Lab 5: Classification

(10% of final score, due by Mar. 26 11:59 PM)

Upload your images (both input and output) and **.py* files to the folder of **Lab 5** under **Assessments > Assignment** in the D2L system. Name the **.py* files according to the series numbers of the questions. If on Google CoLab, please keep the code file output when saving the notebook and share the code with: abbas.salehitangrizi@gmail.com. The **.py* files should follow the PEP-8 Style Guide for Python Code. [https://www.python.org/dev/peps/pep-0008/]

1. Texture Classification and Feature Selection.

   https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_glcm.html#sphx-glr-auto-examples-features-detection-plot-glcm-py

   This example uses GLCM correlation and GLCM dissimilarity to describe features of sky and grass. Use this example to descript features of other textures: dessert and rocks. Find images of dessert and rocks online. For different textures, tune the parameters of function greycomatrix() to separate the feature points in the feature space. If the two properties (correlation and dissimilarity) do not work very well, try other properties of GLCM skimage.feature.greycoprops(), such as contrast, homogeneity, and energy. Output the image of feature space with feature points representing different textures: sky, grass, dessert, and rocks.

2. Classification Evaluation. Download the file < plot_face_recognition.py > from the link:

   https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

   Run the *py* file and answer the following questions:

   a. Explain the definitions of precision, recall, and F_1-score in the results of classification.
   b. Explain the confusion matrix, the meaning of diagonal values and non-diagonal values.
   c. Change the test_size from 0.25 to 0.4 and describe the change of results of classification.
   d. In the part < # Train a SVM classification model >, replace SVC classifier with RandomForestClassifier. [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html]. Set parameters ***n_estimators***, ***criterion, and max_depth*** for grid search using **model_selection.GridSearchCV**. Print out ***best_estimator_*.** Compare the classification performance of SVC classifier and Random Forest classifier.

3. Classify images of clothing

   https://www.tensorflow.org/tutorials/keras/classification

   Read the example file <classification.ipynb> and run it in Google CoLab. Implement the following tasks in the file <classification.ipynb> and do not change the model.

   - Use model.summary() to check out the model details.

a) How is the number of total parameters 101,770 calculated?

- Use **TensorBoard** to visualize accuracy and loss function. Here is a tutorial on tensorboard: https://www.tensorflow.org/tensorboard/get_started

  In the <classification.ipynb> file, add some codes to implement the following functions:

  b) Load the TensorBoard notebook extension

  c) Clear any logs from previous runs

  d) When training with Keras's Model.fit(), adding the tf.keras.callbacks.TensorBoard callback ensures that logs are created and stored.

  e) Start TensorBoard

  The TensorBoard / Scalars would show how the loss and accuracy change with every epoch.

  f) Set epochs=20 in the model.fit(), observe the loss and accuracy changes.

- Hyperparameter Tuning with the **HParams Dashboard**. Read the tutorial:

  https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams

  g) Experiment with three hyperparameters in the model: number of units in the first dense layer, activation function in the first dense layer, and optimizer.

  h) Log hyperparameters and metrics.

  i) Use a grid search to try multiple experiments and log all hyperparameters under one parent directory.

  j) Visualize the results in TensorBoard's HParams plugin.

- Log a custom learning rate. Read the tutorial:
  https://www.tensorflow.org/tensorboard/scalars_and_keras

  k) Create a file writer, using tf.summary.create_file_writer().
  l) Define a custom learning rate function. This will be passed to the Keras LearningRateScheduler callback.
  m) Inside the learning rate function, use tf.summary.scalar() to log the custom learning rate.
  n) Pass the LearningRateScheduler callback to Model.fit(). Set epochs=100.

  The TensorBoard / Scalars would show the learning rate changing with every epoch.