje no supervisado (competitivo o autoorganizado):* Se dispone de un conjunto de patrones de entrenamiento pero no se conoce la salida que debe generar. La red por sí misma buscará su comportamiento más adecuado atendiendo a cierto criterio para encontrar estructuras o prototipos en el conjunto de patrones.

- *Aprendizaje por refuerzo:* basado en un proceso de prueba y error que busca maximizar el valor esperado de una función criterio conocida como una señal de refuerzo.

Aplicaciones

Las características de los modelos de sistema neurocomputacionales permiten que sean utilizadas en una gran variedad de aplicaciones, las cuales se puede enmarcar dentro de las siguientes categorías:

- Aproximación de funciones o análisis de regresión, incluyendo predicción de series temporales.

- Clasificación, comprendiendo reconocimiento de patrones y secuencias además de toma de decisiones.

- Procesamiento de datos, abarcando filtrado, agrupación, separación y compresión.

- Robótica, implicando manipulación directa y diseño de prótesis.

- Control, incluyendo control numérico por ordenador

Algunas de las aplicaciones en las que las ANN son utilizadas son: predicciones del mercado financiero debido al comportamiento no lineal del mismo [Bahrammirzaee (2010)]; diagnóstico médico para la predicción de enfermedades como el cáncer [Baena et al. (2013)]; tratamiento de imágenes como compresión de imágenes y video [Palomo et al. (2013)] o reconocimiento facial [Zhao et al. (2003)]; robótica [Chaoui et al. (2009)]; planificación y toma de decisiones en la teoría de juegos [Memmert y Perl (2009)], modelado de sistemas complejos en el sector energético [Kalogirou (2001)].

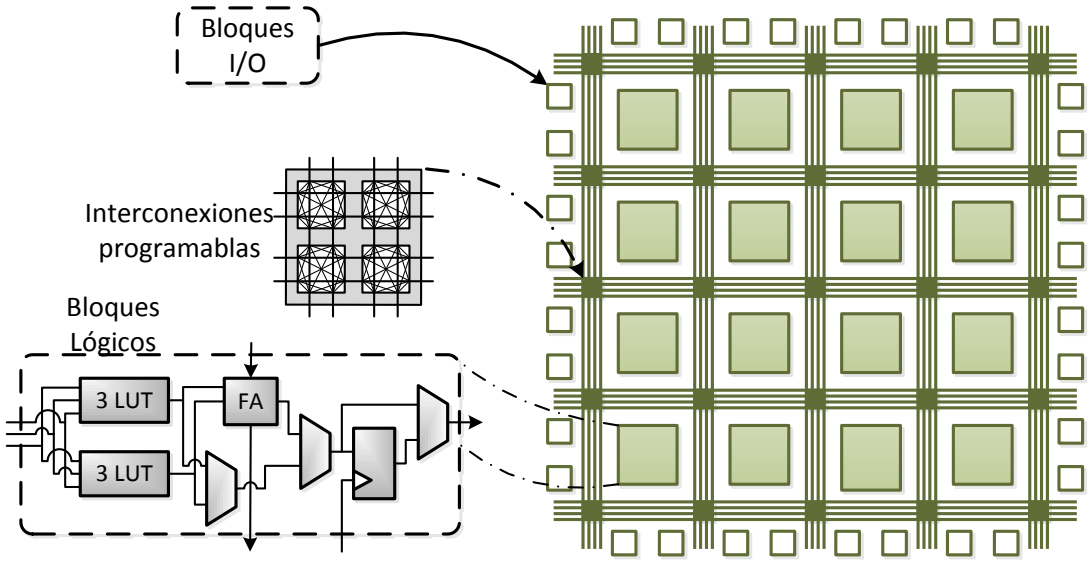

Figura 1.3: Representación de la estructura básica de una FPGA, describiendo los elementos básicos que la componen como son los bloques lógicos, las interconexiones programables y los bloques de entrada salida

1.1.2. FPGA

Una FPGA (Field Programmable Gate Array) es un circuito integrado semiconductor basado en una matriz de bloques de lógica configurables (CLBs) conectados entre sí, y a su vez, con celdas de entrada y salida (I/O) como puede observarse en la Fig. 1.3. Dichas interconexiones (también programables) forman una matriz de enrutado modificable según la funcionalidad necesaria por parte del usuario. Los CLBs se componen generalmente de varias tablas LUTs (Look-Up Table) cuyas salidas están multiplexadas y los bloques I/O comunican a la FPGA con el exterior.

A diferencia de una ASIC (Application-Specific Integrated Circuit), en el que el dispositivo está diseñado a medida para una aplicación determinada, las FPGAs se pueden programar cambiando su configuración para un uso determinado. La lógica programable puede reproducir desde funciones tan sencillas, como las llevadas a cabo por una puerta lógica, hasta sistemas combinacionales complejos. Las FPGAs se utilizan en aplicaciones similares a los ASICs, y aunque son más lentas y presentan un mayor consumo de potencia, proporcionan la gran ventaja de ser reprogramables. Esta característica añade una enorme flexibilidad al flujo de diseño y minimiza tiempos, coste de desarrollo y adquisición para pequeñas cantidades de dispositivos.

Los lenguajes de descripción de hardware más empleados en el diseño de FPGAS son VHDL y Verilog. Ambos son lenguajes que permiten diseñar la FPGA desde un punto de vista abstracto, funcional, aunque también se puede definir la estructura del hardware a bajo nivel. Existen además componentes predefinidos, los IPs, descritos en estos lenguajes para simplificar el diseño de la FPGA. Los principales fabricantes facilitan las herramientas para hacer más sencillo su proceso de diseño.

Reseña histórica

Las FPGA son el resultado de la convergencia de dos tecnologías diferentes, los dispositivos lógicos programables (PLDs Programmable Logic Devices) y los circuitos integrados de aplicación específica (ASIC). La tecnología de circuitos programables se desarrolló en los años 80 a partir de diferentes trabajos que derivaron en patentes, como la de Steve Casselman en 1992, el cual desarrolló un computador con 600.000 puertas programables, o las de David W. Page and LuVerne R. Peterson en 1985 que introdujeron conceptos fundamentales en puertas lógicas programables y bloques lógicos. Sin embargo fue la empresa Xilinx de manos de sus fundadores Ross Freeman y Bernard Vonderschmit la primera en comercializar una FPGA (modelo XC2064) que apenas contaba con 64 bloques lógicos programables con 2 LUTs de 3 entradas. Actualmente, la empresa Xilinx sigue siendo el principal fabricante de placas FPGAs, aunque existen otros fabricantes como Altera (principal competidor), Lattice Semiconductor, Actel ,etc. El mercado de FPGA a nivel mundial en 2013 fue de 5.386 millones de dólares, esperando llegar a 9.552 millones en 2020 [FPG (2014)].

Aplicaciones

Las FPGAs pueden implementar cualquier circuito específico siempre que se disponga de recursos para esa implementación. Están siendo utilizadas cada vez en más aplicaciones, sobre todo en las que es necesario disponer de sistemas de tiempo real y/o sistemas con un alto grado de paralelismo. Un ejemplo de estas aplicaciones pueden ser automoción, sistemas aeroespaciales y de defensa, prototipos de ASICs, procesamiento de imágenes, equipamiento médico, sistemas de visión para computadoras, instrumentación científica, bioinformática, emulación de hardware de computadora, entre otras.

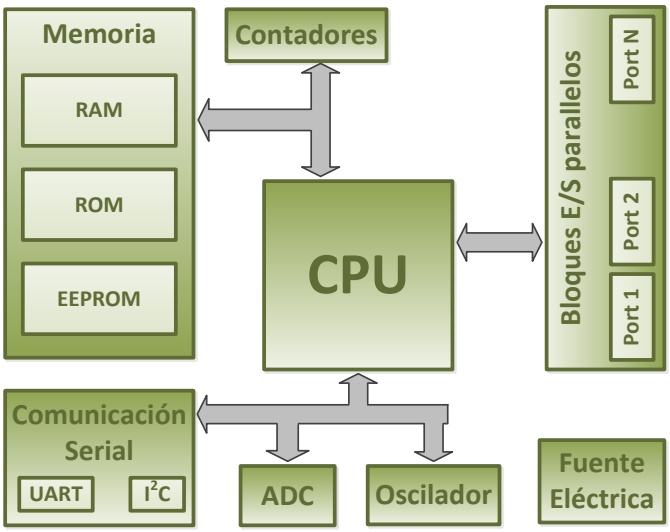

Figura 1.4: Representación gráfica de la estructura genérica de un microcontrolador, pudiéndose observar las diferentes elementos que lo componen

1.1.3. Microcontrolador

Un microcontrolador (μC, UC o MCU) es un circuito integrado programable capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto por las siguientes unidades funcionales cada una con una función específica (Ver Fig. 1.5):

- *Una unidad central de procesamiento (CPU):* Se encarga principalmente de leer la instrucción seleccionada, decodificarla y finalmente ejecutarla. También se encarga de conectar cada parte del microcontrolador.

- *Memoria:* Se utiliza para almacenar datos y programas. Un microcontrolador tiene generalmente una cierta cantidad de memoria RAM y ROM (EEPROM: Electrically Erasable Programmable Read-Only Memory, EPROM: Erasable Programmable Read-Only Memory, etc.) o memorias flash para almacenar los códigos fuente del programa.

- *Puertos paralelos de Entrada/Salida (I/O):* Se utilizan principalmente para manejar o leer interfaces externos, como diferentes tipos de sensores, LCDs, motores, etc.

- *Contadores:* Entre sus principales funciones se encuentran actuar como reloj, modulador, generador de pulso, medidor de frecuencia, generador de oscilación, etc.

- *Comunicaciones serie:* La función principal de esta unidad es dotar de un interfaz serie entre el microcontrolador y otros periféricos mediante comunicación UART o bus I2C.

- *Oscilador:* Genera una señal periódica normalmente sinusoidal o cuadrada para ser empleada a modo de reloj del sistema.

En el momento de fabricación la memoria de un microcontrolador no contiene datos, por lo que es necesario cargar el programa que se desee ejecutar en la memoria EEPROM (o similar) del microcontrolador. Para facilitar el proceso de carga del programa la mayoría de los fabricantes ofrecen un "Bootloader", programa que se graba en el microcontrolador una sola vez para facilitar la carga de posteriores programas sin necesidad de extraerlo del zócalo de la placa de desarrollo, permitiendo grabar los programas por los puertos de comunicaciones. El programa puede ser escrito en diferentes lenguajes de programación, desde lenguajes de bajo nivel como el ensamblador hasta más complejos como JAVA, aunque el más utilizados por los desarrolladores de este tipo de tecnología es el lenguaje C. No obstante, el programa debe ser codificado en sistema numérico hexadecimal antes de ser grabado en la memoria del microcontrolador.

Reseña histórica

El primer microcontrolador data del año 1971, cuando Texas Instruments crea el TMS 1000, que no comercializa hasta 1974. En esa misma fecha Intel lanza el Intel 4004, que sin embargo éste necesitaba circuitos adicionales para funcionar, con lo que poco tiempo después comercializó el Intel 8048, con el que logró un gran volumen de ventas gracias a que combinaba memoria RAM y ROM.

En 1993 la empresa General Instrument dio un salto de calidad en los microcontroladores al desarrollar la familia PIC, los cuales empezaron por primera vez a incluir memoria EEPROM (Electrically Erasable Programmable Read-Only Memory), permitiendo en ese momento un borrado eléctrico y rápido sin necesidad del complicado proceso que precisaban hasta la fecha.

En la actualidad, la reducción de costes provocada por el auge de la tecnología y abaratamiento del diseño está permitiendo la aparición de una gran variedad de microcontroladores, que se clasifican en función de la longitud de palabra utilizada (4, 8, 16, 32, 64 o 128 bits), las diferentes frecuencias de trabajo permitidas (desde varios kilohercios hasta frecuencias elevadas) y los consumos (del orden de milivatios e incluso de microvatios). Por lo general todos incluyen la función de espera, que reduce significativamente tanto la funcionalidad del microcontrolador como el consumo, estado inducido y/o anulado por algún evento como una interrupción o la pulsación de un botón.

Aplicaciones

Los microcontroladores son dispositivos que no sólo se han introducido en todos los ámbitos de la vida cotidiana, como la mayoría de electrodomésticos comunes, aparatos portátiles y de bolsillo, teléfonos móviles, inclusive en muchos elementos de juguetería electrónica, sino que también forman parte de todos los dispositivos de periféricos y dispositivos auxiliares de computación. Además, se utilizan en componentes industriales de instrumentación, sistemas de automoción o elementos diversos sistemas de seguridad y domótica en general y están involucrados en los elementos de tecnología más avanzadas, como sistemas de navegación espacial, en electromedicina, sistemas de control industrial y robótica.

1.2. Motivación

En muchos de los campos de la ciencia se ha generalizado la utilización de algoritmos neurocomputacionales, y en todos ellos van aparecido nuevas aplicaciones donde la utilización de la programación tradicional sobre ordenadores convencionales no es suficiente para poder implementar de manera eficiente una solución a un problema dado, ya sea por el elevado tiempo de cómputo de los ordenadores convencionales en problemas complejos o por su consumo y dimensiones en sistemas empotrados. Los sistemas en tiempo real y las redes de sensores son dos exponentes de aplicaciones donde la utilización de ANN debe realizarse sobre dispositivos y con programación diferente a los convencionales, como las FPGA o los microcontroladores que resultan más eficientes a la hora de una implementación de un modelo neurocomputacional.

1.2.1. Sistemas tiempo real

En ciencias de la computación los sistemas en tiempo real son aquellos sistemas hardware y software que están sujetos a unas limitaciones temporales que vienen dadas por naturaleza del propio sistema. Estos sistemas controlan o actúan sobre un entorno mediante la recepción de datos, el procesamiento de los mismos y la devolución con la suficiente rapidez para actuar sobre el entorno. Las respuestas en tiempo real a menudo son del orden de milisegundos, y en ocasiones microsegundos. Por el contrario, un sistema sin instalaciones en tiempo real no puede garantizar una respuesta dentro de cualquier período de tiempo.

La estructura y operatividad de las FPGAs ofrecen la posibilidad de realizar diseños eficientes de sistema en tiempo real debido a que se puede implementar funciones complejas para que sean ejecutadas de forma paralela, superando la potencia de cómputo de los procesadores digitales convencionales con paradigma de ejecución secuencial. De esta forma, se pueden controlar entradas y salidas (I/O) a nivel hardware ofreciendo tiempos de respuesta más veloces y funcionalidades especializadas que coinciden con los requerimientos de una aplicación en tiempo real.

Debido a la estructura intrínsecamente paralela de la FPGA, este tipo de dispositivos son muy adecuados para implementaciones de algoritmos de redes neuronales, ya que el tratamiento de información se realiza de forma paralela, lo que motiva su uso sobre todo en aplicaciones en tiempo real donde las restricciones temporales son un factor determinante. Existen numerosos ejemplos de aplicaciones en tiempo real [Kuo et al. (2006)] y en las que se emplean ANN [bin Huang et al. (2006)].

En este tipo de implementaciones es determinante el algoritmo utilizado para generar el modelo de ANN puesto que define la arquitectura empleada en el proceso de aprendizaje influyendo directamente en el tiempo de cómputo empleado y en la complejidad del diseño resultante.

Por lo tanto, una de las principales motivaciones de esta tesis es estudiar el uso de las FPGAS como dispositivo en tiempo real y conocer las limitaciones de estos dispositivos en este tipo de aplicaciones, además de diseñar algoritmos neurocomputacionales eficientes desde el punto de vista de los recursos empleados con el fin de poder implementar arquitecturas de ANN cada vez más complejas.

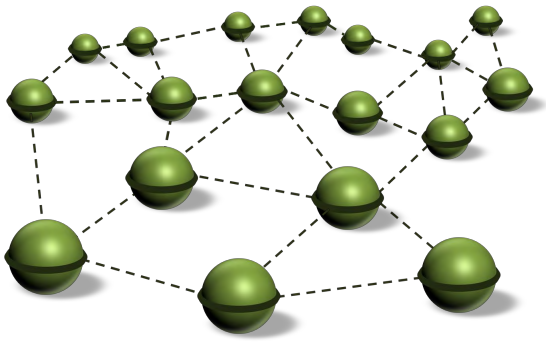

Figura 1.5: Esquema básico de interconexiones generadas por una red de sensores entre sus nodos

1.2.2. Redes de sensores

Una de las tecnologías que más ha avanzado en los últimos años ha sido las redes de sensores debido en mayor medida a los avances tecnológicos registrados en la última década respecto a la capacidad de los microcontroladores usados en este tipo de tecnología, así como la reducción en su costo y consumo energético. Esto ha permitido la utilización a gran escala de redes de este tipo de dispositivos, de hecho, fuentes especializadas estiman en un trillón la cifra de sensores que se alcanzará en 2015 monitorizando todo tipo de actividad [Kharif (2013)].

En los casos en que las condiciones ambientales evolucionan con el tiempo, la programación original de sensores/actuadores puede conducir a decisiones incorrectas, por lo que es necesario cambiar o adaptar el proceso de toma de decisiones a las nuevas condiciones [Sayed-Mouchaweh y Lughofer (2012)]. Por dicho motivo es necesario dotar de "inteligencia" a las redes de sensores, considerando inteligencia en este ámbito la aportación de información adicional a los datos obtenidos para dar soporte a la toma de decisiones y al procesamiento distribuido [Frank (2000)]. En este ámbito, la incorporación a este tipo de sensores de mecanismos cada vez más complejos de adaptación al entorno requiere del diseño de algoritmos específicos, o modificaciones de los ya existentes, para ser implementados en estos dispositivos que presentan limitaciones de recursos en comparación con sistemas de computación comunes (ej. PC).

A pesar del gran avance tecnológico producido en los últimos años en el campo de la microelectrónica, que permite disponer actualmente de microcontroladores con una capacidad de cómputo muy elevada a precios accesibles, los recursos de memoria y cómputo siguen siendo una importante limitación para la implementación de algoritmos de aprendizaje sofisticados sobre este tipo de dispositivos, ya que el tamaño y precio de las motas son aspectos importantes a tener en cuenta a la hora de su implementación física. En este aspecto se centra una de las motivaciones principales en esta tesis, diseñar modelos neurocompuatacionales eficientes con los que dotar de inteligencia a las redes de sensores.

1.3. Estado del arte

1.3.1. Sistemas neurocomputacionales en tiempo real

Los sistemas neurocomputacionales han tenido una gran repercusión en el ámbito científico desde su creación, pero no fue hasta los años 80 y principios de los 90 cuando se produjo un importante avance gracias en su mayor parte al desarrollo de la red de Hopfield, y en especial al algoritmo de aprendizaje de retropropagación (BackPropagation) ideado por Rumelhart y McLellan en 1986 que fue aplicado en el desarrollo de los perceptrones multicapa. Durante estos años se dedicaron grandes esfuerzos a la implementación hardware de sistemas neurocomputacionales sin demasiado éxito. La principal razón de este fracaso se puede atribuir a que la mayoría del trabajo realizado estaba dedicado a la implementación sobre circuitos específicos con tecnología ASIC. El rendimiento obtenido no fue competitivo para justificar la realización de circuitos específicos a gran escala, con lo que las implementaciones hardware sobre ASIC se descartaron en estos inicios. En aquellos momentos la tecnología FPGA no estaba lo suficientemente desarrollada para ser una opción competitiva a la hora de realizar implementaciones neurocomputacionales.

A principios de siglo hubo un resurgimiento de las implementaciones de sistemas neurocomputacionales en tiempo real, con la proliferación de nuevas tecnologías que daban lugar a nuevas familias de FPGAs con más poder de cómputo que permitían la implementación eficiente de este tipo de sistemas.

El libro [Omondi y Rajapakse (2006)] aportó una visión global de los trabajos realizados hasta el momento debido a que se trata de una recopilación de publicaciones de diferentes autores. Los trabajos tratan diferentes cuestiones a la hora de diseñar implementaciones hardware, siendo la más importante la representación del tipo de datos. La naturaleza intrínseca de la FPGA da lugar a utilizar representación entera (punto fijo si es necesaria parte fraccionaria) debido a la eficiencia de este tipo de representación, otro tipo de representación como punto flotante puede ser utilizada pero es necesario el uso de bloques específicos para su utilización [Ho et al. (2009); Jovanovic y Milutinovic (2012)], el trabajo [Savich et al. (2007)] describe un interesante análisis de la implementación de algoritmos neuronales en punto flotante con aritmética de punto fijo.

En estos primeros trabajos ya aparecieron dos posibles alternativas como solución al problema de implementación de modelos de aprendizaje en dispositivos hardware, el concepto de entrenamiento "on-chip" y "off-chip", en función de si incluye o no el proceso de aprendizaje dentro del dispositivo.

En implementaciones "off-chip" el aprendizaje del modelo de red neuronal se realiza generalmente en un ordenador personal (PC) y sólo los pesos sinápticos se transmiten a la FPGA que actúa como un acelerador de hardware. La principal ventaja de este esquema es la simplicidad del diseño hardware ya que todo el proceso de aprendizaje puede realizarse de forma convencional en un ordenador personal estándar y sólo es necesario hacer la implementación hardware de la red resultante. La principal desventaja es la imposibilidad de re-ajustar el modelo directamente en el dispositivo [Himavathi et al. (2007); Jung y Kim (2007); Orlowska-Kowalska y Kaminski (2011)].

Por el contrario, las implementaciones de aprendizaje "on-chip" incluyen el proceso completo de entrenamiento y ejecución en la propia FPGA, permitiendo entrenar los modelos de forma automática independiente del PC, ofreciendo además una gran

flexibilidad y eficiencia en los sistemas resultantes. Como contrapartida, este tipo de implementaciones requieren mucho más recursos de la FPGA [Dinu et al. (2010); Gomperts et al. (2010)].

En la actualidad, se ha extendido el uso de sistemas neurocomputacionales con esquema de aprendizaje "on-chip" en implementaciones hardware debido a que ha demostrado su superioridad en cuanto a la velocidad de cómputo ofreciendo sistemas más eficientes, aunque se hace evidente un incremento en la complejidad del diseño de las aplicaciones resultantes.

Una vez decidido el esquema de aprendizaje "on-chip" como el más adecuado, se plantean diferentes estrategias para implementar ANN en dispositivos FPGAs. Las dos estrategias principales para este tipo de implementaciones son:

1. Mejorar y sacar mayor rendimiento de los algoritmos clásicos ampliamente utilizados en múltiples aplicaciones. Ejemplos de trabajos donde se adoptan esta estrategia son:

   - *Lotrič y Bulić (2012):* Realiza una variación del algoritmo "BackPropagation" para conseguir un incremento de la potencia de cálculo y eficiencia del algoritmo. Para ello se sustituyen los multiplicadores, necesarios en este tipo de algoritmos, por una función aproximada más simple que requiere menos circuitería y que permite obtener mejores resultados sobre diferentes conjuntos de datos.
   - *Gomperts et al. (2011):* Implementa una arquitectura del algoritmo de perceptrón multicapa "BackPropagation" minimizando los costes del hardware y maximizando el rendimiento, la precisión y la parametrización. Presenta además la implementación real de una aplicación de sistemas neurocomputacionales en tiempo real

2. Diseñar nuevos algoritmos que se adapten mejor al tipo de implementación utilizada optimizando la utilización de recursos disponibles. Algunos ejemplos son los trabajos de:

   - *Shawash y Selviah (2013):* Describe una nueva implementación del algoritmo Levenberg-Marquardt, el cual es un algoritmo de aprendizaje no lineal que converge con precisión y rapidez. Como ejemplo, la función XOR es resuelta con tan sólo 13 iteraciones.
   - *Pan y Lan (2014):* Presenta una variación del MLP en la que combina una tercera capa con un algoritmo genético y el método máxima pendiente para hacer una búsqueda global de los pesos sinápticos de forma más rápida y eficiente.
   - *Bahoura (2014):* Propone una red neuronal dinámica con arquitectura en pipeline de alta velocidad para el modelado del comportamiento del amplificador de potencia. La novedad de la arquitectura propuesta reside en una mayor frecuencia de funcionamiento, la latencia de salida inferior y menor grado de utilización de recursos.

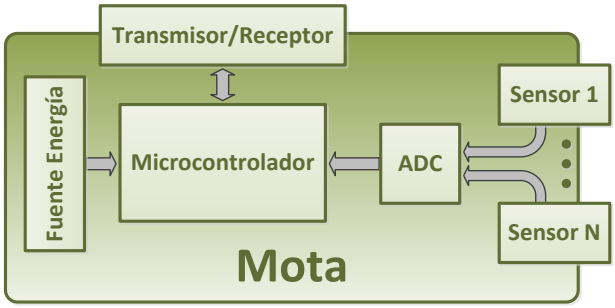

Figura 1.6: Representación gráfica de una mota en la que los sensores envían información al microcontrolador para procesarla y este se comunica con el exterior mediante un Transmisor/Receptor

1.3.2. Redes de sensores inteligentes

Los sensores (o detectores) son dispositivos que permiten la medición de variables químicas o físicas y las transforman en señales eléctricas, que son transmitidas por diferentes medios. Estos dispositivos son unidades autónomas que constan de un microcontrolador, una fuente de energía (usualmente una batería), un transmisor/receptor y un elemento sensor (Fig. 1.6).

Las redes de sensores inalámbricas (WSN) están formadas por un grupo de sensores con ciertas capacidades sensitivas y de comunicación inalámbrica los cuales permiten formar redes ad hoc sin infraestructura física preestablecida ni administración central.

Se conoce poco sobre el nacimiento de esta tecnología, ya que se produjo bajo el auspicio de la industria militar. El desarrollo moderno de pequeñas redes de sensores se remonta al proyecto Smartdust en 1998 y el proyecto de la NASA Sensor Webs. Aunque estos proyectos no tuvieron demasiada repercusión, el proyecto Smartdust desencadenó la ejecución de otros proyectos de investigación, NEST y CENS, desarrollados en la universidad de Berkeley, pioneros en este área de conocimiento y responsables de acuñar el término mota (mote) para referirse a los dispositivos nodo sensor.

Hoy en día el esfuerzo en el desarrollo de las WSN se centra en solucionar los siguientes problemas:

- *Cobertura:* Existen diferentes estrategias que abordan el problema en la fase de implementación [Wang (2011)]: (i) estrategia basada en la fuerza bruta que consiste en distanciar o acercar las motas hasta que queden en una situación de equilibrio en el que se abarque el máximo espacio posible; (ii) estrategia del punto de cuadrícula, que consiste en dividir el área en cuadrículas y poner cada mota en el centro de una; (iii) estrategia de geometría computacional para la optimización, que usa principalmente el diagrama de Voroi y la triangulación de Delaunay.

- *Desarrollo:* Existen 3 niveles de abstracción en cuanto al diseño de la tecnología de WSN para facilitar la labor de los desarrolladores [Laukkarinen et al. (2012)]: nodo, red e infraestructura, siendo ésta última la más importante debido a que se refiere típicamente al "middleware" de la red de sensores. El propósito principal de la abstracción de infraestructura es separar al usuario final de la heterogeneidad de una red de sensores, haciendo el desarrollo de aplicaciones más rápido y fácil.

- *Consumo energético:* La eficiencia energética de una WSN es, quizás, el aspecto más importante a centrarse en el desarrollo de este tipo de tecnología, dado que la recarga de baterías en una red de cierto tamaño puede resultar una tarea altamente costosa en tiempo, por lo que un reto claro en este tipo de redes es reducir significativamente el consumo energético de los dispositivos [Rault et al. (2014)]. Existen diferentes estrategias para reducir el consumo energético, siendo la más habitual la reducción de la transmisión de datos entre los nodos de la red, aunque existen otras vías para reducir el consumo como un enrutamiento energéticamente eficiente, la optimización radio o la estrategia reposo/activación.

Recientemente, en muchas aplicaciones se está considerado el equipamiento de los nodos sensores con un actuador, el cual convierte una señal eléctrica en un movimiento físico. Ejemplo de actuadores son algunos que incluyen válvulas que controlan el agua o la salida de gas, motores eléctricos que abran/cierren puertas y ventanas, interruptores para encender/apagar algún instrumento eléctrico o señales de alarma [Salarian et al. (2012)].

La red sensor/actuador inalámbrica (WSAN) resultante no sólo se compone de nodos de sensores que miden parámetros específicos sino que algunos de ellos tienen la capacidad de actuar sobre el medio ambiente, siendo una característica de este tipo de nodos la necesidad de una cantidad superior de recursos [Melodia et al. (2007)].

En ciertos escenarios donde las variables a sensar evolucionan con el tiempo, la programación original de sensores/actuadores puede conducir a decisiones incorrectas, por lo que es necesario cambiar o adaptar el proceso de toma de decisiones a las nuevas condiciones [Sayed-Mouchaweh y Lughofer (2012)]. La solución tradicional pasa por enviar los datos sensados a una unidad central, donde una persona los interpreta y vuelve a programar el microcontrolador con el nuevo conjunto de reglas Han et al. (2005); Wang et al. (2006); Shaikh et al. (2010).

Se han propuesto diferentes técnicas de reprogramación dinámica para cambiar el comportamiento de los sensores sin tener que reprogramar manualmente, ya que la reprogramación tradicional requiere en la mayoría de los casos la interrupción del proceso para cargar el nuevo código binario, con la consiguiente pérdida de tiempo y energía que participan en el proceso de comunicación entre la unidad central [Rassam et al. (2013); Aiello et al. (2011)].

Un primer paso hacia la reducción de los efectos anteriores ha sido la incorporación de sistemas de aprendizaje automático en el proceso de toma de decisiones, la automatización de la respuesta del microcontrolador sin interrumpir su ejecución y el envío de sólo una pequeña fracción de código para el microcontrolador [Urda et al. (2012); E. Cañete et al. (2012); Farooq et al. (2010)]. Sin embargo, los recientes avances en la potencia de cálculo de los microcontroladores permite la inclusión de sistemas de aprendizaje "on-chip", adaptando el comportamiento de sensores/actuadores dinámicamente según los datos sensados [Aleksendrić et al. (2012); Mahmoud et al. (2013)].

1.4. Objetivos

Esta tesis doctoral tiene como objetivo avanzar en el conocimiento científico y tecnológico necesario para el diseño e implementación de modelos neurocomputacionales en dispositivos hardware específicos (microcontroladores y FPGAs), con el fin de permitir su utilización en aplicaciones de sistemas en tiempo real y redes de sensores.

Se investigan y desarrollan estrategias de diseño para los dispositivos que maximicen la eficiencia en la utilización de recursos, por lo que se evalúan posibles alternativas al clásico algoritmo de aprendizaje para redes neuronales artificiales (algoritmo Backpropagation) utilizado habitualmente en las aplicaciones que precisan neurocomputación. Una alternativa clara parece ser los algoritmos constructivos de red neuronal (CoNN: Constructive Neural Networks) [Franco et al. (2010)], debido a que generan de forma automáticas arquitecturas de red compactas que consumen significativamente menos recursos del dispositivo. De forma más detallada, esta tesis busca alcanzar los siguientes objetivos parciales:

- Analizar el algoritmo de red neuronal "BackPropagation" con la finalidad de evaluar las fortalezas y deficiencias de éste para su diseño e implantación en los dos dispositivos hardware a analizar (FPGA y Microcontroladores). Desarrollar y evaluar diferentes técnicas de optimización del algoritmo para cada dispositivo y comparar dichos resultados con una programación tradicional realizada sobre un PC, con el fin de comprobar la mejoría en eficiencia de las técnicas propuestas.

- Evaluar una alternativa al algoritmo "BackPropagation" para realizar diferentes implementaciones sobre los dos dispositivos hardware estudiados con el objetivo de maximizar el uso de recursos en cada uno de ellos. (i) Analizar los algoritmos constructivos de red neuronal que generan arquitecturas de red de forma automática como alternativa para los modelos neurocomputacionales. (ii) Estudiar el nuevo algoritmo C-Mantec [Subirats et al. (2012)] con la función mayoría como función de salida de la red para comprobar si existe una reducción de la complejidad del modelo que repercuta en la simplicidad de la implementación hardware.

- Realizar un estudio exhaustivo de una implementación eficiente en una FPGA, sobre VHDL, del algoritmo C-Mantec a fin de evaluar su posible utilización en aplicaciones de sistemas en tiempo real. Desarrollar estrategias de diseño del algoritmo para reducir los recursos empleados en su implementación que, sin alterar la integridad del algoritmo, modifiquen la complejidad de los procedimientos. Además realizar un análisis de los tiempos empleados en el proceso de aprendizaje así como de la explotación del modelo, dado que estas variables van a determinar la viabilidad de utilización de la implementación resultante como sistema en tiempo real.

- Evaluar una implementación del algoritmo C-Mantec sobre una placa microcontroladora específica (Arduino UNO) para su utilización en redes de sensores. El objetivo principal es conseguir un aumento significativo en el tamaño de las arquitecturas resultantes así como reducir la memoria utilizada del dispositivo. Otro objetivo es evaluar dicha implementación en aplicaciones reales de redes de sensores como la predicción microclimática, la gestión de alarmas de emergencia y los sensores de caídas.

1.5. Estructura de la tesis

Tras este capítulo introductorio, la memoria de esta tesis se encuentra estructurada en los siguientes cinco capítulos.

El capítulo 2 expone la implementación del algoritmo "BackPropagation" en dos dispositivos hardware como son el microcontrolador y la FPGA, especificando las singularidades de dicho algoritmo para poder diseñar diferentes técnicas con el objetivo de conseguir implementaciones eficientes desde el punto de vista de la velocidad de cómputo y recursos. Se realiza además una comparativa para conocer si esta implementación mejora a las tradicionales de este algoritmo realizadas en un PC. En el apéndice B se describe detalladamente una de las técnicas utilizadas para maximizar la eficiencia, la implementación por medio de tablas de búsqueda e interpolación lineal de las funciones de transferencias usadas en este tipo de algoritmos.

El capítulo 3 describe la implementación de un nuevo algoritmo de red neuronal constructivo, C-Mantec, en una FPGA con el fin de mejorar la velocidad de los cálculos neurocompuacionales. Las características específicas de este nuevo algoritmo hace de él un candidato idóneo para la implementación hardware debido a la reducción de las arquitecturas que genera y a la simplicidad de su estructura, sin perder capacidad de generalización respecto a los algoritmos tradicionales y comprobando que las velocidades de cómputo son más elevadas que las conseguidas en un PC. En el apéndice C se describe una comparativa de la implementación hardware del algoritmo C-Mantec en comparación con la del algoritmo Backpropagation, describiendo las ventajas y desventajas de las dos implementaciones hardware.

El capítulo 4 estudia y diseña una red de sensores inteligentes para su uso en aplicaciones donde las condiciones ambientales evolucionan con el tiempo. Se implementa el novedoso algoritmo de red neuronal constructivo C-Mantec en un microcontrolador Arduino UNO, con el fin de dotar de inteligencia a los nodos sensores y que éstos puedan reprogramarse de forma automática sin necesidad de interacción exterior ahorrando un coste en tiempo y energía. Este proceso está basado en trabajos preliminares recogidos en el apéndice A. Como resultado se demuestra que esta implementación es más eficiente al ser desarrollada con éxito para para tres casos de estudios específicos de la vida real.

Finalmente, el capítulo 5 expone las conclusiones derivadas de esta tesis doctoral junto a las posibles líneas futuras de trabajo en este campo.

Capítulo 2

Implementaciones eficientes del algoritmo Backpropagation en FPGAs y microcontroladores

Francisco Ortega-Zamorano, José M. Jerez, Daniel Urda, Rafael Luque-Baena and Leonardo Franco: Efficient implementation of the Backpropagation algorithm in FPGAs and microcontrollers. **IEEE Transactions on Neural Networks and Learning Systems** (IN PRESS).

Posición JCR: Q1 (7/121) en Computer science: Artificial Intelligence.
Q1 (11/248) en Engineering: Electrical & Electronic.
Factor de impacto: 4,370

Resumen: El algoritmo de aprendizaje BackPropagation, el cual ha sido ampliamente analizado en múltiples estudios, se ha implementado en una FPGA y en un microcontrolador de forma eficiente en términos de uso de recursos y velocidad de cómputo. El planteamiento utilizado en ambos casos para evitar el sobreentrenamiento ha sido la estrategia entrenamiento/validación/testeo. Para el caso específico de la implementación hardware en la FPGA se ha introducido un nuevo diseño de una neurona que combina los elementos de la entrada con las funcionalidades de la primera capa oculta, permitiendo reducir drásticamente el número de recursos hardware utilizados; además, se ha introducido un esquema de división en el tiempo para realizar todas las multiplicaciones implicadas en el algoritmo con un solo bloque multiplicador para cada neurona. Para ambas implementaciones se ha redefinido la representación del tipo de datos utilizados pasando de la clásica representación en punto flotante, utilizada en este tipo de modelos, a la representación de punto fijo, consiguiendo así una reducción en la memoria utilizada para almacenar las variables del proceso y un aumento en la velocidad de procesamiento. Los resultados muestras que las modificaciones propuestas producen un claro incremento de la velocidad de cómputo en comparación con la implementaciones estándar realizadas en un PC, demostrando la utilidad del paralelismo intrínseco de una FPGAs en las tareas neurocomputacionales y la idoneidad de ambas implementaciones del algoritmo para problemas del mundo real.

Capítulo 3

Implementación en una FPGA del Algoritmo constructivo de red neuronal C-Mantec

Francisco Ortega-Zamorano, José M. Jerez and Leonardo Franco: FPGA Implementation of the C-Mantec Neural Network Constructive Algorithm. **IEEE Transactions on Industrial Informatics** 10(2): 1154-1161 (2014).

Posición JCR: Q1 (1/102) en Computer science: Interdisciplinary applications.
Q1 (1/43) en Engineering: Industrial.
Factor de impacto: 8,785

Resumen: Se ha realizado la implementación hardware en una FPGA del algoritmo recientemente propuesto de red neuronal constructivo (CoNN: Constructive Neural Networks) C-Mantec que genera arquitecturas muy compactas y con una muy buena capacidad de generalización. Una clara diferencia del algoritmo C-Mantec con respecto a la mayoría de las implementaciones de redes neuronales basadas en el algoritmo Backpropagation es que el algoritmo C-Mantec genera la arquitectura de red automáticamente, añadiendo neuronas conforme el modelo requiera arquitecturas más grandes para aprender el conjunto de datos de la aplicación; además, no precisa de capa de salida con lo que resta complejidad a la estructura de la red. Todos los pasos involucrados en la ejecución (incluyendo la fase de aprendizaje) se han descrito con detalle realizando un análisis en profundidad de los resultados que presentan claramente un incremento en la velocidad de cálculo en comparación con las implementaciones estándares realizadas mediante un ordenador personal, lo que demuestra la utilidad del paralelismo intrínseco de FPGAs en las tareas neurocomputacionales y la idoneidad de la versión de hardware del algoritmo de C-Mantec para su aplicación a los problemas del mundo real.

Capítulo 4

Reprogramación de sensores inteligentes en entornos cambiantes usando un modelo de red neuronal

Francisco Ortega-Zamorano, José M. Jerez, José Luis Subirats, Ignacio Molina and Leonardo Franco: Smart sensor/actuator node reprogramming in changing environments using a neural network model. **Engineering Applications of Artificial Intelligence** 30: 179-188 (2014).

Posición JCR: Q2 (31/121) en Computer science: Artificial Intelligence.
Q1 (15/87) en Engineering: Multidisciplinary.
Factor de impacto: 1,962

Resumen: Las técnicas desarrolladas en la actualidad como método para la actualización de los nodos de sensores requieren generalmente del uso de reprogramación; si éstos se encuentran en entornos cambiantes dicha reprogramación podría llegar a ser habitual, incrementando de forma sustancial los costes en términos de energía y tiempo. El trabajo presenta una alternativa al enfoque tradicional para la reprogramación de nodos sensores/actuadores en entornos cambiantes basada en un esquema de aprendizaje en el propio sensor para que de forma automática adapte el comportamiento a las condiciones del entorno. El modelo de aprendizaje propuesto se basa en el C-Mantec, un nuevo algoritmo de red neuronal constructivo especialmente adecuado para las implementaciones del microcontrolador, ya que genera arquitecturas de tamaño muy compacto reduciendo significativamente el uso de memoria del microcontrolador. La placa seleccionada ha sido el Arduino UNO, una placa microcontroladora muy popular de código abierto, económica y eficiente. Además este trabajo aporta un análisis en profundidad de las soluciones adoptadas para superar las limitaciones de recursos hardware en la implementación del algoritmo de aprendizaje junto con una evaluación de la eficiencia de este enfoque, probando el algoritmo en un conjunto de datos de funciones de referencia. Finalmente la utilidad y versatilidad del sistema se prueban en tres casos de estudios de diferente naturaleza en los cuales las condiciones ambientales evolucionan con el tiempo, cambiando el comportamiento del sistema.

Capítulo 5

Conclusiones y líneas de trabajo futuras

5.1. Conclusiones

Una vez estudiadas las diferentes implementaciones neurocomputacionales realizadas para los dos dispositivos hardware analizados (FPGA y microcontroladores), se exponen tanto las conclusiones derivadas del trabajo realizado en esta tesis doctoral como las posibles líneas de investigaciones futuras.

Sistemas neurocomputacionales en tiempo real

A la hora de implementar sistemas neurocomputacionales en tiempo real sobre FPGAs existen al menos dos estrategias posibles: modificar un algoritmo ya existente y optimizar su implementación con el objetivo de maximizar el uso de recursos, y una segunda opción que consiste en desarrollar un algoritmo nuevo que se adapte a las características de los dispositivos. A tal efecto se han analizado dos algoritmos diferentes, Backpropagation y C-Mantec, como exponentes de estas estrategias para estudiar su idoneidad y rendimiento en este tipo de sistemas.

El algoritmo de Backpropagation se ha implementado de forma eficiente en una FPGA bajo un paradigma de aprendizaje "on-chip", incluyendo además un sistema de validación del error para evitar el efecto de sobreajuste, e incorporando un diseño con modificaciones estructurales que permite minimizar los recursos utilizados. Se ha introducido un nuevo tipo de neurona, denominada "primera capa" que incorpora tanto las funciones de los elementos de entrada ("inputs"), como de las neuronas de la primera capa oculta, reduciendo drásticamente los recursos necesarios en su desarrollo, principalmente en arquitecturas con un gran número de entradas. Además, se han realizado mejoras adicionales en el diseño como son la multiplexación por división en el tiempo del bloque multiplicador y la interpolación de valores tabulados para aproximar la función sigmoidea. Estas modificaciones han permitido un ahorro de más de un 25 % en el uso de celdas lógicas y de un 50 % en bloques específicos en comparación con los resultados obtenidos en otros trabajos publicados en relación a la implementación del mismo algoritmo. La principal ventaja de las reducciones obtenidas es que permiten un incremento sustancial en el número máximo de neuronas que pueden crearse en la arquitectura de red neuronal. Un análisis detallado de la implementación realizada y de

los recursos de la placa utilizada muestra que la principal limitación en cuanto al número máximo de neuronas que pueden incorporarse viene dada por el número de bloques específicos DSP de la placa FPGA ya que se precisa un bloque específico DSP para cada una de las neuronas, haciéndose notar que de todos los bloques DSP existentes un bloque debe guardarse para su uso en el proceso de validación.

A fin de probar el correcto diseño de la implementación en FPGA, se han comparado los valores obtenidos en el proceso de entrenamiento de la red neuronal frente a los valores correspondientes de una implementación en un ordenador utilizando programación tradicional. Se observa que el error cuadrático medio evoluciona de forma similar para una serie de conjuntos de datos de prueba en ambos dispositivos. Los errores en la placa FPGA para el subconjunto de entrenamiento son ligeramente superiores a los obtenidos mediante el PC debido a los efectos producidos en el redondeo ocasionado por el tipo de representación de datos, pero sin embargo el modelo resultante final no se ve afectado como demuestran los valores obtenidos para los errores del subconjunto de test. En este primer análisis realizado se aprecia que los tiempos de entrenamiento son aproximadamente 100 veces menores en el caso de las implementaciones en FPGA.

Como conclusión de la estrategia en implementaciones hardware de algoritmos conocidos, decir que en el caso considerado fue posible incorporar modificaciones al algoritmo de Backpropagation para adecuarlo al dispositivo utilizado sin afectar la estructura del modelo, consiguiéndose optimizar los recursos disponibles y reducir drásticamente los tiempos de cómputo.

Como una alternativa al algoritmo de Backpropagation se ha estudiado un modelo de red neuronal de tipo constructivo para su implementación hardware en una FPGA. El algoritmo seleccionado ha sido C-Mantec, ya que posee una muy buena capacidad de predicción y permite seleccionar la arquitectura de forma automática conforme se realiza el proceso de entrenamiento. Este algoritmo se ha implementado de manera íntegra (con estructura de aprendizaje "on-chip") en una placa FPGA. Con el fin de evaluar su correcto diseño, los valores obtenidos en generalización para una serie de conjuntos de datos booleanos frecuentemente utilizados han sido comparados con los valores teóricos, constatando, además del correcto funcionamiento por parte de la implementación en FPGA, una disminución del tiempo empleado en el aprendizaje en relación al PC. Un examen pormenorizado de los tiempos de cómputo refleja que el número de veces que una FPGA es más rápida que un PC es proporcional al tamaño de la arquitectura resultante. Este interesante resultado puede entenderse en base a que el tiempo de cómputo en un ordenador crece de forma exponencial conforme se añaden neuronas a la capa oculta, mientras que lo hace de manera lineal en una placa FPGA, dando lugar a implementaciones hasta 47 veces más rápidas en los problemas más complejos.

Varias pruebas realizadas sobre los conjuntos de datos anteriormente mencionados muestran que la representación de datos utilizada (representación en punto fijo) es en la mayoría de los casos suficiente para lograr resultados comparables a los ejecutados en un PC con representación en punto flotante desde el punto de vista de la capacidad de generalización del algoritmo y del tamaño de las arquitecturas resultantes, siendo este algoritmo muy robusto en cuanto al tamaño de palabra utilizado en la representación de los datos.

En comparación con una implementación hardware del algoritmo de Backpropagation, la implementación del algoritmo C-Mantec resulta un 15 % más eficiente en cuanto a los recursos hardware utilizados; analizando los resultados se confirma que

las arquitecturas permitidas para el algoritmo C-Mantec son mayores en número de neuronas que las permitidas para el algoritmo Backpropagation en la misma placa FPGA. Además, las ejecuciones del algoritmo C-Mantec precisan una menor longitud en la representación de los datos para ser precisas, por lo que se constata que dicha implementación es menos sensible al tamaño de palabra utilizada, lo que supone un ahorro de recursos al reducir la complejidad de la implementación. Para finalizar la comparación destacar que el tiempo empleado para el aprendizaje de un conjunto de datos es inferior en las dos implementaciones hardware en comparación al empleado en un PC, destacando que es sensiblemente inferior para el algoritmo C-Mantec. Además, el tiempo de ejecución de la red donde sólo se calcula la salida de la red sin el proceso de aprendizaje es notablemente menor para el algoritmo C-Mantec, lo que supone una ventaja en la fase de explotación del modelo.

A la luz de los resultados observados se puede concluir que el presente análisis demuestra la idoneidad del algoritmo C-Mantec como modelo neurocomputacional para este tipo de dispositivos. Además, este estudio demuestra las ventajas potenciales de las tarjetas FPGA para actuar aceleradores hardware para su aplicación en problemas industriales del mundo real que precisen de la implementación de modelos neurocomputacionales.

Redes de sensores inteligentes

Se ha investigado también en esta tesis doctoral la posibilidad de incluir los dos algoritmos mencionados anteriormente (Backpropagation y C-Mantec) como posibles soluciones a modelos neurocomputacionales en redes de sensores inteligentes con los que dotar de cierto razonamiento el proceso de toma de decisiones necesaria para adaptarse a entornos cambiantes. Para ello se analizan las mejores estrategias a la hora de implementar redes neuronales artificiales en una placa microcontroladora. Se ha seleccionado para llevar a cabo los experimentos la placa Arduino por su versatilidad y eficiencia, siendo además de código abierto.

Los algoritmos Backpropagation y C-Mantec han sido implementados en el microcontrolador con estructura de aprendizaje "on-chip". Se ha cambiado el paradigma de representación de datos del tradicional punto flotante usado en este tipo de dispositivos a la representación en punto fijo. Este cambio ha reducido considerablemente la memoria utilizada para el almacenamiento de variables, permitiendo que el número de neuronas máximas del que disponer para la construcción de la arquitectura de red se vea incrementado en ambos algoritmos.

Las correctas implementaciones de los algoritmos en un microcontrolador han sido verificadas con diferentes conjuntos de datos, un conjunto para cada algoritmo, en los que los resultados obtenidos en el aprendizaje han sido comparados con los resultados teóricos para cada conjunto. Se observa una pequeña reducción de la precisión en la generalización de los datos, debido principalmente a los efectos producidos por el redondeo ocasionado por la representación del tipo de datos sin que ello condicione la eficacia del modelo neurocomputacional resultante. Además, para el caso concreto del algoritmo C-Mantec para el cual la arquitectura de red se determina de forma automática se observa que las arquitecturas resultantes son ligeramente más grandes a las obtenidas en un PC conforme crece el número de entradas, efecto también causado por el redondeo. Este incremento no afecta como ya se ha visto a la generalización de los datos pero repercute negativamente en la memoria utilizada sin que suponga un

perjuicio significativo.

Se han analizado detenidamente los tiempos de cómputo del aprendizaje para los dos tipos de representación de datos en ambos algoritmos. Para el Backpropagation se consiguen incrementos en la velocidad de entre 8 y 18 veces más rápido para las representaciones en punto fijo (8 bits en parte decimal) mientras que para el algoritmo C-Mantec se consiguen mejoras de hasta cinco veces más velocidad en comparación con la de punto flotante. La razón de este incremento se debe a que el tamaño de la arquitectura de la red repercute negativamente en la velocidad de cómputo de forma más severa para las representaciones de punto flotante ya que el manejo de memoria y el uso de la unidad aritmético lógica de este tipo de representaciones son más complejas. Estos resultados posibilitan estructuras de aprendizaje "on-chip" también en este dispositivo para diseñar sistemas neurocomputacionales en el propio microcontrolador, permitiendo su implantación en sensores remotos que precisan de un modo de trabajo autónomo.

Además, el algoritmo C-Mantec se ha empleado en una red de sensor/actuador para tres casos de estudios con el fin de demostrar la eficiencia y la versatilidad del sistema resultante. Los casos de estudios son problemas definidos en entornos cambiantes por tanto la toma de decisiones del sensor/actuador ha de adaptarse en consecuencia a los cambios observados, necesitando una reconversión del modelo de red neuronal que controla el proceso de decisión.

Los tiempos de reprogramación observados son relativamente bajos en los tres casos de estudio siendo, en consecuencia, el consumo de energía del dispositivo también bastante reducido. Incluso sin una comparación exhaustiva con el caso tradicional en el que el nuevo código tiene que ser transmitido desde una unidad de control central, los resultados hacen evidente una reducción muy importante en el gasto energético, cualidad muy importante en este tipo de tecnología debida a la corta duración de las baterías que lo alimentan. Los resultados han demostrado la idoneidad del algoritmo C-Mantec para su aplicación en tareas reales donde se precisen sensores/actuadores, notándose que se ha utilizado un microcontrolador Arduino UNO y que su utilización en casos reales es altamente factible dada la existencia en el mercado de dispositivos con prestaciones superiores a la placa considerada.

Conclusiones generales

Como conclusión global de esta tesis doctoral se puede afirmar que los resultados obtenidos en ella dan una visión detallada de implementaciones hardware de modelos neurocomputacionales en dos tecnologías muy extendidas como son los sistemas en tiempo real y las redes de sensores. Después del análisis exhaustivo realizado se puede concluir que las ANN pueden utilizarse en diversas aplicaciones de las tecnologías citadas, empleando dispositivos más apropiados que los ordenadores convencionales. Se ha demostrado que es posible la implementación de diferentes modelos en FPGAs y microcontroladores consiguiendo unos resultados muy superiores a los mismos modelos desarrollados en un PC.

Además se ha demostrado que si bien modificaciones efectuadas en los algoritmos conocidos los hacen más eficientes para su implementación en dispositivos específicos, existen otros modelos de redes neuronales más adecuados para el tipo de dispositivo analizado que los tradicionales con los que es posible construir arquitecturas de mayor tamaño y obtener tiempos de respuesta más cortos.

5.2. Líneas de trabajo futuras

Dada la importancia de las tecnologías estudiadas en esta tesis, las posibles líneas de investigación y aplicaciones derivadas son numerosas, algunas ya han comenzado a desarrollarse por el grupo de investigación y otras pueden llevarse a cabo en el futuro para ampliar y mejorar los resultados de la tesis. Algunos ejemplos de futuros trabajos son:

- Progresar en la implementación hardware del algoritmo Backpropagation con el fin de poder utilizar arquitecturas de red con N capas ocultas, donde N es un número elevado no predefinido. Para ello se precisa diseñar una sola capa oculta que haga la funcionalidad de todas las capas y que permita la iteración de dicha capa (N veces) para simular la arquitectura completa. Esta implementación podría ser utilizada junto con un algoritmo evolutivo para encontrar arquitecturas de red óptimas en función de diferentes conjuntos de entrada.

- Analizar la posibilidad de usar otros modelos neurocomputacionales con diferentes reglas de aprendizaje para diseñar nuevas toma de decisiones en redes de sensores inteligentes. El mapa auto-organizado (SOM: Self-Organizing Map) es un tipo de ANN no supervisado cuyo entrenamiento produce una representación discreta del espacio de las muestras de entrada. Este modelo genera la red en función de la estructura de los datos de entrada sin necesidad de tener un conocimiento previo de su salida con lo que no precisa guardar el conjunto de datos con el consiguiente ahorro de recursos y tiempo en el uso de memoria.

- Investigar la posibilidad de utilizar las placas FPGAs como acelerador hardware de otros sistemas complejos con una necesidad de cómputo muy elevada. El modelo de Ising es un modelo físico propuesto para estudiar el comportamiento de materiales ferromagnéticos que evalúa el comportamiento de cada partícula en función de sus vecinas. En sistemas con interacciones de largo alcance, en los que se necesite conocer la evolución de una partícula a partir del conjunto total de las mismas, la implementación algorítmica en una FPGA puede reducir drásticamente el tiempo de modelado del sistema.

- Explorar la posibilidad de realizar nuevas aplicaciones con redes de sensores inteligentes, como la utilización de modelos neurocomputacionales para el control de huertos urbanos que precisan de un control exhaustivo de los recursos hídricos. Cada semilla y cada planta necesitan un riego diferente en función de las condiciones microclimáticas y en este escenario los modelos neurocomputacionales podrían suprimir la necesidad de estudios previos costosos, proporcionando un control automático para un sistema de plantación complejo.

- Analizar la posibilidad de utilizar sistemas de neurocomputación en tiempo real en áreas complejas, como los sistemas de estabilización de un cuadricóptero que permita una rápida adaptación a cambios estructurales en el dispositivo en caso, por ejemplo, de pérdida de un rotor o de condiciones de vuelo bajo inclemencias meteorológicas.

Capítulo 6

Conclusions and future lines of research

6.1. Conclusions

We have analyzed in this thesis the implementation of two different neural network models (C-Mantec and Back-propagation algorithms) in FPGA and microcontroller devices, and now we present in this chapter the final conclusions and possible future extensions of the work done.

Neurocomputational models for real time systems

At the time of the implementation of neurocomputational models for real time systems on a FPGA board there are at least two possible strategies: adapt an existing neural model to the hardware resources trying to optimize resource usage, and secondly develop a new algorithm taking into account the characteristics of the FPGA. With these two different strategies in mind, we have analyzed in this work the Backpropagation and C-Mantec neural models, studying their adaptation and performance for FPGA boards.

The Backpropagation algorithm has been efficiently implemented in a FPGA board using an "on-chip" learning scheme that includes a validation process to avoid overfitting; and several optimization strategies have been developed in order to minimize resource usage. In particular, a new type of neuron named "Inp-Hid" layer has been created. This new element incorporates the functions from both the input elements and the first hidden layer neurons, reducing drastically the needed resources, especially in architectures with a large number of inputs. Furthermore, the use of a time-division multiplexing scheme for implementing a multiplier block, together with a scheme based on tabulation plus interpolation of values for the computation of the sigmoid function has enabled a saving of more than 25 % in the number of logic cells used and 50 % in specific blocks in comparison to previous published results. An important aspect of the implemented modifications is that they allow for a substantial increase in the maximum number of neurons that can be included in a network architecture. An analysis of the developed implementation in terms of the constraints imposed by the available resources of the used FPGA board (Xilinx Virtex V) indicates that the main limitation regarding the maximum number of neurons that can be created is the number of DSP

specific blocks included in the board, as each neuron requires one DSP block for its implementation.

In order to check for the correct implementation of the Backpropagation algorithm in the FPGA board, a comparison with the original PC implementation of the algorithm is carried out. It is observed that the mean square error evolves in the same manner in both cases, existing small differences as training errors are lower for the PC implementations, fact that can be justified by rounding errors present in the FPGA implementation that do not affect the predictive accuracy. Regarding computational times, training a neural model using the FPGA code is about 100 times faster than using the PC one.

As a partial conclusion regarding hardware implementations based on existing algorithms we say that neural models should be adapted and modified to suit the device but taking care of not affecting the model structure. In particular in our specific case, it has been possible to maximize the available resources and at the same time reduce significantly the computational times.

As an alternative to the traditional Backpropagation algorithm, we have studied a second neurocomputational model for its FPGA hardware implementation. The selected algorithm is C-Mantec, a neural network constructive model that generates the architecture automatically as training is performed leading to compact neural architectures with very good predictive capabilities. The algorithm has been completely implemented ("on-chip" learning scheme) in a FPGA board, adapting it specifically for this device. The correct implementation of the algorithm has been checked by analyzing and comparing the training and generalization error curves for well-known Boolean data sets, observing besides its proper operation a decrease in the training times in relation to the PC implementation. A detailed examination of these times shows that the number of times an FPGA is faster than a computer is proportional to the size of the resultant architecture, and as the computing time for a PC grows exponentially as neurons are added to the hidden layer while linearly for the FPGA, in some cases, results up to 47 times faster can be observed for the more complex architectures. Several tests with the previously mentioned data sets confirm that the used data type representation (fixed point) is sufficient to achieve similar results to those obtained from a PC using a floating point representation.

In comparison with the hardware implementation of the Backpropagation algorithm, C-Mantec performance is 15 % more efficient in terms of used hardware resources permitting to create for a given FPGA board larger neural networks. In addition, the execution of the C-Mantec algorithm needs shorter length for the data representation in order to operate correctly, indicating as expected that C-Mantec is less sensitive to number precision than Backpropagation. To complete the comparison between both algorithms FPGA implementation training time are significantly lower for the C-Mantec algorithm, and in particular the runtime of the network (after the training phase is done) is significantly lower for this algorithm, an important advantage at the exploitation phase of the algorithm.

The observed results confirm the suitability of the C-Mantec algorithm as a valid neurocomputational model for its implementation in FPGA boards, demonstrating the potential advantages of FPGAs to act as hardware accelerator devices for application to real-world industrial problems.

Smart sensor networks

Smart wireless sensor networks is another type of technology that we have analyzed for the application of neurocomputational models, with the aim of providing intelligence to the decision making process that it is needed in order for systems to adapt to changing environments. The implementations of the two previously studied algorithms have been analyzed in order to demonstrate their suitability for the application in sensor networks. Backpropagation and C-Mantec algorithms have been implemented in an Arduino board with "on-chip" learning scheme. The data type representation has been changed from traditional floating-point representation used in this type of algorithms to fixed-point representation with the idea of reducing memory storage and in order to build as large as possible neural architectures. To verify the correct implementation two data sets have been employed and the obtained results have been compared with the theoretical results. A small reduction in accuracy is observed for the generalization of data sets due mainly to the effects of rounding in the data type representation without altering the effectiveness of the resultant neurocomputational model. In addition, for the specific case of the C-Mantec algorithm where the network architecture is generated automatically it can be observed that the resultant architectures are slightly larger than for the PC implementation, being this effect caused by rounding. Nevertheless, the increase does not affect the generalization ability. Further, we have carefully analyzed the learning computational times for the two types of data representation in both algorithms. An increase in speed between 8 and 18 is obtained for the Backpropagation algorithm, while up 5 times improvement is obtained for the C-Mantec algorithm using 8 bits in the decimal part of the fixed point representation in comparison to the floating point one.

The obtained results enable to build "on-chip" learning schemes that permits to implement the neurocomputational models in the microcontroller, that allows for their use in wireless sensor networks in autonomous mode. Furthermore, the C-Mantec algorithm has been employed in a sensors/actuators network for three case studies to demonstrate the efficiency and versatility of the resultant application. Case studies are defined problems in changing environments where a sensor/actuator should be adjusted accordingly to the observed changes, requiring a re-training of the neural network model that controls the decision process. The observed reprogramming times are significantly short in the three cases, and therefore the power consumption of the device is also quite low. Even without a thorough comparison with the traditional case where the new code should be transmitted from a central control unit, the results highlight a noteworthy reduction in energy supply, very important quality in this type of technology due to the short life of batteries. As part of result, it has demonstrated the suitability of C-Mantec algorithm to operate with using a microcontroller Arduino UNO on concrete applications for complex tasks. Even more, given the availability of similar devices with much more hardware resources, this study confirms the potential of the proposed algorithms for their application in real tasks where sensors/actuators are required.

Final Conclusion

As an overall conclusion of this thesis, it can be said that the problems analyzed together with the obtained results give a complete overview of the features to take into account for carrying out neurocomputational implementations in the two studied technologies (real-time systems and sensor networks). After the detailed analysis done,

it can be concluded that neurocomputational models can be used in real-time systems and sensor networks applications, in cases where is better to use alternative devices rather than traditional PCs. It has been shown that it is possible to implement different neural network models on FPGA boards and microcontrollers obtaining much faster computational times than when the same models are implemented on a PC.

Further, it has also been demonstrated that while adaptations made to known algorithms (the Backpropagation algorithm in our case) can make them much more efficient for their implementation in non PC hardware device, the use of alternative models like C-Mantec can be more suitable for FPGAs and microcontrollers, permitting to build larger network architectures with shorter response times.

6.2. Lines of future research

Given the variety of applications which can be used according to the studied implementations, many possible lines of research can be carried on in the future. Some of them have already begun and others could start in the future in order to improve the exposed data. Some items could be the following:

- Progressing in the hardware implementation of the Backpropagation algorithm in order to allow architectures of neural network of N hidden layers, where N is a non-predetermined large number. This process requires designing a single hidden layer with the functionality of every layer, and then this layer would be able to simulate the whole architecture with N iterations. This implementation could be used with an evolutionary algorithm in order to generate optimal network architectures for different input data sets.

- Analyzing the possibility of using other neurocomputational models with other learning rules to design decision-making in smart sensor networks. A self-organizing map (SOM), that is trained using unsupervised learning, seems a suitable neural network for these devices because the model does not require knowing the output of data sets since the model is modified according to the structure of these sets. So it is not necessary to store the above data, being a saving of resources and time by not using the memory and its management.

- Investigating the possibility of using FPGAs as hardware accelerator boards of other complex systems with a need for very high computation. The Ising model is proposed to study the behavior of ferromagnetic materials that evaluates the behavior of each particle as a function of its neighbors'physical model. In systems that need to know the status of all the neighbors to know the behavior of a single particle, processing can take a huge amount of time. A application based on an FPGA system can reduce the time taken to model the ferromagnetic behavior.

- Exploring the possibility of new real applications to networks of intelligent sensors. An application startup is to use neurocomputational models for the control of urban gardens that require a very thorough control of water resources. Every seed and every plant require different irrigation based on microclimatic conditions; neurocomputational models eliminate the need for a prior study and can make automatic control of such a complex system of plantation.

- Analyzing the possibility of using neurocomputational systems in real time on complex areas. One possibility would be to develop a system of stabilization in quadricopter that allows quick adaptation to structural changes in the device such as the loss of a rotor or inclement weather.

Apéndice A

Implementación del algoritmo de red neuronal constructivo en un microcontrolador Arduino UNO

Francisco Ortega-Zamorano, José Luis Subirats, José M. Jerez, Ignacio Molina, Leonardo Franco: Implementation of the C-Mantec Neural Network Constructive Algorithm in an Arduino Uno Microcontroller. **Lecture Notes in Computer Science** 7902, pp. 80-87, (2013). ISBN: 978-3-642-38678-7.

Resumen:

Un algoritmo constructivo propuesto recientemente de red neuronal, denominado C-Mantec, se diseña de forma íntegra (proceso de aprendizaje incluido) para una placa microcontroladora Arduino UNO. Dicho algoritmo genera arquitecturas de red muy compactas con buenas capacidades de predicción que lo hacen idóneo para ser implementado en un microcontrolador a fin de ser usado como dispositivo neurocomputacional sin necesidad de transmitir información a una unidad de control central para efectuar el proceso de aprendizaje.

Se detalla la implementación de los procesos más complejos y se realiza un análisis del correcto funcionamiento de la aplicación resultante mediante la verificación del aprendizaje de un conjunto de datos de referencia usados normalmente en el diseño de circuitos.

Apéndice B

Implementaciones FPGA de alta precisión de funciones de transferencia de redes neuronales

Francisco Ortega-Zamorano, José M. Jerez, Gustavo Júarez, Jorge O. Pérez and Leonardo Franco: High precision FPGA implementation of neural network activation functions. **Proceedings of the IEEE Symposium Series on Computatitional Intelligence (SSCI'2014),**pp. 55-60, (2014). ISBN: 978-1-4799-4486-6.

Resumen:

Las implementaciones hardware de modelos neurocomputacionales en FPGAs requieren afrontar varios problemas que afectan en gran medida al resultado final del sistema para disponer de aplicaciones eficientes. Uno de los más evidentes es el cálculo de la función de transferencia de la neurona. Se ha efectuado un análisis de las implementaciones de las funciones Sigmoidea y exponencial en las que se ha utilizado una estructura que consiste en tabular los valores de la función combinada con un procedimiento de interpolación lineal.

Además se ha utilizado un esquema de división en el tiempo para el bloque multiplicador con el objetivo de implementar un solo bloque por neurona, para ejecutar todas las multiplicaciones asociadas al algoritmo para ahorrar recursos de la placa.

Los resultados se han evaluado en términos de error absoluto y relativo obtenidos para la aproximación y a través de un factor de calidad, demostrando una clara mejoría en relación a los trabajos publicadas con anterioridad.

Apéndice C

Comparativa de implementaciones hardware dos algoritmos de aprendizaje de redes neuronales

F. Ortega-Zamorano, José M. Jerez, Gustavo Juárez and Leonardo Franco:FPGA implementation comparison between C-Mantec and Back-Propagation neural network algorithms. **Lecture Notes in Computer Science** 9095, pp. 197-208, (2015). DOI:10.1007/978-3-319-19222-2_17

Resumen:

Las implementaciones hardware de los modelos neurocomputacionales pueden orientarse a dos estrategias diferentes para conseguir diseños eficientes que reduzcan el consumo de recursos: modificar el algoritmo Backpropagation usado normalmente en este tipo de implementaciones o desarrollar nuevos algoritmos que se adapten mejor a los dispositivos.

Se ha realizado una comparación de las dos estrategias de diseño representadas por dos implementaciones diferentes. Una implementación ha sido una modificación del algoritmo Backpropagation con su estructura modificada al definir una nueva primera capa para reducir los recursos utilizados; y la otra implementación ha sido el algoritmo C-Mantec que es un modelo de red neuronal constructivo que genera la arquitectura de red de forma automática.

Se analizan varios aspectos fundamentales de la aplicaciones sobre FPGAs, como son los recursos hardware utilizados, utilización de bloque específicos DSP, tiempos de Cómputo, número de LUTs, implementación de la función de transferencia, etc. Además se estudian las ventajas y desventajas de ambos métodos en el contexto de aplicaciones prácticas.

Bibliografía

FPGA (Field-Programmable Gate Array) Market Analysis By Application (Automotive, Consumer Electronics, Data Processing, Industrial, Military And Aerospace, Telecom) And Segment Forecasts To 2020, 2014.

Aiello, F., Bellifemine, F. L., Fortino, G., Galzarano, S. y Gravina, R. An agent-based signal processing in-node environment for real-time human activity monitoring based on wireless body sensor networks. *Eng. Appl. Artif. Intell.*, vol. 24(7), páginas 1147–1161, 2011.

Aleksendrić, D., Jakovljević, I. y Irović, V. Intelligent control of braking process. *Expert Syst. Appl.*, vol. 39(14), 2012.

Baena, R. M. L., Urda, D., Subirats, J. L., Franco, L. y Jerez, J. M. Analysis of cancer microarray data using constructive neural networks and genetic algorithms. En *IWBBIO* (editado por I. Rojas y F. M. O. Guzman), páginas 55–63. Copicentro Editorial, 2013.

Bahoura, M. Fpga implementation of high-speed neural network for power amplifier behavioral modeling. *Analog Integrated Circuits and Signal Processing*, vol. 79(3), páginas 507–527, 2014.

Bahrammirzaee, A. A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, vol. 19(8), páginas 1165–1195, 2010.

Chaoui, H., Sicard, P. y Gueaieb, W. Ann-based adaptive control of robotic manipulators with friction and joint elasticity. *Industrial Electronics, IEEE Transactions on*, vol. 56(8), páginas 3174–3187, 2009.

Dinu, A., Cirstea, M. y S.E.Cirstea. Direct neural-network hardware-implementation algorithm. *IEEE Transactions on Industrial Electronics*, vol. 57(5), páginas 1845–1848, 2010.

E. Cañete, E., Chen, J., Luque, R. y Rubio, B. Neuralsens: A neural network based framework to allow dynamic adaptation in wireless sensor and actor networks. *J. Network and Computer Applications*, vol. 35(1), páginas 382–393, 2012.

Farooq, U., Amar, M., ul Haq, E., Asad, M. U. y Atiq, H. M. Microcontroller based neural network controlled low cost autonomous vehicle. En *Proceedings of the 2010 Second International Conference on Machine Learning and Computing*, ICMLC '10, páginas 96–100. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-3977-5.

FRANCO, L., ELIZONDO, D. A. y JEREZ, J. *Constructive neural network*, vol. 258. Springer, 2010.

FRANK, R. *Understanding Smart Sensors, Second Edition*. Artech House, Inc., Norwood, MA, USA, 2nd edición, 2000. ISBN 0890063117.

GOMPERTS, A., UKIL, A. y ZURFLUH, F. Implementation of neural network on parameterized fpga. En *AAAI Spring Symposium: Embedded Reasoning: Stanford University,*, páginas 45–51. AAAI Spring Symposium 2010 - Embedded Reasoning: Stanford University, CA, USA, 2010.

GOMPERTS, A., UKIL, A. y ZURFLUH, F. Development and implementation of parameterized fpga-based general purpose neural networks for online applications. *IEEE Transactions on Industrial Informatics*, vol. 7(1), páginas 78–89, 2011.

HAN, C.-C., KUMAR, R., SHEA, R. y SRIVASTAVA, M. Sensor network software update management: a survey. *Int. J. Netw. Manag.*, vol. 15(4), páginas 283–294, 2005.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edición, 1998.

HIMAVATHI, S., ANITHA, D. y MUTHURAMALINGAM, A. Feedforward neural network implementation in fpga using layer multiplexing for effective resource utilization. *IEEE Transactions on Neural Networks*, páginas 880–888, 2007.

HO, C. H., YU, C. W., LEONG, P., LUK, W. y WILTON, S. J. E. Floating-point fpga: Architecture and modeling. *IEEE Trans. VLSI Syst.*, vol. 17(12), páginas 1709–1718, 2009.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *National Academy of Sciences of the USA*, vol. 79(8), páginas 2554–2558, 1982.

BIN HUANG, G., MEMBER, S., YU ZHU, Q. y KHEONG SIEW, C. Real-time learning capability of neural networks. *IEEE Trans. Neural Networks*, páginas 863–878, 2006.

JOVANOVIC, Z. y MILUTINOVIC, V. Fpga accelerator for floating-point matrix multiplication. *IET Computers & Digital Techniques*, vol. 6(4), páginas 249+, 2012.

JUNG, S. y KIM, S. S. Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems. *IEEE Transactions on Industrial Electronics*, vol. 54(1), páginas 265–271, 2007.

KALOGIROU, S. A. Artificial neural networks in renewable energy systems applications: a review. *Renewable and Sustainable Energy Reviews*, vol. 5(4), páginas 373–401, 2001.

KHARIF, O. Trillions of smart sensors will change life. bloomberg news. http://www.bloomberg.com/news/2013-08-05/trillions-of-smart-sensors-will-change-life-as-apps-have.html, 2013.

KUO, S. M., LEE, B. H. y TIAN, W. *Real-Time Digital Signal Processing: Implementations and Applications*. Wiley, 2006.

LAUKKARINEN, T., SUHONEN, J. y HANNIKAINEN, M. A survey of wireless sensor network abstraction for application development. *International Journal of Distributed Sensor Networks*, vol. 2012, páginas 1–12, 2012.

LOTRIČ, U. y BULIĆ, P. Applicability of approximate multipliers in hardware neural networks. *Neurocomputing*, vol. 96, páginas 57–65, 2012.

MAHMOUD, S., LOTFI, A. y LANGENSIEPEN, C. Behavioural pattern identification and prediction in intelligent environments. *Appl. Soft Comput.*, vol. 13(4), páginas 1813–1822, 2013.

MCCULLOCH, W. S. y PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, vol. 5(4), páginas 115–133, 1943.

MELODIA, T., POMPILI, D., GUNGOR, V. y AKYILDIZ, I. Communication and coordination in wireless sensor and actor networks. *Mobile Computing, IEEE Transactions on*, vol. 6(10), páginas 1116–1129, 2007.

MEMMERT, D. y PERL, J. Game creativity analysis using neural networks. *Journal of Sports Sciences*, vol. 27(2), páginas 139–149, 2009.

MINSKY, M. y PAPERT, S. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

OMONDI, A. y RAJAPAKSE, J. *FPGA Implementations of Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387284850.

ORLOWSKA-KOWALSKA, T. y KAMINSKI, M. Fpga implementation of the multilayer neural network for the speed estimation of the two-mass drive system. *IEEE Transactions on Industrial Informatics*, vol. 7(3), páginas 436–445, 2011.

PALOMO, E. J., DOMINGUEZ, E., LUQUE-BAENA, R. M. y MUÑOZ, J. Image compression and video segmentation using hierarchical self-organization. *Neural Processing Letters*, vol. 37(1), páginas 69–87, 2013.

PAN, S.-T. y LAN, M.-L. An efficient hybrid learning algorithm for neural network-based speech recognition systems on fpga chip. *Neural Computing & Applications*, vol. 24(7–8), páginas 1879–1885, 2014.

RASSAM, M., ZAINAL, A. y MAAROF, M. An adaptive and efficient dimension reduction model for multivariate wireless sensor networks applications. *Appl. Soft Comput.*, vol. 13(4), páginas 1978–1996, 2013.

RAULT, T., BOUABDALLAH, A. y CHALLAL, Y. Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks*, vol. 67(0), páginas 104 – 122, 2014.

ROSENBLATT, F. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan Books Washington, 1962.

RUMELHART, D., HINTONT, G. y WILLIAMS, R. Learning representations by back-propagating errors. *Nature*, vol. 323(6088), páginas 533–536, 1986.

SALARIAN, H., CHIN, K.-W. y NAGHDY, F. Coordination in wireless sensor/actuator networks: A survey. *Journal of Parallel and Distributed Computing*, vol. 72(7), páginas 856 – 867, 2012.

SAVICH, A. W., MOUSSA, M. y AREIBI, S. The impact of arithmetic representation on implementing mlp-bp on fpgas: A study. *IEEE Transactions on Neural Networks*, vol. 18, páginas 240–252, 2007.

SAYED-MOUCHAWEH, M. y LUGHOFER, E. *Learning in non-stationary environments: Methods and Applications*. Springer, New York, 2012.

SHAIKH, R., THAKARE, V. y DHARASKAR, R. Efficient code dissemination reprogramming protocol for wsn. *International Journal of Computer and Network Security*, vol. 2(2), páginas 116–122, 2010.

SHAWASH, J. y SELVIAH, D. Real-time nonlinear parameter estimation using the levenberg-marquardt algorithm on field programmable gate arrays. *IEEE Transactions on Industrial Electronics*, vol. 60(1), páginas 170–176, 2013.

SUBIRATS, J., FRANCO, L. y JEREZ, J. C-mantec: A novel constructive neural network algorithm incorporating competition between neurons. *Neural Networks*, vol. 26, páginas 130–140, 2012.

URDA, D., CANETE, E., SUBIRATS, J. L., FRANCO, L., LLOPIS, L. y JEREZ, J. M. Energy-efficient reprogramming in wsn using constructive neural networks. *International Journal of Innovative, Computing, Information and Control*, vol. 8, páginas 7561–7578, 2012.

WANG, B. Coverage problems in sensor networks: A survey. *ACM Comput. Surv.*, vol. 43(4), páginas 32:1–32:53, 2011.

WANG, Q., ZHU, Y. y CHENG, L. Reprogramming wireless sensor networks: challenges and approaches. *Network, IEEE*, vol. 20(3), páginas 48–55, 2006.

ZHAO, W., CHELLAPPA, R., PHILLIPS, P. J. y ROSENFELD, A. Face recognition: A literature survey. *ACM Comput. Surv.*, vol. 35(4), páginas 399–458, 2003.