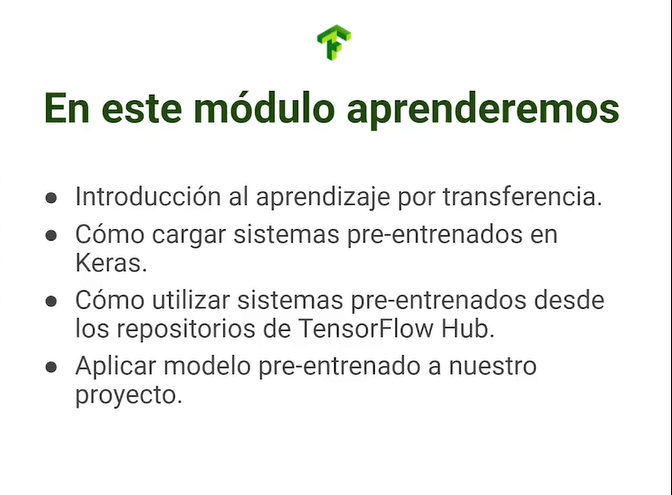
El transfer learning es una técnica valiosísima que nos permite aprovechar modelos ya entrenados por otros devs para usarlos en nuestros datasets.

A través de este módulo comprenderemos qué es el aprendizaje por transferencia, cómo cargar sistemas pre-entrenados, cómo consumirlos desde fuentes de terceros y cómo aplicarlos a nuestros proyectos.



**¿Cómo funciona el aprendizaje por transferencia?**

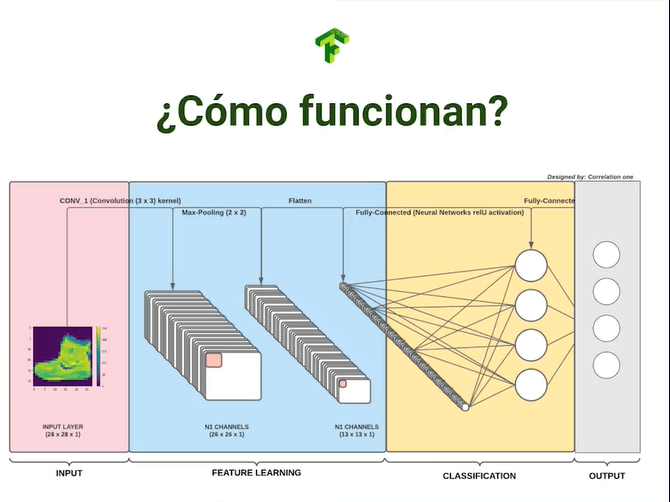
Para entender cómo funciona el transfer learning vamos a hacer una analogía: cuando eras niño aprendiste a manejar bicicleta, durante ese aprendizaje entendiste los conceptos de equilibrio, fuerza, velocidad y demás. En el futuro, cuando aprendas a manejar moto, podrás trasladar gran parte de los conceptos que ya aprendiste para obtener ventaja en este nuevo aprendizaje.

A nivel de máquina el transfer learning se hace a través de las features o características del modelo que entrenaste originalmente. Supón que tienes un detector de manzanas y ahora quieres detectar otras frutas (naranjas, piñas, etc).

No será necesario volver a entrenar un modelo desde el inicio dado que ya posees una configuración que detecta formas y colores, bastaría con hacer algunas iteraciones sobre este modelo principal con los nuevos datos y obtendrás un modelo igual de funcional.

En la red puedes encontrar decenas de configuraciones que han sido entrenadas por meses de la mano de grandes protagonistas de la investigación del deep learning.

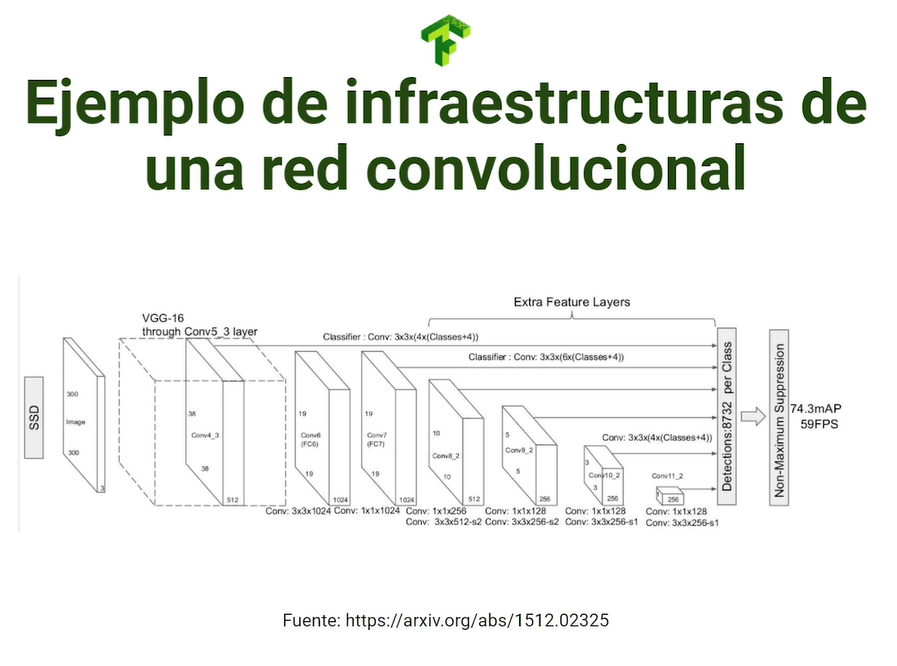
El proceso de configuración será el de eliminar la capa final de la red que vamos a aprovechar (la capa de predicciones original) y la sustituiremos por nuestra propia configuración de salida.



**Usando una red ya entrenada**

Antes de usar un modelo pre-entrenado es fundamental entender su arquitectura.

La arquitectua MobileNet V2 fue diseñada para detección de objetos en dispositivos embebidos y móviles, su entrada es una imágen de 300x300 pixeles y a través de una serie de capas convolucionales con max pooling se adquieren todas las features a clasificar con un sistema de redes neuronales. Si quisieramos usarla, bastaría con eliminar la última capa y personalizarla a nuestra necesidad.



Para esta ocasión cargaremos el modelo Inception versión 3 (otra arquitectura altamente conocida de redes convolucionales). Importaremos nuestras dependencias de Keras y cargaremos la configuración desde la locación donde estén guardadas en disco, crearemos un modelo secuencial y lo inyectaremos desde la primer capa (nota que no se incluye la capa de salida).

Se añade nuestra nueva capa de salida y se configura al modelo como no entrenable.

from tensorflow.keras.layers import Dense

from tensorflow.keras.Model import Sequential

from tensorflow.keras.applications.inception\_v3 import InceptionV3

## URL incepction model

weights\_file = "/tmp/inception\_v3\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5"

new\_model = Sequential()

new\_model.add(InceptionV3(include\_top = False,

weights = weights\_file))

new\_model.add(Dense(num\_classes, activation = "softmax"))

new\_model.layers[0].trainable = False

Con esto podemos aprovechar cientos de modelos ya entrenados por otros IA devs.

**Contribución creada por** Sebastián Franco Gómez.