## Summary

This guide shows how to get TensorFlowJS Model and go about converting the model trained with TensorFlowJS to TensorFlow Lite FlatBuffers.

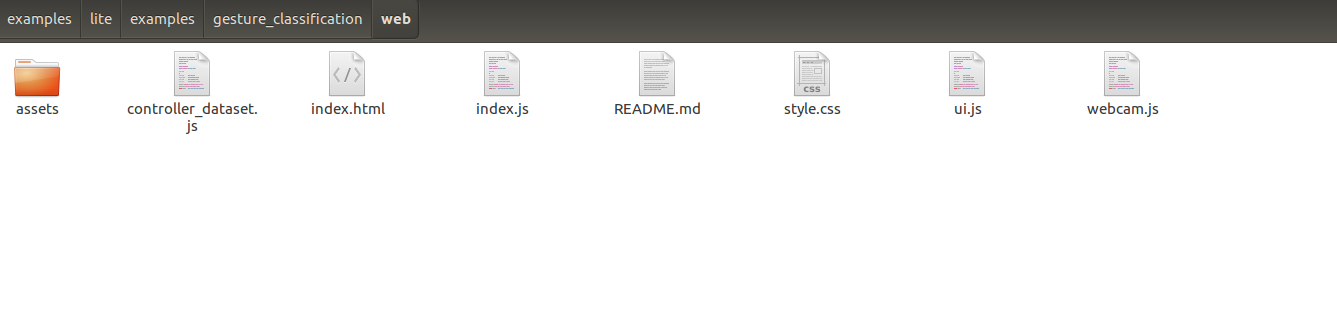
## How to get TensorFlowJS Model

**1.Download gesture\_classification web project from github.**

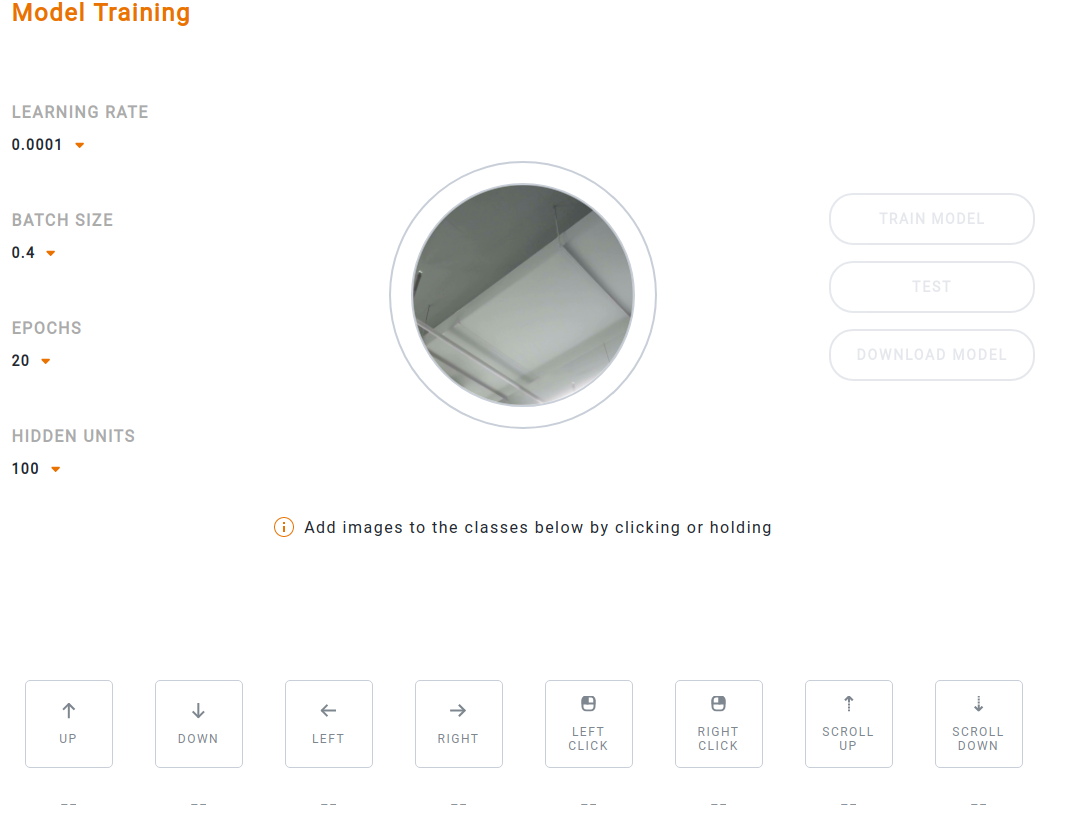
URL:https://github.com/tensorflow/examples/tree/master/lite/examples/gesture\_classification/web

**2.Run gesture\_classification web project.**

Insert an USB camera and use a browser to open ../gesture\_classification/web/index.html



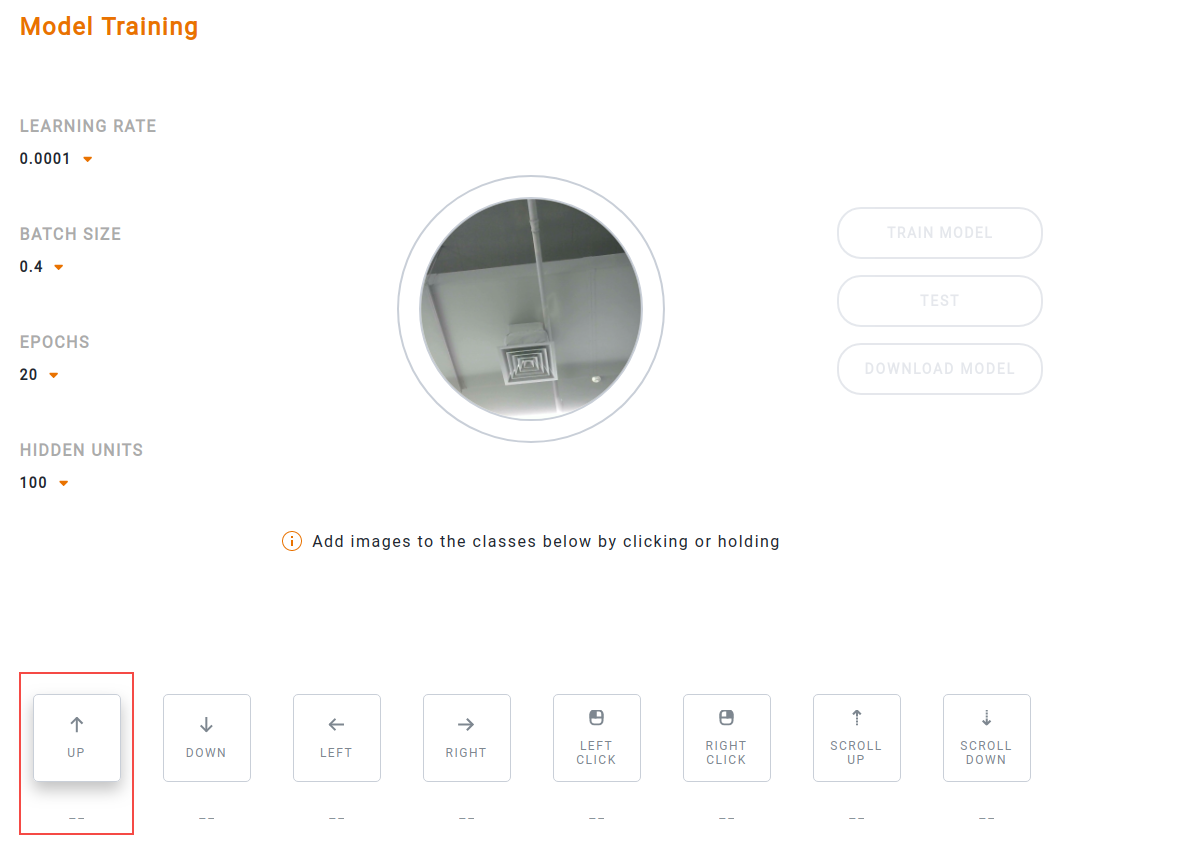
Enter Model Training page, set some parameters if needed

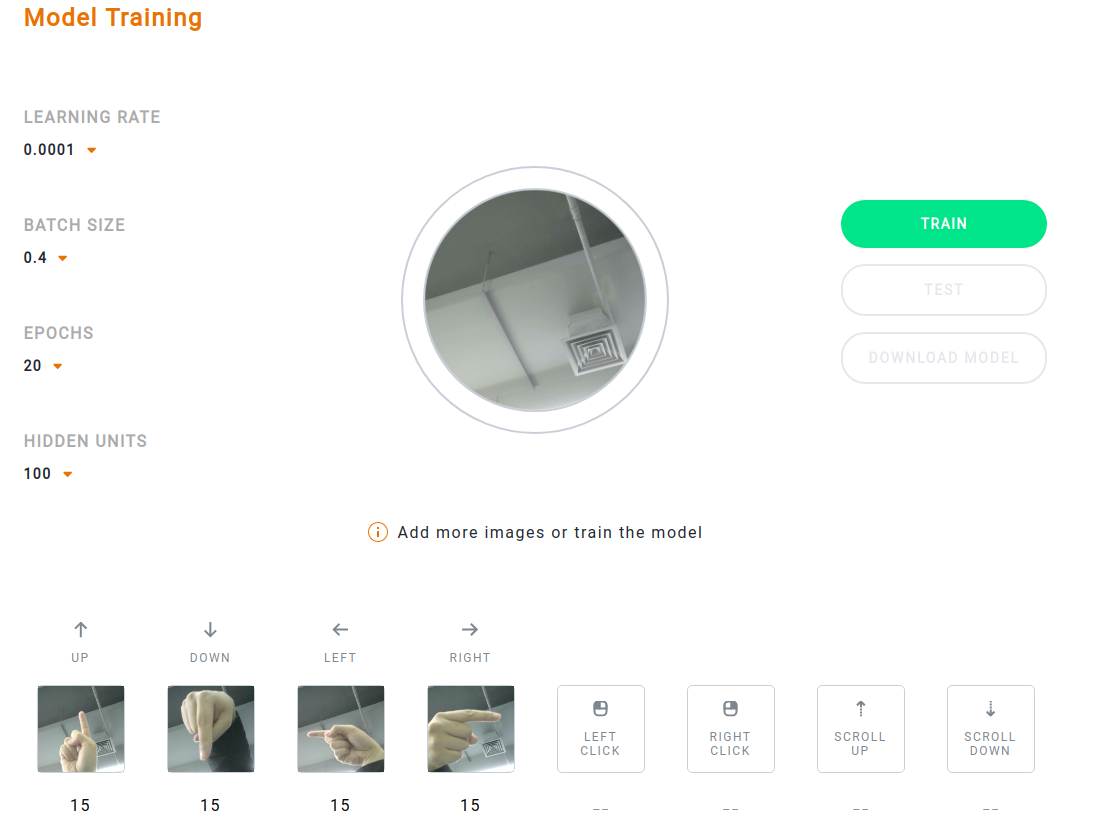


**3.Train and get TensorFlowJS Model.**

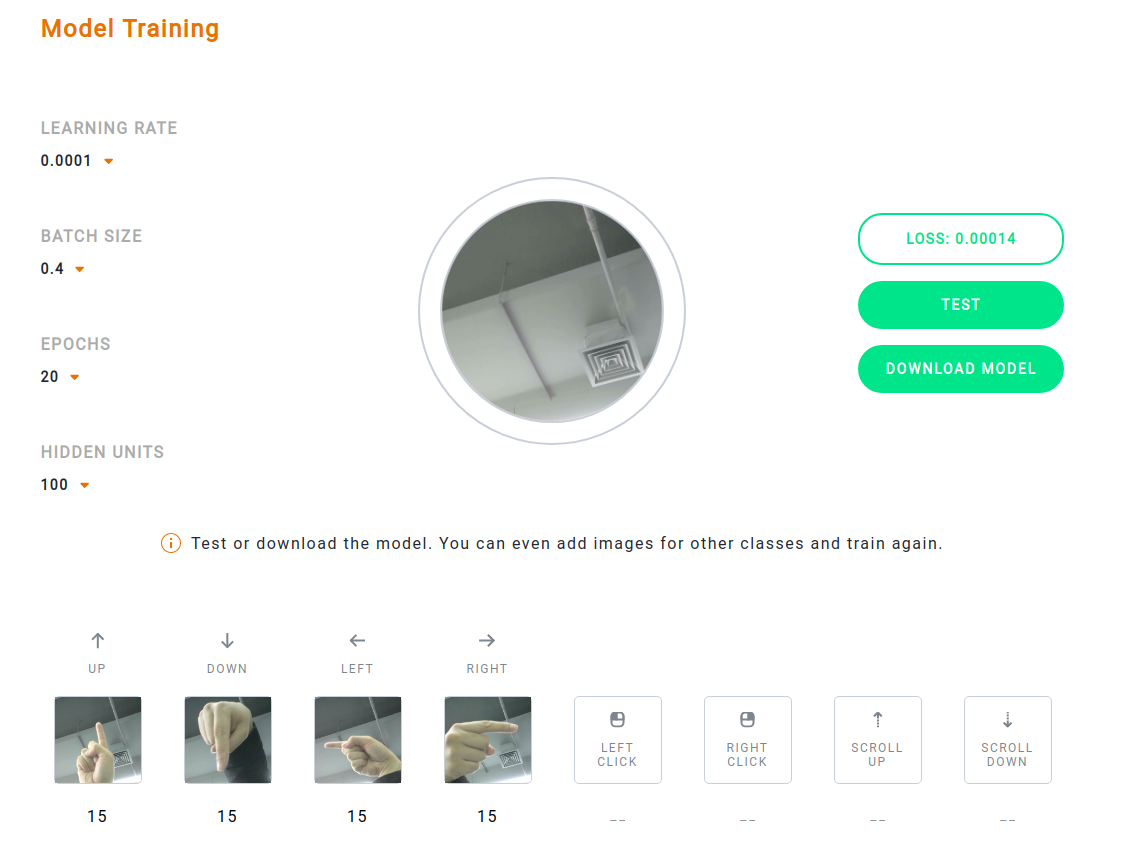
Add the corresponding data set (i.e. take a few corresponding pictures), and then click the “**TRAIN”** button.

We should add data sets with "up", "down", "left" and "right" directions

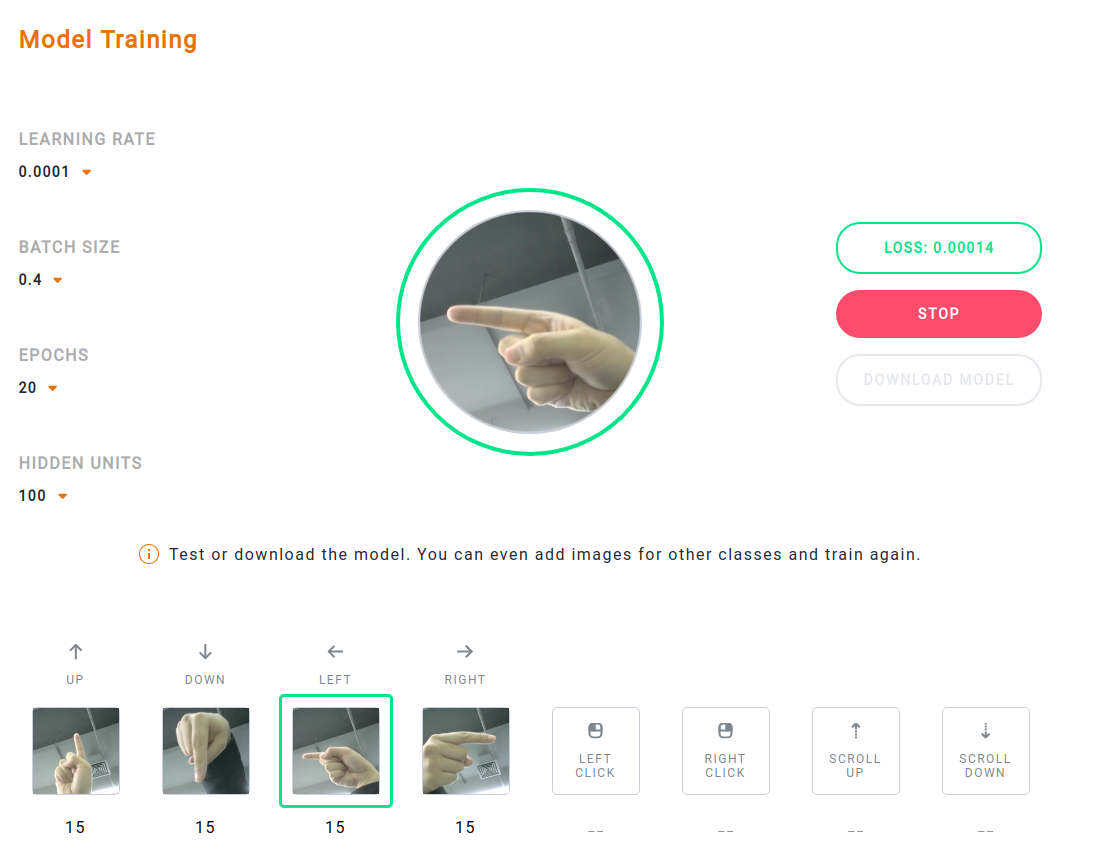




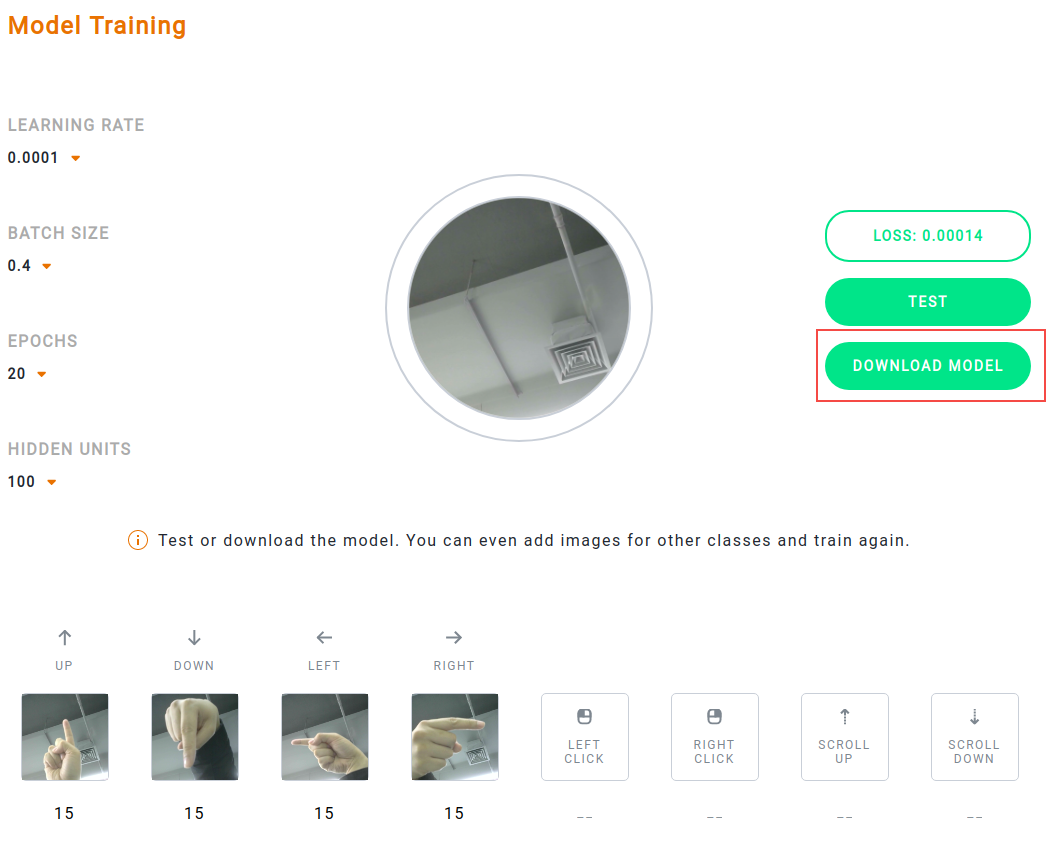
Train the model



Test the model



Download the model



## How to use Google Colab to convert

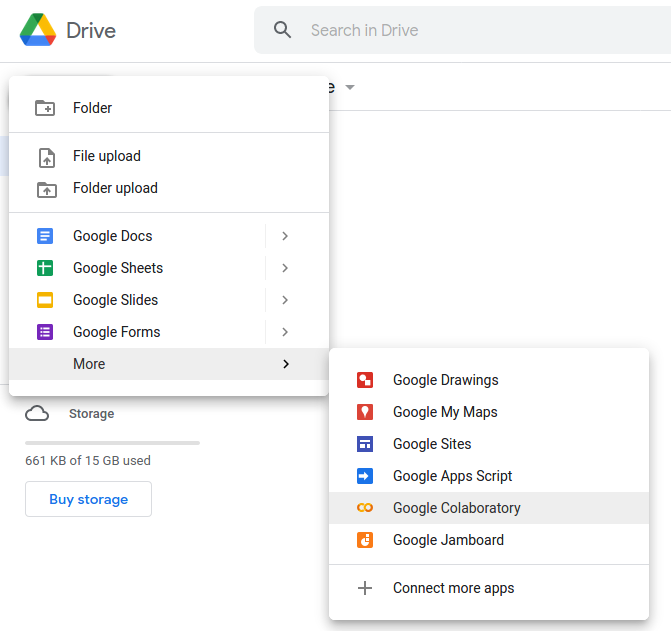
#### 2.1 Create a Colab notebook on Google Drive

**1.Login in Google Drive.**

https://drive.google.com

**2.First associate Colab with Google Drive**

Click New -> More -> Connect more apps, find Google Colaboratory, and connect.

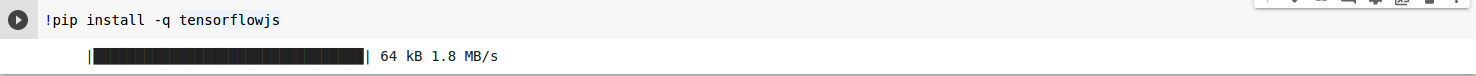


Appears following page after creation



#### 2.2 Run command at Colab notebook

**1.Install Dependencies**



!pip install -q tensorflowjs

## 

import traceback

import logging

import tensorflow as tf

import os

from google.colab import files

from tensorflow import keras

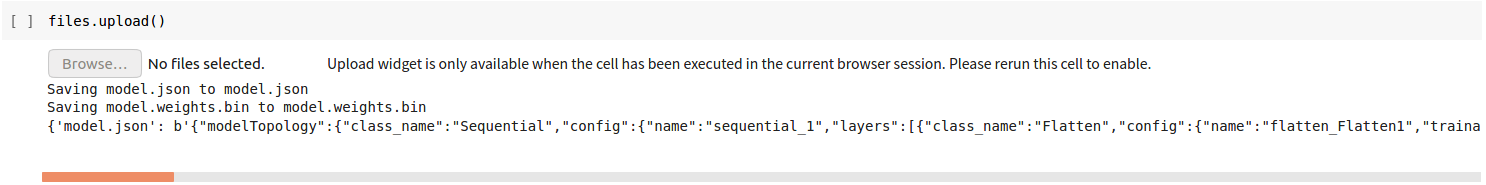
from tensorflowjs.converters import load\_keras\_model

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

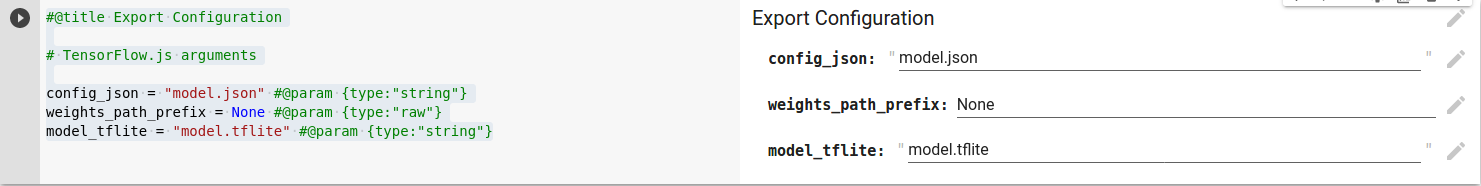
**2.Upload your Tensorflow.js Artifacts Here**

i.e., The weights manifest **model.json** and the binary weights file **model-weights.bin**



files.upload()

**3.Export Configuration**



#@title Export Configuration

# TensorFlow.js arguments

config\_json = "model.json" #@param {type:"string"}

weights\_path\_prefix = None #@param {type:"raw"}

model\_tflite = "model.tflite" #@param {type:"string"}

**4.Model Converter**

The following class converts a TensorFlow.js model to a TFLite FlatBuffer



class ModelConverter:

"""

Creates a ModelConverter class from a TensorFlow.js model file.

Args:

:param config\_json\_path: Full filepath of weights manifest file containing the model architecture.

:param weights\_path\_prefix: Full filepath to the directory in which the weights binaries exist.

:param tflite\_model\_file: Name of the TFLite FlatBuffer file to be exported.

:return:

ModelConverter class.

"""

def \_\_init\_\_(self,

config\_json\_path,

weights\_path\_prefix,

tflite\_model\_file

):

self.config\_json\_path = config\_json\_path

self.weights\_path\_prefix = weights\_path\_prefix

self.tflite\_model\_file = tflite\_model\_file

self.keras\_model\_file = 'merged.h5'

# MobileNet Options

self.input\_node\_name = 'the\_input'

self.image\_size = 224

self.alpha = 0.25

self.depth\_multiplier = 1

self.\_input\_shape = (1, self.image\_size, self.image\_size, 3)

self.depthwise\_conv\_layer = 'conv\_pw\_13\_relu'

def convert(self):

self.save\_keras\_model()

self.\_deserialize\_tflite\_from\_keras()

logger.info('The TFLite model has been generated')

def save\_keras\_model(self):

top\_model = load\_keras\_model(self.config\_json\_path, self.weights\_path\_prefix,

weights\_data\_buffers=None,

load\_weights=True,

)

base\_model = self.get\_base\_model()

self.\_merged\_model = self.merge(base\_model, top\_model)

logger.info("The merged Keras has been generated.")

def merge(self, base\_model, top\_model):

"""

Merges base model with the classification block

:return: Returns the merged Keras model

"""

logger.info("Initializing model...")

layer = base\_model.get\_layer(self.depthwise\_conv\_layer)

model = keras.Model(inputs=base\_model.input, outputs=top\_model(layer.output))

logger.info("Model created.")

return model

def get\_base\_model(self):

"""

Builds MobileNet with the default parameters

:return: Returns the base MobileNet model

"""

input\_tensor = keras.Input(shape=self.\_input\_shape[1:], name=self.input\_node\_name)

base\_model = keras.applications.MobileNet(input\_shape=self.\_input\_shape[1:],

alpha=self.alpha,

depth\_multiplier=self.depth\_multiplier,

input\_tensor=input\_tensor,

include\_top=False)

return base\_model

def \_deserialize\_tflite\_from\_keras(self):

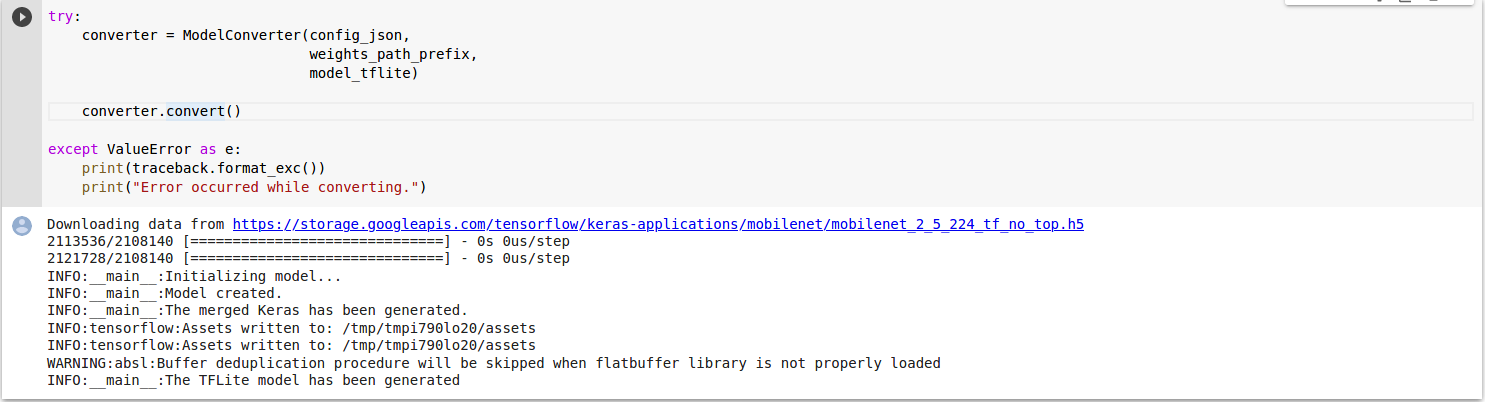
converter = tf.lite.TFLiteConverter.from\_keras\_model(self.\_merged\_model)

tflite\_model = converter.convert()

with open(self.tflite\_model\_file, "wb") as file:

file.write(tflite\_model)

**5.Getting TFLite model.**



try:

converter = ModelConverter(config\_json,

weights\_path\_prefix,

model\_tflite)

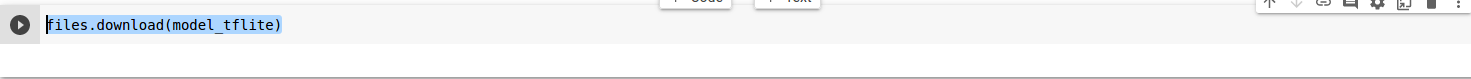
converter.convert()

except ValueError as e:

print(traceback.format\_exc())

print("Error occurred while converting.")

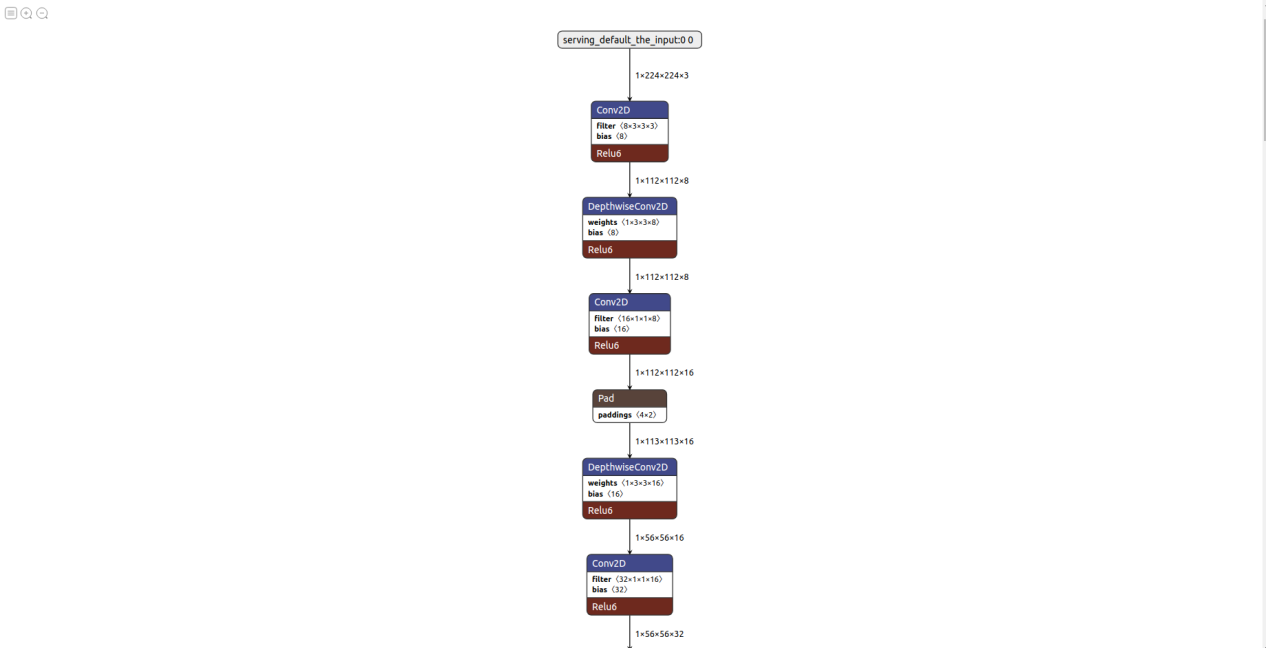
**6.Download TFLite model**



files.download(model\_tflite)

**7.Check TFLite model**

Access <https://netron.app/> and import tflite model file:



Please check <Project Dir>/C865DK-VisionTest1.0/model.tflite.svg to check full layer structure.