**UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS**



**CURSO:**

**INTELIGENCIA ARTIFICIAL**

**USO DE REDES NEURONALES PARA LA IDENTIFICACIÓN DE PLANTAS**

**Presentado por:**

Alex Santiago Rojas

Paul Cueva Gonzales

Juan Francisco Valencia Martinez

Alberto Milla San Martín

Josué Espinoza Arroyo

**Profesor:**

Bugarín Peche, José Luis

**LIMA – PERÚ**

**2020**

**ÍNDICE**

[RESUMEN 3](#_Toc56463782)

[1. INTRODUCCIÓN 4](#_Toc56463783)

[2. MACHINE LEARNING 4](#_Toc56463784)

[3. REDES NEURONALES 4](#_Toc56463785)

[4. PLANTEAMIENTO DEL PROBLEMA 5](#_Toc56463786)

[5. ESTADO DEL ARTE 6](#_Toc56463787)

[5.1. Fundamentos matemáticos sobre redes neuronales 6](#_Toc56463788)

[5.2. Función XOR 7](#_Toc56463789)

[5.3. Inicialización De La Red: 8](#_Toc56463790)

[5.3.1 Propagación Hacia Adelante: 8](#_Toc56463791)

[5.3.2 Propagación hacia atrás: 10](#_Toc56463792)

[5.4. Calculando el error total: 10](#_Toc56463793)

[6. IMPLEMENTANDO ESTA TEORÍA: 15](#_Toc56463794)

[6.4. Como funciona una Red Neuronal 16](#_Toc56463795)

[6.2 Algoritmo o modelo seleccionado 19](#_Toc56463796)

[7. DESARROLLO DE LA SOLUCIÓN 19](#_Toc56463797)

[7.1. Describir los datos y la fuente 19](#_Toc56463798)

[7.2 Tecnologías y Herramientas 19](#_Toc56463799)

[7.3 Aplicación del algoritmo seleccionado 21](#_Toc56463800)

[7.4. Variantes del algoritmo 28](#_Toc56463801)

[8. Referencias Bibliográficas 30](#_Toc56463802)

# RESUMEN

La aplicación que presentamos hace uso de un algoritmo de machine learning de clasificación con aprendizaje supervisado, para la construcción de un modelo basado en redes neuronales capaz de identificar plantas ya sea con flor o sin flor mediante una imagen.

# INTRODUCCIÓN

El término de inteligencia artificial fue adoptado en 1956, pero se ha vuelto más popular hoy día gracias al incremento de volúmenes de datos, algoritmos avanzados y mejoras en el poder de cómputo y el almacenaje. Históricamente el termino de inteligencia artificial se ha plantado desde distintos puntos de vista: la capacidad de pensamiento o la habilidad de actuar inteligentemente. Centrándose en la aproximación a una idea humana de inteligencia, en la que las máquinas piensen y sean racionales. El segundo planteamiento se basa no tanto en el proceso como el resultado considerando la inteligencia artificial a la capacidad de actuar y emular que sería resultado de una acción estrictamente racional.

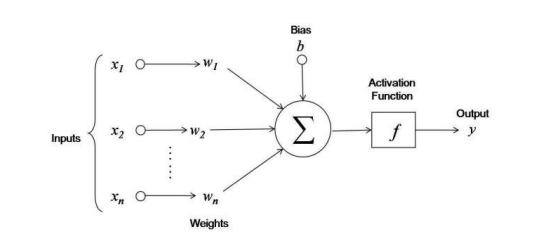
# MACHINE LEARNING

La inteligencia tiene como parte fundamental en el aprendizaje, proceso por el cual, mediante información, estudio y experiencia se logra una determinada formación. Es por ello por lo que surge la necesidad dentro del marco de la Inteligencia Artificial de dotar a los sistemas de conocimiento. De esta manera nace el término de aprendizaje automático o machine learning que será aplicado en nuestro proyecto que gracias al procesado de datos (imágenes) busca la identificación de patrones comunes que permitirán la elaboración de predicciones cada vez más perfeccionadas.

# REDES NEURONALES

Las redes neuronales artificiales o neural networks son modelos matemáticos que tratan de emular el comportamiento natural de las redes neuronales biológicas.

Se establece una red de unidades lógicas o neuronales interconectadas entre sí que procesan la información recibida y emiten un resultado ala siguiente capa determinado por una función de activación que tiene en cuenta el peso de cada entrada, dotando así de mayor importancia a conexiones entrantes concretas.

  
*Fig. 1: Esquema Neurona Artificial*

En este modelo seleccionado, la salida neuronal y vendría dada por:

Durante la fase de entrenamiento de una red neuronal los parámetros Weights y Bias son reajustados con el fin de adaptar el modelo a una tarea concreta y mejorar las predicciones. La función de activación será seleccionada de acuerdo al problema que se pretende resolver.

# PLANTEAMIENTO DEL PROBLEMA

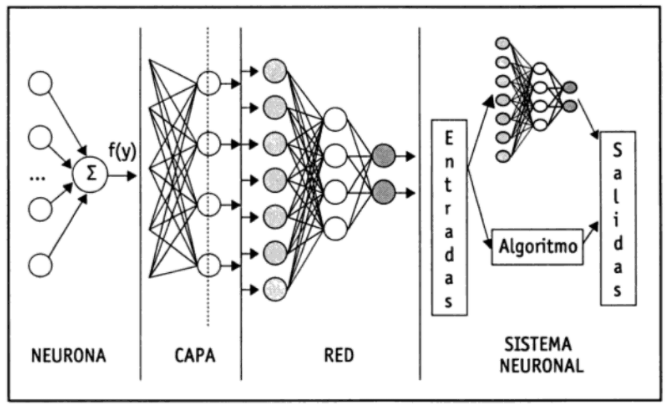
Actualmente existen muchos casos estudiados en los cuales mediante el análisis de imágenes se puede llegar a una conclusión, toma de decisiones, etc. Por ejemplo, tenemos el caso de la detección de enfermedades como es la tuberculosis que es necesario la toma de una serie de imágenes con rayos x y posterior se realiza la consulta con experto lo cual esto implicaría un ahorre de costes para el paciente para poder llegar a la detección de la enfermedad. Por otro lado, tenemos también la aplicación de la empresa Ebay que utiliza la búsqueda de artículos mediante una fotografía utilizando la cámara del dispositivo (celular). La forma que funciona el reconocimiento de imágenes, por lo general, implica la creación de una red neuronal que procesa los píxeles individuales de una imagen que es lo que hemos planteado para la presentación del proyecto en el reconocimiento de plantas mediante el uso de redes neuronales. Finalmente, es bueno indicar que esta solución es totalmente aplicable en varias situaciones o para diversas áreas de estudio, para nuestro proyecto será de total beneficio para el sector agrónomo.

# ESTADO DEL ARTE

## Fundamentos matemáticos sobre redes neuronales

Las redes neuronales tienen su fundamento en la forma biológica del cerebro. El cual compuesto por neuronas que se comunican entre sí por sinapsis. Las RNA fueron fundamentadas por Donald Hebb en 1949 quien dice que los conceptos y el conocimiento están representados por un conjunto de neuronas activas simultáneamente y que la memoria está en las conexiones entre estas neuronas. Además, sostiene que el uso de estas conexiones refuerza las mismas. Estos sistemas son de gran interés hoy en día debido a su capacidad de aprendizaje automático y la habilidad de funcionar adecuadamente aun en presencia de información inexacta, incompleta o con fallos en sus componentes. Además, la investigación de métodos de procesamiento paralelo tal cual cómo funciona el cerebro humano podría llevar a un importante intercambio de conocimiento entre la neurociencia y los modelos de RNA de la inteligencia artificial.

Cada procesador pondera las entradas que recibe, es entonces factor importante en el aprendizaje de la red la modificación de estas ponderaciones. Así, la RNA puede aprender de sus propios errores a través de un proceso inductivo que se basa en la presentación de un conjunto de patrones informativos que permiten al sistema la generalización de conceptos a partir de casos particulares.



Una red neuronal puede definirse como un grafo dirigido con las siguientes propiedades:

A cada nodo j se le asocia una variable de estado xj,

A cada conexión (i, j), entre los nodos i y j, se le asocia un peso wij ∈ R. En muchos casos a cada nodo se le asocia un umbral de disparo θj.

Para todo nodo j se define una función fj (Xi, Wij , θj), que depende del estado de todos los nodos unidos a él, de los pesos de sus conexiones y del umbral de activación para proporcionar un nuevo estado.

En cuanto a comparación entre una neurona biológica y una neurona artificial, podemos decir lo siguiente:

|  |  |
| --- | --- |
| **Neurona biológica** | **Neurona artificial** |
| Señales que llegan a las sinapsis | Entradas a la neurona |
| Carácter excitador o inhibidor de las sinapsis de entrada | Pesos de entrada |
| Estímulo total de la neurona |  |
| Activación o no de la neurona | Función de activación |
| Respuesta de la neurona | Función de salida |

**Aprendizaje supervisado**

Se caracteriza por la presencia de un agente externo (supervisor o maestro) que controla el proceso de entrenamiento, estableciendo la respuesta de debería generar la red (output del sistema) a partir de una entrada determinada. De esta forma, el supervisor comprueba la salida de la red, la compara con la salida deseada y, si existen diferencias, ajusta iterativamente los pesos hasta que su salida tiende a ser la deseada, utilizando para ello información detallada del error cometido en cada paso. Este proceso se reitera hasta que el resultado se aproxime al esperado con cierto grado de confianza.

Desde el punto de vista formal, sea E(W) la función que representa el error esperado de la red expresado en base a sus pesos sin´apticos. El aprendizaje supervisado tiene como objetivo hallar una función multivariable desconocida, f*:* R*N* → R*M* a partir de submuestras de patrones de entrada-salida (x, y), donde *x* ∈ R*N e* *y* ∈ R*M* .

El modelado de esta función se basa en la minimización iterativa de E(W) mediante algún algoritmo de aproximación.

El tipo de algoritmo de aproximación empleado permite distinguir tres tipos de aprendizaje supervisados: por corrección de error, por refuerzo o de tipo estocástico.

**Aprendizaje por corrección de error:** Constituye el tipo de aprendizaje supervisado más utilizado en la práctica. Su funcionamiento se basa en el ajuste de los pesos de las conexiones de la red a partir de la diferencia entre los valores deseados y los obtenidos por el sistema, esto es, en función del error cometido en la salida.

Una de las reglas más sencillas de aprendizaje por corrección de error es la siguiente:

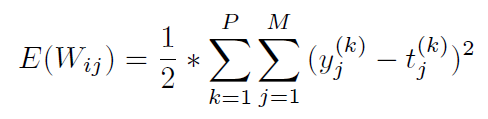
∆*wij* = *α* ∗ *xi*(*tj* − *yj* )

donde:

* + - * + ∆*wij* es la variación en el peso de la conexi´on entre las neuronas *i* y *j*,
        + *xi* es la *i*-´esima entrada a la neurona *j*-´esima,
        + *tj* es la salida deseada para la neurona *j*,
        + *yj* es la salida obtenida en la *j*-´esima neurona, y
        + *α* es el factor o taza de aprendizaje

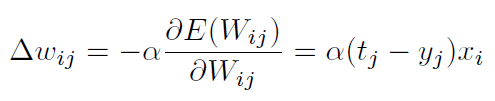
Esta regla presenta la restricción de no considerar la magnitud del error global cometido durante el proceso de aprendizaje.

Para superar esta limitación, Widrow y Hoff desarrollaron un nuevo algoritmo de aprendizaje más rápido y con mayor campo de aplicación. Se lo conoce como “Regla del error mínimo cuadrado” (“Least-Mean-Square-Error”) o “Regla de Widrow-Hoff” para las funciones de activación de tipo lineal; y con el nombre de “Regla delta” en el caso de funciones de activación de tipo sigmoideo.

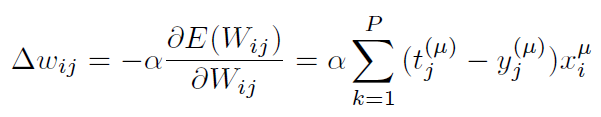
El método parte de la función de error global cometido por una red durante su enrutamiento 

siendo P es el número de patrones que debe aprender la red y M el número de neuronas de salida.

En base a la ecuación anterior, la variación relativa del error puede calcularse de la siguiente manera:

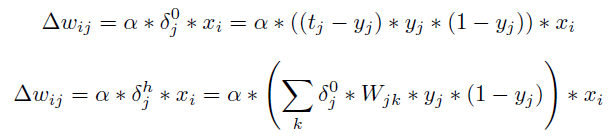


O bien de forma acumulativa para todos los patrones:



La generalización de la regla delta constituye el denominado algoritmo de retropropagación del error (‘backpropagation’).

Suponiendo funciones de activación sigmoidales, este método emplea los siguientes mecanismos de ajuste de los pesos de la red, el primero en caso de ser j una neurona de salida y el segundo en caso de ser una neurona oculta:

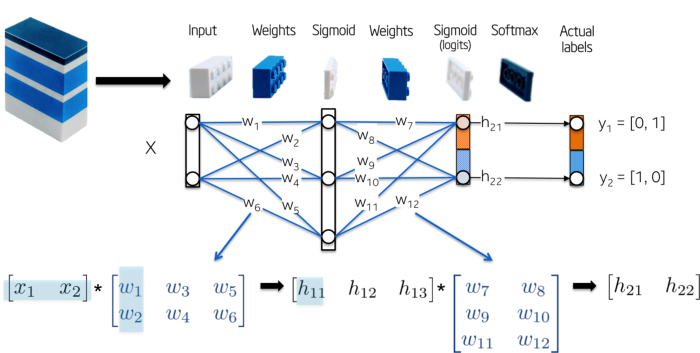


donde k hace referencia a todas las neuronas de la capa inmediatamente superior de la neurona j.

**Aprendizaje por refuerzo:** En este aprendizaje la tarea del supervisor se limita a indicar mediante una señal de refuerzo (éxito = 1, fracaso = -1) si la salida obtenida por la red se ajusta o no a la deseada. En función de ello se procede al ajuste de los pesos utilizando un mecanismo basado en probabilidades.

**Aprendizaje estocástico:** Se basa en la introducción de cambios aleatorios en los valores de los pesos de la red, evaluando su efecto a partir de la salida deseada y de una determinada distribución de probabilidad. El aprendizaje consiste en minimizar la energía del sistema a través del ajuste de los pesos: se realizan cambios aleatorios de los valores de los pesos y se determina la energía de la red tras estas modificaciones. Si la energía es menor después del cambio, se acepta la modificación, en caso contrario, la inclusión del cambio depende de la distribución de probabilidad preestablecida.

Se identificó las funciones matemáticas que sirven de soporte al algoritmo para el desarrollo del proyecto:



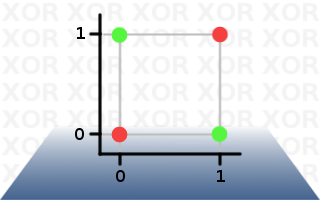
La figura de arriba muestra la matemática usada para entrenar una red neuronal. Puede encontrar interesante que una red neuronal es una pila de módulos con diferentes propósitos:

* + Entrada X alimenta la red neuronal con datos sin procesar, la cual se almacena en una matriz en la cual las observaciones con filas y las dimensiones son columnas
  + Pesos W1 proyectan entrada X a la primera capa escondida h1. Pesos W1 trabajan entonces como un kernel lineal
  + Una función Sigmoid que previene los números de la capa escondida de salir del rango 0–1. El resultado es un array activaciones neuronales h1 = Sigmoid(WX)

Hasta este punto estas operaciones solo calculan un sistema general lineal, el cual no tiene la capacidad de modelar interacciones no lineales. Esto cambia cuando ponemos otro elemento en el pila, añadiendo profundidad a la estructura modular. Mientras más profunda sea la red, más interacciones no lineales podremos aprender y problemas mas complejos podremos resolver, lo cual puede explicar en parte la popularidad de redes neuronales.

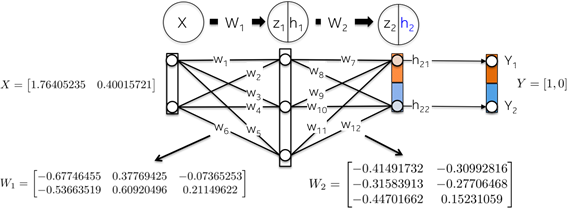
## Función XOR

La función XOR es una función binaria muy simple pero muy usada en computación.  
  
Se trata de una función f:{0,1}x{0,1}→{0,1} que vale 0 si ambos argumentos son iguales, es decir, f(1,1) = f(0,0) = 0 y vale 1 en caso contrario, es decir, f(1,0) = f(0,1) = 1  
  
Cuando tenemos un problema de clasificación real, podemos tratar de definir los espacios de entrada y las clases de salida a la medida del problema de forma que podamos construir una [función de clasificación](https://www.revolucionia.com/2017/08/Clasificadores_definicion_basica.html) que nos resuelva el problema.

En estas condiciones, es imposible reproducir la función XOR porque una sola neurona usa un hiperplano (recta en este caso) para dividir el espacio de entrada (plano en este caso) en dos mitades y no se puede hacer tal división de forma que los puntos en los que la función XOR vale 0 queden separados de los que resultan en valor 1  
  
En la ilustración adjunta queda patente la imposibilidad de separar con una recta los puntos de la región de entrada en función de su valor para la función XOR (distintos colores en la ilustración).

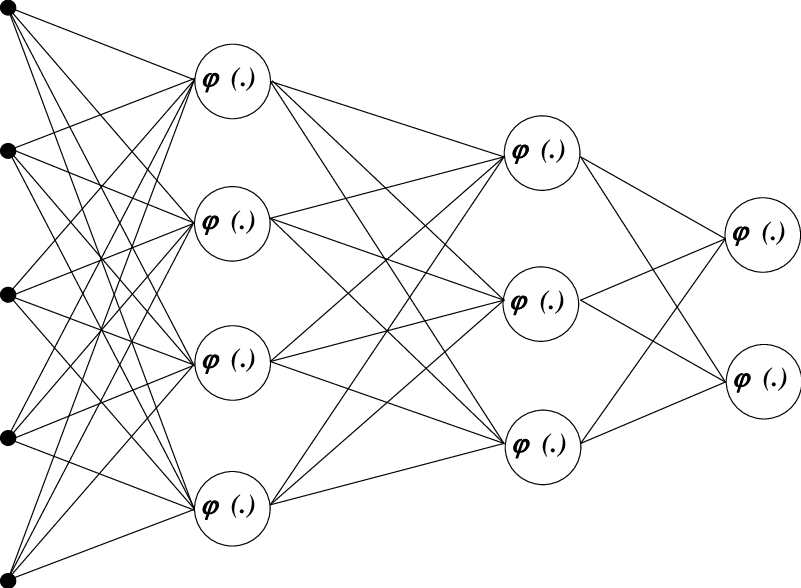
## Inicialización De La Red:

Inicialicemos \*\*los pesos de la red \*\* con valores aleatorios:



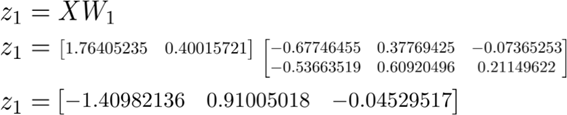
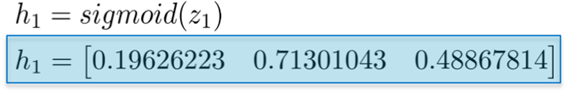
### Propagación Hacia Adelante:

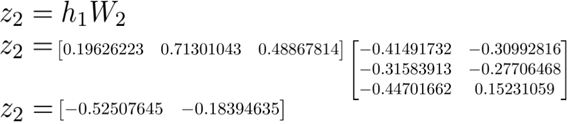
Cuando el grafo que representa a la red es acíclico, la red se denomina hacia adelante (las que trataremos en este tema).



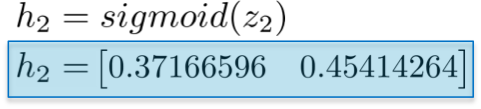
Las unidades en una red hacia adelante suelen estructurarse en capas, tal que cada capa recibe sus entradas de unidades de la capa inmediatamente anterior Capa de entrada, capas ocultas y capa de salida Hablamos entonces de redes multicapa Otras arquitecturas: redes recurrentes, en la que las unidades de salida retroalimentan a las de entrada.

* Se proyecta linealmente la entrada X usando pesos W1 a manera de kernel:

  
  
   
Se escala esta suma z1 con una función Sigmoid para obtener valores de la primera capa escondida. Note que el vector original de 2D ha sido proyectado ahora a 3D.  
   
Un proceso similar toma lugar para la segunda capa h2. Calculemos primero la suma z2 de la primera capa escondida, la cual es ahora un vector de entrada.



* Y luego calculemos su activación Sigmoid. Este vector [0.37166596 0.45414264] representa el logaritmo de la probabilidad o vector predecido calculado por la red dado los datos de entrada X.



### Propagación hacia atrás:

Es un método de cálculo del gradiente utilizado en [algoritmos](https://es.wikipedia.org/wiki/Algoritmos) de [aprendizaje supervisado](https://es.wikipedia.org/wiki/Aprendizaje_supervisado) utilizados para entrenar [redes neuronales artificiales](https://es.wikipedia.org/wiki/Redes_neuronales_artificiales). El método emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

El modelo de una neurona artificial es una imitación del proceso de una neurona biológica, tal como se ve en la figura:



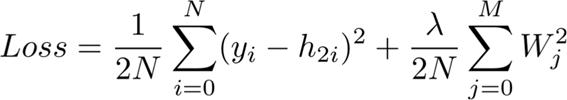
La salida de la neurona luego es modificada mediante la función de transferencia f:



## Calculando el error total:

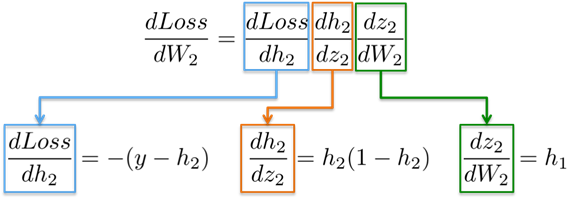
También conocido como “valor real menos predecido”, el objetivo de la función Loss es cuantificar la distancia entre el vector predecido h2 y la etiqueta real proveída por un ser humano, y.

Note que la función Loss contiene un componente de regularización que penaliza valores de los pesos muy altos a manera de una regresión L2. En otras palabras, grandes valores cuadrados de los pesos incrementaran la función Loss, una métrica de error que en realidad queremos reducir.



**dLoss/dW2:**

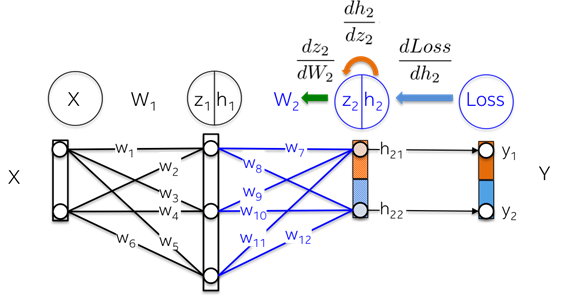
La regla de la cadena dice que podemos descomponer el cálculo de gradientes de una red neuronal en funciones diferenciables:



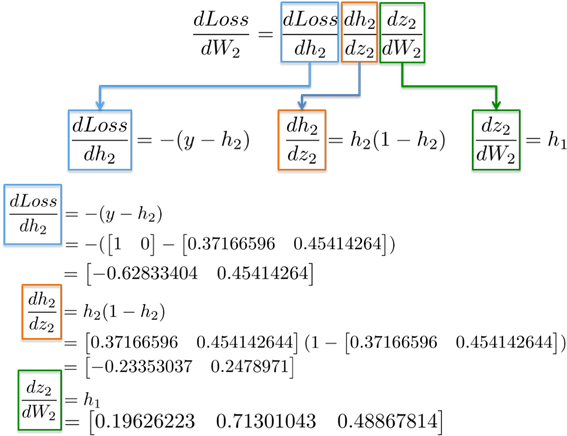
Aquí están las definiciones de funciones usadas arriba y sus primeras derivadas:

FunciónPrimera derivadaLoss = (y-h2)²dLoss/dW2 = -(y-h2)h2 = Sigmoid(z2)dh2/dz2 = h2(1-h2)z2 = h1W2dz2/dW2 = h1z2 = h1W2dz2/dh1 = W2

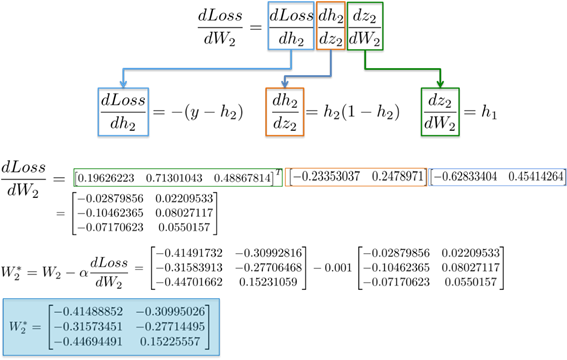
Mas visualmente, queremos actualizar los pesos W2 (en azul) en la figura de abajo. Para eso necesitamos calcular tres derivadas parciales a lo largo de la cadena.



Insertando esos valores esas derivadas parciales nos permite calcular gradientes con respecto a los pesos W2 como sigue.

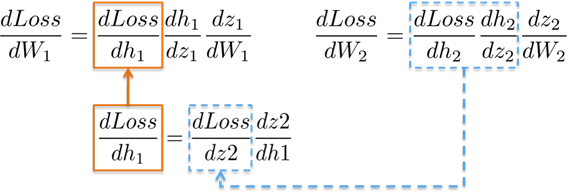


El resultado es una matriz de 3x2 llamada dLoss/dW2, la cual actualizara los valores originales de W2 en una dirección que minimiza la función Loss.

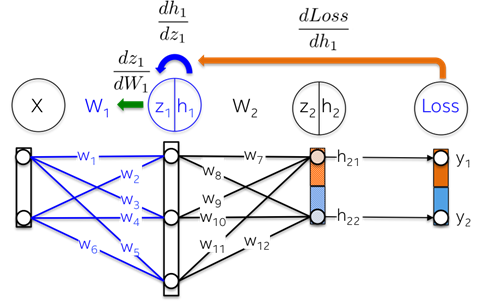


1. dLoss/dW1:

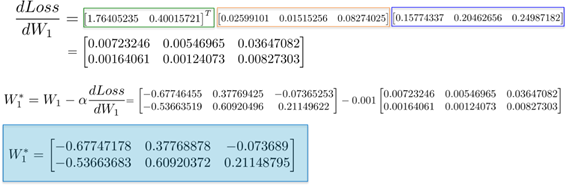
Calculando la regla de la cadena para actualizar los pesos de la primera capa escondida W1 exhibe la posibilidad de reutilizar cálculos existentes.



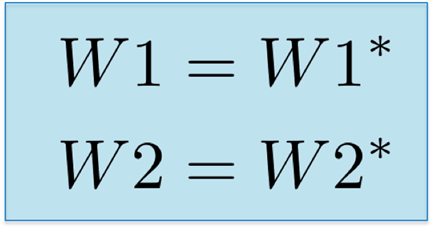
Más visualmente, el camino desde la capa de salida hasta los pesos W1 toca derivadas parciales ya calculadas en capas más superiores.



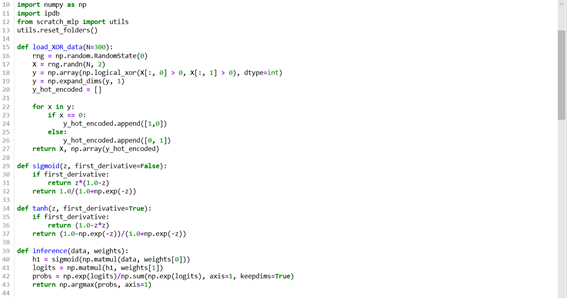
Ubicando todas las derivadas juntas, podemos ejecutar la regla de la cadena de nuevo para actualizar los pesos de la capa escondida W1:

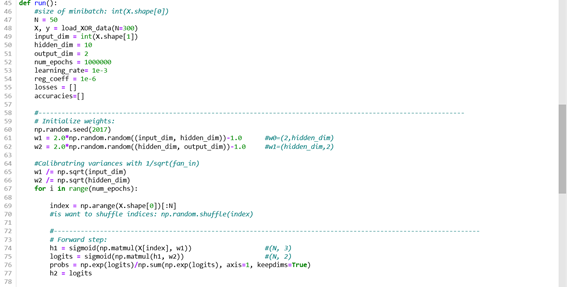


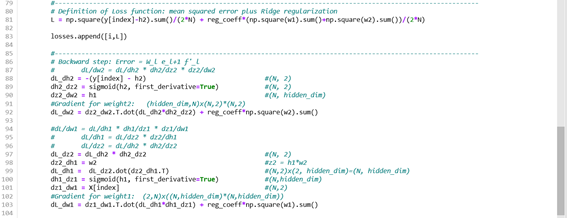
Finalmente, asignamos los nuevos valores de los pesos y hemos completado una iteración del entrenamiento de la red neuronal.

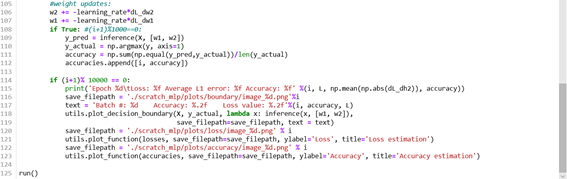


# IMPLEMENTANDO ESTA TEORÍA:



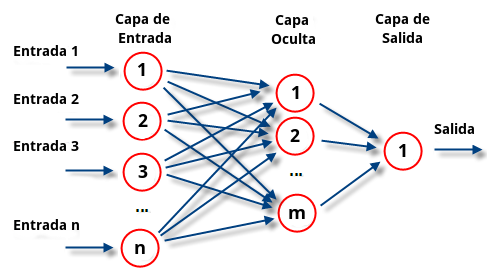


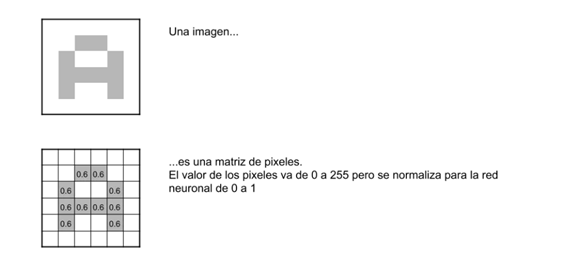


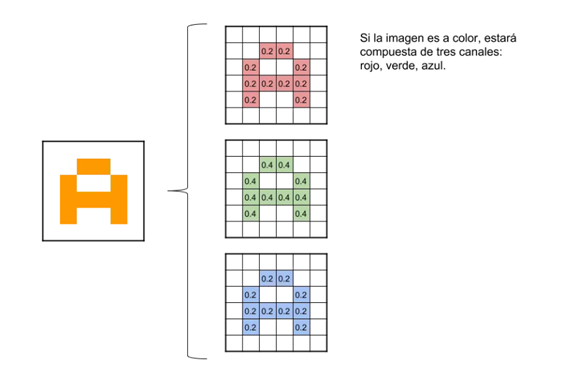


## Como funciona una Red Neuronal

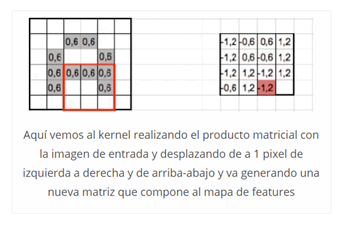
El funcionamiento de las redes se asemeja al del cerebro humano. Las redes reciben una serie de valores de entrada y cada una de estas entradas llega a un nodo llamado neurona. Las neuronas de la red están a su vez agrupadas en capas que forman la red neuronal. Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red. Este funcionamiento puede observarse de forma esquemática en la siguiente imagen.



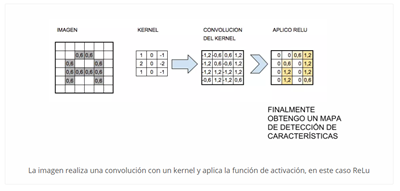
La Red Neuronal deberá aprender por sí sola a través de un entrenamiento (se necesitan miles de imágenes) a reconocer una diversidad de objetos dentro de imágenes con la finalidad de poder clasificarlas en base a las etiquetas indicadas.  
   
Para comenzar, la red toma como entrada los pixeles de una imagen. Si tenemos una imagen con apenas 28×28 pixeles de alto y ancho, eso equivale a 784 neuronas. Y eso es si sólo tenemos 1 color (escala de grises). Si tuviéramos una imagen a color, necesitaríamos 3 canales (red, green, blue) y entonces usaríamos 28x28x3 = 2352 neuronas de entrada. Esa es nuestra capa de entrada. Para continuar con el ejemplo, supondremos que utilizamos la imagen con 1 sólo color.



En realidad, no aplicaremos 1 sólo kernel, sino que tendremos muchos kernel (su conjunto se llama filtros). Por ejemplo, en esta primera convolución podríamos tener 32 filtros, con lo cual realmente obtendremos 32 matrices de salida (este conjunto se conoce como feature mapping), cada una de 28x28x1 dando un total del 25.088 neuronas para nuestra PRIMER CAPA OCULTA de neuronas.



A medida que vamos desplazando el kernel y vamos obteniendo una “nueva imagen” filtrada por el kernel. En esta primera convolución y siguiendo con el ejemplo anterior, es como si obtuviéramos 32 “imágenes filtradas nuevas”. Estas imágenes nuevas lo que están “dibujando” son ciertas características de la imagen original. Esto ayudará en el futuro a poder distinguir un objeto de otro (por ej. gato o perro).



## Algoritmo o modelo seleccionado

El algoritmo seleccionado es crear una Red Neuronal Convencional la cual es capaz de analizar imágenes para obtener un resultado final.

Dentro de las ventajas de este algoritmo podemos mencionar las siguientes

* Detección y categorización de objetos
* Clasificación y segmentación de imágenes
* Pueden sintetizar algoritmos a través de un proceso de aprendizaje.

# DESARROLLO DE LA SOLUCIÓN

## Describir los datos y la fuente

* **Código Fuente:**
* **Lenguaje de Programación**: Python 3.0
* **Fuente:** <https://www.aprendemachinelearning.com/clasificacion-de-imagenes-en-python/>
* **Datos:** Imágenes de plantas
* **Dataset:** Las imágenes de cada planta se obtuvieron a partir de YouTube usando FFMpeg. Las imágenes están en tamaño diminuto de 28x24 pixeles en color y son un total de 14607

## Tecnologías y Herramientas

* **Dataset**: Parte indispensable del aprendizaje automático es la información que se pretende analizar pues en ella se basa todo el desarrollo posterior. Para nuestro caso de estudio se ha optado por el reconocimiento de plantas, enfocada en machine learning.
* **Jupyter Notebook**: Es una aplicación cliente-servidor lanzada en 2015 por la organización sin ánimo de lucro Proyecto Jupyter. Permite crear y compartir documentos weben formato JSON que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida. Estas celdas albergan, entre otras cosas, código, texto (en formato Markdown), fórmulas matemáticas y ecuaciones, o también contenido multimedia.

El programa se ejecuta desde la **aplicación web cliente que funciona en cualquier navegador estándar.**El requisito previo es instalar y ejecutar en el sistema el servidor Jupyter Notebook. Los documentos creados en Jupyter pueden exportarse, entre otros formatos, **a HTML, PDF, Markdown o Python**y también pueden compartirse con otros usuarios por correo electrónico, utilizando Dropbox o GitHub o mediante el visor integrado de Jupyter Notebook.

* **TensorFlow:** Desarrollada por Google con el fin de satisfacer sus necesidades en el ámbito del machine learning. TensorFlow es una biblioteca para cálculos numéricos mediante el uso de diagramas de flujos de datos. Publicada bajo licencia de código abierto en 2015 se ha situado desde entonces como uno de los referentes en el desarrollo de sistemas de aprendizaje profundo y redes neuronales.

TensorFlow deriva de las operaciones que tales redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como "tensores". En una primera versión los computos de TensorFlow se expresaban como [stateful](https://es.wikipedia.org/wiki/Estado_(inform%C3%A1tica)" \o "Estado (informática)) [dataflow](https://es.wikipedia.org/w/index.php?title=Dataflow_programming&action=edit&redlink=1" \o "Dataflow programming (aún no redactado)) [graphs](https://es.wikipedia.org/wiki/Grafo_dirigido" \o "Grafo dirigido) , sin embargo, con la evolución de la biblioteca la forma de codificar estas redes se ha vuelto más imperativa. En junio de 2016, [Jeff Dean](https://es.wikipedia.org/w/index.php?title=Jeff_Dean_(computer_scientist)&action=edit&redlink=1) de Google declaró que 1,500 repositorios en [GitHub](https://es.wikipedia.org/wiki/GitHub) mencionaron TensorFlow, de los cuales solo 5 eran de Google.

* **Keras:** Keras nace con la flexibilidad y usabilidad en mente, se trata de una biblioteca para redes neuronales de alto nivel desarrollada para Python que busca simplificar en gran medida su desarrollo. Permite ejecutarse sobre bibliotecas como TensorFlow, Theano o CNTK como backend, entendiéndose más como una interfaz más que como un framework. En el 2017 Keras fue integrado en el código fuente de TensorFlow permitiendo su desarrollo con un nivel mayor de abstracción.

Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de [Aprendizaje Profundo](https://es.wikipedia.org/w/index.php?title=Aprendizaje_Profundo&action=edit&redlink=1). Sus fuertes se centran en ser amigable para el usuario, modular y extensible.

Inicialmente fue desarrollada como parte de los esfuerzos de investigación del proyecto [ONEIROS](https://es.wikipedia.org/w/index.php?title=ONEIROS&action=edit&redlink=1) (Open-ended Neuro-Electronic Intelligent Robot Operating System). Su autor principal y mantenedor ha sido el ingeniero de Google [François Chollet](https://es.wikipedia.org/w/index.php?title=Fran%C3%A7ois_Chollet&action=edit&redlink=1).

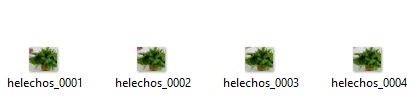
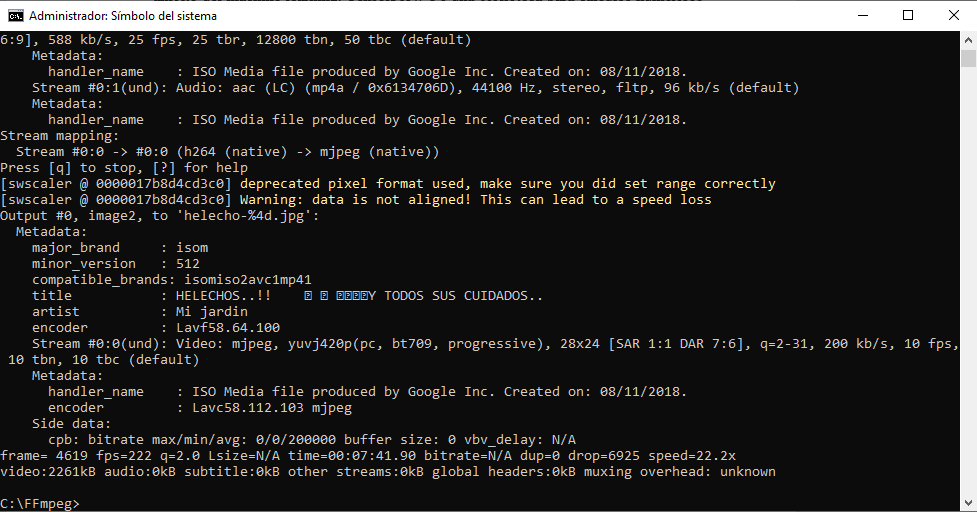
**Nota:**

Para la data de las imágenes de plantas se ha considerado instalar el software “FFMPEG”

Se ejecutó el siguiente comando para cada video de cada planta que fue obtenido por medio de YouTube.



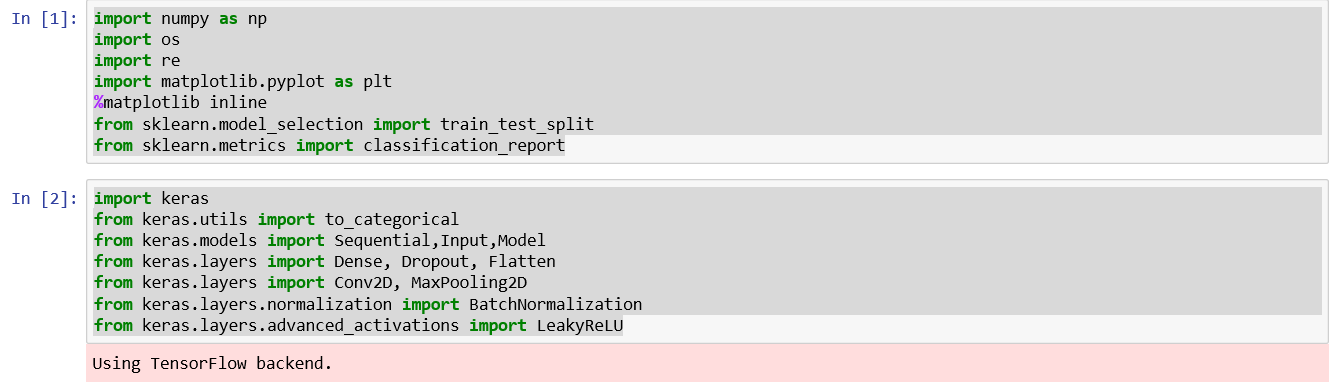
Se obtiene el siguiente resultado

## Aplicación del algoritmo seleccionado

Ejecutaremos el algoritmo a través de la suite de Anaconda con Jupyter.

Lo primero que debemos hacer es importar las librerías necesarias para la ejecución del algoritmo de Machine Learning “Keras” que nos ayudara a simplificar el desarrollo del proyecto.

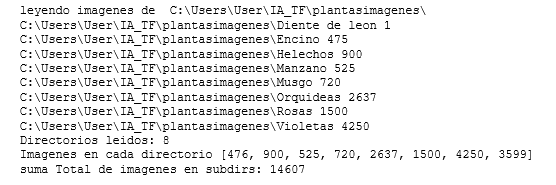


Lo que realizamos en las líneas de código descritas en la imagen, es buscar dentro de la raíz donde se encuentra ubicada nuestra carpeta **“plantasimagenes”** para cargar las imágenes en memoria. La carpeta **plantasimagenes** cuenta con “7” carpetas por cada planta seleccionado para este ejercicio y en cada una de ellas más de 500 imágenes para realizar el reconocimiento y pruebas de las mismas.



Este proceso cargará a memoria en un array las 77mil imágenes, por lo que puede tomar varios minutos y consumirá algo de memoria RAM del ordenador.

Como resultado del algoritmo donde realiza la búsqueda de carpetas de plantas encontró las siguientes carpetas: Diente de león, encino, helechos, manzano, musgo, orquídeas, rosa y violetas, cargó un total de 14607 archivos en memoria para los propósitos y pruebas que se realizarán tal y como se detalla en el punto anterior.



En este código crearemos las etiquetas en , le daremos el valor de 0 a 7 a cada planta El algoritmo de clasificación seleccionado trabaja sobre el aprendizaje supervisado, en la última línea de código “In[6]” se crean las etiquetas para cada carpeta de plantas. En total como verán tenemos 8 etiquetas (8 carpetas de plantas). Como sabemos en el lenguaje Python el iniciador del array recorrido es 0.

Luego convertimos las etiquetas y las imágenes en numpy array con 

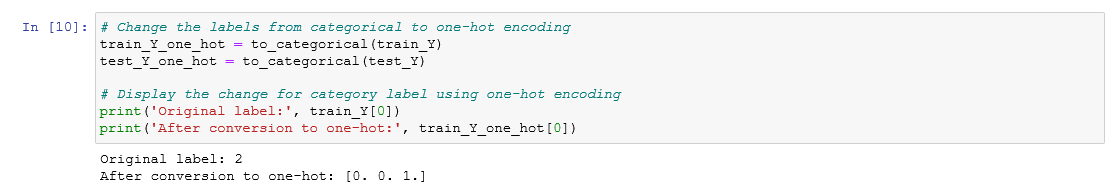


En estas líneas de código se realiza la creación de entrenamiento y testeo “Training y Testing”, dividido como está establecido en un 80-20.

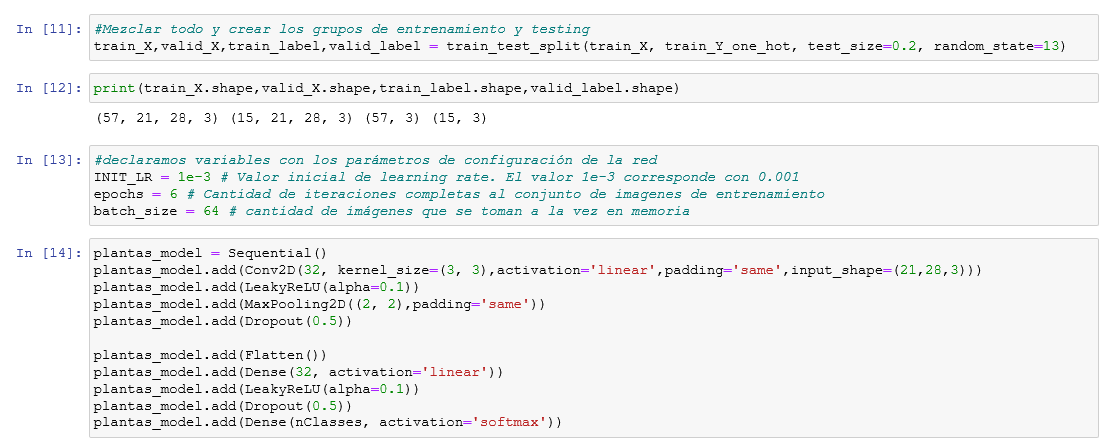




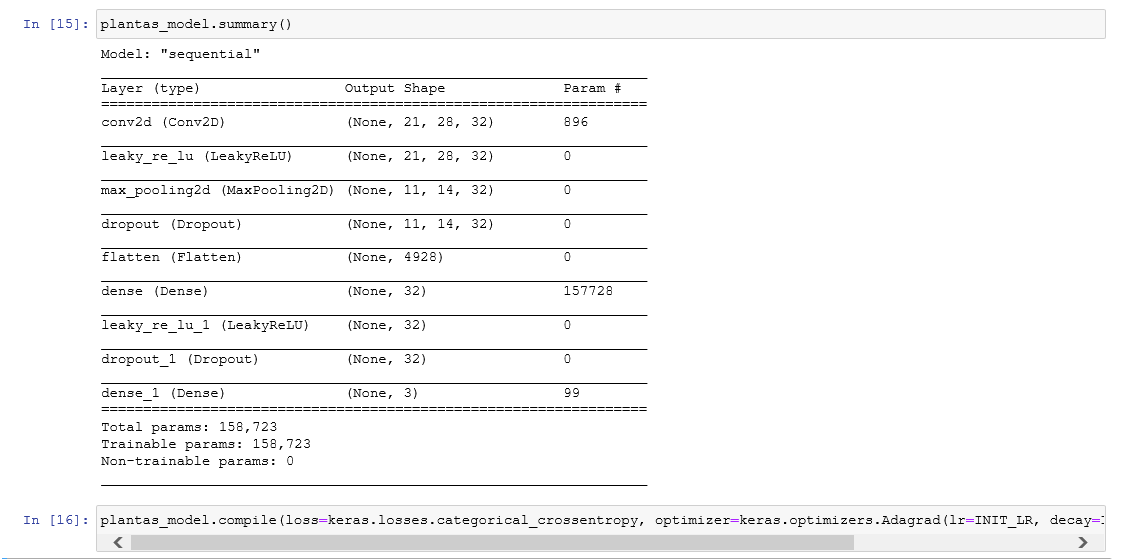
Realizamos el procesamiento de las imágenes.

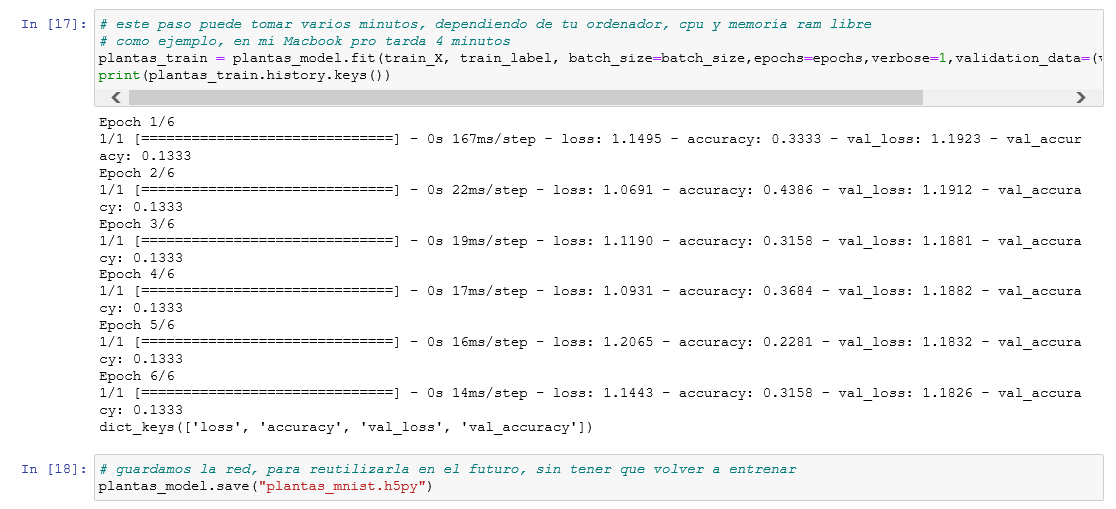


Se realiza la ejecución de One – Hot Encoding para la red. Esta técnica descrita se refiere básicamente a convertir las etiquetas antes establecidas, es decir; dientes de león = 1 a una salida de tipo 01100011 ya que es la manera en cómo funcionan mejor las redes neuronales para su clasificación.

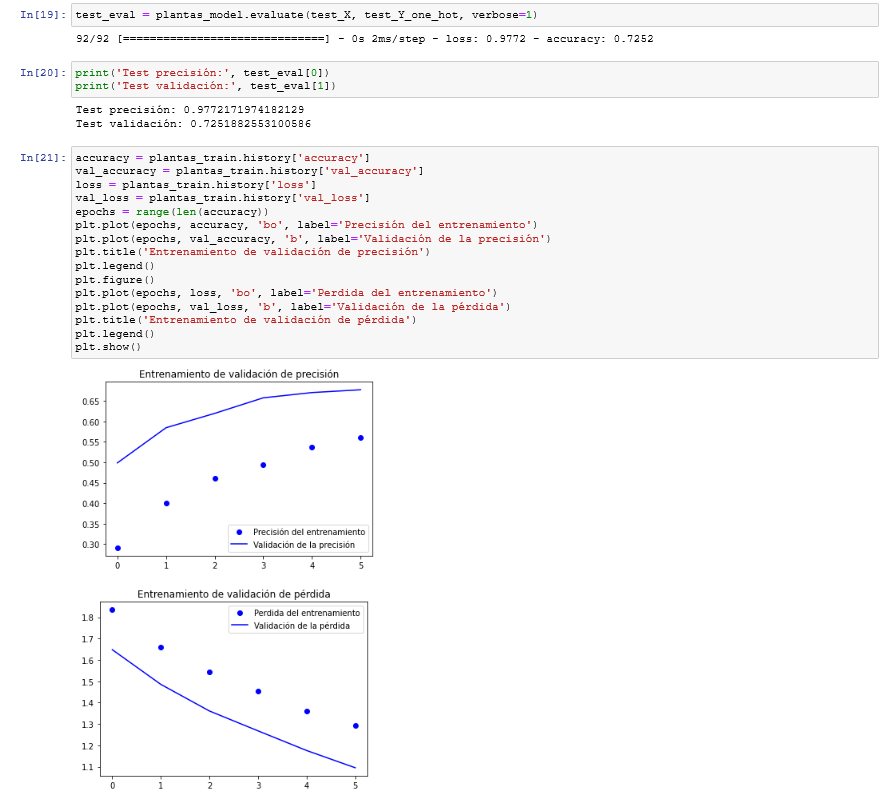
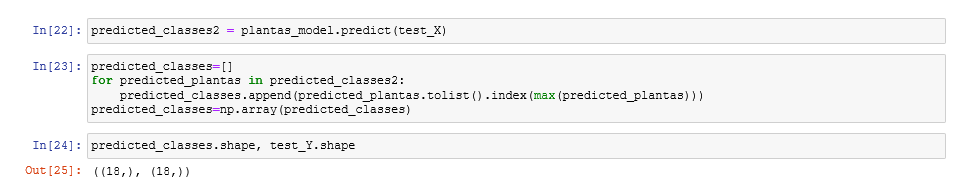


Se realizará la creación de la red neuronal. Se creará una primera capa de neuronas de 2 dimensiones donde entrarán las imágenes 24x28x32, se aplicará 32 filtros de tamaño 3x3 que detectan las características de la imagen. La función “leaky\_re\_lu\_1” tiene la labor de realizar la activación de las neuronas. Se finaliza la capa de salida con 3 neuronas con activación Softmax para que se corresponda con el Hot Encoding especificado líneas anteriores



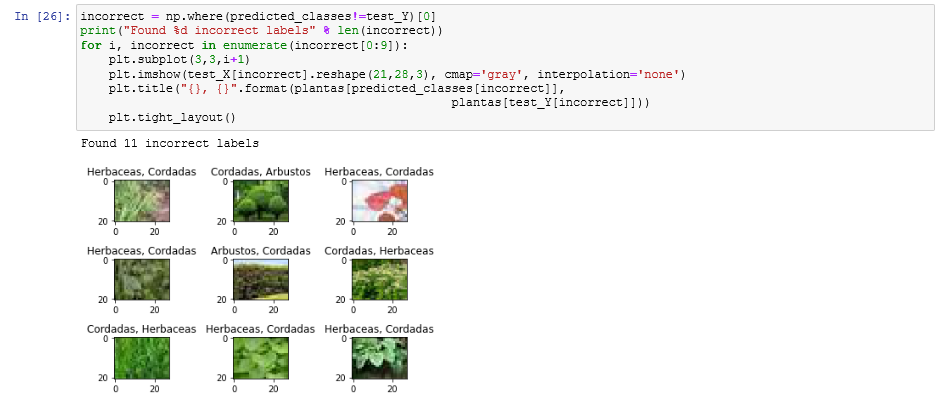


Se realiza el entrenamiento del modelo para el reconocimiento de imágenes.

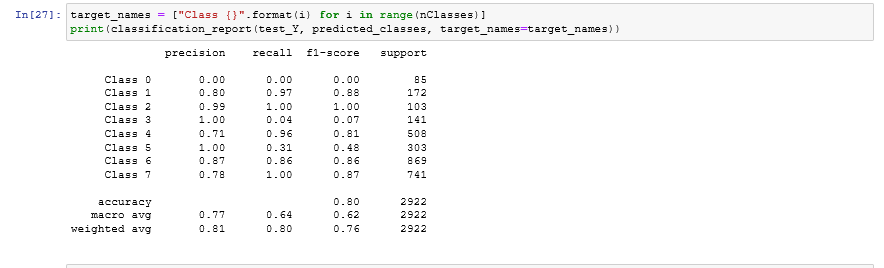
 

Tras 6 iteraciones completas al set de entrenamiento podemos notar que logramos un valor de precisión de 97% y en el set de validación de 72%





Como podemos observar, el algoritmo una precisión de 81% reconociendo las imágenes de deportes. Podríamos argumentar que a más cantidad de iteraciones de entrenamiento el algoritmo alcanza un mayor nivel de precisión.

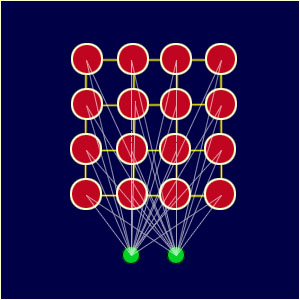


## Variantes del algoritmo

* **Mapas de Konohen – Redes Neuronales Autoorganizativas**

Los mapas Auto-Organizados son un tipo de red neuronal entrenada como aprendizaje no supervisado, de forma que se modifican repetidamente los pesos de dicha red en respuesta a patrones de activación hasta que una configuración final queda desarrollada.

El modelo, creado por Teuvo Kohonen en 1982, consiste en establecer una correspondencia entre los datos de entrada y un espacio bidimensional, creando mapas topológicos, de manera que datos similares entre si activen neuronas en zonas próximas. Con ello, se pueden clasificar datos para los que no se conoce a priori por ningún tipo de ordenación.

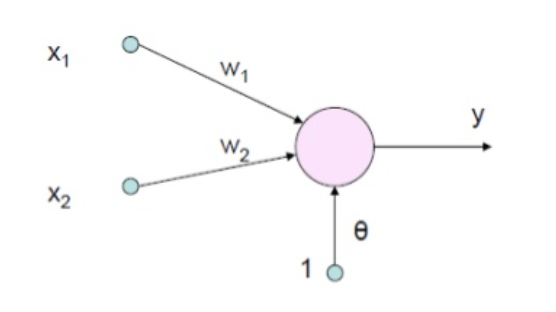
  
Fig.2. Estructura de dos capas en SOM

Las redes de Konohen son redes bicapas, es decir, de dos capas tal y como se muestra en la Fig. 2. La capa de entrada formada por N neuronas (una neurona por cada dato de entrada), que se encarga de recibir y transmitir a la capa de salida la información procedente del exterior, y la capa de salida formada por M neuronas, que es la encargada de procesar la información, crear patrones e identificar las posibles categorías. Los mapas de Kohonen nos servirán para reducción de dimensionalidad, visualización y clustering.

* **Perceptrón Simple**

El perceptrón simple es un modelo unidireccional, consta de una red con una capa de salida de n neuronas y otra capa de salida de m neuronas también utiliza señales binarias.

Las neuronas de entrada son discretas, la función de activación de las neuronas de la capa de salida es de tipo escalón además es un dispositivo entrenable que puede determinar automáticamente los pesos sinápticos que clasifican un conjunto de patrones etiquetados.



El perceptrón es un clasificador que asigna a un vector de N valores un valor binario, usando transformación no lineal. Así cada vector pertenece a una de las particiones que crea el perceptrón. Asimismo, el perceptrón es una máquina de computación universal y posee la capacidad de expresividad equivalente ala lógica binaria ya que se puede realizar la creación de un perceptrón que tenga el mismo comportamiento que una función booleana NAND a partir de esta función se puede crear cualquier otra función booleana.

Características:

* Es un sistema capaz de realizar tareas de clasificación de forma automática.
* A partir de un número de ejemplos etiquetados, el sistema determinada la ecuación de plano discriminante.
* Puede usarse como neurona dentro de otro tipo de red de neurona artificial.
* Aprendizaje supervisado y conexiones en un solo sentido.
* Es reconocido por su capacidad de aprender a reconocer patrones.
* Es un modelo unidireccional compuesto por dos capas de neuronas
* Con respecto al entrenamiento las neuronas de entrada son discretas y la función de activación de las de salida son de tipo escalón.

# Referencias Bibliográficas

* Los perceptrones y la función XOR. (2017, 4 octubre). revolucionia. https://www.revolucionia.com/2017/10/Perceptrones\_funcion\_XOR.html
* Florez, O. U. (2018, 19 junio). Un LEGO a la vez: Explicando la Matemática de Como las Redes Neuronales Aprenden. Medium. https://planetachatbot.com/un-lego-a-la-vez-explicando-la-matem%C3%A1tica-de-como-las-redes-neuronales-aprenden-9f5e21239f2d
* U. (2017, 12 mayo). Redes de propagación hacia atrás. redesneuronalescipanuevos. http://redesneuronalescipanuevos.blogspot.com/p/redes-de-propagacion-hacia-atras.html
* N. (2020a, junio 25). Clasificación de Imágenes en Python. Aprende Machine Learning. https://www.aprendemachinelearning.com/clasificacion-de-imagenes-en-python/
* N. (2020b, junio 25). ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. Aprende Machine Learning. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/
* Mapas de Kohonen. Redes Neuronales Autorganizativas. java. (s. f.). redes-neuronales. <http://www.redes-neuronales.com.es/tutorial-redes-neuronales/redes-neuronales-autoorganizadas-mapas-de-kohonen.htm>
* Flórez, R. y Fernández J. (2008). Las Redes Neuronales Artificiales:Fundamentos Teóricos y Aplicaciones Prácticas. La Coruña, España: Editorial Netbiblo