Lab package 4: Image classification

Wednesday, November 10, 2021 6:08 PM

Labs 1-3 applied spatial analysis to identify spatial patterns or conceptual objects that carry geographic meanings in response to questions on what we have here, how things are distributed, where are things of interest (such as hotspots, high service call types, or zones of demographic similarity).

This lab (Lab 4) differs from the previous labs in its goal to classify what is in images; or which image has geospatial things of interest to us. This is a vibrant field of research in machine learning or computer vision. This is the most basic image classificaiton. Most computer vision examples aim at classifying images with different animals (cats, dogs, etc.) or things (clothes, shoes, etc.). In Spatial Data Science, common applications classify images with geospatial things of interest, like buildings, traffic signs, bridges, trails, trees, vehicles, pedestrians, etc. Advances in image classification lays the ground work for self driving cars and robotic navigation. Following image classificaiton is image segmentation (also called semantic segmentation) which aims to delineate objects in individual images, such as identify individual tress, trails, or cars.

Our lab will work to classify images with or without buildings. The lab will give you a simple exercise on two basic approaches to deep learning: (1) a neural network with multiple hidden layers; and (2) a convolutionary neural network. Like the previous labs, the course emphasizes intuitions on thinking through the problem and how the algorithms work. Graduate students who would like to include machine learning in thesis projects will need to dig into the math that grounds the chosen machine learning algorithm in your thesis. However, the math is not required for the graduate projects in the class.

Acknowledgment: TA Yalin prepared the Tiff images and label files. He also provided the first draft of the codes below.

```
[1]  import tensorflow as tf
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import os
     import cv2
```

```
[2]  np.version.version
```

```
[3]  labels = pd.read_excel(r'/content/drive/MyDrive/package4/TiffLabels.xlsx')
```

```
[4]  labels
```

```
[5]  image_dir = '/content/drive/MyDrive/package4/Tiff'
```

```
[6]  image_list = list(labels.file_name)
     droplist = []
     for item in image_list:
       img = cv2.imread(os.path.join(image_dir, item))
       if img.shape != (64, 64, 3):
         print("removing..", item, "from labels dataframe")
         droplist.append(item)
     labels = labels[~labels.file_name.isin(droplist)]
```

Q# 1: Why do we need to check image dimensions?

Q#2: How many images need to be dropped from the list? _______

```
[7]  labels
```

```
[8]  from sklearn.model_selection import train_test_split
     X = labels.file_name
     y = labels.y_label
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=15, stratify=y)
```

```
[9]  X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=25, stratify=y_train)
```

```
[10] X_train.shape, X_val.shape, X_test.shape, y_train.sum(), y_val.sum(), y_test.sum()
```

Q#3 What is the difference between X_train and X_train_img:

A. X_train and X_train_img are the same
B. X_train has the data for classification
C. X_train_img has the data for classification
D. X_train_img lists all images in the training data set.

```
[11] X_train_img = [cv2.imread(os.path.join(image_dir, x)) for x in X_train]
     X_train_img = np.array(X_train_img)
```

```
[12] X_val_img = [cv2.imread(os.path.join(image_dir, x)) for x in X_val]
     X_val_img = np.array(X_val_img)
```

```
[13] X_test_img=[cv2.imread(os.path.join(image_dir, x)) for x in X_test]
     X_test_img = np.array(X_test_img)
```

```
[14] y_train = np.asarray(y_train)
     y_val = np.asarray(y_val)
     y_test = np.asarray(y_test)
```

```
[15] fig = plt.figure(figsize=(15, 15))
     for i in range(25):
       plt.subplot(5, 5, i+1)
       plt.xticks([])
       plt.yticks([])
       plt.grid(False)
       plt.imshow(X_train_img[-i])
       plt.xlabel(y_train[-i])
     plt.show()
```

Q#4: Why don't we have y_train_img:

A. y_train is the same as y_train_img
B. y_train _img is part of X_train_img
C. y_train is a list of labels, not images
D. None of the above

```
[16] from tensorflow import keras
     from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
     from keras.losses import SparseCategoricalCrossentropy
     from keras.metrics import SparseCategoricalAccuracy
```

```
[17] X_train_img = X_train_img/255.0
     X_val_img = X_val_img/255.0
     X_test_img = X_test_img/255.0
```

Q#5: How many hidden layers are in the neural network? _________

```
[18] model = keras.Sequential()
     model.add(Flatten(input_shape=(64, 64, 3)))
     model.add(Dense(256, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(128, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(64, activation='softplus'))
     model.add(Dense(32, activation='sigmoid'))
```

Q#5: How many hidden layers are in the neural network? ________

```
[18] model = keras.Sequential()
     model.add(Flatten(input_shape=(64, 64, 3)))
     model.add(Dense(256, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(128, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(64, activation='softplus'))
     model.add(Dense(32, activation='sigmoid'))
     model.add(Dense(2, activation='softmax'))
```

```
[19] opt = keras.optimizers.SGD(learning_rate=0.01)
```

```
[20] model.compile(optimizer=opt,
                   loss=SparseCategoricalCrossentropy(),
                   metrics=SparseCategoricalAccuracy())
```

```
[21] history = model.fit(X_train_img, y_train, epochs=200, verbose=1, validation_data=(X_val_img, y_val))
```

```
[22] history.history
```

Q#6: How many images does each epoch consider? _____

Q#7: If the training accuracy increases but the validation accuracy more or less remain the same, what does it imply about the model we are building? Select all possible answers

A. Overfitting
B. Underfitting
C. High bias
D. High variance
E. All of the above

```
[23] acc = history.history['sparse_categorical_accuracy']
     val_acc = history.history['val_sparse_categorical_accuracy']
     loss = history.history['loss']
     val_loss = history.history['val_loss']

     epochs_range = range(200)

     plt.figure(figsize=(15, 15))
     plt.subplot(2, 2, 1)
     plt.plot(epochs_range, acc, label='Training Accuracy')
     plt.plot(epochs_range, val_acc, label='Validation Accuracy')
     plt.legend(loc='lower right')
     plt.title('Training and Validation Accuracy')

     plt.subplot(2, 2, 2)
     plt.plot(epochs_range, loss, label='Training Loss')
     plt.plot(epochs_range, val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.title('Training and Validation Loss')
     plt.show()
```

```
[24] model.summary()
```

Q#8. What is the total number of parameters that the training process attempts to estimate? __________________

```
[25] test_loss, test_acc = model.evaluate(X_test_img, y_test, verbose=2)

     13/13 - 0s - loss: 0.8001 - sparse_categorical_accuracy: 0.5875 - 166ms/epoch - 13ms/step
```

```
[26] y_pred = model.predict(X_test_img)
```

```
[27] y_pred[0:10]
```

```
[28] plt.figure(figsize=(15,15))
     for i in range(30):
         plt.subplot(6,5,i+1)
         plt.xticks([])
         plt.yticks([])
         plt.grid(False)
         plt.imshow(X_test_img[-i], cmap=plt.cm.binary)
         plt.xlabel(f'Observed : {y_test[-i]}' + '\n' + f'Predict: {bool(np.argmax(y_pred[-i]))}' )
     plt.show()
```

```
[29] cnnmodel = tf.keras.Sequential()

     cnnmodel.add(Conv2D(filters=10, kernel_size=3, padding='same', activation='relu', input_shape=(64, 64, 3)))
     cnnmodel.add(MaxPool2D(pool_size=2))

     cnnmodel.add(Conv2D(filters=10, kernel_size=3, padding='same', activation='relu'))
     cnnmodel.add(MaxPool2D((2, 2)))
     cnnmodel.add(Dropout(0.2))

     cnnmodel.add(Flatten())
     cnnmodel.add(Dense(128, activation='relu'))
     cnnmodel.add(Dense(2, activation='softmax'))
```

Q#9: How many hidden layers are in the convolutionary neural network? _______________________

```
[30] opt = keras.optimizers.Adam(learning_rate=0.0001)
     cnnmodel.compile(optimizer=opt, loss=keras.losses.SparseCategoricalCrossentropy(), metrics=SparseCategoricalAccuracy())
```

```
[31] cnnhistory = cnnmodel.fit(X_train_img, y_train, epochs=200, verbose=1, validation_data=(X_val_img, y_val))
```

```
[32] cnnmodel.summary()
```

Q#10: What is the total number of parameters that the CNN training attempt to estimate? ____________

```
[33] test_loss, test_acc = cnnmodel.evaluate(X_test_img, y_test, verbose=2)
```

```
[34] acc = cnnhistory.history['sparse_categorical_accuracy']
     val_acc = cnnhistory.history['val_sparse_categorical_accuracy']
     loss = cnnhistory.history['loss']
     val_loss = cnnhistory.history['val_loss']

     epochs_range = range(200)

     plt.figure(figsize=(15, 15))
     plt.subplot(2, 2, 1)
     plt.plot(epochs_range, acc, label='Training Accuracy')
     plt.plot(epochs_range, val_acc, label='Validation Accuracy')
     plt.legend(loc='lower right')
     plt.title('Training and Validation Accuracy')

     plt.subplot(2, 2, 2)
     plt.plot(epochs_range, loss, label='Training Loss')
     plt.plot(epochs_range, val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.title('Training and Validation Loss')
     plt.show()
```

```
[35] y_pred = cnnmodel.predict(X_test_img)
```

```
[36] plt.figure(figsize=(15,15))
     for i in range(30):
         plt.subplot(6,5,i+1)
         plt.xticks([])
         plt.yticks([])
         plt.grid(False)
         plt.imshow(X_test_img[-i], cmap=plt.cm.binary)
         plt.xlabel(f'Observed : {y_test[-i]}' + '\n' + f'Predict: {bool(np.argmax(y_pred[-i]))}' )
     plt.show()
```