1. In this question, you have to compare the performance on a binary classification task of the following models:

- VGG (1 block)
- VGG (3 blocks)
- VGG (3 blocks) with data augmentation
- Transfer learning using VGG16 or VGG19 with tuning all layers (including tuning convolution layers)
- Transfer learning using VGG16 or VGG19 with tuning only final MLP layers (excluding convolution layers)

[Refer this article](https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/) You do not need to write your own code (but it might be easier if you do so!). You can reuse the code from the post. Or, you could roll out your own implementation. Either way, you should be able to explain your code during the viva.

You need to create the dataset on your own based on your first names of any two members from the group. For instance if the first name of the team members are: **V**aidehi and **R**aghav, they can choose a dataset of their liking based on any names, place, animal or thing. As examples:

- Vulture v/s Rat
- Vada Pav v/s Roti

You can refer to [resource 1](https://python.plainenglish.io/how-to-automatically-download-bulk-images-for-your-dataset-using-python-f1efffba7a03) or [resource 2](https://github.com/JorgePoblete/DuckDuckGoImages) or plainly download 100 images of both classes (total 200 images). Of these 100 images of each class, we will use 80 for training and 20 for testing. You get 1 mark for dataset creation [1 mark]

Create a table with models as rows and the following columns **[2.5 marks (0.5 marks for each model)]**

- Training time
- Training loss
- Training accuracy
- Testing accuracy
- Number of model parameters

We will now be using Tensorboard for visualizing network performance. You are suggested to refer to:

- [PyTorch + Tensorboard](https://www.youtube.com/watch?v=RLqsxWaQdHE)
- [Tensorflow + Tensorboard](https://www.youtube.com/watch?v=k7KfYXXrOj0)

Use Tensorboard to log the following and present screenshots/images **[1 mark]**

Scalars
- Training loss v/s iterations (and not epochs)
- Training accuracy v/s iterations (and not epochs)
- Testing accuracy v/s iterations (and not epochs)

Images
- Show all images from the test set and their predictions

Now you have to present various insights. For instance, you should discuss the following: **[1 marks (0.25 marks for each question)]**

- Are the results as expected? Why or why not?
- Does data augmentation help? Why or why not?
- Does it matter how many epochs you fine-tune the model? Why or why not?
- Are there any particular images that the model is confused about? Why or why not?

Now, create an MLP model with parameters comparable to VGG16 and compare your performance with the other models in the table. You can choose the distribution of the number of neurons and the number of layers. What can you conclude? **[0.5 marks]**

Finally, use any image generation tool of your choice. Provide a prompt that helps easily classify your image; and provide a prompt that creates a hard to correctly classify image. Show the performance of various models on these sets of images. Minimum 4 such images can be created. Class A: easy, Class A: hard; Class B: easy, Class B: hard **[1 mark]**

2. In this question you will implement and compare the following different KNN variants (see: https://youtu.be/C9HQLyXwEw0?t=3382)

   a. LSH **(1 marks)**

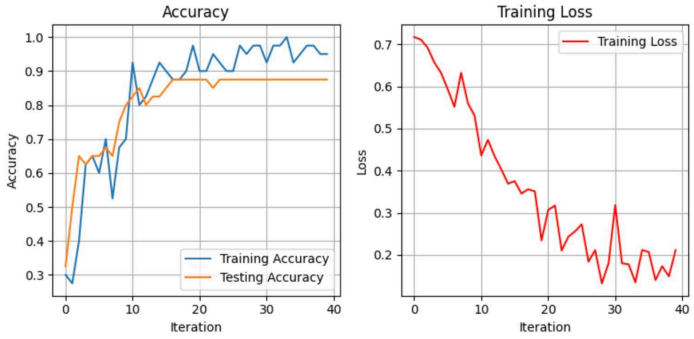
b. KD-tree **(1 marks)**
c. Naive version of KNN **(0.5 marks)**

Vary dataset size N, number of dimensions D to do training and testing time and memory comparison for finding K nearest neighbours. **(1 mark)**
Now, in a 2d randomly generated dataset visually show how many of the K closest neighbors appx. K NN methods miss out due to their approximate nature.

Also show the partitions in the 2d space. **(0.5 marks)**

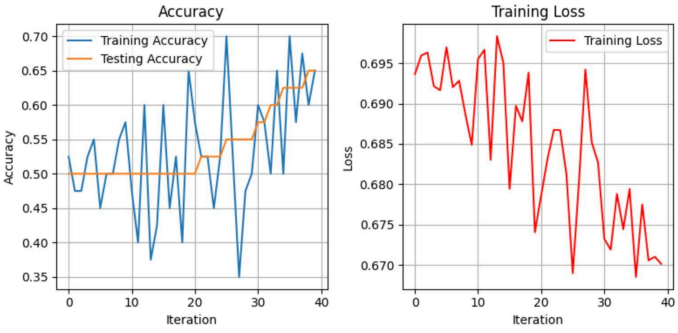
Model	Training Accuracy	Training Loss	Testing Accuracy	Training Time	No. of Parameters
VGG (1 block)	0.95	0.211507	0.875	54.113453	51,381,377
VGG (3 block)	0.65	0.67015	0.65	58.915615	12,938,561
VGG (3 block) with data augmentation	0.60	0.6543	0.625	141.28	12,938,561
VGG16 with tuning all layers	1.0	0.2639	0.975	1214.63	134,264,641
VGG16 with tuning only final MLP	0.975	0.1184	1.0	290.70	119,549,953

VGG (1 block)



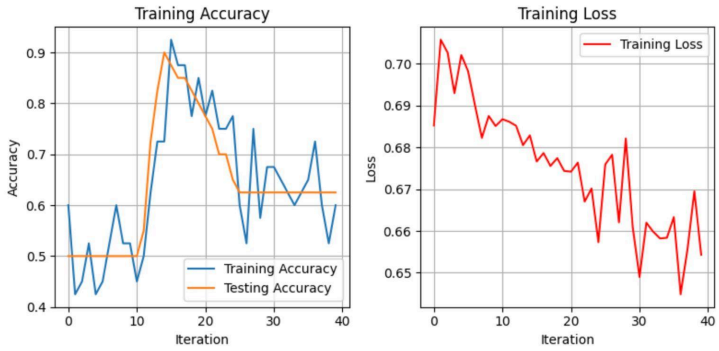

```
Training Accuracy : 0.95
Training Loss : 0.21150735020637512
Testing Accuracy : 0.875
Training time : 54.11345291137695
No. of Parameters : 51381377
```

VGG (2 block)



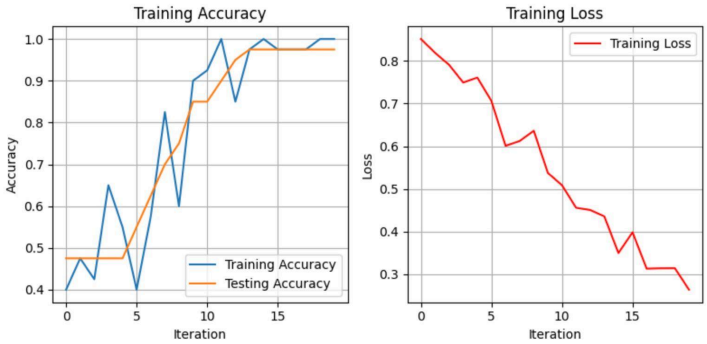

```
Training Accuracy : 0.65
Training Loss : 0.6701499223709106
Testing Accuracy : 0.65
Training time : 58.91561484336853
No. of Parameters : 12938561
```

VGG (3 Block - 2nd)



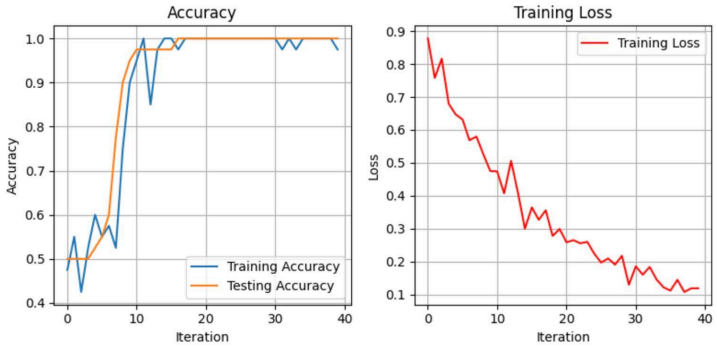

```
Training Accuracy : 0.6
Training Loss : 0.6542922258377075
Testing Accuracy : 0.625
Training time : 141.28253841400146
No. of Parameters : 12938561
```

VGG16 with tuning all layers




```
Training Accuracy : 1.0
Training Loss : 0.2639283835887909
Testing Accuracy : 0.975
Training time : 1214.6277890205383
No. of Parameters : 134264641
```

VGG16 with tuning only final MLP




```
Training Accuracy : 0.975
Training Loss : 0.11839500814676285
Testing Accuracy : 1.0
Training time : 290.69995379447937
No. of Parameters : 119549953
```