Mount Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Prepare the Dataset

Import library

```
!pip install tensorflowjs
```

```
import tensorflow as tf
import subprocess
```

Copy dataset from drive

```
rm -rf /content/food-dataset-500
```

```
cp -R /content/drive/MyDrive/indo_food_datasets/jadi/food-dataset-500 /content/
```

Unzip file

```
import zipfile

# Extract the archive
```

```
local_zip = './food-dataset-500.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('tmp/food-dataset')
zip_ref.close()

# local_zip = './rps-test-set.zip'
# zip_ref = zipfile.ZipFile(local_zip, 'r')
# zip_ref.extractall('tmp/rps-test')
# zip_ref.close()
```

Delete unused dataset

```
food_classes = ['soto','pepes', 'martabak', 'lumpia', 'mendoan']

for food_class in food_classes:
  subprocess.run(["rm", "-rf", "/content/food-dataset-500/test/"+food_class])
  subprocess.run(["rm", "-rf", "/content/food-dataset-500/train/"+food_class])
```

```
cp -R /content/food-dataset-500/train /content/drive/MyDrive/indo_food_datasets/jadi/food-dataset-500
```

```
ls /content/drive/MyDrive/indo_food_datasets/jadi/food-dataset-500/train/klepon | wc -l
```

```
     416
```

Model

Build Model Layer

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
```

```
    tf.keras.layers.MaxPooling2D(),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    # The fourth convolution
    # tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    # tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    #tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Print the model summary
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 148, 148, 32)      896

 max_pooling2d_28 (MaxPoolin  (None, 74, 74, 32)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 72, 72, 32)        9248

 max_pooling2d_29 (MaxPoolin  (None, 36, 36, 32)       0
 g2D)

 conv2d_30 (Conv2D)          (None, 34, 34, 64)        18496

 max_pooling2d_30 (MaxPoolin  (None, 17, 17, 64)       0
 g2D)
```

```
 flatten_9 (Flatten)         (None, 18496)             0

 dense_18 (Dense)            (None, 128)               2367616

 dense_19 (Dense)            (None, 10)                1290

=================================================================
Total params: 2,397,546
Trainable params: 2,397,546
Non-trainable params: 0
_________________________________________________________________
```

Compile Model

```
# Set the training parameters
model.compile(loss = 'categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(), metrics=['accuracy'])
```

Prepare the ImageDataGenerator

```
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/content/food-dataset-500/train"
training_datagen = ImageDataGenerator(
      rescale = 1./255,
        rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

VALIDATION_DIR = "/content/food-dataset-500/test"
validation_datagen = ImageDataGenerator(rescale = 1./255)
```

```
train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(150,150),
    class_mode='categorical',
  batch_size=126
)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150,150),
    class_mode='categorical',
  batch_size=126
  #batch_size=126
)
```

```
Found 4160 images belonging to 10 classes.
Found 1000 images belonging to 10 classes.
```

Train the model and evaluate the results

Define Callback

```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    '''
    Halts the training after reaching 60 percent accuracy

    Args:
      epoch (integer) - index of epoch (required but unused in the function definition below)
      logs (dict) - metric results from the training epoch
    '''

    # Check accuracy
    # if(logs.get('loss') < 0.4):

    #   # Stop if threshold is met
```

```
    #   print("\nLoss is lower than 0.4 so cancelling training!")
    #   self.model.stop_training = True
    if(logs.get('val_accuracy') > 0.75 and logs.get('accuracy') > 0.8):
      # Stop if threshold is met
      print("\nVal_accuracy is higher than 0.8 so cancelling training!")
      self.model.stop_training = True

# Instantiate class
callbacks = myCallback()
```

Train Model

```
history = model.fit(train_generator, epochs=60, validation_data = validation_generator, verbose = 1, validation_steps=3, cal
```

```
Epoch 26/60
34/34 [==============================] - 24s 715ms/step - loss: 1.0570 - accuracy: 0.6334 - val_loss: 1.1749 - val_ac
Epoch 27/60
34/34 [==============================] - 24s 698ms/step - loss: 0.9979 - accuracy: 0.6413 - val_loss: 1.2454 - val_ac
Epoch 28/60
34/34 [==============================] - 24s 702ms/step - loss: 0.9038 - accuracy: 0.6745 - val_loss: 1.1551 - val_ac
Epoch 29/60
34/34 [==============================] - 24s 701ms/step - loss: 0.9142 - accuracy: 0.6724 - val_loss: 1.0333 - val_ac
Epoch 30/60
34/34 [==============================] - 24s 705ms/step - loss: 0.8573 - accuracy: 0.6954 - val_loss: 1.2463 - val_ac
Epoch 31/60
34/34 [==============================] - 24s 719ms/step - loss: 1.0730 - accuracy: 0.6315 - val_loss: 1.0088 - val_ac
Epoch 32/60
34/34 [==============================] - 24s 715ms/step - loss: 0.8241 - accuracy: 0.7101 - val_loss: 1.1773 - val_ac
Epoch 33/60
34/34 [==============================] - 24s 707ms/step - loss: 0.8388 - accuracy: 0.7000 - val_loss: 0.8250 - val_ac
Epoch 34/60
34/34 [==============================] - 24s 702ms/step - loss: 0.8426 - accuracy: 0.7012 - val_loss: 0.9307 - val_ac
Epoch 35/60
34/34 [==============================] - 24s 706ms/step - loss: 0.7607 - accuracy: 0.7368 - val_loss: 1.1119 - val_ac
Epoch 36/60
34/34 [==============================] - 24s 701ms/step - loss: 0.7741 - accuracy: 0.7262 - val_loss: 0.8998 - val_ac
Epoch 37/60
34/34 [==============================] - 24s 716ms/step - loss: 0.7107 - accuracy: 0.7481 - val_loss: 1.1494 - val_ac
```

```
Epoch 38/60
34/34 [==============================] - 24s 708ms/step - loss: 0.7689 - accuracy: 0.7356 - val_loss: 0.9506 - val_ac
Epoch 39/60
34/34 [==============================] - 24s 706ms/step - loss: 0.8433 - accuracy: 0.7084 - val_loss: 0.8750 - val_ac
Epoch 40/60
34/34 [==============================] - 24s 702ms/step - loss: 0.8406 - accuracy: 0.7202 - val_loss: 1.1267 - val_ac
Epoch 41/60
34/34 [==============================] - 24s 703ms/step - loss: 0.9244 - accuracy: 0.6671 - val_loss: 0.9425 - val_ac
Epoch 42/60
34/34 [==============================] - 24s 702ms/step - loss: 0.7646 - accuracy: 0.7337 - val_loss: 0.8536 - val_ac
Epoch 43/60
34/34 [==============================] - 24s 715ms/step - loss: 0.7466 - accuracy: 0.7337 - val_loss: 1.0359 - val_ac
Epoch 44/60
34/34 [==============================] - 24s 702ms/step - loss: 0.9920 - accuracy: 0.6697 - val_loss: 1.0067 - val_ac
Epoch 45/60
34/34 [==============================] - 24s 699ms/step - loss: 0.7361 - accuracy: 0.7430 - val_loss: 0.9324 - val_ac
Epoch 46/60
34/34 [==============================] - 24s 702ms/step - loss: 0.7249 - accuracy: 0.7435 - val_loss: 0.7962 - val_ac
Epoch 47/60
34/34 [==============================] - 24s 701ms/step - loss: 0.7457 - accuracy: 0.7450 - val_loss: 0.8388 - val_ac
Epoch 48/60
34/34 [==============================] - 24s 701ms/step - loss: 0.7586 - accuracy: 0.7320 - val_loss: 0.9208 - val_ac
Epoch 49/60
34/34 [==============================] - 24s 708ms/step - loss: 0.6651 - accuracy: 0.7692 - val_loss: 0.9436 - val_ac
Epoch 50/60
34/34 [==============================] - 24s 707ms/step - loss: 0.6048 - accuracy: 0.7868 - val_loss: 0.9196 - val_ac
Epoch 51/60
34/34 [==============================] - 24s 708ms/step - loss: 0.6043 - accuracy: 0.7988 - val_loss: 0.8776 - val_ac
Epoch 52/60
34/34 [==============================] - ETA: 0s - loss: 0.5679 - accuracy: 0.8048
Val_accuracy is higher than 0.8 so cancelling training!
34/34 [==============================] - 24s 700ms/step - loss: 0.5679 - accuracy: 0.8048 - val_loss: 0.6065 - val_ac
```

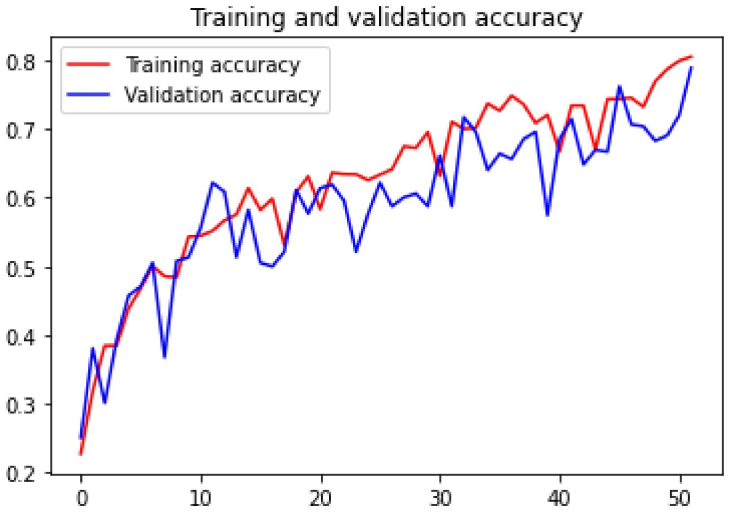
```
import matplotlib.pyplot as plt

# Plot the results
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()

plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

Model Prediction

```
## CODE BLOCK FOR NON-SAFARI BROWSERS
## SAFARI USERS: PLEASE SKIP THIS BLOCK AND RUN THE NEXT ONE INSTEAD

import numpy as np
from google.colab import files
from keras.preprocessing import image
```

```
uploaded = files.upload()

for fn in uploaded.keys():

  # predicting images
  path = fn
  img = image.load_img(path, target_size=(150, 150))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  print(fn)
  print(classes)
```

Choose Files 10 files

- **904.png**(image/png) - 7325 bytes, last modified: 5/13/2022 - 100% done
- **908.png**(image/png) - 7151 bytes, last modified: 5/13/2022 - 100% done
- **934.png**(image/png) - 6714 bytes, last modified: 5/13/2022 - 100% done
- **945.png**(image/png) - 11828 bytes, last modified: 5/13/2022 - 100% done
- **950.png**(image/png) - 5814 bytes, last modified: 5/13/2022 - 100% done
- **952.png**(image/png) - 8734 bytes, last modified: 5/13/2022 - 100% done
- **964.png**(image/png) - 6929 bytes, last modified: 5/13/2022 - 100% done
- **967.png**(image/png) - 5321 bytes, last modified: 5/13/2022 - 100% done
- **970.png**(image/png) - 7833 bytes, last modified: 5/13/2022 - 100% done
- **986.png**(image/png) - 10146 bytes, last modified: 5/13/2022 - 100% done

```
Saving 904.png to 904 (1).png
Saving 908.png to 908 (1).png
Saving 934.png to 934 (4).png
Saving 945.png to 945 (2).png
Saving 950.png to 950 (2).png
Saving 952.png to 952 (2).png
```

```
import time
saved_model_path = "./saved_model/{}.h5".format(int(time.time()))

model.save(saved_model_path)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
```

```
!tensorflowjs_converter --input_format=keras {saved_model_path} ./saved_model/js/
```

```
934.png
```

```
!zip -r model10Class.zip saved_model
```

```
  adding: saved_model/ (stored 0%)
  adding: saved_model/js/ (stored 0%)
  adding: saved_model/js/group1-shard3of3.bin (deflated 7%)
  adding: saved_model/js/group1-shard2of3.bin (deflated 7%)
  adding: saved_model/js/model.json (deflated 82%)
  adding: saved_model/js/group1-shard1of3.bin (deflated 7%)
  adding: saved_model/1653807976.h5 (deflated 18%)
  adding: saved_model/.ipynb_checkpoints/ (stored 0%)
```

```
970.png
```

Finish


2s
completed at 2:26 PM