Deep Learning, Spring 2021
Assignment 6 (Optional)
Image Classification Competition

NOTE: This is an entirely optional assignment.

You will not lose marks if you do not attempt it. The assignments in total are worth 45% of the course grade. Full marks in this assignment will count towards an additional 5% that will be added to whatever score you have already earned. If this addition takes your total percentage over 45%, then it will be cut off at 45%.

UPDATE: The total percentage of assignments is not finalized yet, this assignment will still add a max of 5%.

Submission Guidelines:
Submit all your code, predictions and report as separate files with the naming conventions defined below:.

- You will be submitting three files in total
  (a) Code (b) Report (c) Predictions file (csv)
- There should be a report named **RollNumber_06.pdf** detailing your experience and highlighting any interesting result. Kindly don't explain your code in the report, just explain the results. Your report should include your comments on the results.
- The assignment is only acceptable in Jupyter Notebook format (.ipynb).
- Your code notebooks should be named as **'rollNumber_06.ipynb'**
- Your prediction file should be named as **'rollNumber_06.csv'**
- Please carefully follow all the naming conventions. If your conventions do not match ours, your assignment might not be accepted or graded
- Email instructor or TAs if there are any questions. You cannot look at others code or use others code, however you can discuss with each other. Plagiarism will lead to a straight zero with additional consequences as well.

Due Date: August 3rd, 2021

Note: Do not share code or look at the other student's codes. You should not be in possession of any implementation related to the assignment, other than your own. In case of any confusion please reach out to the TA's (email them or visit them during their office hours).

- **You are strongly advised to implement this assignment using Google Colaboratory's free GPU if you don't have a high end GPU optimized local machine**
- **Please go through the code uploaded on Google Classroom before starting this assignment. It contains code to load pretrained models and how you can freeze and unfreeze layers of the model. Starter code file**

Objectives:

- This assignment is a competition between the whole class. Students performing better will be given more credit than others. The top certain ratio of students will be awarded with full credit and the next certain percentage will be graded a little lower and so on. So, do not share any code with each other to assure your place in the top rank.
- Objective of this assignment is to allow students to employ their knowledge of Deep Learning to solve the problem. In this assignment you will be finetuning a network of your choice on the provided dataset and extending it as you wish to solve the problem.
- Objective of this assignment is to familiarize students with the transfer learning for image classification, data-augmentation, etc.

Assignment Task

Basic purpose of this assignment is to make students familiar with transfer learning for image classification. To make it interesting we have made it a competition. You will be training an image classification network of your choice and finetune it on the provided dataset.

Dataset:

In this assignment you are required to use a custom dataset that we have particularly designed for this assignment. The custom dataset is a mixture of two public datasets, one part of the dataset is taken from Tiny-ImageNet and the second part is taken from the kaggle's Fish dataset. Hence, you are advised not to use any model that is pre-trained on these datasets. There are 59 classes in the given dataset. We have renamed folders of class names with digits. So names of classes in the given dataset are from 0 to 58. You need to split this dataset into two separate sets of training and validation. On the final day we will provide you with test images and your accuracy on the testing dataset will be compared with your other class members. Students with better accuracy will get more marks. Dataset Link.

Note: On the last day of the submission, we will provide you with a test dataset and a csv file format in which you will submit your predicted results against each test image. You will have to follow the class labels of the dataset as given in the train data folder. In case the conventions do not match and we are unable to compute your test accuracy metrics, your assignment might get discarded.

Steps for transfer learning:

- Load a DNN, pre-trained on ImageNet, to extract basic features from images.
- Change the classification layers in order to make the model work with the given dataset. Classification networks are pretrained on imagenet dataset. It has a thousand classes, so the FC final layer of pre-trained networks has 1000 neurons. You have to change the final layer according to the number of classes in the provided dataset.
- Fine-tune the entire or partial network for improving its accuracy for the given dataset. You can use the pre-trained network as the feature extractor too.

- The image sizes in the dataset are different for several classes. While loading the dataset, make sure if the model requires you to rescale them on the dimensions of the pre-trained model input size.

Data Augmentation:

You are required to use data augmentation because the collected dataset does not contain enough images of each class for fine tuning large models. You will have to write augmentation code yourself and you need to find the best augmentation technique for this dataset. Some simple data augmentation techniques are listed below. You can use built in library functions for this task.

1. Image Rotation
2. Image flipping
3. Changing brightness

In this assignment you will be working on large classification networks. We have described some of the networks below. We went through the architecture of some of these networks in detail in class lectures. You are allowed to pursue any of the following or you can choose anything other than these.

Resnet:

Resnet is a deep convolutional neural network. It is used as a feature extractor and backbone for a lot of computer vision tasks like object detection, instance segmentation. It uses residual connections and it comes with different depths like 18, 34, 52, 101 layers. For more information go through lecture slides.

VGG16:

VGG16 is another deep convolutional neural network used for classification and feature extraction. It can also be used in this assignment. Its architecture is a little simpler than other complex DNNs. For more details go through course slides.

Vision Transformer:

Vision transformers use the attention modules for extracting features from images. Mostly transformers are used in NLP but Vision Transformer presents a model which uses transformers for vision applications. For more details about architecture see the link above.

You can also go through other DNNs like **mobilenet, squeezenet** etc. These networks are more optimized in terms of training and inference time and computation cost than above mentioned networks but they sacrifice accuracy for it.

Fine-tuning in Pytorch:

In PyTorch, each layer's weights are stored in a Tensor. Each tensor has an attribute called ‘requires_grad’, which specifies if a layer needs training or not. In fine-tuning tasks, we freeze our pre-trained networks to a certain layer and update all the bottom layers. In PyTorch we can loop through our network layers and set ‘requires_grad’ to False for all the layers that we want to freeze. We will set ‘requires_grad’ to True for any layer we want to fine-tune.

It is upto you which network to choose and how many layers to fine tune. Fine tuning more layers might improve your final accuracy at the expense of more time during training. This is a decision that you have to make yourself. We are aware that all of you do not have access to a GPU and most of you will be using colab for training. We advise you to go through the above networks in detail and choose one that is suitable for your situation then work on it to improve its performance.

NOTE: YOU CANNOT BORROW OR USE THE NETWORK TRAINED ON THE TINY-IMAGENET.

Extra Steps:

You can get help on different techniques from papers/blogs/books to enhance your model's performance. A few techniques are shared above. You can also search for ensemble voting, few shot learning, different advanced optimization and regularization approaches, etc.

Deliverables.

There are three deliverables for this assignment.

- First deliverable is your predictions on testing images. You will submit your predictions in a csv file. There should be three columns, first one for image names, second one for the labels that your model assigned to it and third one is for the probability that model is assigned to that particular image. While submitting the csv file do not make any convention mistakes.
- For the second deliverable you need to submit a table in the report containing information about the resources you used for training along with your training and validation accuracy. You need to submit a table of the following form. Information that you provide, will be used for evaluation.

Batch Size used for training	
DNN Model used	
Total trained parameters in the model[1]	
GPU/Mem	
Time taken for training the network	
Training dataset size after augmentation	
Validation dataset size after augmentation	
Training Accuracy	
Validation Accuracy	
Validation F1 score	

- Third deliverable in the report are the following points:
    - Indicating your strategy, diagram of model you used.
    - Examples where prediction was correct and where it was wrong
    - ROC-curve

Note: Your final accuracy is not the only thing that matters. How efficiently you train the model and resources you utilised also matter to compute your final score. We might try to create a mechanism to distribute the weightages among the accuracies and the hardware used, however, this is not decided yet . Please note that Google Colab randomly provides a maximum of 16GB memory GPU with a time constraint but this is kind of unofficial standard for now.

Grades distributions:

- Students who are in the top 5% in the final score list will be awarded full marks.
- Students in the range of top 5% to 20% will be awarded 80% of total marks.
- Students in the range of top 20% to 30% will be awarded 60% of total marks.
- Students in the range of top 30% to 40% will be awarded 40% of total marks.
- Students in the range of top 40% to 50% will be awarded 20% of total marks.
- Students below 50% of the total class will be awarded 10% of total marks.

The top students might be asked to explain their techniques in a presentation. Please note that your scores need to be better than the baseline results to be considered for grading. Similarly, if your score is 20 points lower than the top score, you might be assigned a zero score. If your report is not adequate, similarly you might be assigned a zero score.

Programming Helpful Commands:

Commands to find total parameters of the model:

- `total_params = sum(p.numel() for p in vgg16.parameters())`
- `print(f'{total_params:,} total parameters.')`

Commands to find trainable parameters of the model:

- `total_trainable_params = sum(p.numel() for p in vgg16.parameters() if p.requires_grad)`
- `print(f'{total_trainable_params:,} training parameters.')`