ISSN 1870-4069

Diseño de una red neuronal distribuida entre dispositivos Raspberry Pi conectados a Internet por medio de XMPP

Alberto Martínez Contreras[1], David Tinoco Varela[2], Fernando Gudiño Peñaloza[2]

[1] Instituto Politécnico Nacional, ESIME Zacatenco, México
[2] Universidad Nacional Autónoma de México, Facultad de Estudios Superiores Cuautitlán, Departamento de Ingeniería, ITSE, México
phama_contra26@hotmail.com, dativa19@hotmail.com, ilciarmin@gmail.com

Resumen. Actualmente se tienen diferentes esquemas de comunicación que buscan la interconexión de la mayor cantidad de dispositivos posibles, tal como el Internet de las Cosas, para así poder controlar actuadores u obtener datos desde diferentes puntos geográficos. Cuando se tienen situaciones tales como el estudio de los comportamientos ambientales, análisis de los comportamientos de consumo de individuos, o análisis de comportamientos poblacionales generales, es necesario obtener información de diferentes zonas y analizarlas en conjunto para obtener el resultado adecuado. Por este motivo, la generación de sistemas que puedan trabajar en conjunto desde distintas latitudes se vuelve un tema de estudio importante. En este trabajo se describirán diferentes modelos para la generación de una red neuronal distribuida, o segmentada, entre diferentes dispositivos conectados a Internet comunicándose mediante el protocolo de mensajería XMPP. Estos modelos logran identificar patrones de entrenamiento con datos provenientes desde distintas fuentes. El modelo utiliza tarjetas de desarrollo Raspberry Pi 3B+ para procesar y entrenar la red neuronal.

Palabras clave: internet de las cosas, redes neuronales distribuidas, cómputo distribuido, Raspberry Pi, XMPP.

Design of a Distributed Neural Network between Raspberry Pi Devices Connected to the Internet through XMPP

Abstract. Currently there are different communication schemes that seek the interconnection of any kind of devices, such as the Internet of Things, in order to control actuators or obtain data from different geographical points. When there are situations such as the study of environmental behaviors, or analysis of general population behaviors, it is necessary to obtain information from different areas and analyze such data together to obtain the correct result. For this reason, the generation of systems that can work together from different latitudes becomes an important topic of study. In this paper we will describe different models for the generation of a distributed or segmented neural network between different de-

vices connected to the Internet through the XMPP messaging protocol. Such models are able to identify training patterns with data from different sources. The model uses Raspberry Pi 3B + development boards to process and train the neural network.

Keywords: internet of things, distribuited neural networks, Raspberry Pi, XMPP.

1. Introducción

Actualmente la adquisición, procesamiento, transmisión y análisis de datos se ha vuelto de gran valor científico, tecnológico, e incluso social, debido a la cantidad de aplicaciones que los resultados de tales análisis puedan generar. El *Internet de las cosas* (IoT, por sus siglas en inglés) es un esquema de comunicaciones que busca la recolección y el análisis de la mayor cantidad posible de datos.

El IoT ha ido tomando relevancia tecnológica y científica en los últimos años, este esquema busca la conexión de todo tipo de dispositivos electrónicos a Internet, y que estos se puedan comunicar, de forma automática, más aún, este esquema busca que estos dispositivos puedan tomar decisiones o actuar de forma "inteligente". Esto con la finalidad de lograr un entorno más cómodo, se pueden ver las tecnologías, y las direcciones que se espera que este esquema llevara en el futuro en [1, 2].

Muchos dispositivos comerciales o sistemas embebidos conectados a estos esquemas informáticos adquieren y analizan información proveniente de un solo usuario o zona geográfica, pero ¿Y si los datos de entrada no estuvieran dados en la misma maquina o ubicación, sino separadas una gran distancia, como por ejemplo el nivel de tráfico en distintos puntos de una ciudad, la cantidad de personas que entran a en distintos puntos de un determinado transporte?

Si obtenemos información desde distintos puntos, se podría crear una serie de patrones de comportamiento avanzado y preciso que tendrían un sinfín de aplicaciones. Dicho de otra forma, se tienen grandes cantidades de datos provenientes de distintos lugares, pero actuando como datos de un mismo comportamiento a analizar o un mismo problema a resolver. Para estas posibilidades, es necesario generar no solo modelos de comunicación, sino también dispositivos tecnológicos que analicen datos distribuidos, es decir, que se transmitan información entre ellos y puedan analizar la información desde puntos diferentes pero que actúen como si fueran un solo dispositivo.

Es precisamente para analizar la información proveniente de diferentes zonas geográficas, entornos y usuarios, que en este trabajo se proponen diferentes modelos neuronales para adquirir, procesar y analizar datos a través de diferentes tarjetas de desarrollo tipo *Raspberry Pi*. Se han propuesto modelos basados en redes neuronales, que se distribuyen en diferentes dispositivos electrónicos para analizar de forma inteligente los datos que cada dispositivo adquiere de forma distribuida, y obtener una respuesta como si de un solo dispositivo se tratara. Con estos modelos se ha logrado identificar patrones de entrenamiento con datos provenientes desde diferentes puntos, demostrando así que se pueden identificar patrones por medio de redes neuronales seccionadas en dispositivos de baja potencia.

2. Preliminares

2.1. Raspberry Pi

Raspberry Pi es considerada una computadora de placa reducida, todos sus componentes, incluidos el procesador, la memoria y los puertos de entrada y salida, están integrados en una sola placa. Cuenta con 1GB de memoria RAM, un procesador Quad Core ARM Cortex-A53 corriendo a 1.2 Ghz, cuenta con puertos Ethernet y Wi-Fi para poder conectarse a Internet, todo esto a un costo reducido de aproximadamente $40 USD. Uno de los aspectos más importantes por los cuales se decidió utiliza esta placa de desarrollo, es que esta se ha analizado y probado como un elemento que se puede vincular rápidamente dentro de la red IoT [3]. También puede ser una herramienta de alto impacto en los procesos de enseñanza y aprendizaje [4].

Este dispositivo ha sido utilizado en distintos proyectos tecnológicos y de desarrollo, tales como sistemas de seguridad basados en la detección de rostros [5], o sistemas de alarmas que envía videos y fotografías cuando un movimiento es detectado en el inmueble a proteger [6].

Pero lo más importante para fines de este proyecto, la Raspberry Pi ha sido el centro de operaciones de muchas propuestas tecnologías que buscan la conexión de dispositivos dentro del IoT, como por ejemplo la generación de sistemas de monitoreo ambiental [7], así como el control y monitoreo de sistemas domóticos tal como se ve en [8] y [9], incluso se ha propuesto el monitoreo del entorno urbano [10].

Las características mencionadas, así como las aplicaciones dentro y fuera del IoT, hacen de Raspberry Pi, la mejor opción para el desarrollo de este proyecto.

2.2. XMPP

XMPP (*Extensible Messaging and Presence Protocol*) es un protocolo abierto que se basa en XML (*eXtensible Markup Language*), diseñado para proporcionar soluciones relacionadas con las comunicaciones en tiempo real.

Algunas de las ventajas más notables de XMPP son: descentralización, seguridad, extensibilidad, estándar abierto y gratuito. Debido a sus ventajas, el protocolo XMPP es útil en el desarrollo de interfaces hombre-máquina que pueden controlar dispositivos tecnológicos de forma remota [11].

Cada entidad XMPP requiere un identificador Jabber (JID), que contiene el nombre de usuario y el dominio en el que se conecta el recurso. JID utiliza un formato que es similar a la dirección de correo electrónico "usuario @ dominio / recurso".

XMPP es multilenguaje y multiplataforma pudiendo implementarse en Windows, en Linux, Android y casi cualquier sistema operativo, también puede implementarse en diferentes lenguajes de programación.

Adicionalmente, cuenta con algoritmos de encriptación que permite proteger los datos transmitidos.

2.3. Redes neuronales y aplicaciones distribuidas

Tratar de imitar la inteligencia de los seres vivos utilizando circuitos y maquinas sin alma es actualmente uno de los objetivos más ambiciosos de los investigadores y aunque se han logrado grandes avances e innovaciones, pensar que en un futuro próximo tendremos maquinas más inteligentes que nosotros es aún un sueño incierto.

Uno de los paradigmas de la computación que trata de asemejar el funcionamiento del cerebro de los seres vivos son las redes neuronales artificiales (RNA), que están avanzando rápidamente gracias a las nuevas tecnologías, y el desarrollo de equipos de cómputo con mayores capacidades de procesamiento y almacenamiento; aún más, se cuenta con el desarrollo de sistemas especiales para ejecutar procesos en paralelo como en el caso de las GPU's (*Graphics Processing Unit*).

Las redes neuronales son ampliamente utilizadas para la clasificación y detección de patrones, en donde se da una serie de entradas a la red neuronal y está dependiendo del entrenamiento previo arrojara un resultado, el esquema básico de una red neuronal puede verse en la figura 1.

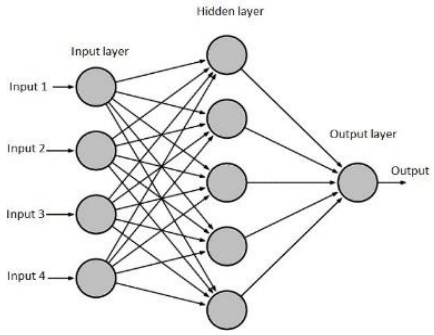


Fig. 1. Esquema básico de una red neuronal que contiene 4 entradas, una salida y una capa oculta.

Las RNA son sistemas que necesitan gran cantidad de recursos computacionales, aunque pueden ser implementadas en sistemas de baja potencia computacional tal como Raspberry [12, 13]. Sin embargo, cuando se tienen grandes cantidades de datos de entrada, un solo dispositivo puede no tener la capacidad necesaria para procesar tales cantidades de información, por este motivo, las RNA podrían ejecutarse en diferentes dispositivos de forma distribuida para aminorar la carga de cómputo en cada uno, logrando así una red distribuida de procesamiento de información. Para lograr esto de manera práctica, debemos considerar un protocolo que nos permita distribuir la información entre los diferentes nodos de la red, tal como XMPP.

En la literatura científica, existen experimentos en donde se han seccionado RNA en diferentes dispositivos electrónicos. En [14], *Krizhevsky* y otros autores mostraron la forma de entrenar una red neuronal con grandes cantidades de datos de entrada, para este fin, ellos distribuyeron la red en dos GPU.

Un ejemplo similar al anterior es el experimento llevado a cabo por *Dean* y otros autores en [15], donde se entrenó una red neuronal profunda con un billón de parámetros de entrenamiento utilizando 16000 CPU en solo 3 días.

En [16] los autores utilizan un clúster de GPU's para entrenar RNA con millones de parámetros de entrenamiento. Según los autores, se puede entrenar una red con un billón de parámetros usando solo tres GPUs en 3 días.

Los experimentos mencionados han sido realizados sobre unidades de gran nivel de procesamiento, sin embargo, en el proyecto presentado, se muestra la idea de generar RNA distribuidas sobre elementos de bajo procesamiento, en este caso la tarjeta Raspberry. En la literatura se tienen pocos experimentos relacionados a esta tarjeta de desarrollo, en [17], los autores afirman que Raspberry Pi tiene recursos limitados, por lo que la RNA se procesa y se entrena en la Nube, y la Raspberry sólo se encarga de obtener los datos relacionados al reconocimiento facial.

En [18] se genera una RNA distribuida, pero esta se distribuye en dos secciones, por un lado, se genera una RNA en una Raspberry que se entrena para casos prioritarios, y por otro lado se genera una red neuronal en la nube para casos más específicos, en este esquema, las RNA son entrenadas de forma independiente.

Caso similar sucede en [19], donde se genera una RNA distribuida también dividida en dos secciones, la red comienza a procesarse en el dispositivo restringido como la Raspberry y solo se basa en la parte remota cuando la parte local no proporciona un resultado lo suficientemente preciso.

Como se puede ver en los experimentos mencionados, solo se han desarrollado casos particulares en donde la Raspberry solo ha sido utilizada para un entrenamiento parcial, y trabaja de forma independiente al elemento de mayor procesamiento. Sin embargo, en la presente propuesta, se definen modelos en los cuales las tarjetas de desarrollo trabajan conjuntamente para solucionar un mismo problema. A conocimiento de los autores, no existe otro experimento que utilice estas placas de desarrollo para generar un procesamiento neuronal distribuido.

3. Implementación

En esta sección se presentarán los modelos propuestos para el procesamiento de los datos mediante una red neuronal artificial distribuida entre placas tipo Raspberry, así como los experimentos realizados para evaluar su operación.

3.1. Modelos de redes neuronales distribuidas

La idea principal para este proyecto es proponer un modelo para implementar una red neuronal distribuida a través de Internet, que pueda analizar datos que son recolectados por diferentes dispositivos desde diferentes posiciones geográficas. Actualmente los modelos existentes se basan en arquitecturas centralizadas o descentralizadas (Figura 2-a y 2-b), en el que los dispositivos periféricos envían los datos a uno o varios servidores y estos se encargan de aplicar los algoritmos necesarios para el análisis de los datos. En este trabajo se propone un modelo que no dependerá de servidores para su funcionamiento, sino en la cooperación e interacción de todos los nodos en la red.

El modelo que se pondrá a prueba en este trabajo es distribuir una red neuronal entre diferentes dispositivos físicos separados entre sí y conectados a través de Internet, como se muestra en la figura 2-c. Algunas de las ventajas de este modelo son las siguientes:

a) Escalabilidad. Se pueden añadir dispositivos a la red incrementando su alcance y potencia.
b) Confiabilidad. Al estar distribuida la carga de trabajo entre todos los dispositivos, la falla o desconexión de unos pocos no afecta a las demás y el sistema sobrevive como un todo.
c) Economía. Es mucho más barato añadir dispositivos clientes que servidores.
d) Independencia. Cada dispositivo tiene un sistema de procesamiento independiente, pero entre todos realizan acciones en conjunto.
e) Recursos compartidos. Se satisfacen las necesidades de muchos usuarios a la vez al compartir la información y los recursos de cada dispositivo.

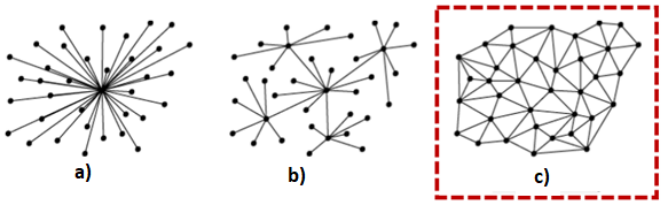


Fig. 2. Distintos tipos de arquitecturas de conexión: a) centralizada, b) descentralizada y c) distribuida.

Actualmente el modelo distribuido está tomando mucha importancia en campos como el IoT.

Para este tipo de distribución se proponen tres modelos de procesamiento de información, cada modelo tiene un flujo de información diferente. A continuación, se describen los tres modelos y la forma en la que las RNA son distribuidas entre los dispositivos electrónicos. Hay que recordar que se considera que cada dispositivo se encuentra en puntos geográficos diferentes, y obtienen la información de elementos sensoriales independientes entre sí.

Modelo 1. Cada dispositivo tendrá pocas neuronas y solo en conjunto se formará la red. Cuenta con una corrección de errores colectiva entre todos los dispositivos, el intercambio de información es masivo. Todos los dispositivos pueden tener entradas de datos y recolectar información de otros dispositivos, pero solo los finales tendrán una salida válida para la red. Para el entrenamiento de la red se intercambia información sobre los pesos de los enlaces, justo como si fuera una red neuronal clásica. Este modelo puede apreciarse en la figura 3.

Modelo 2. Cada dispositivo tendrá una red neuronal completa que se entrenará localmente, y analizará los datos recolectados en el mismo dispositivo, pero también enviará sus resultados a otros dispositivos en la red neuronal. Cuenta con corrección de errores interna, la red neuronal del dispositivo corregirá errores y también tendrá una corrección de errores colectiva pues todos los dispositivos corregirán errores entre sí. El intercambio de información es moderado, necesita pocos dispositivos para completarse. En este modelo cada dispositivo tiene un resultado particular correspondiente al análisis de sus entradas, pero la salida de la red puede ser consultada en cada dispositivo. Para la salida total, solo se podrá consultar en los dispositivos finales. En la figura 4, puede visualizarse la estructura de este modelo.

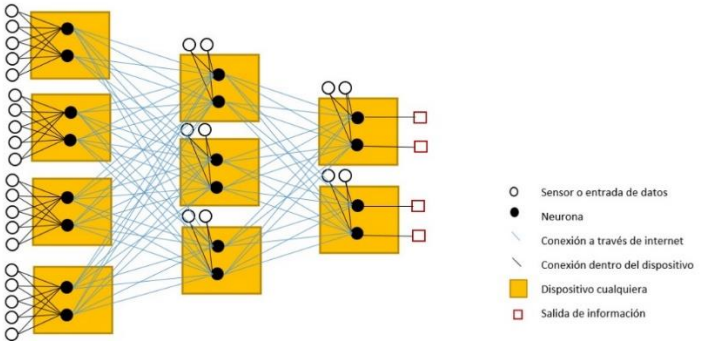


Fig. 3. Modelo 1 de procesamiento distribuido de información sobre una cantidad indefinida de dispositivos electrónicos.

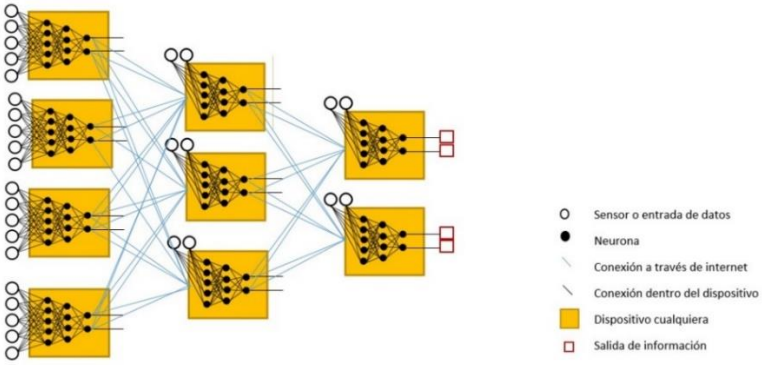


Fig. 4. Modelo 2 de procesamiento distribuido de información sobre una cantidad indefinida de dispositivos electrónicos.

Modelo 3. Cada dispositivo tendrá una red neuronal completa interna que analiza los datos de sus entradas, pero también analiza los resultados de la salida de otros dispositivos. Cuenta con corrección de errores interna, no se hace corrección de errores colectiva, cada dispositivo corrige sus propios errores. El intercambio de información es poco, basta un dispositivo para estar completa, pero entre más dispositivos, su capacidad de procesamiento será mejor. Esta red no tiene jerarquía como los modelos anteriores, solo se envía la información necesaria a los dispositivos que la soliciten no como en los modelos anteriores que la salida de unos e enviada a todos los de la siguiente capa. La arquitectura de este modelo se puede ver en la figura 5.

De los tres modelos este último sería el más adecuado de implementar en un escenario real, pues limita el flujo de datos, y presenta simplicidad en su implementación, pero los tres modelos pueden ser implementados dependiendo el escenario particular.

Sería especialmente útil en escenarios donde un nodo recolecta información de diferentes tipos, cada tipo de información puede ser útil para una red neuronal diferente, y cada una de estas redes neuronales se estarían ejecutando en el mismo dispositivo, pero en diferentes instancias, y podría o no haber comunicación entre estas, como se muestra en la figura 6.

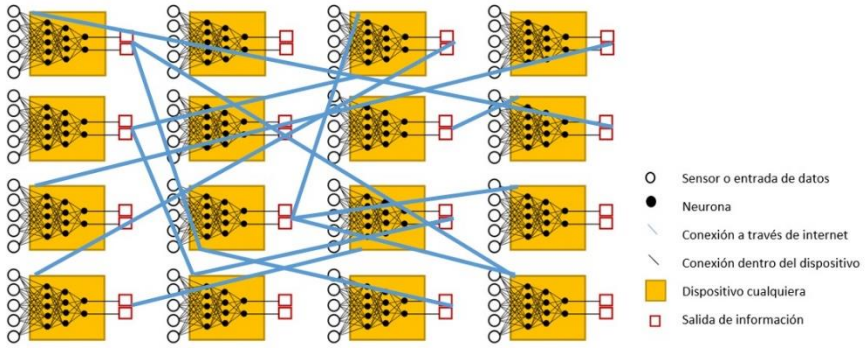


Fig. 5. Modelo 3 de procesamiento distribuido de información sobre una cantidad indefinida de dispositivos electrónicos.

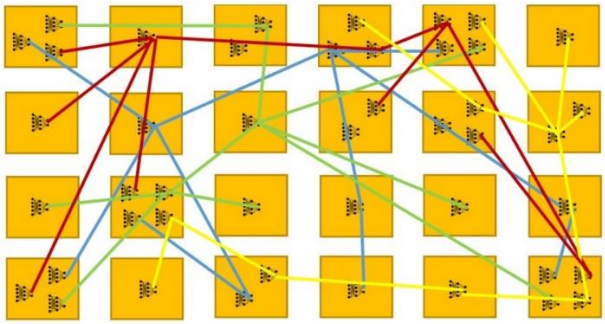


Fig. 6. Ejemplo de un dispositivo con múltiples instancias neuronales, dónde cada línea de color representa distintas redes.

En la figura 6 se puede observar que dentro del mismo dispositivo se pueden tener múltiples instancias de RNA, cada una conectándose a través de Internet con los demás dispositivos de su red, un dispositivo puede formar parte de varias redes que analicen diferentes flujos de datos cada una.

El objetivo de estos modelos es analizar información recolectada por diferentes dispositivos que estén conectados a través de Internet.

Un ejemplo es analizar datos recolectados por los *smartphones* de las personas, los datos pueden ser imágenes, datos de los sensores, datos de comportamiento, etc. Estos datos al ser analizados en una red neuronal podrían dar patrones del comportamiento de los usuarios.

Con los ejemplos anteriores queda en claro las aportaciones y el uso que se le podría dar a un sistema como el descrito en este artículo, principalmente para IoT.

3.2. Pruebas

Para realizar las pruebas de los modelos planteados se tomaron los tres escenarios descritos anteriormente y se toma en cuenta una red de pocas neuronas y pocos dispositivos para hacer la demostración, el modelo de neurona propuesto en este trabajo fue

programado en Phyton, las pruebas se llevaron a cabo en Raspberry Pi 3B+, un dispositivo móvil con *Android* y una PC con *Windows* para demostrar la compatibilidad del lenguaje y protocolo de mensajería en diferentes dispositivos. El protocolo para establecer comunicación entre los dispositivos será XMPP, pues nos permite conectar usando servidores gratuitos y en cualquier parte del mundo que se tenga una conexión a Internet. El modelo básico será el mostrado en la figura 7.

Fig. 7. Modelo básico de prueba de las modelos de redes neuronales distribuidas.

Cada modelo de prueba tendrá neuronas individuales que funcionan con el siguiente algoritmo base, las entradas serán diferentes dependiendo del modelo.

Evento 0. Creación de la neurona. Este proceso lo lleva a cabo únicamente una vez cuando se comienza la ejecución.

1. Poner con valor de cero la salida y la señal de error.
2. Inicializa con un valor aleatorio o precargado los pesos y el offset.
3. Inicializa con un valor aleatorio o precargado el factor de aprendizaje.
4. Queda a la espera de percibir un cambio en las entradas.

Evento 1. Cambio en la entrada. Este proceso se ejecuta de forma concurrente, cada vez que la neurona detecta un cambio en sus entradas.

1. Multiplica cada entrada por su respectivo peso.
2. Realiza la sumatoria de todas las entradas y suma el valor del offset.
3. Aplica la función programada, puede ser rampa, sigmoidea o escalón.
4. Cambia el valor de la salida.

Evento 2. Cambio en la señal de error. Este proceso se ejecuta de forma concurrente cada vez que la neurona detecta un cambio en la señal de error.

1. Calcula el valor esperado considerando todas las neuronas que tiene a su salida.
2. Calcula el valor de error comparando el valor esperado contra el obtenido.
3. Hace la corrección de los pesos, ya sea aumentándolos o disminuyéndolos en proporción al error que aporto cada entrada y el factor de aprendizaje.
4. Hace la corrección del offset.
5. En base al error obtenido, hace el cálculo del valor ideal en cada entrada y lo pasa a la neurona en esa entrada.

El sistema se ha probado con un patrón binario sencillo de 4x4 casillas donde cada posición podrá tener valor de (0,1). Las neuronas después de ser entrenadas deberán identificar un patrón entre los múltiples patrones que se les presentaran, se contaran las veces que el patrón es identificado correctamente y los errores que se cometan.

También se contarán los paquetes intercambiados en la red en el tiempo de entrenamiento más el tiempo de reconocimiento, el número de paquetes es un promedio no el número exacto de cada reconocimiento.

Para el porcentaje de éxito se toma una base de 100 repeticiones con patrones similares.

Modelo 1. El patrón se reconoce por medio de dos dispositivos, la mitad de un patrón es entrenada en un dispositivo, la otra mitad en un dispositivo diferente y el final en el tercer dispositivo. Los dos dispositivos que leen el patrón no pueden reconocer su parte. El resultado puede consultarse en el tercer dispositivo únicamente, el esquema de este modelo se muestra en la figura 8. En la tabla 1, pueden verse los resultados experimentales del modelo representado en la figura 8.

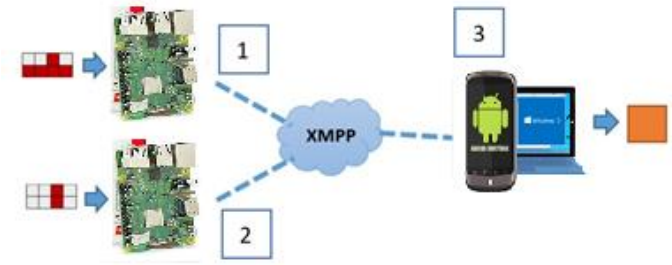
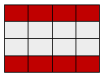
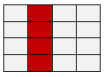
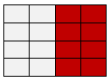
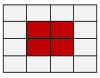
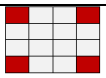
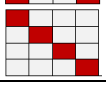


Fig. 8. Dos dispositivos reconocen la mitad de un patrón definido, solo el tercer dispositivo logrará identificar el patrón final.

En las tablas 1, 2 y 3, se muestran los patrones a entrenar, la cantidad de paquetes XMPP enviados a través de Internet entre los dispositivos durante el entrenamiento, y la cantidad de mensajes recibidos, para los modelos 1, 2 y 3, respectivamente. Para cada modelo se muestra la cantidad de veces que cada dispositivo logró identificar el patrón correcto.

Tabla 1. Comportamiento del modelo representado en la figura 8. Se puede visualizar que solo el dispositivo 3 logra identificar con diferentes porcentajes los patrones de entrenamiento.

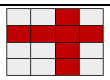
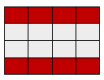
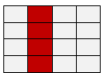
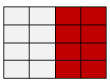
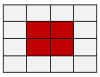
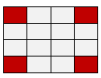
Patrón	Enviados	Recibidos	Dispositivo 1	Dispositivo 2	Dispositivo 3
	360	352	NA	NA	87%
	341	402	NA	NA	83%
	328	298	NA	NA	93%
	268	354	NA	NA	98%
	413	352	NA	NA	84%

Patrón	Enviados	Recibidos	Dispositivo 1	Dispositivo 2	Dispositivo 3
	278	368	NA	NA	72%
	401	323	NA	NA	79%

Modelo 2. El patrón se reconoce entre los tres dispositivos, cada dispositivo puede reconocer su parte del patrón, pero el resultado final se puede ver únicamente en el tercer dispositivo, la estructura de este modelo puede verse en la figura 9. En la tabla 2, pueden verse los resultados experimentales del modelo representado en la figura 9.

Fig. 9. Cada dispositivo reconoce el patrón a identificar, pero solo el dispositivo 3 lo puede mostrar.

Tabla 2. Comportamiento del modelo representado en la figura 9. Se puede visualizar que los 3 dispositivos logran identificar con diferentes porcentajes los patrones de entrenamiento.

Patrón	Enviados	Recibidos	Dispositivo 1	Dispositivo 2	Dispositivo 3
	114	119	88%	89%	85%
	121	124	87%	85%	84%
	106	129	92%	93%	90%
	114	122	99%	97%	95%
	144	131	87%	95%	86%
	160	120	74%	79%	71%
	149	128	81%	82%	77%

Modelo 3. El patron completo esta repartido entre los tres dispositivos, cada dispositivo puede reconocer su parte del patron y preguntar a los otros dispositivos por la parte faltante del patron (a los tres se les da el mismo patron al mismo tiempo), la estructura de este modelo puede verse en la figura 10. En la tabla 3, pueden verse los resultados experimentales del modelo representado en la figura 10.

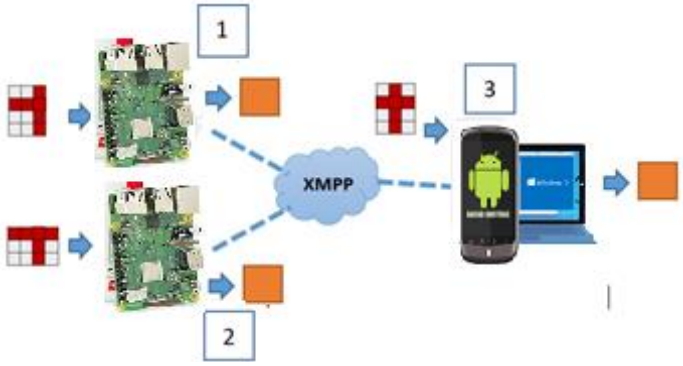
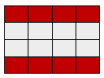
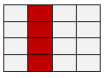
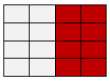
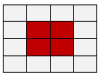
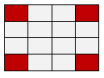


Fig 10. Cada dispositivo puede reconocer su parte del patrón e interactuar con los demás dispositivos para identificar cada sección del patrón.

Una vez analizadas las tablas 1 a 3 con las pruebas experimentales, se puede notar que los tres modelos tienen un porcentaje de reconocimiento similar, la diferencia es la complejidad de los patrones reconocidos y el tamaño de la red neuronal implementada, así como la cantidad de mensajes intercambiados por los dispositivos en el entrenamiento y reconocimiento. Las pruebas son únicamente para ver el intercambio de información entre los dispositivos, en otro entorno se implementará una red neuronal con pesos ya probados y con un porcentaje de éxito mayor.

Tabla 3. Comportamiento del modelo representado en la figura 10. Se puede visualizar que los 3 dispositivos logran identificar con diferentes porcentajes los patrones de entrenamiento.

Patrón	Enviados	Recibidos	Dispositivo 1	Dispositivo 2	Dispositivo 3
	40	12	84%	85%	86%
	33	15	83%	87%	85%
	50	8	96%	97%	96%
	38	16	95%	99%	97%
	48	8	80%	80%	82%
	43	13	73%	70%	71%

Patrón	Enviados	Recibidos	Dispositivo 1	Dispositivo 2	Dispositivo 3
	39	8	77%	79%	78%

Para contar los mensajes intercambiados únicamente se activa un contador que incrementa con cada mensaje XMPP enviado o recibido, no se hace distinción sobre el tipo de mensaje, pues se envían mensajes de configuración propios del protocolo además de los de la red neuronal, pero en un entorno real estos mensajes de configuración también se intercambiaran.

4. Conclusiones

Distribuir una red neuronal en diferentes dispositivos es rentable únicamente si la información que circulara entre los dispositivos no excede la capacidad del canal que los conecta. En el caso de Internet se debe tener en cuenta que la información de muchos otros servicios viaja por el mismo canal, por lo que la cantidad de información que puede fluir entre los dispositivos es limitada. La red neuronal debe configurarse de tal manera que los nodos solo intercambien la información más relevante, como el modelo 3 presentado en este trabajo.

Esta propuesta permite obtener información desde diferentes puntos o usuarios, y trabajar con toda esa información como si se estuviera ejecutando en un solo dispositivo, esta parte es de suma importancia, porque podemos generar redes de reconocimiento de patrones que se entrenaran con datos de todo el mundo, esto claro, permite generar una red neuronal que pueda interactuar directamente dentro del esquema del Internet de las cosas, comunicando diversos dispositivos entre sí, y procesando información de manera distribuida a través de él.

Se propone una comunicación basada en XMPP por su simplicidad de implementación y su compatibilidad con múltiples lenguajes y dispositivos, pero se puede implementar este modelo usando cualquier otro protocolo de comunicaciones instantáneo.

La ventaja que nos da el uso de XMPP, es que se vuelve una comunicación gratuita, y al tener protocolos criptográficos inherentes, permite que los datos transmitidos estén seguros.

Agradecimientos. Proyecto apoyado por PAPIIT IN 105219, PAPIME PE111519, PIAPIME 4.31.05.19, PIAPI 1824.

Referencias

1. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey, Computer networks, 54(15), 2787–2805 (2010)
2. Gubbi, M., Buyya, J., Marusic, R., Palaniswami, S.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems 29(7), 1645–1660 (2013)
3. Maksimović, M., Vujović, V., Davidović, N., Milošević, V., Perišić, B.: Raspberry Pi as Internet of Things hardware: Performances and Constraints. In: Proceedings of 1st International

Conference on Electrical, Electronic and Computing Engineering IcETRAN 2014, pp. ELI1.6.1–6 Vrnjačka Banja, Serbia (2014)
4. Zhong, X., Liang, Y.: Raspberry Pi: An Effective Vehicle in Teaching the Internet of Things in Computer Science and Engineering. Electronics, 5(3), p. 56 (2016)
5. Gunawan, T.S., Gani, M.H.H., Rahman, F.D.A., Kartiwi, M.: Development of face recognition on raspberry pi for security enhancement of smart home system. Electri. Eng. Inf.(IJEEI), 5, pp. 317–325 (2017)
6. Ansari, A.N., Sedky, M., Sharma, N., Tyagi, A.: An Internet of things approach for motion detection using Raspberry Pi. In: Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, pp. 131–134. IEEE (2015)
7. Ibrahim, A., Elgamri, M., Babiker, A., Mohamed, S.: Internet of things based smart environmental monitoring using the Raspberry-Pi computer. In: 2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC), pp. 159–164 (2015)
8. Patchava, V., Kandala, H.B., Babu, P.R.: A Smart Home Automation technique with Raspberry Pi using IoT. In: 2015 International Conference on Smart Sensors and Systems, IC-SSS 2015, pp. 1–4, IEEE (2017)
9. Sandeep, V., Gopal, K.L., Naveen, S., Amudhan, A., Kumar, L.S.: Globally accessible machine automation using Raspberry pi based on Internet of Things. In: 2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015, pp. 1144–1147, IEEE (2015)
10. Shete, R., Agrawal, S.: IoT based urban climate monitoring using Raspberry Pi. In: International Conference on Communication and Signal Processing, ICCSP 2016, pp. 2008–2012. IEEE (2016)
11. Ramírez, D., Franco, J.A.R., Tinoco Varela, E.M.: Fuzzification of facial movements to generate human-machine interfaces in order to control robots by XMPP internet protocol, In: MATEC Web of Conferences, vol. 125, p. 04020. EDP Sciences (2017)
12. Bechtel, H., McEllhiney, M.G., Kim, E., Yun, M.: Deeppicar: A low-cost deep neural network-based autonomous car. In: 2018 IEEE 24th Int. Conf. Embed. Real-Time Comput. Syst. Appl., pp. 11–21, IEEE (2018)
13. Kumar, V.S., Gogul, I., Raj, M.D., Pragadesh, S.K., Sebastin, J.S.: Smart Autonomous Gardening Rover with Plant Recognition Using Neural Networks. In: Procedia Computer Science, 93, pp. 975–981 (2016)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: ImageNet Classification with Deep Convolutional Neural Networks, pp. 1097–1105 (2012)
15. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., V Le, Q.: Large scale distributed deep networks. Advances in neural information processing systems, pp. 1223–1231 (2012)
16. Coates, A., Huval, B., Wang, T., Wu, D.J., Ng, A.Y.: Deep learning with COTS HPC systems. In: International conference on machine learning, pp. 1337–1345 (2012)
17. Sajjad, M., Nasir, M., Muhammad, K., Khan, S., Jan, Z., Sangaiah, A.K., Elhoseny, M., Baik, S.W.: Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities. Future Generation Computer Systems (2017)
18. De Coninck, E., Verbelen, T., Vankeirsbilck, B., Bohez, S., Simoens, P., Demeester, P., Dhoedt, B.: Distributed neural networks for internet of things: The big-little approach. In: International Internet of Things Summit, pp. 484–492. Springer, Cham (2015)
19. Leroux, S., Bohez, S., De Coninck, E., Verbelen, T., Vankeirsbilck, B., Simoens, P., Dhoedt, B.: The cascading neural network: building the Internet of Smart Things. Knowl. Inf. Syst. 52(3), 791–814 (2017)