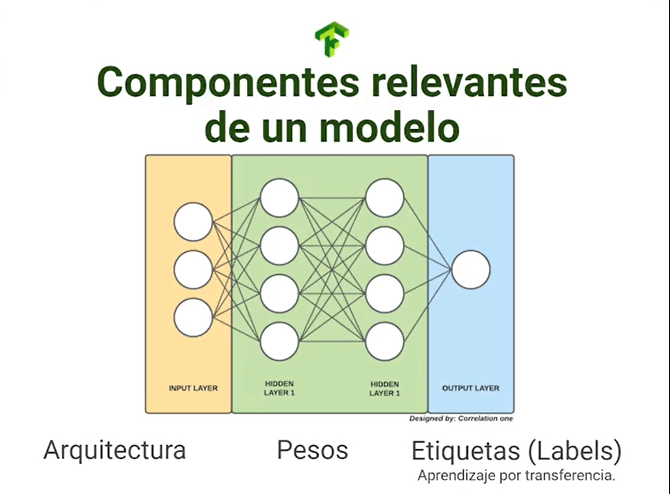
Aunque logres entrenar la mejor arquitectura de todas y encontrar los mejores parámetros posibles, si cierras tu notebook, pierdes todo el progreso. Es de vitar importancia conocer cómo cargar y descargar nuestros modelos.

Los modelos tienen 3 componentes principales: La arquitectura (que define la cantidad de capas, neuronas y entradas de la red), los pesos (que son los valores que se entrenan a la red) y las etiquetas (estas se usan especialmente en transfer learning para dar contexto al modelo).



Indaguemos en el código sobre cómo cargar y descargar modelos.

**Cargando y descargando arquitecturas sin pesos**

Puedes usar la arquitectura de un modelo para basarte a la hora de entrenar otros modelos, esto no traerá los pesos, por lo que no será útil para realizar predicciones.

Con el método get\_config de tus modelos puedes adquirir un JSON completo con la información de la arquitectura de tu red.

python config\_dict = hypermodel.get\_config()

Para cargar un modelo con base en esta configuración basará con usar el método from\_config de los modelos secuenciales de Keras enviando como parámetro el JSON de configuración.

python model\_same\_config = tf.keras.Sequential.from\_config(config\_dict)

Si resumimos el nuevo modelo nos encontraremos con las mismas dimensiones que el original.

python model\_same\_config.summary()

**Descargando arquitecturas con pesos**

Para guardar arquitecturas con pesos es necesario usar el callback de ModelCheckpoint que nos permitirá guardar en disco el modelo con sus pesos.

python from tensorflow.keras.callbacks import ModelCheckpoint

Crearemos un modelo vacío que guardaremos después.

python model\_weight = get\_model() model\_weight.compile(optimizer = "adam", loss = "categorical\_crossentropy", metrics = ["accuracy"])

Crearemos la configuración para nuestro callback. Definimos el path donde se guardará el modelo, cada cuando se guardará, si solo guardará los pesos y el output de texto a recibir.

```python checkpoint\_path = "model\_checkpoints/checkpoint"

checkpoint\_weight = ModelCheckpoint( filepath = checkpoint\_path, frecuency = "epoch", save\_weight\_only = True, verbose = 1 ) ```

Entrenaremos el modelo sin olvidar agregar el callback.

python history\_weight = model\_weight.fit( train\_generator, epochs = 20, callbacks = [checkpoint\_weight], validation\_data = validation\_generator )

Con esto hemos guardado un historial entero de nuestro modelo, puedes revisarlo en el directorio model\_checkpoints/checkpoint.

Si deseas guardar manualmente los pesos de tu red lo puedes haces con el método save indicando el directorio de salida. Esta manera únicamente guardará la última iteración, por lo que si por alguna razón la red sufrió un daño en esta etapa, no podrás revertirlo (a comparación del callback que guarda el historial entero).

python model\_weight.save("model\_manual/my\_model")

**Cargando arquitectura con pesos**

Crearemos un nuevo modelo hueco con la misma estructura que el original.

python model\_weights2 = get\_model() model\_weights2.compile(optimizer = "adam", loss = "categorical\_crossentropy", metrics = ["accuracy"])

Para cargar el modelo desde disco nos basta con usar el método load\_weights indicando la locación a cargar.

python model\_weights2.load\_weights(checkpoint\_path)

Si evaluamos el desempeño del modelo original y el cargado nos daremos cuenta que son literalmente el mismo, reafirmando que el proceso de carga fue correcto.

```python model\_weights2.evaluate(test\_generator) 57/57 [==============================] - 3s 41ms/step - loss: 0.9109 - accuracy: 0.8494 [0.9108972549438477, 0.8494144082069397]

model\_weight.evaluate(test\_generator) 57/57 [==============================] - 2s 31ms/step - loss: 0.9109 - accuracy: 0.8494 [0.9108973145484924, 0.8494144082069397] ```

Ya conoces cómo cargar y descargar arquitecturas huecas o con pesos (en historial o de una sola época), ahora indagaremos sobre las mejores decisiones a la hora de cargar modelos.

**Contribución creada por** Sebastián Franco Gómez.

En la documentación de Tensorflow nos dan otra opcion para guardar modelos:

## Guardar el Modelo

model.save(‘path\_to\_my\_model.h5’)

## Recrea exactamente el mismo modelo solo desde el archivo

new\_model = keras.models.load\_model(‘path\_to\_my\_model.h5’)