CS-5007 Deep Learning

Assignment 5 - CNNs

13 April, 2021

Instructions

1. Submit the assignment as a single notebook(*.ipynb).
2. Follow the given naming format strictly: ⟨*name_roll*⟩_Assignment_5.ipynb
3. Try to solve each sub question in a new cell wherever possible. This improves the notebook's readability.
4. Please note down your observations in text blocks below the corresponding code blocks.
5. Mention and justify your assumptions if any.
6. You can use all the packages used in the demo. And these are sufficiently enough.
7. Maintain separate cells each for loaded libraries, configuration data, utility functions, main code block and so on.
8. You don't have a separate report, so try to present your notebooks in a more readable way.

Introduction

- In this assignment, our focus will be on building a simple Convolution Neural Network.
- Remember that a CNN consists of two important parts- *feature extractor* and *feature classifier*.
- For the given dataset, construct a small CNN model (i.e with less layers, filters and parameters, etc.), for image classification.
- Again remember, why small? Because the dataset is simple and so huge model is not required (*And yes! It is easy to train :)*)

Dataset details:

- Link: https://drive.google.com/file/d/1wCBE5YamOKAFNxvzYXlHOcGaycT_k8gQ/view?usp=sharing
- We will be using a subset of Facial Expression Recognition dataset. We will just have two classes, namely- 'Happy' and 'Neutral'.
- The images are of size 48×48. The images are given in two folders: 'Train' and 'Test'.
- Train folder further contains two sub-folders, one for each class of images.
- Use these folder names as the class names (You don't need a separate labels file!).
- *(Note: this is the storage convention used by ImageNet dataset.)*
- The test folder contain a '.npy' file. It contains the test images.
- Load it as follows:
  import numpy as np
  data = np.load('test_data.npy')
  # This will be an array of size (200,48,48,1) with 200 test images.
- **(Note: DONT shuffle the test data!)**

Tasks

1. Load the images and their corresponding labels in your train variables : Train_X and Train_y.
2. Build a CNN for binary classification using basic functions (like Convolution, Pooling, Batch Normalization, FNNs, etc.).
3. Decide the appropriate hyper-parameters and pre-processing that you will be using. Try to maintain these details in a separate code cell. (like the 'Config. cell' in the demo)
4. Train the model on the given dataset using **'Image Data Augmentation'**:
   (a) 'Model 1': Using 'Flatten' layer.
   (b) 'Model 2': Using 'Global Average Pooling' layer.
5. Train another model ('Model 3') using **Pretrained VGG** network. (i.e freeze the convolution weights and just fine tune the feature classifier.)
6. For VGG, you can use the function defined in the demo, by changing the FNN part.
7. Plot the Training curves with loss and accuracy on the Y-axis and Number of Epochs on the X-axis. Plot these for all the three models.
8. Compare these models on metrics like Total number of parameters, Number of trainable parameters, Training Accuracy, average F1 score, etc.
9. Test your model on the test dataset. Submit a csv file with 'Image_index' and 'Pred_labels' as the two columns. 'Image_index' is nothing but the index of the image in the test data array. 'Pred_labels' will be either 0 (for 'happy') or 1 (for 'neutral').
10. Models will be evaluated on the labels generated for the test dataset.
11. {Bonus} Inclusion of GradCAM interpretability atleast for 10 images (5 for each class), will attract a bonus mark.