DA5402: Assignment 5

We have learned how to use DVC for setting up continuous integration. DVC allows us to track and version large and several data files. DVC allows us to setup experimental pipeline DAGs and ensure that only the relevant pieces get executed when some part of the pipeline gets updated. Moreover, DVC allows us to compare, reproduce and rerun experiements just by the click of the button through the VSCode plugin. Let's put whatever we studied into an exercise to strengthen our understanding of continuous integration using DVC.

Let's build a multi-classifier on CIFAR-10 dataset using the DVC's CI pipelining. We may use the CNN method to learning the classification model. One such example implementation is here.

*** A formal logging mechanism is mandatory**

*** Failing to adhere to Python coding standards will attract penalty**

*** Elaborate code comments are mandatory**

Task 1 [5 points]

Download the entire CIFAR dataset into a folder with 10 sub-folders, representing each class. Every sub-folder should contain the 32x32 images in JPG or PNG formats. Strictly, dataframe style of dataset handling is not allowed under this task.

Task 2 [10 points]

Draw a random sample (20000 items without replacement) from the available 60000 images to create the first dataset. Likewise, create two more datasets. Essentially, the datasets are three partitions of the original downloaded content. Push those three partitions into DVC and labels them as v1, v2 and v3. When the data is checked into DVC/Git, you are checking in the images files. You are not allowed to use dataframes yet.

Task 3 [25 points]

Setup the DVC pipeline with the following stages:

- *Pull Data from DVC:* Pull an arbitrary version of the dataset from DVC. The version string should be a configuration parameter via YAML.
  - Input: nothing
  - Output: 20k partition (this can be dataframe)
- *Data Preparation*: Create a random train, val and test splits, whose proportions are configured via YAML configuration.
  - Input: 20k partition.
  - Output: <tr, va, te> triplet datasets.
- *Model Training & Tuning*: Setup the model training process with the train dataset. Allow different hyperparameters such as but not limited to {lr, #conv_layers,

#conv_filters, #kernel_sizes} via the YAML config. Tune the model using the val-dataset, where you may choose any two hyperparameters with at least 3 choices for the tuning setup.

    - Input: <tr, va> triplet
    - Output: Tuned model

- *Evaluate Performance*: Load the test dataset and evaluate the class-wise accuracy of the learned classifier model. Generate the confusion matrix also.
    - Input: Model, <te>
    - Output: Evaluation Report
- Random seed value should be a parameter used by all stages.

Task 4 [10 points]

Remember v1, v2, v3 are partitions of the full dataset. Now, run the experiment pipeline with v1, v2, v3, v1+v2, v1+v2+v3 and generate their performance metrics. Repeat the runs for two more times by changing only the random seed value. You should be able to see all the experiments run on the output of DVC EXP SHOW or in VSCode.

Brownie [10 points]

Identify the hard to learn images from the v1 partition across the models built using v1, v1+v2, v1+v2+v3. We are trying to find the images that none of the 3 classifiers were able to learn. Identify the class distribution of the hard to learn images. Also, report the misclassification table.