parth.pandey13103447@gmail.com_21

March 11, 2021

1 Transfer Learning Assignment

1.1 Imports

[ ]:
```
import pandas as pd
import os,sys, gc
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Conv2D, BatchNormalization,␣
↪Dropout, MaxPool2D, Input, LeakyReLU
from tensorflow.keras.initializers import he_normal, glorot_normal
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,␣
↪ReduceLROnPlateau
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import datetime
```

1.2 Tensorboard

[ ]:
```
%load_ext tensorboard
```

1.3 Model Encapsulation

[ ]:
```
class transferlearning:
    def __init__(self, assign_dir, data_dir, file):
        self.data_dir = data_dir

        if not (os.path.exists(self.data_dir) and os.path.isfile(os.path.
↪join(self.data_dir, file))):
            print("Creating Data On Colab Disk")
            print(os.getcwd())
            print(os.listdir(assign_dir))
            cmd = "mkdir {} ".format(self.data_dir)
            print(cmd)
            ret = os.system(cmd)
            print("Return {}".format(ret))
```

```
            cmd = "cp -r '{}' {} ".format(os.path.join(assign_dir,
                                                       file), self.data_dir)
            print(cmd)
            ret = os.system(cmd)
            print("Return {}".format(ret))
            cmd = "sudo unrar x {} {} ".format(
                os.path.join(self.data_dir, file), self.data_dir)
            print(cmd)
            ret = os.system(cmd)
            print("Return {}".format(ret))

        df = pd.read_csv(os.path.join(self.data_dir, "labels_final.csv"))
        print(df.head())
        os.chdir(os.path.join(self.data_dir, "data_final"))
        print(os.getcwd())

        x, y = df[['path']], df[['label']]
        x_train, x_test, y_train, y_test = train_test_split(
            x, y, stratify=y, test_size=0.2)

        print(x_train.shape)
        print(y_train.shape)
        print(x_test.shape)
        print(y_test.shape)

        self.df_train = pd.DataFrame()
        self.df_test = pd.DataFrame()

        self.df_train['path'] = x_train['path']
        self.df_train['label'] = y_train['label'].apply(lambda x: str(x))
        self.df_test['path'] = x_test['path']
        self.df_test['label'] = y_test['label'].apply(lambda x: str(x))
        print(self.df_train.head())

        del x, y, x_train, x_test, y_train, y_test

    def prepare_data_generator(self, batch_size, val_batch_size):
        tf.keras.backend.clear_session()

        datagen = ImageDataGenerator(
            samplewise_center=True,
            samplewise_std_normalization=True,
            rotation_range=360,
            rescale=(1 / 255),
            fill_mode='nearest'
        )
```

```
        train_generator = datagen.flow_from_dataframe(
            self.df_train,
            x_col='path',
            y_col='label',
            class_mode='sparse',
            batch_size=batch_size,
            target_size=(224, 224),
            shuffle=True,
            seed=42,
            save_format='tif',
            validate_filenames=False,
        )

        test_data_gen = ImageDataGenerator(
                            samplewise_center=True,
                            samplewise_std_normalization=True,
                            rescale=(1/255),
        )
        test_generator = test_data_gen.flow_from_dataframe(
            self.df_test,
            x_col='path',
            y_col='label',
            class_mode='sparse',
            batch_size=val_batch_size,
            target_size=(224, 224),
            shuffle=True,
            seed=42,
            save_format='tif',
            validate_filenames=False,
        )
        ds_train = tf.data.Dataset.from_generator(lambda: train_generator,
                                                  output_types=(
                                                      tf.float32, tf.float32),
                                                  output_shapes=(
                                                      [None, 224, 224, 3],␣
↪[None])
                                                  )
        ds_train = ds_train.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

        ds_test = tf.data.Dataset.from_generator(lambda: test_generator,
                                                 output_types=(
                                                     tf.float32, tf.float32),
                                                 output_shapes=(
                                                     [None, 224, 224, 3],␣
↪[None])
                                                 )
        ds_test = ds_test.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
```

```
        return (ds_train, ds_test)


    def prepare_data_generator2(self, batch_size, val_batch_size):
        tf.keras.backend.clear_session()

        datagen = ImageDataGenerator(
            samplewise_center=True,
            samplewise_std_normalization=True,
            rotation_range=360,
            rescale=(1 / 255),
            fill_mode='nearest'
        )

        train_generator = datagen.flow_from_dataframe(
            self.df_train,
            x_col='path',
            y_col='label',
            class_mode='sparse',
            batch_size=batch_size,
            target_size=(224, 224),
            shuffle=True,
            seed=42,
            save_format='tif',
            validate_filenames=False,
        )

        # test_data_gen = ImageDataGenerator(
        #                     samplewise_center=True,
        #                     samplewise_std_normalization=True,
        #                     rescale=(1/255),
        # )
        test_generator = datagen.flow_from_dataframe(
            self.df_test,
            x_col='path',
            y_col='label',
            class_mode='sparse',
            batch_size=val_batch_size,
            target_size=(224, 224),
            shuffle=True,
            seed=42,
            save_format='tif',
            validate_filenames=False,
        )
        ds_train = tf.data.Dataset.from_generator(lambda: train_generator,
                                                  output_types=(
                                                      tf.float32, tf.float32),
```

```
                                                output_shapes=(
                                                    [None, 224, 224, 3],␣
↪[None])
                                                )
        ds_train = ds_train.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

        ds_test = tf.data.Dataset.from_generator(lambda: test_generator,
                                                output_types=(
                                                    tf.float32, tf.float32),
                                                output_shapes=(
                                                    [None, 224, 224, 3],␣
↪[None])
                                                )
        ds_test = ds_test.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
        return (ds_train, ds_test)

    @staticmethod
    def load_model_from_file(file_name: str, model: tf.keras.Model):
        model.load_weights(file_name)
        return model

    @staticmethod
    def define_model1():
        """
        Model Definition
        INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool␣
↪Layer --> 2 FC layers --> Output Layer
        """
        base_model = tf.keras.applications.VGG16(
            include_top=False, weights='imagenet', input_tensor=None,␣
↪input_shape=(224, 224, 3),
            pooling=None, classes=1000)
        print(base_model.summary())
        base_model.trainable = False
        # Input
        input_layer = tf.keras.Input(shape=(224, 224, 3), name='Input_Layer')
        # Base Model
        x = base_model(input_layer)

        # Conv Layer1
        x = Conv2D(filters=256, kernel_size=(2, 2), padding="valid",␣
↪data_format="channels_last",
                   activation='relu', kernel_initializer=he_normal(seed=9),␣
↪name='block_6_conv1')(x)

        # Maxpooling Layer2
```

```
        x = MaxPool2D(pool_size=(2, 2), strides=(
            1, 1), padding='valid', data_format='channels_last',␣
↪name='block6_pool1')(x)

        # Flatten
        x = Flatten(
            data_format='channels_last', name='Flatten')(x)

        # FC layer
        x = Dense(units=256, activation='relu',
                  kernel_initializer=glorot_normal(seed=32), name='FC1')(x)

        # FC layer
        x = Dense(units=128, activation='relu',
                  kernel_initializer=glorot_normal(seed=33), name='FC2')(x)

        # output layer
        output = Dense(
            units=16, activation='softmax',␣
↪kernel_initializer=glorot_normal(seed=3), name='Output')(x)

        # Creating a model
        model = tf.keras.Model(
            inputs=input_layer, outputs=output, name='model_1')

        print(model.summary())
        return model

    @staticmethod
    def define_model2():
        """
        Model Definition
        INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to␣
↪FC --> Output Layer
        """
        base_model = tf.keras.applications.VGG16(include_top=False,␣
↪weights='imagenet', input_tensor=None, input_shape=(224, 224, 3),
                                                 pooling=None, classes=1000)
        print(base_model.summary())
        base_model.trainable = False



        input_layer = Input(shape=(224, 224, 3), name='InputLayer')
        x = base_model(input_layer)

        x = BatchNormalization()(x)
```

```
        lrlu = LeakyReLU(alpha=0.9)
        x = Conv2D(filters=4096, kernel_size=(4, 4), padding="valid",
                   activation=lrlu, kernel_initializer=he_normal(),␣
↪name='FC1')(x)

        # x = Dropout(0.4)(x)

        lrlu = LeakyReLU(alpha=0.9)
        x = Conv2D(filters=4096, kernel_size=(4, 4), padding="valid",
                   activation=lrlu, kernel_initializer=he_normal(),␣
↪name='FC2')(x)
        x = BatchNormalization()(x)
        x = Flatten(name='Flatten')(x)
        output = Dense(units=16, activation="softmax",␣
↪kernel_initializer=he_normal(), name="OutputLayer")(x)

        model = tf.keras.Model(inputs=input_layer, outputs=output,␣
↪name="model_2")
        print(model.summary())
        return model

    @staticmethod
    def define_model3():
        """
        Model Definition
        INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to␣
↪FC --> Output Layer
        """

        base_model = tf.keras.applications.VGG16(include_top=False,␣
↪weights='imagenet', input_tensor=None, input_shape=(224, 224, 3),
                                                 pooling=None, classes=1000)
        print(base_model.summary())

        for layer in base_model.layers[:13]:
            layer.trainable = False
        for layer in base_model.layers:
            print(layer.name, layer.trainable)

        input_layer = Input(shape=(224, 224, 3), name='InputLayer')
        x = base_model(input_layer)

        x = BatchNormalization()(x)

        lrlu = LeakyReLU(alpha=0.9)
```

```
        x = Conv2D(filters=4096, kernel_size=(4, 4), padding="valid",
                   activation=lrlu, kernel_initializer=he_normal(),␣
↪name='FC1')(x)

        # x = Dropout(0.4)(x)

        lrlu = LeakyReLU(alpha=0.9)
        x = Conv2D(filters=4096, kernel_size=(4, 4), padding="valid",
                   activation=lrlu, kernel_initializer=he_normal(),␣
↪name='FC2')(x)
        x = BatchNormalization()(x)
        x = Flatten(name='Flatten')(x)
        output = Dense(units=16, activation="softmax",␣
↪kernel_initializer=he_normal(), name="OutputLayer")(x)

        model = tf.keras.Model(inputs=input_layer, outputs=output,␣
↪name="model_3")
        print(model.summary())
        return model

    def compile_and_train(self, model, **kwargs):
        parameters = kwargs["parameters"]
        model.compile(optimizer=parameters["optimizer"],
                      loss='SparseCategoricalCrossentropy',␣
↪metrics=['SparseCategoricalCrossentropy','accuracy'])


        # Creating Log Directory for tensorboard
        log_dir = f"{os.path.join(self.data_dir, model.name)}/logs/{datetime.
↪datetime.now().strftime('%Y%m%d-%H%M%S')}"
        tensorboard_callback_model_1 = tf.keras.callbacks.TensorBoard(
            log_dir=log_dir,
            histogram_freq=1,
            write_graph=True,
            write_grads=True
        )


        # Get Data
        ds_train, ds_test = self.
↪prepare_data_generator2(parameters["batch_size"],␣
↪parameters["val_batch_size"])

        # Cleaning Extra Memory for better env clean up
        n = gc.collect()
        print(f"Number of unreferenced elements deleted {n}")
```

```
        # Fitting Model
        history = model.fit(ds_train,
                            batch_size=parameters["batch_size"],
                            epochs=parameters["epoch"],
                            validation_data=ds_test,
                            validation_batch_size=parameters["val_batch_size"],
                            validation_steps=self.df_test.shape[0]/␣
↪parameters["val_batch_size"],
                            max_queue_size=1000,
                            workers=10,
                            steps_per_epoch=self.df_train.shape[0] /␣
↪parameters["batch_size"],
                            callbacks=[
                                    #    model_1_checkpoint,
                                       tensorboard_callback_model_1,
                                    #    early_stopping_callback,
                                    #    reduce_lr
                                       ]
                            )
        print(history.history)
        # Evaluate Model
        # model.evaluate(x=ds_test, verbose=1, batch_size=128, steps=75)
        return model
```

1.4 Declaring Variables

[ ]:
```
assign_dir = "/content/drive/MyDrive/Colab Notebooks/AppliedAICourse/Assignment/
↪assignment21/"
data_dir   = "/content/sample_data/assignment21/"
file = "rvl-cdip-002.rar"
```

1.5 Prepaing Model

[ ]:
```
tl = transferlearning(assign_dir=assign_dir, data_dir=data_dir, file=file)
```

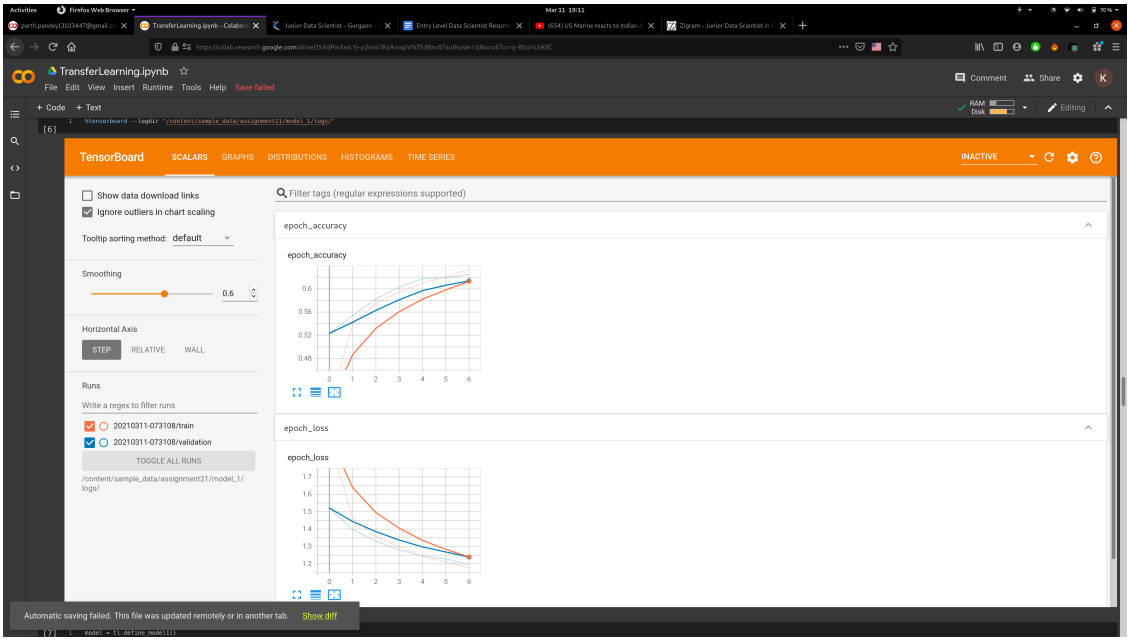
```
Creating Data On Colab Disk
/content
['labels_final.csv', 'rvl-cdip-002.rar', 'TransferLearning.ipynb']
mkdir /content/sample_data/assignment21/
Return 0
cp -r '/content/drive/MyDrive/Colab
Notebooks/AppliedAICourse/Assignment/assignment21/rvl-cdip-002.rar'
/content/sample_data/assignment21/
Return 0
sudo unrar x /content/sample_data/assignment21/rvl-cdip-002.rar
```

```
/content/sample_data/assignment21/
Return 0
                                          path  label
0  imagesv/v/o/h/voh71d00/509132755+-2755.tif      3
1        imagesl/l/x/t/lxt19d00/502213303.tif      3
2       imagesx/x/e/d/xed05a00/2075325674.tif      2
3  imageso/o/j/b/ojb60d00/517511301+-1301.tif      3
4       imagesq/q/z/k/qzk17e00/2031320195.tif      7
/content/sample_data/assignment21/data_final
(38400, 1)
(38400, 1)
(9600, 1)
(9600, 1)
                                          path label
40703    imagesv/v/f/a/vfa79c00/50191124-1126.tif    14
14640       imagesw/w/m/m/wmm30f00/0012211440.tif     0
23205  imagese/e/k/s/eks80d00/522868445+-8446.tif     0
28349       imagesw/w/l/z/wlz26e00/2047433504.tif     8
23350         imagesq/q/e/r/qer01e00/86003028.tif    11
```

1.6 Training Model1

[ ]:
```
%tensorboard --logdir "/content/sample_data/assignment21/model_1/logs/"
```

```
<IPython.core.display.Javascript object>
```



[ ]:
```
model = tl.define_model1()
training_parameters = {
```

```
    "batch_size": 256,
    "val_batch_size":64,
    "epoch": 7,
     "optimizer": tf.keras.optimizers.Adam(learning_rate=0.001)
}
tl.compile_and_train(model=model,parameters=training_parameters)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
```

```
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (InputLayer)     [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
block_6_conv1 (Conv2D)       (None, 6, 6, 256)         524544
_________________________________________________________________
block6_pool1 (MaxPooling2D)  (None, 5, 5, 256)         0
_________________________________________________________________
Flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
FC1 (Dense)                  (None, 256)               1638656
_________________________________________________________________
FC2 (Dense)                  (None, 128)               32896
_________________________________________________________________
Output (Dense)               (None, 16)                2064
=================================================================
Total params: 16,912,848
Trainable params: 2,198,160
Non-trainable params: 14,714,688
_________________________________________________________________
None
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
Found 38400 non-validated image filenames belonging to 16 classes.
Found 9600 non-validated image filenames belonging to 16 classes.
Number of unreferenced elements deleted 935
Epoch 1/7
150/150 [==============================] - 730s 5s/step - loss: 2.3901 -
sparse_categorical_crossentropy: 2.3901 - accuracy: 0.2837 - val_loss: 1.5215 -
val_sparse_categorical_crossentropy: 1.5215 - val_accuracy: 0.5231
Epoch 2/7
150/150 [==============================] - 678s 5s/step - loss: 1.4950 -
sparse_categorical_crossentropy: 1.4950 - accuracy: 0.5265 - val_loss: 1.3962 -
val_sparse_categorical_crossentropy: 1.3962 - val_accuracy: 0.5541
```

```
Epoch 3/7
150/150 [==============================] - 676s 5s/step - loss: 1.3797 -
sparse_categorical_crossentropy: 1.3797 - accuracy: 0.5674 - val_loss: 1.3299 -
val_sparse_categorical_crossentropy: 1.3299 - val_accuracy: 0.5826
Epoch 4/7
150/150 [==============================] - 675s 5s/step - loss: 1.3137 -
sparse_categorical_crossentropy: 1.3137 - accuracy: 0.5886 - val_loss: 1.2798 -
val_sparse_categorical_crossentropy: 1.2798 - val_accuracy: 0.6025
Epoch 5/7
150/150 [==============================] - 693s 5s/step - loss: 1.2455 -
sparse_categorical_crossentropy: 1.2455 - accuracy: 0.6091 - val_loss: 1.2455 -
val_sparse_categorical_crossentropy: 1.2455 - val_accuracy: 0.6177
Epoch 6/7
150/150 [==============================] - 678s 5s/step - loss: 1.1982 -
sparse_categorical_crossentropy: 1.1982 - accuracy: 0.6268 - val_loss: 1.2279 -
val_sparse_categorical_crossentropy: 1.2279 - val_accuracy: 0.6191
Epoch 7/7
150/150 [==============================] - 653s 4s/step - loss: 1.1731 -
sparse_categorical_crossentropy: 1.1731 - accuracy: 0.6341 - val_loss: 1.1958 -
val_sparse_categorical_crossentropy: 1.1958 - val_accuracy: 0.6248
{'loss': [1.929975152015686, 1.4639347791671753, 1.3563119173049927,
1.2994931936264038, 1.2430107593536377, 1.2125444412231445, 1.176021933555603],
'sparse_categorical_crossentropy': [1.929975152015686, 1.4639347791671753,
1.3563119173049927, 1.2994931936264038, 1.2430107593536377, 1.2125444412231445,
1.176021933555603], 'accuracy': [0.39921873807907104, 0.5386458039283752,
0.5753385424613953, 0.5946614742279053, 0.610338568687439, 0.6204166412353516,
0.6339322924613953], 'val_loss': [1.521492600440979, 1.3962492942810059,
1.3298829793930054, 1.2797954082489014, 1.2455145120620728, 1.2279083728790283,
1.1958224773406982], 'val_sparse_categorical_crossentropy': [1.521492600440979,
1.3962492942810059, 1.3298829793930054, 1.2797954082489014, 1.2455145120620728,
1.2279083728790283, 1.1958224773406982], 'val_accuracy': [0.5231249928474426,
0.5540624856948853, 0.582604169845581, 0.6025000214576721, 0.6177083253860474,
0.6190624833106995, 0.62479168176651]}
```

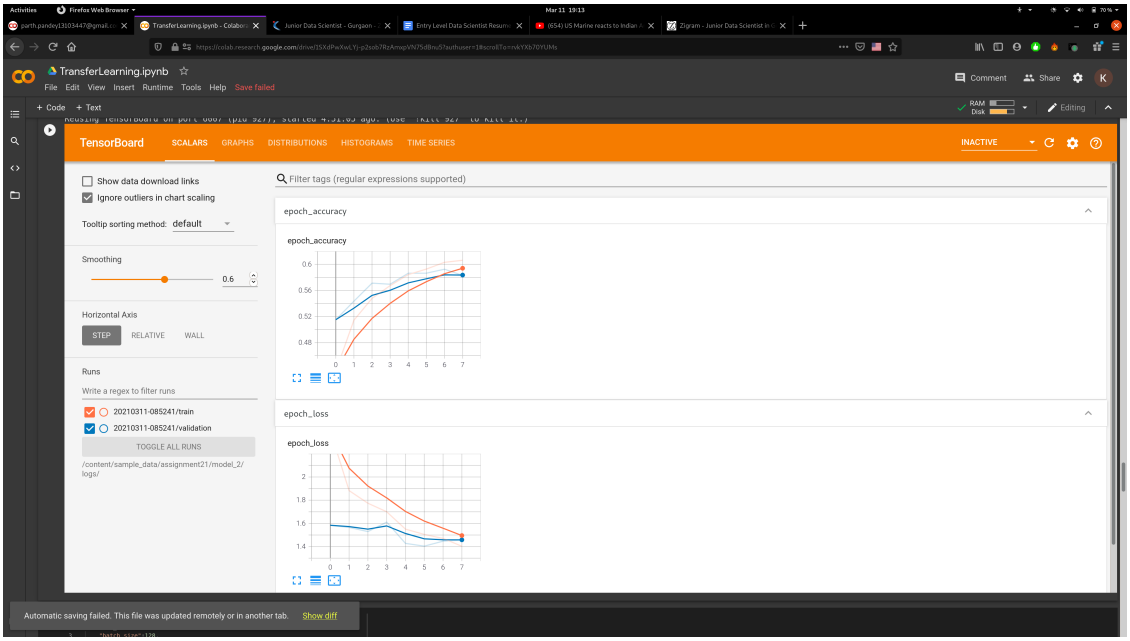
[ ]:
```
<tensorflow.python.keras.engine.functional.Functional at 0x7fdc5014d490>
```

1.7 Training Model2

[12]:
```
%tensorboard --logdir "/content/sample_data/assignment21/model_2/logs/"
```

```
Reusing TensorBoard on port 6007 (pid 927), started 4:51:05 ago. (Use '!kill␣
↪927' to kill it.)
```

```
<IPython.core.display.Javascript object>
```



[9]:
```
model = tl.define_model2()
training_parameters = {
    "batch_size":128,
    "val_batch_size":32,
   "epoch": 8,
   "optimizer": tf.keras.optimizers.RMSprop(learning_rate=3e-4, momentum=0.9)
}
tl.compile_and_train(model=model,parameters=training_parameters)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
```

```
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
None
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
InputLayer (InputLayer)      [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
batch_normalization (BatchNo (None, 7, 7, 512)         2048
_________________________________________________________________
FC1 (Conv2D)                 (None, 4, 4, 4096)        33558528
_________________________________________________________________
FC2 (Conv2D)                 (None, 1, 1, 4096)        268439552
_________________________________________________________________
batch_normalization_1 (Batch (None, 1, 1, 4096)        16384
_________________________________________________________________
Flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
OutputLayer (Dense)          (None, 16)                65552
=================================================================
Total params: 316,796,752
```

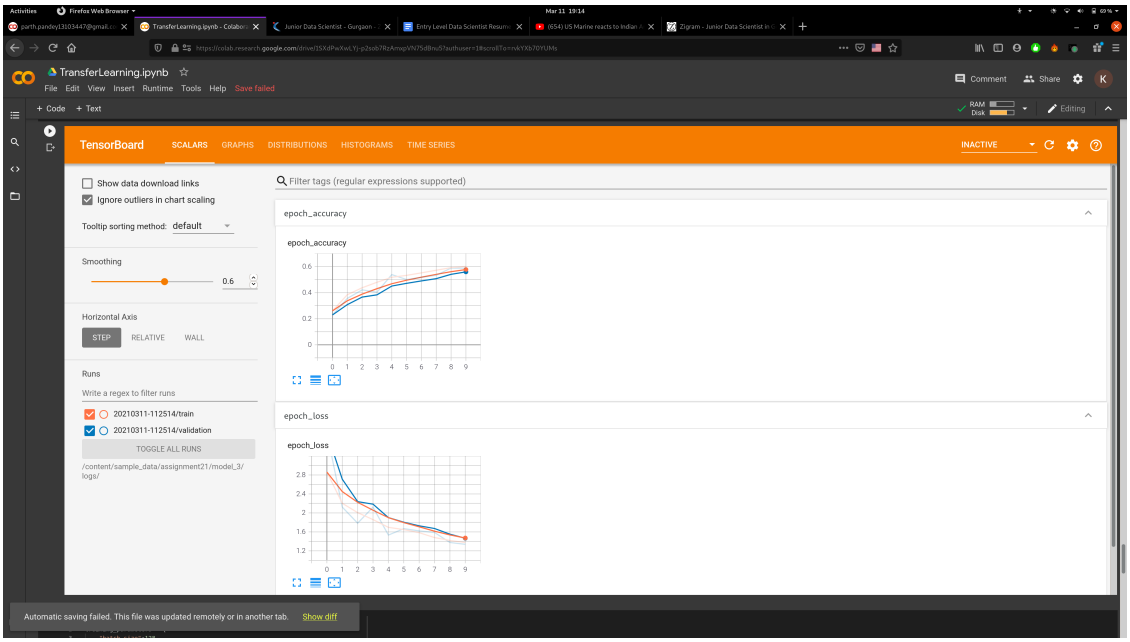
```
Trainable params: 302,072,848
Non-trainable params: 14,723,904
__________________________________________________________________
None
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
Found 38400 non-validated image filenames belonging to 16 classes.
Found 9600 non-validated image filenames belonging to 16 classes.
Number of unreferenced elements deleted 869
Epoch 1/8
300/300 [==============================] - 802s 3s/step - loss: 3.4390 -
sparse_categorical_crossentropy: 3.4390 - accuracy: 0.3820 - val_loss: 1.5837 -
val_sparse_categorical_crossentropy: 1.5837 - val_accuracy: 0.5151
Epoch 2/8
300/300 [==============================] - 761s 3s/step - loss: 1.9205 -
sparse_categorical_crossentropy: 1.9205 - accuracy: 0.5033 - val_loss: 1.5651 -
val_sparse_categorical_crossentropy: 1.5651 - val_accuracy: 0.5435
Epoch 3/8
300/300 [==============================] - 768s 3s/step - loss: 1.7333 -
sparse_categorical_crossentropy: 1.7333 - accuracy: 0.5467 - val_loss: 1.5293 -
val_sparse_categorical_crossentropy: 1.5293 - val_accuracy: 0.5713
Epoch 4/8
300/300 [==============================] - 767s 3s/step - loss: 1.6578 -
sparse_categorical_crossentropy: 1.6578 - accuracy: 0.5727 - val_loss: 1.6110 -
val_sparse_categorical_crossentropy: 1.6110 - val_accuracy: 0.5696
Epoch 5/8
300/300 [==============================] - 780s 3s/step - loss: 1.5703 -
sparse_categorical_crossentropy: 1.5703 - accuracy: 0.5861 - val_loss: 1.4276 -
val_sparse_categorical_crossentropy: 1.4276 - val_accuracy: 0.5865
Epoch 6/8
300/300 [==============================] - 792s 3s/step - loss: 1.4995 -
sparse_categorical_crossentropy: 1.4995 - accuracy: 0.5942 - val_loss: 1.4036 -
val_sparse_categorical_crossentropy: 1.4036 - val_accuracy: 0.5866
Epoch 7/8
300/300 [==============================] - 772s 3s/step - loss: 1.4772 -
sparse_categorical_crossentropy: 1.4772 - accuracy: 0.6048 - val_loss: 1.4463 -
val_sparse_categorical_crossentropy: 1.4463 - val_accuracy: 0.5926
Epoch 8/8
300/300 [==============================] - 722s 2s/step - loss: 1.4019 -
sparse_categorical_crossentropy: 1.4019 - accuracy: 0.6078 - val_loss: 1.4574 -
val_sparse_categorical_crossentropy: 1.4574 - val_accuracy: 0.5831
{'loss': [2.398589611053467, 1.8827733993530273, 1.7740107774734497,
1.7002365589141846, 1.550517201423645, 1.5049656629562378, 1.4700520038604736,
1.4028819799423218], 'sparse_categorical_crossentropy': [2.398589611053467,
1.8827733993530273, 1.7740107774734497, 1.7002365589141846, 1.550517201423645,
1.5049656629562378, 1.4700520038604736, 1.4028819799423218], 'accuracy':
[0.4362500011920929, 0.5143489837646484, 0.5477864742279053, 0.5674739480018616,
0.5843489766120911, 0.5928124785423279, 0.6031510233879089, 0.6063281297683716],
```

```
'val_loss': [1.5837442874908447, 1.565141201019287, 1.5293341875076294,
1.6109524965286255, 1.4275779724121094, 1.403571367263794, 1.4463320970535278,
1.457356333732605], 'val_sparse_categorical_crossentropy': [1.5837442874908447,
1.565141201019287, 1.5293341875076294, 1.6109524965286255, 1.4275779724121094,
1.403571367263794, 1.4463320970535278, 1.457356333732605], 'val_accuracy':
[0.5151041746139526, 0.543541669845581, 0.5712500214576721, 0.5695833563804626,
0.5864583253860474, 0.5865625143051147, 0.5926041603088379, 0.5831249952316284]}
```

[9]: <tensorflow.python.keras.engine.functional.Functional at 0x7fdbf818ca90>

[10]:
```
%tensorboard --logdir "/content/sample_data/assignment21/model_3/logs/"
```

<IPython.core.display.Javascript object>



[11]:
```
model = tl.define_model3()
training_parameters = {
    "batch_size":128,
    "val_batch_size":32,
   "epoch": 10,
   "optimizer": tf.keras.optimizers.RMSprop(learning_rate=3e-4, momentum=0.9)
}
tl.compile_and_train(model=model,parameters=training_parameters)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
```

```
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
None
input_1 False
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False
```

```
block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 True
block4_pool True
block5_conv1 True
block5_conv2 True
block5_conv3 True
block5_pool True
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
InputLayer (InputLayer)      [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
batch_normalization (BatchNo (None, 7, 7, 512)         2048
_________________________________________________________________
FC1 (Conv2D)                 (None, 4, 4, 4096)        33558528
_________________________________________________________________
FC2 (Conv2D)                 (None, 1, 1, 4096)        268439552
_________________________________________________________________
batch_normalization_1 (Batch (None, 1, 1, 4096)        16384
_________________________________________________________________
Flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
OutputLayer (Dense)          (None, 16)                65552
=================================================================
Total params: 316,796,752
Trainable params: 311,512,080
Non-trainable params: 5,284,672
_________________________________________________________________
None
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
Found 38400 non-validated image filenames belonging to 16 classes.
Found 9600 non-validated image filenames belonging to 16 classes.
Number of unreferenced elements deleted 1964
Epoch 1/10
300/300 [==============================] - 809s 3s/step - loss: 3.4659 -
sparse_categorical_crossentropy: 3.4659 - accuracy: 0.2181 - val_loss: 3.6906 -
val_sparse_categorical_crossentropy: 3.6906 - val_accuracy: 0.2288
Epoch 2/10
300/300 [==============================] - 768s 3s/step - loss: 2.1929 -
```

```
sparse_categorical_crossentropy: 2.1929 - accuracy: 0.3762 - val_loss: 2.1225 -
val_sparse_categorical_crossentropy: 2.1225 - val_accuracy: 0.3558
Epoch 3/10
300/300 [==============================] - 771s 3s/step - loss: 2.0194 -
sparse_categorical_crossentropy: 2.0194 - accuracy: 0.4281 - val_loss: 1.7833 -
val_sparse_categorical_crossentropy: 1.7833 - val_accuracy: 0.4209
Epoch 4/10
300/300 [==============================] - 769s 3s/step - loss: 1.8525 -
sparse_categorical_crossentropy: 1.8525 - accuracy: 0.4749 - val_loss: 2.1217 -
val_sparse_categorical_crossentropy: 2.1217 - val_accuracy: 0.4032
Epoch 5/10
300/300 [==============================] - 777s 3s/step - loss: 1.6870 -
sparse_categorical_crossentropy: 1.6870 - accuracy: 0.5117 - val_loss: 1.5324 -
val_sparse_categorical_crossentropy: 1.5324 - val_accuracy: 0.5383
Epoch 6/10
300/300 [==============================] - 800s 3s/step - loss: 1.5949 -
sparse_categorical_crossentropy: 1.5949 - accuracy: 0.5351 - val_loss: 1.6681 -
val_sparse_categorical_crossentropy: 1.6681 - val_accuracy: 0.5004
Epoch 7/10
300/300 [==============================] - 788s 3s/step - loss: 1.5963 -
sparse_categorical_crossentropy: 1.5963 - accuracy: 0.5495 - val_loss: 1.6204 -
val_sparse_categorical_crossentropy: 1.6204 - val_accuracy: 0.5168
Epoch 8/10
300/300 [==============================] - 730s 2s/step - loss: 1.4865 -
sparse_categorical_crossentropy: 1.4865 - accuracy: 0.5704 - val_loss: 1.5878 -
val_sparse_categorical_crossentropy: 1.5878 - val_accuracy: 0.5328
Epoch 9/10
300/300 [==============================] - 730s 2s/step - loss: 1.4642 -
sparse_categorical_crossentropy: 1.4642 - accuracy: 0.5851 - val_loss: 1.3758 -
val_sparse_categorical_crossentropy: 1.3758 - val_accuracy: 0.5888
Epoch 10/10
300/300 [==============================] - 725s 2s/step - loss: 1.4110 -
sparse_categorical_crossentropy: 1.4110 - accuracy: 0.5916 - val_loss: 1.3377 -
val_sparse_categorical_crossentropy: 1.3377 - val_accuracy: 0.5872
{'loss': [2.8618714809417725, 2.2036497592926025, 2.0062997341156006,
1.8591395616531372, 1.6960029602050781, 1.648950457572937, 1.5829265117645264,
1.4801697731018066, 1.4192566871643066, 1.3746092319488525],
'sparse_categorical_crossentropy': [2.8618714809417725, 2.2036497592926025,
2.0062997341156006, 1.8591395616531372, 1.6960029602050781, 1.648950457572937,
1.5829265117645264, 1.4801697731018066, 1.4192566871643066, 1.3746092319488525],
'accuracy': [0.2600520849227905, 0.3839062452316284, 0.43742188811302185,
0.47953125834465027, 0.5177603960037231, 0.5323697924613953, 0.5516145825386047,
0.5717447996139526, 0.5915104150772095, 0.6009374856948853], 'val_loss':
[3.690594434738159, 2.12249755859375, 1.7833105325698853, 2.1216633319854736,
1.5324236154556274, 1.6681290864944458, 1.6203676462173462, 1.5878342390060425,
1.3757853507995605, 1.337663173675537], 'val_sparse_categorical_crossentropy':
[3.690594434738159, 2.12249755859375, 1.7833105325698853, 2.1216633319854736,
1.5324236154556274, 1.6681290864944458, 1.6203676462173462, 1.5878342390060425,
```

```
1.3757853507995605, 1.337663173675537], 'val_accuracy': [0.22875000536441803,
0.3558333218097687, 0.42093750834465027, 0.4032291769981384, 0.5383333563804626,
0.5004166960716248, 0.5167708396911621, 0.5328124761581421, 0.5887500047683716,
0.5871875286102295]}
```

[11]: `<tensorflow.python.keras.engine.functional.Functional at 0x7fdbee2b8b50>`