# Título del Proyecto:

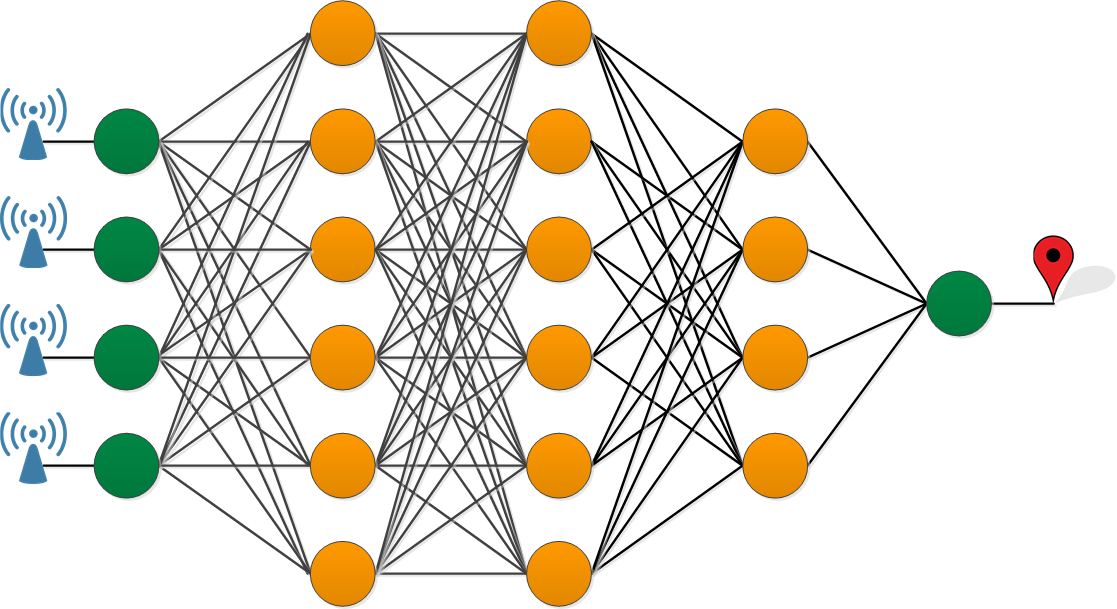
**WiFlow**

# Autor:

Fernández Ippólito, Juan Carlos

# Director:

Dr. Tomás Gironés, Jesús



WiFlow

**TESINA PARA LA**

**OBTENCIÓN DEL TÍTULO DE:**

# Máster en Desarrollo de Aplicaciones Android

**Septiembre de 2018**

Contenido

[Introducción 3](#_Toc421525768)

[Descripción del problema 3](#_Toc421525769)

[Objetivos 3](#_Toc421525770)

[Motivación 3](#_Toc421525771)

[Situación de… / Tecnologías utilizadas 3](#_Toc421525772)

[Arquitectura de la aplicación 3](#_Toc421525773)

[Esquema del diseño 3](#_Toc421525774)

[Modelo de datos 3](#_Toc421525775)

[Vistas 3](#_Toc421525776)

[Conclusiones 3](#_Toc421525777)

[Anexo fuentes 4](#_Toc421525778)

[Listado de fuentes entregadas o enlace a GitHub 4](#_Toc421525779)

[Manual de usuario 4](#_Toc421525780)

# Introducción

## Descripción del problema

Antecedentes y descripción de la problemática a resolver.

La localización derivada de señales satelitales puede volverse muy dificultosa o incluso imposible dentro de una edificación, ya sea un museo, una universidad, etcétera. Esto es debido a la atenuación que supone a esas señales, de por sí ya muy pequeñas, los materiales utilizados en la construcción. Esto es especialmente cierto si se desea discernir pocos metros y la exactitud de la posición empeora debido al efecto de esas atenuaciones.

## Objetivos

El objetivo es encontrar una solución tecnológica que pueda dar respuesta a la pregunta ¿dónde estamos?, en el contexto descripto anteriormente: localización en interiores.

## Motivación

Como proyecto final de la asignatura Android Fundamentos, dictada por la UPV a fin del año 2015, realicé un trabajo denominado WiFi Locus [[1]](#endnote-1). Este proyecto consistía en una aplicación para localización en interiores. Mediante las potencias emitidas por los puntos de acceso WiFi que llegaban al dispositivo móvil, registrados en una en una etapa inicial de entrenamiento haciendo uso de la aplicación en distintos lugares del edificio, intentaba luego calcular la posición en que se encontraba el dispositivo móvil, utilizando fundamentalmente un método de mínima diferencia de cuadrados más algunos otros algoritmos de ajuste, descriptos en la memoria de ese proyecto [[2]](#endnote-2).

¿Sería posible obtener una funcionalidad similar utilizando técnicas de inteligencia artificial, como lo son las redes neuronales con sistemas de aprendizaje automático?

Un desafío interesante, especialmente estando disponible TensorFlow, con la posibilidad entonces de utilizarlo en Android.

# Tecnologías utilizadas

Incluir la información necesaria que facilite la comprensión del proyecto. Solo se incluirá este punto para describir alguna tecnología, paradigma de diseño, plataforma, … no descrita durante el Diploma o Master. Su extensión será siempre reducida.

**Redes neuronales de aprendizaje automático**

Los sistemas de AA (**A**prendizaje **A**utomático) supervisado, aprenden cómo combinar entradas para producir predicciones útiles sobre datos nunca antes vistos, mediante una serie de ejemplos que se les suministra durante la etapa de entrenamiento.

**Tensorflow**

TensorFlow es una biblioteca de código abierto para aprendizaje automático desarrollada por Google, mediante la cual es posible construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones. Fue liberado como software de código abierto en noviembre del 2015.

El nombre TensorFlow deriva de las operaciones qué tales redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como "tensores"

**T**ensor**F**low proporciona una API de Python, así como APIs de C++ y otros lenguajes. La enorme cantidad de ejemplos y documentación de TensorFlow disponible en Python puede hacer que tal vez el lector comience utilizando este lenguaje para familiarizarse con **TF**, aunque Python parezca inicialmente poco amigable. Al menos ese fue mi caso.

**Keras**

La API de alto nivel de Keras, proporciona bloques de construcción para crear, entrenar y utilizar en predicciones a modelos de AA.

Tiene una interfaz simple optimizada para casos de uso común. Proporciona comentarios bastante claros los errores en la escritura del código y en la ejecución.

Los modelos de las redes en Keras se hacen conectando bloques de construcción configurables, con pocas restricciones.

TensorFlow tiene embebida una implementación de la especificación de la API de Keras, de manera que instalando **TF**, Keras ya está disponible.

Trabajar con Keras, en lugar de hacerlo solo con TensorFlow, hace la curva de aprendizaje del trabajo con redes neuronales menos traumática.

Como pasa en muchos otros paradigmas, al aumentar el nivel y ganar en simplicidad, el costo que hay que pagar es alguna pérdida de versatilidad. En particular en este trabajo, la dificultad más grande a superar fue que el formato modelo de red entrenado usando Keras, no puede ser utilizado directamente en Android y el proceso de obtener uno compatible con ese sistema operativo, fue casi artesanal.

**Python**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Es un lenguaje interpretado, usa tipado dinámico (no se comprueban los tipos a medida que se van haciendo los cálculos) y es multiplataforma.

Es muy usado en áreas de ciencia e ingeniería. Tanto en forma nativa como por medio de bibliotecas importadas posee gran potencia para el manejo de datos y cálculos numéricos.

# Arquitectura de la aplicación

NOTA: Los siguientes puntos son orientativos

La idea general de proyecto es colectar cuidadosamente los datos de potencia e identificación (BSSID, es decir el MAC address al que se conecta) de los puntos de acceso que son detectados en un dispositivo móvil, a lo largo y ancho de distintos espacios dentro de una edificación, tomando nota de la ubicación en que se obtuvo cada medición.

A medida que nos desplacemos en cada recinto, el dispositivo móvil registrará distintas potencias e incluso pueden cambiar la constelación de puntos de accesos recibidos: es posible que algunos se pierdan y también que aparezcan otros nuevos.

Estos datos serán utilizados para entrenar y validar el funcionamiento del modelo de la red neuronal.

Las potencias y las BSSIDs constituirán las características utilizadas como entrada en el entrenamiento de la red, mientras que las posiciones correspondientes (una para cada conjunto de potencias y BSSIDs obtenidas en un punto determinado de cada recinto, oficina o espacio) serán las etiquetas que se utilizarán para ajustar los pesos que conectan las neuronas, distribuidas en distintas capas.

## Componentes y herramientas utilizadas

Wifi Locus modificado: https://github.com/jon1721/WiFiLocusModificado.git

JADE Javascript based Database Editor: es un plugin para Chrome, disponible en Chrome Web Store, que permite editar y observar base de datos SQLite: https://chrome.google.com/webstore/detail/javascript-based-database/bponbdjkefbmgkfiiphhabghkkfocook

RebaseData: conversor en línea de SQLite a CSV: https://www.rebasedata.com/convert-sqlite-to-csv-online

Microsoft Excel

PyCharm: IDE para Python desarrollado por JetBrains. Tiene enormes ventajas para quienes han usado Android Studio, pues la GUI y su uso son muy similares. La búsqueda, descarga e instalación de paquetes está muy simplificada. La versión libre Community es totalmente funcional: https://www.jetbrains.com/pycharm/

TensorFlow: https://www.tensorflow.org/

Keras: https://keras.io/

Tensorboard: https://www.tensorflow.org/guide/summaries\_and\_tensorboard

Paso a paso

Obtención de los datos

El área de medición fue la que se muestra en el siguiente plano:

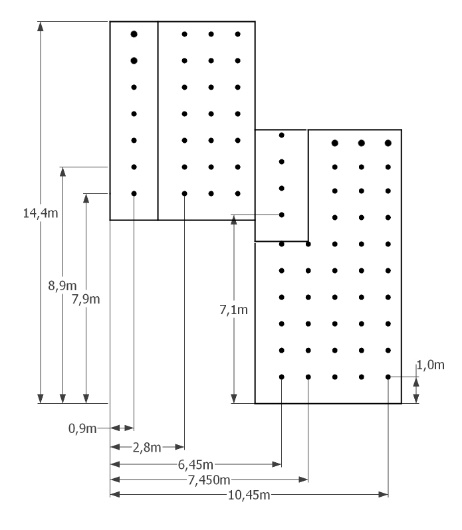


Figura 1

En el plano se observan las dimensiones en metros, el perímetro, la ubicación de las paredes y los puntos en los que realizaron las mediciones, representados por pequeños círculos negros.

La que sigue es una captura de pantalla de la aplicación WiFi Locus modificada[[3]](#endnote-3) que se utilizó para la captura de los datos:

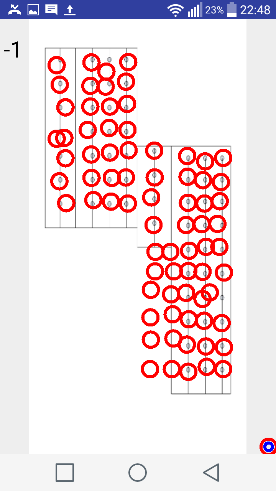


Figura 2

Para realizar cada medición, hay que situarse físicamente en el punto a medir. En esa condición, pulsar la pantalla. Eso dispara unas mediciones sucesivas, cuya cantidad debe haber sido configurada previamente en los ajustes de la aplicación, almacenándose los datos en una base de datos SQLite en la memoria no volátil del dispositivo. El punto medido en cuestión queda indicado en la pantalla mediante un círculo rojo, sumando esa indicación cada vez que se realiza una nueva medición.

En total se tomaron mediciones válidas en 74 puntos, 10 mediciones en cada punto, cada una indicada mediante su ID de registro en la base de datos, el cual con cada medición se mostraba en el ángulo superior izquierdo de la pantalla, con el objetivo de llevar un registro externo del proceso.

La siguiente imagen muestra el registro manual del ID de registro inicial de cada punto:

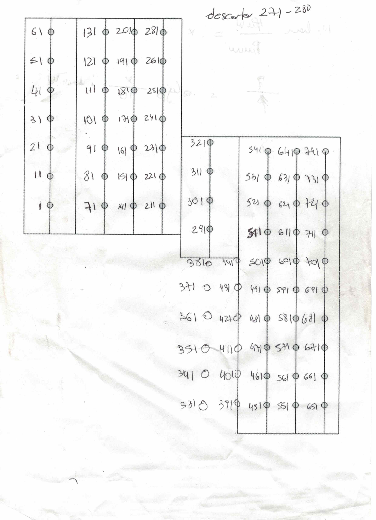


Figura 3

De esta manera, la aplicación generó el archivo Puntoswifi.db[[4]](#endnote-4), en formato SQLite.

En la imagen siguiente se observa una vista de Puntoswifi.db obtenida con la aplicación Jade:

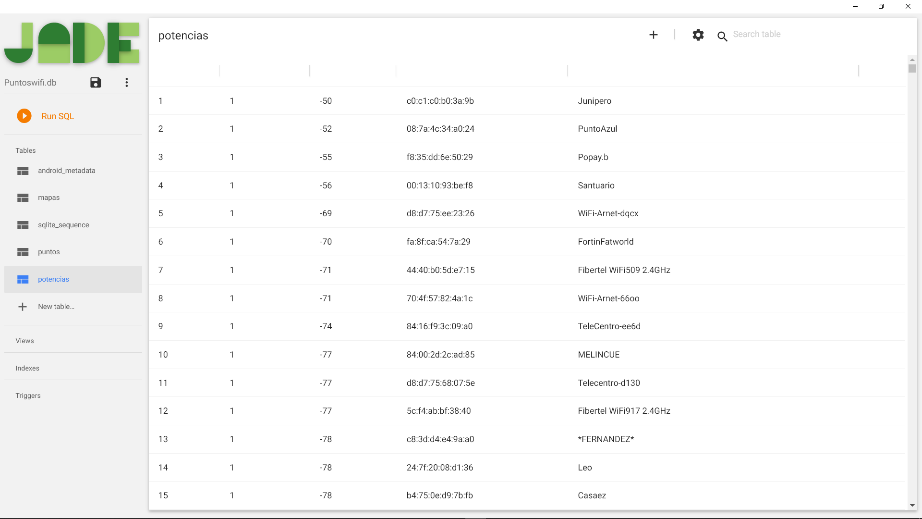


Figura 4

Aunque Python puede manejar archivos en formato SQlite, preferí exportarlo a CSV, para poder ver y analizar los resultados en Excel. Utilicé entonces RebaseData [[5]](#endnote-5)para exportar Puntoswifi.db a CSV.

De los archivos obtenidos en este proceso los importante son potencias.csv y puntos.csv, pues contienen la identificación del punto (su posición dentro del edificio) y la identificación de los puntos de acceso WiFi con el nivel de potencia medido para cada uno de ellos en ese lugar.

A partir de ahora, el proceso de analizar, filtrar y acomodar los datos para su procesamiento posterior en la red neuronal, fue llevado a cabo en el archivo wiflow.xlsx[[6]](#endnote-6)

En ese archivo Excel, la hoja Puntos y la hoja Mediciones son el resultado de importar los archivos CSV mencionados.

Como ejemplo de lo dicho, en la hoja Puntos vemos los resultados de 3 mediciones, en relación a las coordenadas de los puntos de ellas en el mapa en pantalla. Las coordenadas de pantalla x e y coinciden, pues son 3 mediciones tomadas en el mismo lugar. De hecho, cada vez que se pulsaba la pantalla la aplicación fue configurada para tomar 10 mediciones sucesivas.



Figura 5

En la hoja de cálculo se agregaron además las coordenadas físicas en metros del punto en el edificio, para tener una mejor idea si el resultado de la predicción posterior coincidía con la realidad y si su error era significativo.



Figura 6

Simultáneamente y en el mismo proceso del registro anterior, se almacenaron los valores de potencia e identificación de los puntos de acceso WiFI que eran obtenidos por el móvil:



Figura 7

En la tabla justo arriba de este texto solo se ven 8 entradas para el id de medición mediciónID con valor 1, pero si el lector observa la hoja Mediciones del archivoWiFlow.xlsx, podrá ver que en esa tabla para el medicionId igual a 1 se registraron 48 entradas, es decir que, para la primera medición en el punto de coordenadas x=0.9; y=7.9 el móvil midió y registró la potencia de 48 puntos de acceso distintos. Para el caso del medicioID igual a 2, la cantidad ya no es 48, sino 50, a pesar de que el lugar es el mismo y entre una y otra adquisición solo transcurren pocos segundos. Dos puntos de acceso “desaparecieron del radar”.

Un fenómeno casi siempre observado en lugares rodeados de puntos de acceso es que muchas de las señales de los puntos de acceso WiFi recibidas son marginales e inestables. Esto se ve claramente en el siguiente gráfico, obtenido mediante una captura de pantalla de la aplicación WiFi Analyzer[[7]](#endnote-7):

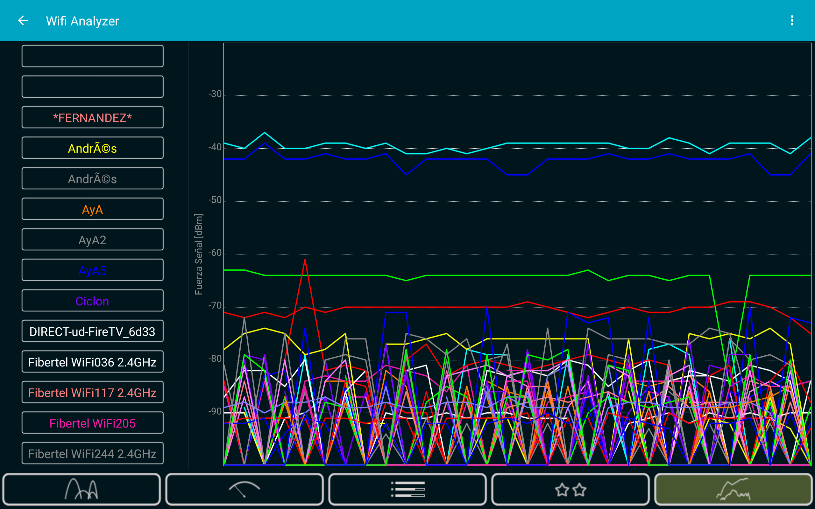


Figura 8

En el gráfico se aprecia que las señales que tiene una presencia relativamente constante están por encima de un umbral de aproximadamente -70. Aunque consideré ese umbral en este proyecto, el mismo es arbitrario y otra persona podría preferir uno un poco distinto.

Es posible que cerca de los -90 otro factor de inestabilidad sea el umbral de recepción de las primeras etapas del receptor WiFi del móvil y que además la señal se confunda con el nivel de ruido térmico o ruido de Johnson-Nyquist.

En cualquier caso y puesto que las features o características que se ingresan en la capa de entrada de la red neuronal, serán las potencias de cada punto de acceso, si consideráramos todas las señales que alguna vez aparecieron, aunque esta aparición sea esporádica, de bajo nivel e inestable, sería necesario disponer de muchas más neuronas en la capa de entrada de la red y alimentarlas con valores nulos cuando la señal no está presente. A su vez, debido a que su aparición (o no) tiene cierta característica aleatoria, sería necesario multiplicar las características de entrada en el entrenamiento de la red tantas veces como combinatorias de esta condición quieran considerarse, a los efectos de atenuar el error introducido por la ausencia absoluta de una característica que alguna vez existió.

Hecho esto con datos sintéticos, los resultados no fueron buenos y por ello preferí descartar por filtrado aquellos puntos de acceso WiFi cuyo nivel recibido fue menor que -70.

Como resultado de ese filtrado obtuve la hoja “> -71 dBm” del archivo WiFlow: en ella, observando la columna potenciaId, se aprecia que muchas mediciones fueron descartadas, pues los valores no son consecutivos (comparar la Figura 7 con la ). La tabla que sigue es una pequeña muestra del total de los resultados al aplicar el filtro.



Figura 9

Ya en esta instancia, fue posible obtener las MACs (BSSID) que aparecían, en un listado sin repetición de las mismas y asignarles un identificador de punto de acceso WiFi más cómodo (un entero consecutivo) para su procesamiento posterior en Python.



Figura 10

nota: Excel permite obtener un listado de celdas (MACs) no duplicadas en forma simple mediante Datos-Quitar duplicados

Las features o características que se ingresan en la capa de entrada de la red neuronal, serán las potencias de cada punto de acceso. Esa información será procesada también por las neuronas de la capa oculta y finalmente la red neuronal, luego del proceso de entrenamiento, terminará dando una predicción en las 2 neuronas de salida: una para la coordenada X y otra para la coordenada Y.

Es decir que habrá tantas neuronas de entrada como puntos de acceso se hayan registrado y dos neuronas de salida.

Es decir que habrá tantas neuronas de entrada como puntos de acceso y para cada medición en cada una de las estas neuronas se ingresará la potencia registrada. Una vez que la red esté entrenada, en las neuronas de salida se obtendrá la predicción que el modelo hace de la posición.

Para lograr esto, la forma en la que están ordenados ahora los datos no es la conveniente. Tal vez sería mejor tener columnas que representen a cada uno de los puntos de acceso y debajo de ellas los registros que correspondan a cada medición, con la potencia que aportó cada punto de acceso bajo la columna del mismo, en el caso de que exista una potencia, para esa medición y para el punto de acceso WiFi que indica la columna. Es dedir, algo asi:



Figura 11

En la Figura 11, para cada medición que se llevó a cabo se ven las potencias registradas para cada uno de los 20 puntos de accesos que resultaron del filtrado de superiores a -70 dB. Aquellos que no tenían indicación de potencia igual requieren un valor a ser aplicado en la neurona correspondiente de la capa de entrada de la red, por lo cual asigné arbitrariamente -100 dB, un valor suficientemente bajo y que no debe estar muy lejos de la verdad.

En el margen derecho vemos las coordenadas x e y en las que cada medición se efectuó.

## Esquema del diseño

Diagrama con los diferentes componentes del diseño y sus interrelaciones.

Justificación de las principales decisiones tomadas en el diseño.

## Modelo de datos

Esquema de la base de datos, incluyendo relaciones entre las tablas.

Listado de servicios webs.

## Vistas

Esquema que muestre las principales pantallas de la aplicación y el diagrama de navegación.

# Capítulos adicionales

Si consideras puedes poner más capítulos adicionales

# Conclusiones

Grado de cumplimiento de los objetivos planteados.

Líneas abiertas.

Consideraciones personales.

# Anexos

## Listado de fuentes entregadas / Código fuente en GitHub

## Manual de usuario

Links

Estoy preparando cierto marco teórico en la memoria del proyecto y me gustaría saber si estás de acuerdo con los siguiente:

**¿Qué es Deep Learning? (Ilustración 1)**

Dentro de la IA hay muchas cosas: razonamiento, planificación, toma de decisiones, procesos de decisión de Márkov, etc.

Una parte (no pequeña) de la IA es el aprendizaje automático. Dentro de Aprendizaje Automático estarías las Redes Bayesianas, las Máquinas Vectores de Soporte, K-medias, Aprendizaje por refuerzo,

Las Redes Neuronales constituyen un área dentro del Aprendizaje Automático

Deep Learning es un área dentro de las RedesNeuronales, que consiste en tomar las Redes Neuronales, aplicarles nuevas técnicas y organizarlas de cierta forma.

Nota: al Aprendizaje Automático, también se lo llama Machine Learning

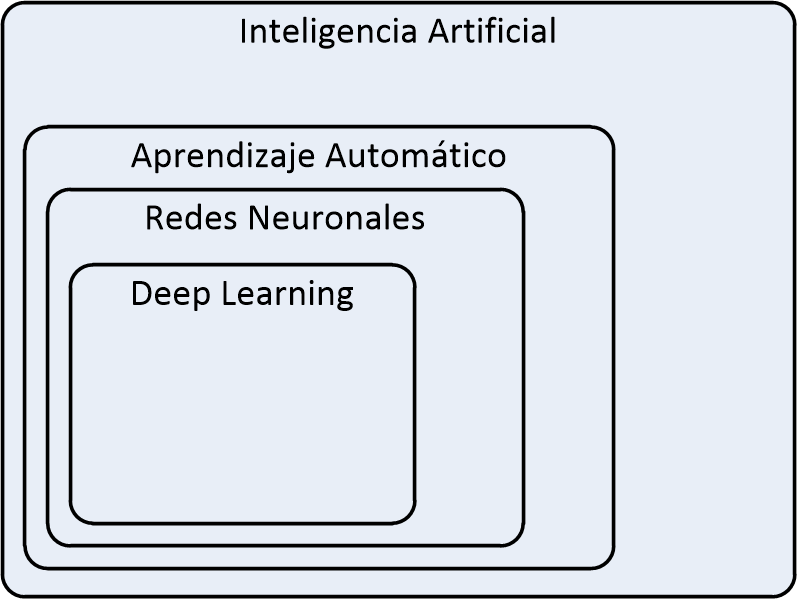


Ilustración 1

Deep Learning (Ilustración 2)

Lógica y SBC:

En Sistemas Basados en el Conocimiento y Lógica (primera columna), escribimos un programa en forma manual: estructura, arquitectura… todo. Son sistemas sin fase de entrenamiento .  
Ingresan datos y dan una salida. Si es incorrecta, hay que rehacer le programa.

Aprendizaje tradicional:

Sistemas probabilísticos, aprendizaje por refuerzo, las primeras redes neuronales. Se divide el problema, pues se diseña manualmente no todo el programa, sino ciertas propiedades. El desarrollador decide qué características son significativas. Por ejemplo, en redes bayesianas, decide cuales son los nodos y cuales son las relaciones entre esos nodos. Lo que se aprende mediante entrenamiento en este caso son las tablas de probabilidad condicional.

En la Ilistración 2, lo que lleva a cabo un humano está dibujado en fondo blanco, mientras que lo que hace la máquina tiene un fondo casi anaranjado (ni verde, ni blanco).

Aprendizaje de Representación:

Se busca que el agente inteligente aprenda las propiedades.

En Redes Neuronales, la importancia de las características la asigna la misma red. Es decir en la fase de entrenamiento usa las propiedades para ver cuanto peso le tiene que asignar a cada una.

Deep Learning:

Poniendo ciertos tipos de redes neuronales en cierto orden se observo que las primeras aprendían algunas características y la siguiente capa, basada en el aprendizaje anterior aprendía otras características más complejas.

Esta “profundidad” de capas de aprendizaje le da el nombre a Deep Learning

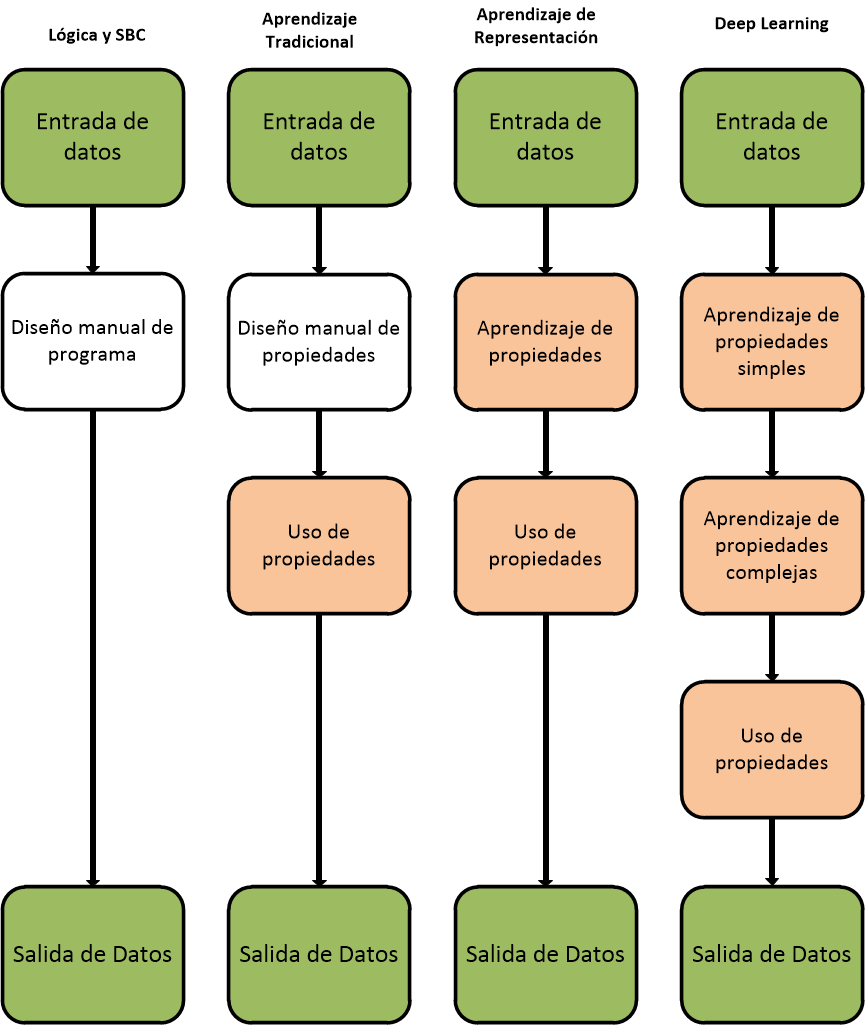


Ilustración 2

1. http://androidcurso.com/index.php/recursos/proyectos/67-proyectos-android-fundamentos/proyectos-android-fundamentos-6-ed?start=5 [↑](#endnote-ref-1)
2. http://www.dcomg.upv.es/~jtomas/android/ProyectosFundamentos2015/WifiLocus.pdf [↑](#endnote-ref-2)
3. https://github.com/jon1721/WiFiLocusModificado.git [↑](#endnote-ref-3)
4. Archivo disponible en https://github.com/jon1721/wiflow.git [↑](#endnote-ref-4)
5. https://www.rebasedata.com/convert-sqlite-to-csv-online [↑](#endnote-ref-5)
6. Archivo disponible en https://github.com/jon1721/wiflow.git [↑](#endnote-ref-6)
7. https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer.classic [↑](#endnote-ref-7)