Task 1.1 Define a problem

Define an image classification problem that may help people better recycle, particularly by reducing contamination.

The Problem we are going to address here is to reduce waste contamination, by classifing diffrent garbage wastes into diffrent classes. In order to diffrentiate the waste into recycle and non-recycle waste. First we need to identify the waste for diffrentiating the items into recycle and non recycle waste. So in this Solution we are going to perform image classification model where we will classify the six Primary Waste items. like Carbaord, plastic, paper, trash , metal and glass.

Describe the desired inputs and outputs, including the target classes.

The inputs will be images and outputs 4 class classification , the Target classes are identified as "Glass, cardboard, paper,trash,metal and plastic"

Task 1.2 Make a plan

What dataset can you use to develop a deep learning solution?

For this solution we are going to use "Garbage Classification dataset" from kaggle . Kaggle is Opensource Data and soultion Sharing platform. The data set contains arround 2527 Total Garbage Classification Dataset contains 6 classifications: cardboard (393), glass (491), metal (400), paper(584), plastic (472) and trash(127).

How many images do you need? How many for training? How many for testing?

Since 2k images is below the nominal value for image classification in, i have decided to whole images for building model.

But however i will be using 2276 images for Training so that the model can learn best knowledge of the classes with number training sets and 251 images for Testing.

Do you need to label the images yourself?

Although the the labels are availble along with dataset, we are going to create labels for our training and test set becuase we are going to split the data randomly to get even distribution of classes in train and Test sets.

How do you determine if your model is good enough?

The model evaluation will be determined using Classification matrix and also the loss and

✓ 0s completed at 10:54 PM

Task 1.3 Implement a solution

Collect relevant data.

```
! pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whee
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
```

```
! mkdir ~/.kaggle
```

```
! cp sample_data/kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
! kaggle datasets download asdasdasasdas/garbage-classification
```

```
Downloading garbage-classification.zip to /content
 89% 73.0M/82.0M [00:03<00:00, 32.4MB/s]
100% 82.0M/82.0M [00:03<00:00, 28.1MB/s]
```

```
! unzip garbage-classification.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard152.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard153.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard154.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard155.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard156.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard157.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard158.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard159.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard16.j
  inflating: Garbage classification/Garbage classification/cardboard/cardboard160.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard161.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard162.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard163.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard164.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard165.
```

```
  inflating: Garbage classification/Garbage classification/cardboard/cardboard166.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard167.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard168.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard169.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard17.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard170.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard171.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard172.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard173.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard174.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard175.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard176.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard177.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard178.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard179.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard18.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard180.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard181.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard182.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard183.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard184.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard185.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard186.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard187.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard188.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard189.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard19.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard190.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard191.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard192.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard193.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard194.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard195.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard196.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard197.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard198.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard199.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard2.jp
  inflating: Garbage classification/Garbage classification/cardboard/cardboard20.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard200.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard201.
  inflating: Garbage classification/Garbage classification/cardboard/cardboard202.
```

Develop a deep learning model.

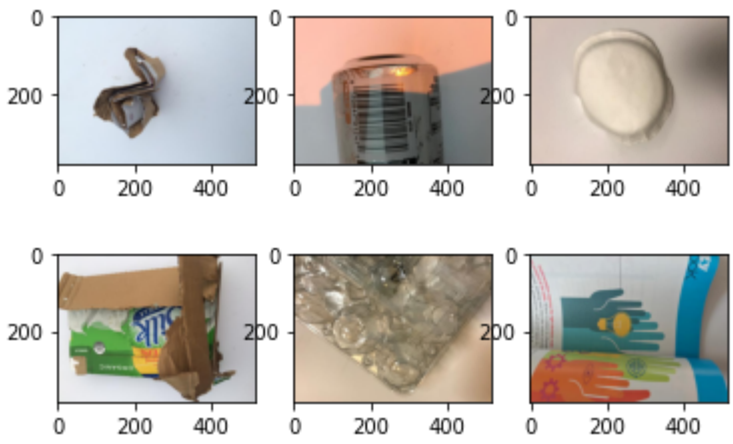
```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array
from keras.layers import Conv2D, Flatten, MaxPooling2D, Dense
from keras.models import Sequential
```

```
import glob, os, random
```

```
base_path = '/content/Garbage classification/Garbage classification'

img_list = glob.glob(os.path.join(base_path, '*/*.jpg'))

print(len(img_list))
```

```
2527
```

```
#Previewing the images

from PIL import Image
for i, img_path in enumerate(random.sample(img_list, 6)):
    img = load_img(img_path)
    img = img_to_array(img, dtype=np.uint8)

    plt.subplot(2, 3, i+1)
    plt.imshow(img.squeeze())
```



```
#Preparing Test, Train and Validation

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.1
)

test_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.1
```

```
)

train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    seed=0
)

validation_generator = test_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    seed=0
)

labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())

print(labels)
```

```
Found 2276 images belonging to 6 classes.
Found 251 images belonging to 6 classes.
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
#Now Lets build our model

model = Sequential([
    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=(3
    MaxPooling2D(pool_size=2),

    Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(6, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

```
model.summary()
```

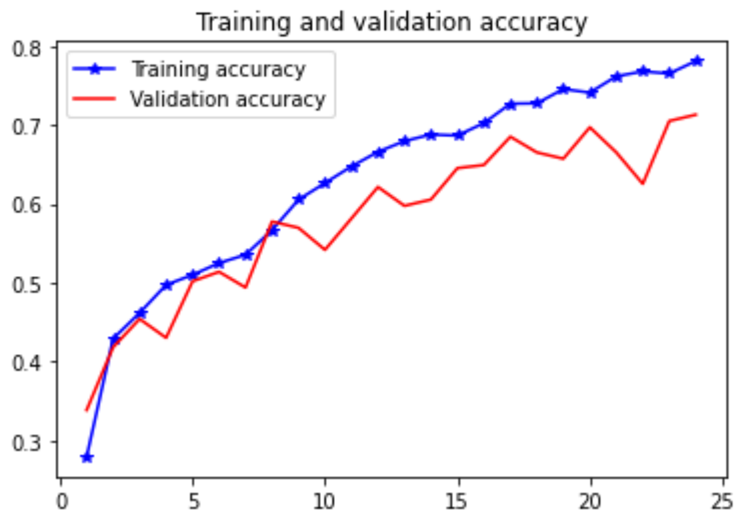
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 300, 300, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 150, 150, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 150, 150, 64)      18496

 max_pooling2d_5 (MaxPooling  (None, 75, 75, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 75, 75, 32)        18464

 max_pooling2d_6 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 37, 37, 32)        9248

 max_pooling2d_7 (MaxPooling  (None, 18, 18, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 10368)             0

 dense_2 (Dense)             (None, 64)                663616

 dense_3 (Dense)             (None, 6)                 390

=================================================================
Total params: 711,110
Trainable params: 711,110
Non-trainable params: 0
_________________________________________________________________
```

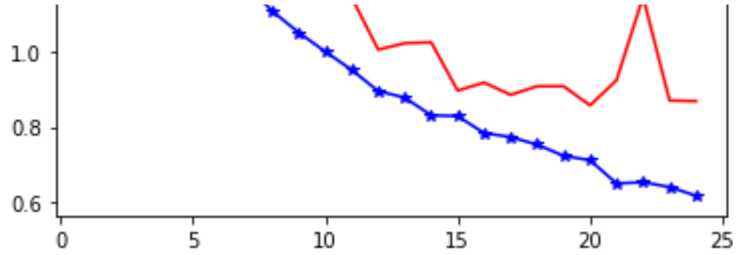
```
callback = tf.keras.callbacks.EarlyStopping(patience=4, restore_best_weights=True)
history = model.fit_generator(train_generator, epochs=50,validation_data=validation_gen
```

```
Epoch 1/50
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Mode

158/158 [==============================] - 13s 81ms/step - loss: 3.0364 - acc: 0.2
Epoch 2/50
158/158 [==============================] - 13s 80ms/step - loss: 1.5692 - acc: 0.3
Epoch 3/50
158/158 [==============================] - 13s 80ms/step - loss: 1.2776 - acc: 0.5
Epoch 4/50
158/158 [==============================] - 13s 80ms/step - loss: 1.0343 - acc: 0.6
Epoch 5/50
158/158 [==============================] - 13s 79ms/step - loss: 0.8178 - acc: 0.6
Epoch 6/50
158/158 [==============================] - 13s 79ms/step - loss: 0.5879 - acc: 0.7
Epoch 7/50
```

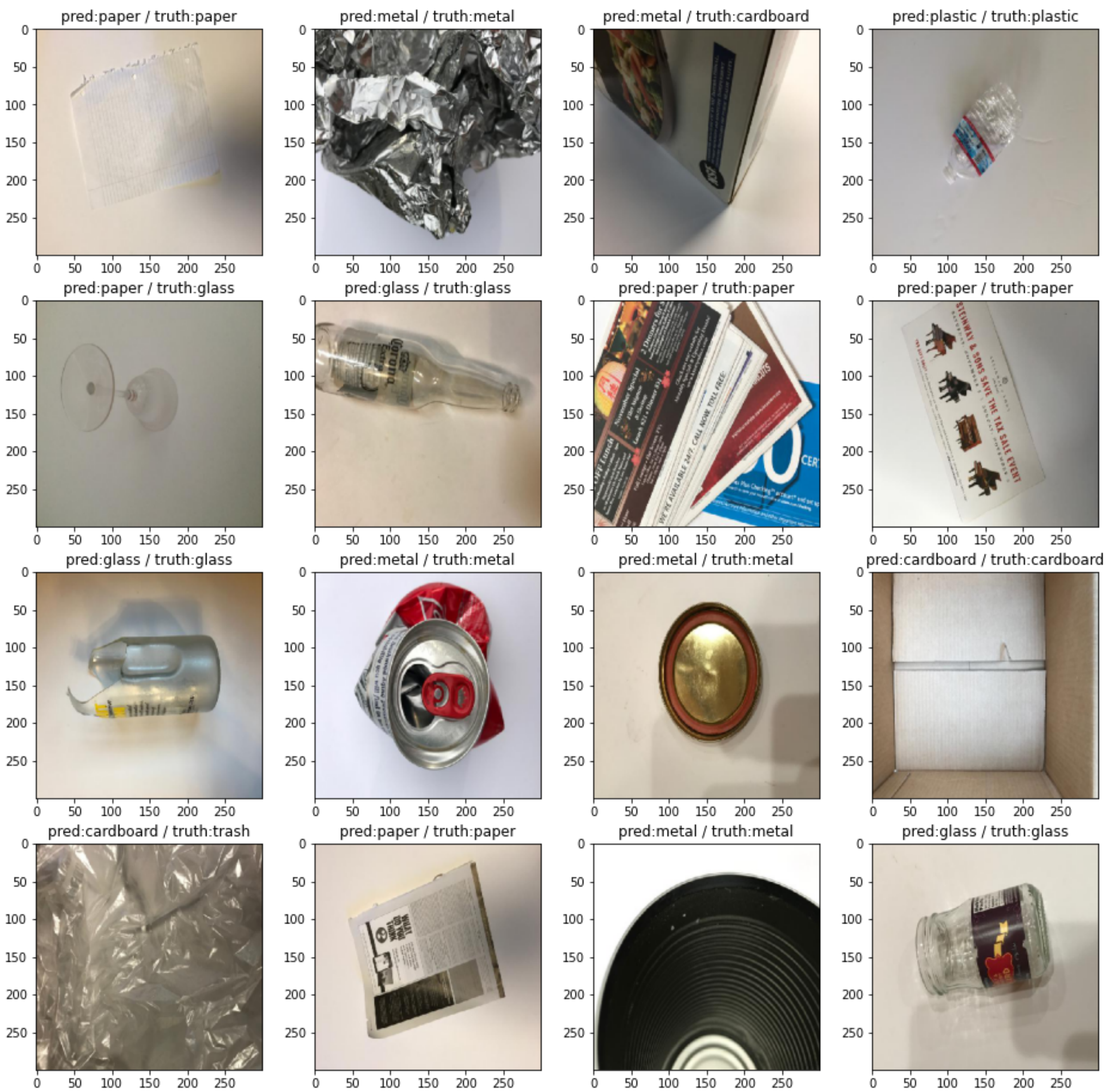
```
Epoch 7/50
158/158 [==============================] - 13s 79ms/step - loss: 0.4811 - acc: 0.8
Epoch 8/50
158/158 [==============================] - 13s 79ms/step - loss: 0.4223 - acc: 0.8
Epoch 9/50
158/158 [==============================] - 13s 80ms/step - loss: 0.4260 - acc: 0.8
Epoch 10/50
158/158 [==============================] - 13s 79ms/step - loss: 0.3269 - acc: 0.8
Epoch 11/50
158/158 [==============================] - 13s 79ms/step - loss: 0.2654 - acc: 0.9
Epoch 12/50
158/158 [==============================] - 13s 79ms/step - loss: 0.2648 - acc: 0.9
```

```
train_acc = history.history['acc']
val_acc = history.history['val_acc']
train_loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(train_acc) + 1)

plt.plot(epochs, train_acc, 'b*-', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label = 'Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, train_loss, 'b*-', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```






```
test_x, test_y = validation_generator.__getitem__(1)

preds = model.predict(test_x)

plt.figure(figsize=(16, 16))
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.title('pred:%s / truth:%s' % (labels[np.argmax(preds[i])], labels[np.argmax(tes
    plt.imshow(test_x[i])
```



```
print("-- Evaluate --")
scores = model.evaluate_generator(validation_generator, steps=5)
print("%s: %.2f%%" %(model.metrics_names[1], scores[1]*100))
```

```
-- Evaluate --
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Mode

acc: 77.50%
```

```
from sklearn.metrics import classification_report
test_true = np.argmax(test_y, axis=1)
test_pred = np.argmax(preds, axis=1)
print(classification_report(test_true, test_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       1.00      0.75      0.86         4
           2       0.80      1.00      0.89         4
           3       0.80      1.00      0.89         4
           4       1.00      1.00      1.00         1
           5       0.00      0.00      0.00         1

    accuracy                           0.81        16
   macro avg       0.68      0.71      0.69        16
weighted avg       0.77      0.81      0.78        16

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Ur
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Ur
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Ur
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.metrics import confusion_matrix

predictions_one_hot = model.predict(test_x)
cm = confusion_matrix(test_true, test_pred)
print(cm)
```

```
[[1 0 1 0 0 0]
 [0 3 0 1 0 0]
 [0 0 4 0 0 0]
 [0 0 0 4 0 0]
 [0 0 0 0 1 0]
 [1 0 0 0 0 0]]
```

```
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)
```

```
# Individual accuracy for class labels
print ("Cardboard Glass Metal Paper Plastic Trash")
ACC = (TP+TN)/(TP+FP+FN+TN)
print (ACC)
```

```
Cardboard Glass Metal Paper Plastic Trash
[0.875  0.9375 0.9375 0.9375 1.     0.9375]
```

Report the model performance against the success criteria that you define.

From the above observation we can see the model has perfomed pretty well with accuracy arround 67%. The Learning curves also showes the model has trained well with validation and Training curves almost converging on both accuracy and loss graphs.

Task 2 (C Task) Analyse and improve the model

Task 2.1 Build an input pipeline for data augmentation

Build a data preprocessing pipeline to perform data augmentation. (You may use Keras ImageDataGenerator or write your own transformations.)

In the above model you would have noticed that i have generated the images in preprocessing section using Imagedatagenerator using various transformation. So we will use the same transformation and observe performance using tensorboard.

```
#Preparing Test, Train and Validation

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.1,
    width_shift_range=0.1,
```

```
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0.1
)

test_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.1
)

train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    seed=0
)

validation_generator = test_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    seed=0
)

labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())

print(labels)
```

```
Found 2276 images belonging to 6 classes.
Found 251 images belonging to 6 classes.
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
!pip uninstall -q tensorboard tb-nightly
!pip install -q tb-nightly tensorboard_plugin_profile
!pip install -U tensorboard_plugin_profile
```

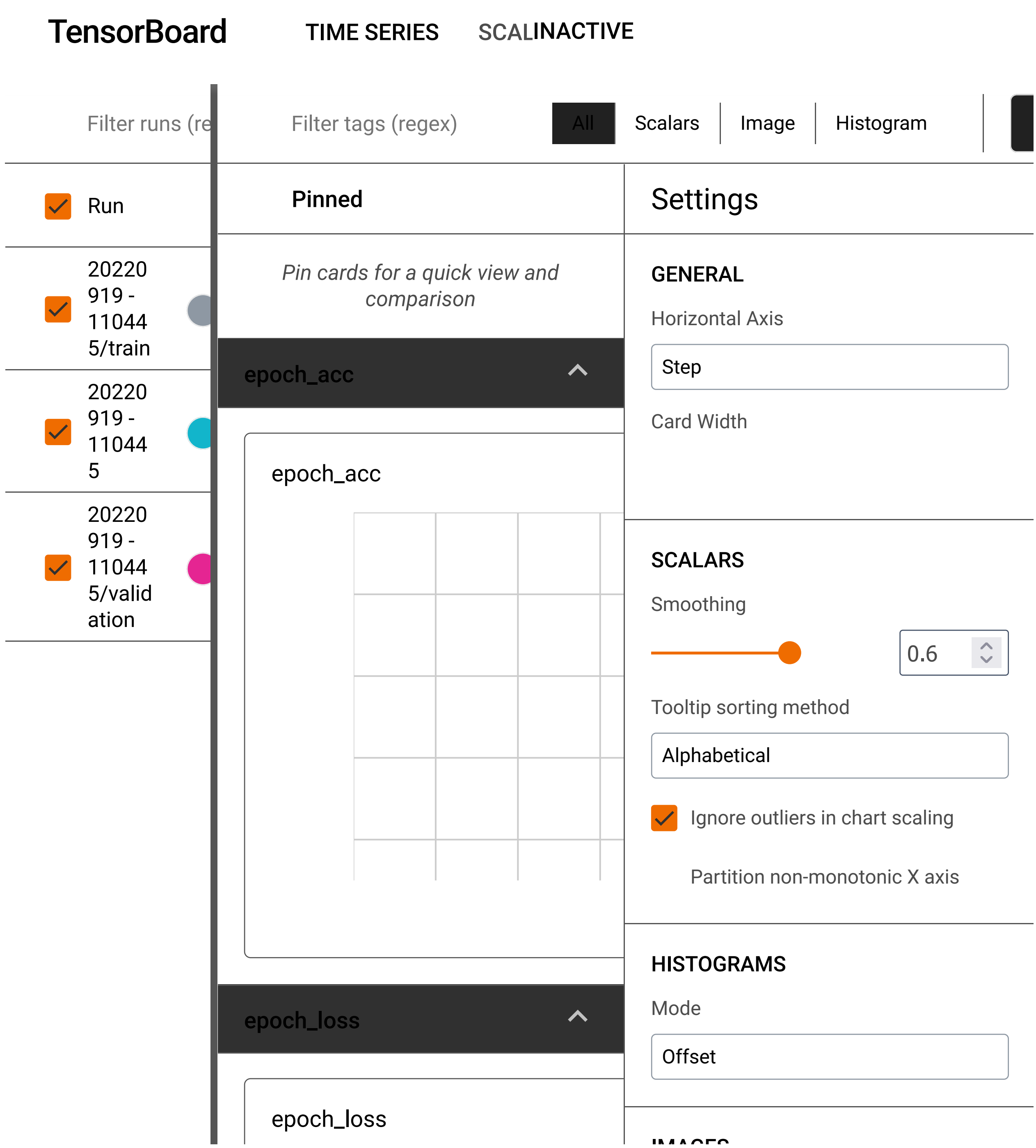
```
Proceed (y/n)? y
WARNING: Skipping tb-nightly as it is not installed.
     |████████████████████████████████| 5.9 MB 28.5 MB/s
     |████████████████████████████████| 5.3 MB 52.3 MB/s
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whee
Requirement already satisfied: tensorboard_plugin_profile in /usr/local/lib/pythor
Requirement already satisfied: gviz-api>=1.9.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packag
```

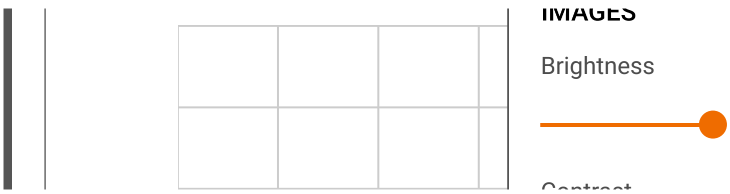
```
!rm -rf ./logs/
from datetime import datetime
import os

logs = 'logs/' + datetime.now().strftime("%Y%m%d - %H%M%S")
CB_TensorBoard = tf.keras.callbacks.TensorBoard(log_dir= logs,profile_batch = '10,15',h


model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])


history = model.fit_generator(train_generator, epochs=50,validation_data=validation_gen
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Mode
  """Entry point for launching an IPython kernel.
Epoch 1/50
143/143 [==============================] - 56s 387ms/step - loss: 0.6787 - acc: 0.
Epoch 2/50
143/143 [==============================] - 46s 324ms/step - loss: 0.6428 - acc: 0.
Epoch 3/50
143/143 [==============================] - 46s 319ms/step - loss: 0.6525 - acc: 0.
Epoch 4/50
143/143 [==============================] - 45s 317ms/step - loss: 0.6225 - acc: 0.
Epoch 5/50
143/143 [==============================] - 46s 320ms/step - loss: 0.6113 - acc: 0.
Epoch 6/50
143/143 [==============================] - 46s 321ms/step - loss: 0.5966 - acc: 0.
Epoch 7/50
143/143 [==============================] - 46s 324ms/step - loss: 0.5723 - acc: 0.
Epoch 8/50
143/143 [==============================] - 46s 320ms/step - loss: 0.5605 - acc: 0.
Epoch 9/50
143/143 [==============================] - 46s 323ms/step - loss: 0.5988 - acc: 0.
Epoch 10/50
143/143 [==============================] - 46s 321ms/step - loss: 0.5420 - acc: 0.
Epoch 11/50
143/143 [==============================] - 46s 319ms/step - loss: 0.5354 - acc: 0.
Epoch 12/50
143/143 [==============================] - 45s 317ms/step - loss: 0.5412 - acc: 0.
Epoch 13/50
143/143 [==============================] - 46s 318ms/step - loss: 0.5512 - acc: 0.
Epoch 14/50
143/143 [==============================] - 45s 317ms/step - loss: 0.5411 - acc: 0.
Epoch 15/50
143/143 [==============================] - 45s 317ms/step - loss: 0.4961 - acc: 0.
Epoch 16/50
143/143 [==============================] - 46s 319ms/step - loss: 0.4790 - acc: 0.
Epoch 17/50
143/143 [==============================] - 45s 317ms/step - loss: 0.4817 - acc: 0.
Epoch 18/50
143/143 [==============================] - 45s 318ms/step - loss: 0.4575 - acc: 0.
Epoch 19/50
143/143 [==============================] - 45s 316ms/step - loss: 0.5069 - acc: 0.
Epoch 20/50
143/143 [==============================] - 45s 316ms/step - loss: 0.4294 - acc: 0.
Epoch 21/50
143/143 [==============================] - 45s 318ms/step - loss: 0.4316 - acc: 0.
```

```
Epoch 22/50
143/143 [==============================] - 45s 317ms/step - loss: 0.4583 - acc: 0.
Epoch 23/50
143/143 [==============================] - 45s 315ms/step - loss: 0.4090 - acc: 0.
Epoch 24/50
143/143 [==============================] - 46s 318ms/step - loss: 0.4191 - acc: 0.
Epoch 25/50
143/143 [==============================] - 45s 316ms/step - loss: 0.4246 - acc: 0.
Epoch 26/50
143/143 [==============================] - 45s 317ms/step - loss: 0.4141 - acc: 0.
Epoch 27/50
143/143 [==============================] - 45s 316ms/step - loss: 0.4086 - acc: 0.
Epoch 28/50
143/143 [==============================] - 45s 318ms/step - loss: 0.3841 - acc: 0.
```

```
%load_ext tensorboard
%tensorboard --logdir=logs
```





From the above the Perfomance Summary in Profile Tab of tensorboard we can see that input time has conumed more time compared to all other operation. The Possible action we could take is reduce the image data to feed in as input but that would evatuallly would reduce the preformance of mode. or we can test the perfomance summary with or with out data augmentaion.

Task 2.2 Compare the performance under equal training time

You may notice that with your pipeline, the model performance improves, but at the cost of a longer training time per epoch. Is the additional training time well spent? Compare the dynamic of model performance (e.g., classification accuracy on the test data) with and without data augmentation, when equal training time is spent in the two scenarios.

The above results in Task 2.1 show observation for data with augmentaion in Preprocessing pipline, now lets try without data augmentaion.

```
#Preparing Test, Train and Validation

train_datagen = ImageDataGenerator()

test_datagen = ImageDataGenerator(rescale=1./255,
    validation_split=0.1)

train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    seed=0
)

validation_generator = test_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    seed=0
)

labels = (train_generator.class_indices)
```

```
labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())

print(labels)
```

```
Found 2527 images belonging to 6 classes.
Found 251 images belonging to 6 classes.
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
!rm -rf ./logs/
from datetime import datetime
import os

logs = 'logs/' + datetime.now().strftime("%Y%m%d - %H%M%S")
CB_TensorBoard = tf.keras.callbacks.TensorBoard(log_dir= logs,profile_batch = '10,15',h
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

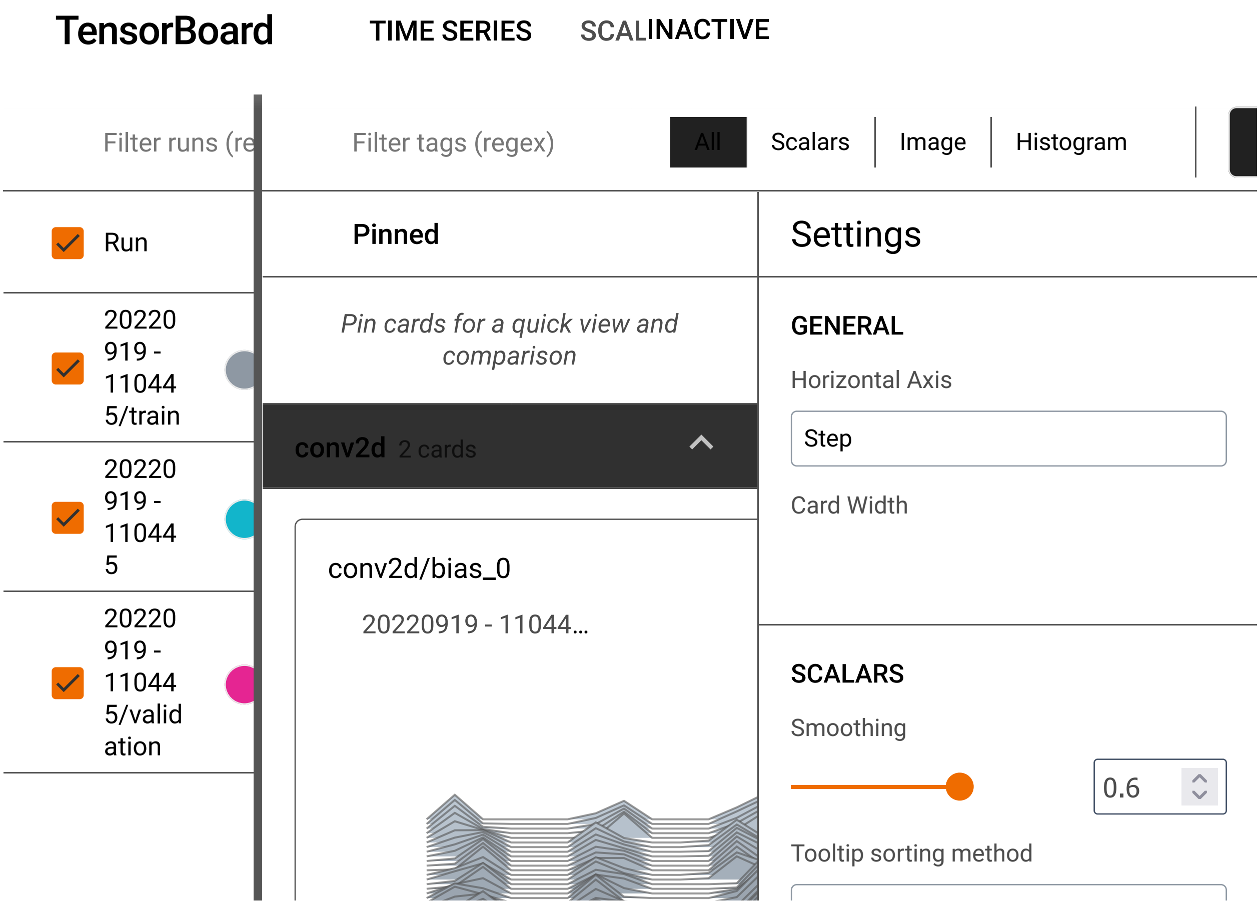
```
history = model.fit_generator(train_generator, epochs=50,validation_data=validation_gen
```

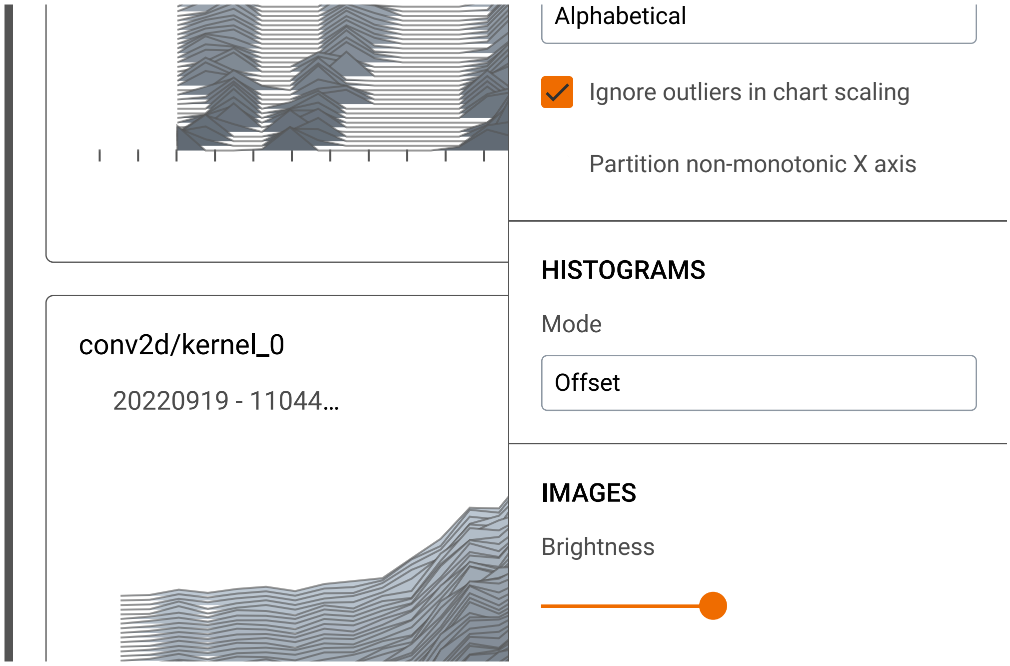
```
Epoch 1/50
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Mode
  """Entry point for launching an IPython kernel.
158/158 [==============================] - 23s 143ms/step - loss: 90.7491 - acc: 0
Epoch 2/50
158/158 [==============================] - 13s 82ms/step - loss: 14.5678 - acc: 0.
Epoch 3/50
158/158 [==============================] - 13s 83ms/step - loss: 7.1742 - acc: 0.7
Epoch 4/50
158/158 [==============================] - 13s 82ms/step - loss: 3.9038 - acc: 0.7
Epoch 5/50
158/158 [==============================] - 13s 81ms/step - loss: 2.3807 - acc: 0.8
Epoch 6/50
158/158 [==============================] - 13s 83ms/step - loss: 1.4501 - acc: 0.8
Epoch 7/50
158/158 [==============================] - 13s 81ms/step - loss: 1.0659 - acc: 0.8
Epoch 8/50
158/158 [==============================] - 14s 90ms/step - loss: 1.0770 - acc: 0.8
Epoch 9/50
158/158 [==============================] - 13s 82ms/step - loss: 0.6542 - acc: 0.9
Epoch 10/50
158/158 [==============================] - 13s 81ms/step - loss: 0.4179 - acc: 0.9
Epoch 11/50
158/158 [==============================] - 13s 82ms/step - loss: 0.4064 - acc: 0.9
Epoch 12/50
158/158 [==============================] - 13s 82ms/step - loss: 0.3646 - acc: 0.9
Epoch 13/50
158/158 [==============================] - 13s 81ms/step - loss: 0.7040 - acc: 0.9
Epoch 14/50
158/158 [==============================] - 13s 82ms/step - loss: 1.1428 - acc: 0.8
Epoch 15/50
158/158 [==============================] - 13s 82ms/step - loss: 0.6459 - acc: 0.9
Epoch 16/50
158/158 [==============================] - 13s 82ms/step - loss: 0.5221 - acc: 0.9
Epoch 17/50
```

```
Epoch 17/50
158/158 [==============================] - 13s 81ms/step - loss: 0.5696 - acc: 0.9
Epoch 18/50
158/158 [==============================] - 13s 81ms/step - loss: 0.8919 - acc: 0.9
Epoch 19/50
158/158 [==============================] - 13s 81ms/step - loss: 1.0972 - acc: 0.9
Epoch 20/50
158/158 [==============================] - 13s 80ms/step - loss: 2.0854 - acc: 0.8
Epoch 21/50
158/158 [==============================] - 13s 81ms/step - loss: 0.8028 - acc: 0.8
Epoch 22/50
158/158 [==============================] - 13s 81ms/step - loss: 0.1993 - acc: 0.9
Epoch 23/50
158/158 [==============================] - 13s 81ms/step - loss: 0.1381 - acc: 0.9
Epoch 24/50
158/158 [==============================] - 13s 81ms/step - loss: 0.0854 - acc: 0.9
Epoch 25/50
158/158 [==============================] - 13s 81ms/step - loss: 0.0599 - acc: 0.9
Epoch 26/50
158/158 [==============================] - 13s 81ms/step - loss: 0.2724 - acc: 0.9
Epoch 27/50
158/158 [==============================] - 13s 81ms/step - loss: 0.3180 - acc: 0.9
Epoch 28/50
158/158 [==============================] - 13s 81ms/step - loss: 0.4873 - acc: 0.9
```

```
%load_ext tensorboard
%tensorboard --logdir=logs
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 1538), started 0:11:12 ago. (Use '!kill
1538' to kill it.)
```

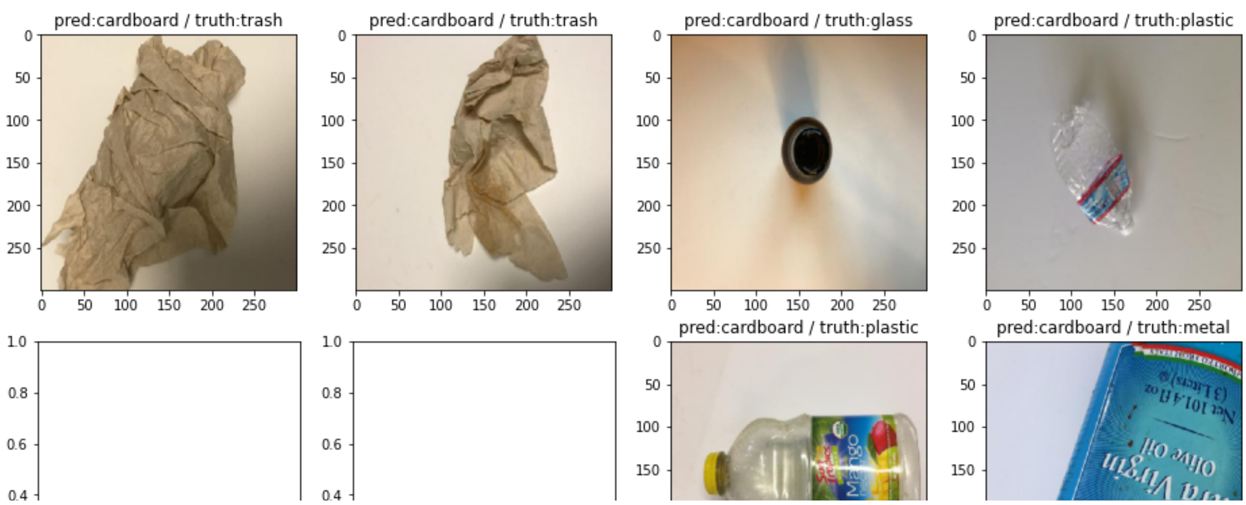


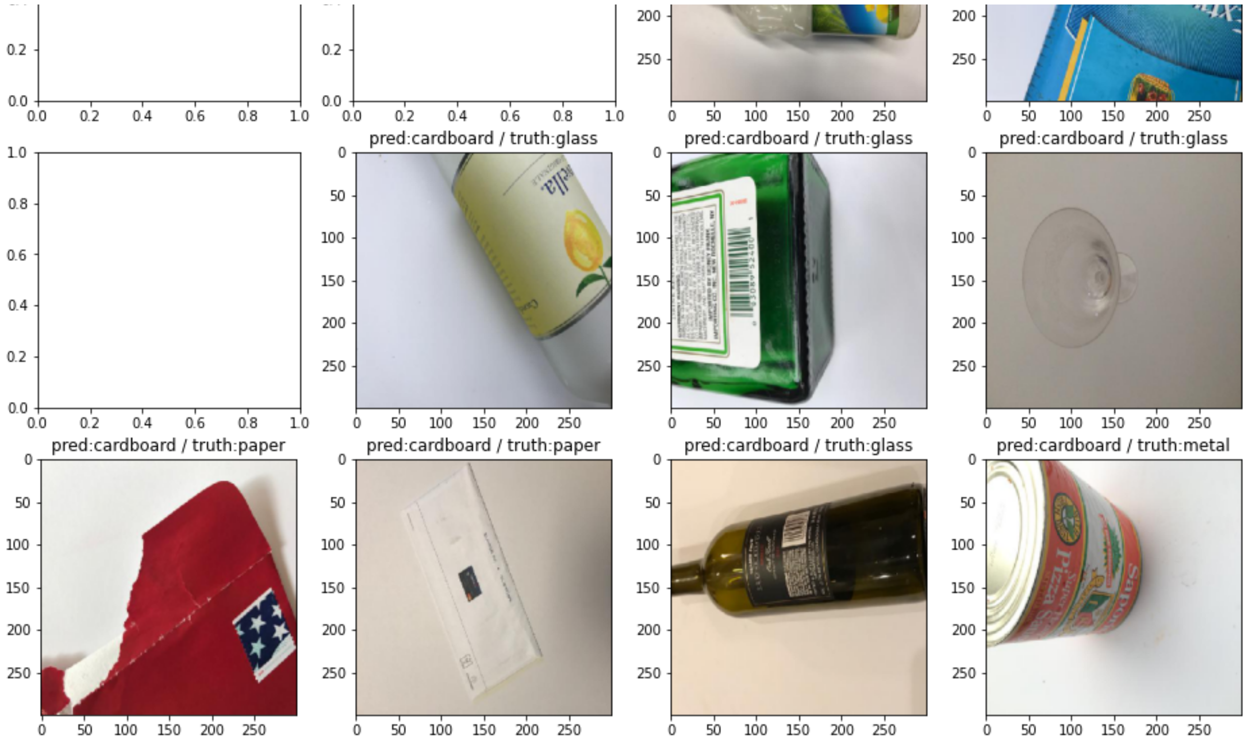


Task 2.3 Identifying model strength and weakness

Identify images that are incorrectly classified by your model. Do they share something in common? How do you plan to improve the model's performance on those images?

```
test_x, test_y = validation_generator.__getitem__(1)

prediction = model.predict(test_x)

plt.figure(figsize=(16, 16))
for i in range(16):
  plt.subplot(4, 4, i+1)
  if np.argmax(prediction[i]) != np.argmax(test_y[i]):
    plt.title('pred:%s / truth:%s' % (labels[np.argmax(prediction[i])], labels[np.argma
    plt.imshow(test_x[i])
```





From the above images we can see the miscalssified images, with thier true Predictive class and truth class. The one common thing we can see in this images is not fully covered images are seen in diffrent angle and objects are not fully covered, as most of our images in training images not cropped and objects are fully seen. This may be one of the reason that these images has been misclassified.

Task 3 (D Task) Improve model generalisability across domains

So far, you have used training and test images from the same source (via random data split). Now collect new test images from a different source. For example, you may take some photos yourself if you used downloaded images before. Otherwise, you may take new photos using a different mobile phone or against a different background.

Show sample images from the original test data and the newly collected test data. In what ways are they different?

Feed the new test data into your model. Report the performance change.

Improve your model so that it generalises better on unseen test images.

```
import cv2
from PIL import Image, ImageOps
import numpy as np

def import_and_predict(image_data, model):
  size = (300,300)
  #image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
  #image1 = np.asarray(image_data)
  #img2 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
  img_resize = (cv2.resize(image_data, dsize=(300, 300),    interpolation=cv2.INTER_CUB
  img_reshape = img_resize[np.newaxis,...]
  prediction = model.predict(img_reshape)

  return prediction
```

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

```
Mounted at /content/gdrive
```

```
from matplotlib import pyplot as plt
img= cv2.imread('/content/gdrive/MyDrive/glass.jpg')

plt.imshow(img)
```

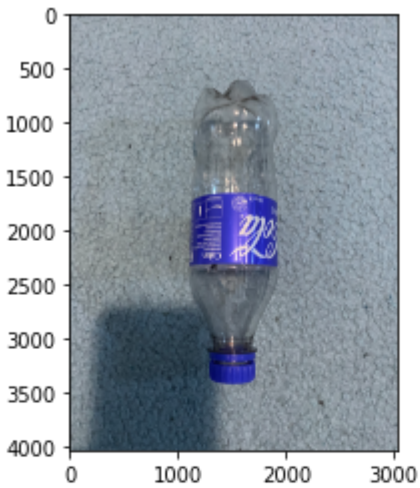
```
<matplotlib.image.AxesImage at 0x7fc738511950>
```

```
predict_test = import_and_predict(img,model)
print(predict_test)
print(np.argmax(predict_test))

labels
```

```
[[0.1491692  0.20592603 0.16559957 0.16978124 0.17849547 0.1310285 ]]
1
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
from matplotlib import pyplot as plt
img= cv2.imread('/content/gdrive/MyDrive/plastic.jpg')

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fc5cfed55d0>
```




```
predict_test = import_and_predict(img,model)
print(predict_test)
print(np.argmax(predict_test))

labels
```

```
[[0.16917731 0.18249026 0.14124504 0.17772408 0.19261447 0.13674879]]
4
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

We can see that my Test image taken from my phone which was uploaded through the google drive is now correctly classified into glass and plastic

Task 4 (HD Task) Build a workable prototype

Build a web/mobile app that people from your city council can use to determine what to recycle. Test your prototype with the target users and report their feedback.

Upload your code into a GitHub repository.

Create a short video presentation about your product.

```
tf.keras.models.save_model(model,'my_model1.hdf5')
```

Once the model is exported , we will create new .py File know as "Webapp.py" which contains the code for running and deploying code in Web application.

After compiling we will run the "Webapp.py" file in comman prompt which will open local host URL , where we can test our model in Web Application