```
# University of St Thomas
# SEIS 764 AI Fall 2022, Prof. Chih Lai
#
# TEAM PROJECT
#
# Jamie Boehme
# Jonathan Ditlevson
# Satya Dampanaboyina
# Swetha Doddi
# Stan Kegel

from datetime import datetime
from tensorflow import keras
from keras import Sequential
from keras.layers import Conv2D, Dense, Flatten, Activation, Dropout, Flatten, MaxPooling2
from keras.losses import CategoricalCrossentropy
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
#from keras.applications.vgg19 import preprocess_input
#from keras.applications.inception_v3 import preprocess_input
#from keras.applications.inception_resnet_v2 import preprocess_input
from keras.applications.resnet import preprocess_input
from PIL import Image
from torchvision import transforms

# parameters for image data

from google.colab import drive
drive.mount('/content/drive')
dir_path = "/content/drive/My Drive/Crops-Clean"


img_class_mode_ = 'categorical'
rotation_range_ = 15       # 15 degrees rotation range for image augmentation
width_shift_range_ = 0.15  # 15% horizontal shift range for image augmentation
height_shift_range_ = 0.15 # 15% pixels veritcal shift range for image augmentation
zoom_range_ = 0.3          # zoom range 0.7 - 1.3 for image augmentation
brightness_range_ = (0.3, 1.0)   # brightness range 50% to 100% for image augmentation
horizontal_flip_ = True    # horizontal flip on for image augmentation
interpolation_mode_ = "lanczos"  # higher quality interpolation for re-scaling (when appli
fill_mode_ = "reflect"  # reflect margin for shifted regions for image augmentation
keep_aspect_ratio_ = True

# parameters for learning
img_batch_size_ = 10
checkpoint_path = "model_checkpoint_" + datetime.now().strftime('%Y%m%d-%H%M%S') + ".h5"

# fix model.py: Line #170 in_channels = inputs.shape[-1]
!git clone https://github.com/rkuo2000/keras-deeplab-v3-plus
%cd keras-deeplab-v3-plus
```

```
# load data and split into training and test/validation data sets
images_data = ImageDataGenerator(
    rescale=1. / 255,
    preprocessing_function = preprocess_input,  # Ref. https://keras.io/api/applications/v
    validation_split = 0.25,
    rotation_range = rotation_range_, width_shift_range = width_shift_range_,
    height_shift_range = height_shift_range_, zoom_range = zoom_range_,
    horizontal_flip = horizontal_flip_, fill_mode = fill_mode_,
    brightness_range = brightness_range_)
trainD_shuffle = images_data.flow_from_directory(
    dir_path, shuffle = True, target_size = (224, 224), interpolation = interpolation_mode
    keep_aspect_ratio = keep_aspect_ratio_,
    class_mode = img_class_mode_, batch_size = img_batch_size_, subset = 'training')
trainD_mask = images_data.flow_from_directory(
    dir_path, shuffle = True, target_size = (224, 224), interpolation = interpolation_mode
    keep_aspect_ratio = keep_aspect_ratio_,
    class_mode = img_class_mode_, batch_size = img_batch_size_, subset = 'training')
trainD_noshuffle = images_data.flow_from_directory(
    dir_path, shuffle = False, target_size = (224, 224), interpolation = interpolation_mod
    keep_aspect_ratio = keep_aspect_ratio_,
    class_mode = img_class_mode_, batch_size = img_batch_size_, subset = 'training')
testD_noshuffle = images_data.flow_from_directory(
    dir_path, shuffle = False, target_size = (224, 224), interpolation = interpolation_mod
    keep_aspect_ratio = keep_aspect_ratio_,
    class_mode = img_class_mode_, batch_size = img_batch_size_, subset = 'validation')
testD_mask = images_data.flow_from_directory(
    dir_path, shuffle = False, target_size = (224, 224), interpolation = interpolation_mod
    keep_aspect_ratio = keep_aspect_ratio_,
    class_mode = img_class_mode_, batch_size = img_batch_size_, subset = 'validation')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
Cloning into 'keras-deeplab-v3-plus'...
remote: Enumerating objects: 375, done.
remote: Total 375 (delta 0), reused 0 (delta 0), pack-reused 375
Receiving objects: 100% (375/375), 5.12 MiB | 33.17 MiB/s, done.
Resolving deltas: 100% (202/202), done.
/content/keras-deeplab-v3-plus/keras-deeplab-v3-plus
Found 635 images belonging to 30 classes.
Found 635 images belonging to 30 classes.
Found 635 images belonging to 30 classes.
Found 194 images belonging to 30 classes.
Found 194 images belonging to 30 classes.
```

```
train_img_class_count = len(trainD_noshuffle.class_indices)
test_img_class_count = len(testD_noshuffle.class_indices)
if (train_img_class_count != test_img_class_count):
    raise Exception("Training and Testing Data Sets Not Aligned.")
img_class_count = test_img_class_count
print("image class count", img_class_count)
```

```
image class count 30
```

```
# get pre-trained CNN
#from keras.applications import InceptionResNetV2
#from keras.applications import InceptionV3
from keras.applications import ResNet50V2
cnn = ResNet50V2(weights = 'imagenet', include_top = False, input_shape = (224, 224, 3))
cnn.trainable = False


opt = keras.optimizers.Adam(learning_rate=0.0001)
epochs_ = 100

model = Sequential([
    cnn,
    Flatten(),
    Dense(500, activation = 'relu'),
    Dense(360, activation = 'relu'),
    Dense(180, activation = 'relu'),
    Dense(90, activation = 'relu'),
    Dropout(.2),
    Dense(img_class_count, activation = 'softmax')
])

model.compile(loss = CategoricalCrossentropy(),
    optimizer=opt,
     metrics = ['accuracy'])


model.summary()
```

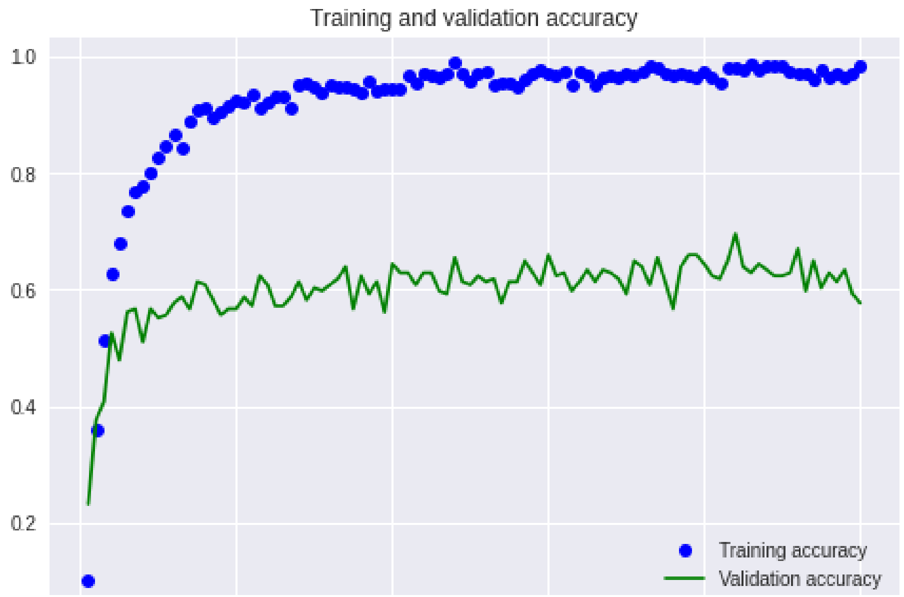
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 flatten_1 (Flatten)         (None, 100352)            0

 dense_5 (Dense)             (None, 500)               50176500

 dense_6 (Dense)             (None, 360)               180360

 dense_7 (Dense)             (None, 180)               64980

 dense_8 (Dense)             (None, 90)                16290

 dropout_3 (Dropout)         (None, 90)                0

 dense_9 (Dense)             (None, 30)                2730

=================================================================
Total params: 74,005,660
Trainable params: 50,440,860
Non-trainable params: 23,564,800
_________________________________________________________________
```

```
# set up logging
# and train nerual net

%load_ext tensorboard

log_dir = 'logs/batch/' + datetime.now().strftime('%Y%m%d-%H%M%S') + '/train'
tensorboard_callback = keras.callbacks.TensorBoard(log_dir = log_dir)
checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath = checkpoint_path, monitor="val_accuracy", batch_size = img_batch_size_,
    verbose=1, mode="max", save_weights_only=True, save_best_only=True)
history = model.fit(trainD_shuffle, epochs = epochs_, validation_data = testD_noshuffle,
    callbacks=[tensorboard_callback, checkpoint_callback])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/100
35/64 [===============>..............] - ETA: 8s - loss: 3.3767 - accuracy: 0.0857/us
  " Skipping tag %s" % (size, len(data), tag)
64/64 [==============================] - ETA: 0s - loss: 3.3256 - accuracy: 0.1008
Epoch 1: val_accuracy improved from -inf to 0.23196, saving model to model_checkpoint
64/64 [==============================] - 29s 398ms/step - loss: 3.3256 - accuracy: 0.
Epoch 2/100
64/64 [==============================] - ETA: 0s - loss: 2.4058 - accuracy: 0.3606
Epoch 2: val_accuracy improved from 0.23196 to 0.37629, saving model to model_checkpo
64/64 [==============================] - 23s 366ms/step - loss: 2.4058 - accuracy: 0.
Epoch 3/100
64/64 [==============================] - ETA: 0s - loss: 1.7696 - accuracy: 0.5134
Epoch 3: val_accuracy improved from 0.37629 to 0.40722, saving model to model_checkpo
64/64 [==============================] - 23s 360ms/step - loss: 1.7696 - accuracy: 0.
Epoch 4/100
64/64 [==============================] - ETA: 0s - loss: 1.4080 - accuracy: 0.6268
Epoch 4: val_accuracy improved from 0.40722 to 0.52577, saving model to model_checkpo
64/64 [==============================] - 23s 353ms/step - loss: 1.4080 - accuracy: 0.
Epoch 5/100
64/64 [==============================] - ETA: 0s - loss: 1.0845 - accuracy: 0.6787
Epoch 5: val_accuracy did not improve from 0.52577
64/64 [==============================] - 22s 343ms/step - loss: 1.0845 - accuracy: 0.
Epoch 6/100
64/64 [==============================] - ETA: 0s - loss: 0.9616 - accuracy: 0.7339
Epoch 6: val_accuracy improved from 0.52577 to 0.56186, saving model to model_checkpo
64/64 [==============================] - 24s 375ms/step - loss: 0.9616 - accuracy: 0.
Epoch 7/100
64/64 [==============================] - ETA: 0s - loss: 0.7936 - accuracy: 0.7669
Epoch 7: val_accuracy improved from 0.56186 to 0.56701, saving model to model_checkpo
64/64 [==============================] - 23s 355ms/step - loss: 0.7936 - accuracy: 0.
Epoch 8/100
64/64 [==============================] - ETA: 0s - loss: 0.7635 - accuracy: 0.7764
Epoch 8: val_accuracy did not improve from 0.56701
64/64 [==============================] - 22s 343ms/step - loss: 0.7635 - accuracy: 0.
Epoch 9/100
64/64 [==============================] - ETA: 0s - loss: 0.7389 - accuracy: 0.8016
Epoch 9: val_accuracy did not improve from 0.56701
64/64 [==============================] - 23s 358ms/step - loss: 0.7389 - accuracy: 0.
Epoch 10/100
64/64 [==============================] - ETA: 0s - loss: 0.5936 - accuracy: 0.8283
Epoch 10: val_accuracy did not improve from 0.56701
64/64 [==============================] - 22s 341ms/step - loss: 0.5936 - accuracy: 0.
```

```
Epoch 11/100
64/64 [==============================] - ETA: 0s - loss: 0.4985 - accuracy: 0.8472
Epoch 11: val_accuracy did not improve from 0.56701
64/64 [==============================] - 22s 343ms/step - loss: 0.4985 - accuracy: 0.
Epoch 12/100
64/64 [==============================] - ETA: 0s - loss: 0.4155 - accuracy: 0.8661
Epoch 12: val_accuracy improved from 0.56701 to 0.57732, saving model to model_checkp
64/64 [==============================] - 23s 355ms/step - loss: 0.4155 - accuracy: 0.
Epoch 13/100
64/64 [==============================] - ETA: 0s - loss: 0.5311 - accuracy: 0.8425
Epoch 13: val_accuracy improved from 0.57732 to 0.58763, saving model to model_checkp
64/64 [==============================] - 23s 354ms/step - loss: 0.5311 - accuracy: 0.
Epoch 14/100
64/64 [==============================] - ETA: 0s - loss: 0.3682 - accuracy: 0.8898
```

```
# CHANGE:  Plot training and validation accuracy over epochs
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label = "Training accuracy")
plt.plot(epochs, val_accuracy, "g", label = "Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label = "Training loss")
plt.plot(epochs, val_loss, "g", label = "Validation loss")
plt.yscale("log")
plt.legend()
plt.show()
```



```
# restore checkpoint model
#model = keras.models.load_model(checkpoint_path)
model.load_weights(checkpoint_path)
```

```
# run prediction based on training data
train_scores = model.predict(trainD_noshuffle)
train_pred_labels = train_scores.argmax(axis = 1)
```

```
    64/64 [==============================] - 17s 255ms/step
```

```
# evaluate trained network
print('')
print('')
print('Model Evaluation Using Training Data:')

print("Accuracy Score")
print(accuracy_score(trainD_noshuffle.labels, train_pred_labels))

print("Confusion Matrix")
print(confusion_matrix(trainD_noshuffle.labels, train_pred_labels))
```

```
  0  0  0  0  0  0]
[ 0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]

[ 0  0  0 18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]
[ 0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]
[ 0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]
[ 0  0  0  0  0  0 29  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 17  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 22  0  0  0  0  0  0  0  0  0  0  1
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 24  0  0  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0
   0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 18  0  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 22  0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  18  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0 18  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0 18  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0 20  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   0  0  0  0 20  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0  0 23]]
```

```
# run prediction based on test data
test_scores = model.predict(testD_noshuffle)
test_pred_labels = test_scores.argmax(axis = 1)
```

```
    20/20 [==============================] - 6s 327ms/step
```

```
# print confusion matrix
print('')
print('')
print('Model Evaluation Using Test Data:')
print("Accuracy Score")
print(accuracy_score(testD_noshuffle.labels, test_pred_labels))

print("Confusion Matrix")
print(confusion_matrix(testD_noshuffle.labels, test_pred_labels))
```

```
Model Evaluation Using Test Data:
Accuracy Score
0.6082474226804123
Confusion Matrix
[[3 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0]
 [0 0 3 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 1 0]
 [0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 4 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 5 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 3 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 0 0 1 0 0 0 2 0 0 0 0 0 1 0 1 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 4 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 1 1 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 4 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 2]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 1 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 4 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 3 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 4 0 0 0]
 [0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 3 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 6]]
```

```
# Create lookup to convert class labels (index-numbers) to string labels
nameToLabelDict = testD_noshuffle.class_indices
labelToNameDict = dict([(value, key) for key, value in nameToLabelDict.items()])


# set up lists of colors and styles for use in plotting
from itertools import cycle
from itertools import product
from sklearn.linear_model import LassoCV
import matplotlib as mpl
```

```
color_list = [\
    "b", "r", "g", "c", "m", \
    "skyblue", "pink", "lime", "cyan", "magenta", \
    "navy", "brown", "olive", "orange", "purple"]
    # note "cyan" is brighter than "c", "magenta" is brighter than "m"

# repeat each style times number of colors
base_style_list = ['solid', 'dotted', 'dashed', 'dashdot']
line_style_list = \
    [cartesian[0] for  cartesian in product(base_style_list, color_list)]

# set up plot styling
mpl.style.use('seaborn')

# Plot ROC (Receiver Operating Characteristic) Curve and compute area under curve for each
# using test data
import matplotlib as mpl
import seaborn as sms
mpl.style.use('seaborn')
import matplotlib.pyplot as plt
# Compute ROC curve and ROC area for each class
from sklearn.metrics import roc_curve, auc
fig, ax = plt.subplots()
for i, c, l in zip(range(img_class_count), cycle(color_list), line_style_list):
    fpr_, tpr_, _ = roc_curve(testD_noshuffle.labels, test_scores[:, i], pos_label=i)
    auc_ = auc(fpr_, tpr_)
    label_ = 'class ' + str(i) + " (" + labelToNameDict[i] + ") AUC = %0.2f)" % auc_
    plt.plot(fpr_, tpr_, marker='.', label=label_, color = c, linestyle = l)
plt.title("Test Data ROC Curve", color='C6')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
ax.legend(bbox_to_anchor = (1.0,1.0), loc = "upper left", fontsize = "x-small")
#plt.legend()
plt.show()
```

Test Data ROC Curve

```
# Print precision, recall, F-score for each class
# using test data
from sklearn.metrics import classification_report
scores = classification_report(testD_noshuffle.labels, test_pred_labels)
print(scores)
```

```
              precision    recall  f1-score   support

           0       1.00      0.38      0.55         8
           1       0.44      0.57      0.50         7
           2       0.75      0.43      0.55         7
           3       1.00      1.00      1.00         5
           4       0.67      0.57      0.62         7
           5       0.45      0.71      0.56         7
           6       1.00      0.78      0.88         9
           7       0.50      0.62      0.56         8
           8       0.23      0.60      0.33         5
           9       0.50      0.29      0.36         7
          10       0.67      0.40      0.50         5
          11       0.67      0.80      0.73         5
          12       0.80      0.57      0.67         7
          13       0.67      0.67      0.67         6
          14       0.86      0.75      0.80         8
          15       0.80      0.67      0.73         6
          16       1.00      0.86      0.92         7
          17       0.38      0.60      0.46         5
          18       0.45      0.71      0.56         7
          19       0.55      0.86      0.67         7
          20       0.50      0.40      0.44         5
          21       1.00      0.83      0.91         6
          22       0.57      0.57      0.57         7
          23       0.22      0.29      0.25         7
          24       0.67      0.67      0.67         6
          25       1.00      0.50      0.67         6
          26       0.80      0.80      0.80         5
          27       1.00      0.33      0.50         6
          28       0.33      0.17      0.22         6
          29       0.60      0.86      0.71         7

    accuracy                           0.61       194
   macro avg       0.67      0.61      0.61       194
weighted avg       0.67      0.61      0.61       194
```

✓ 0s completed at 3:15 PM ● ✕