Include the writeup and all your code in a zip file . (Use one .py file or a Jupyter notebook for each experiment on one model variant. Name the files as minivgg, var1, var2 and var3 and var4.)

Implement a mini-vgg network and several of its variants. Train and test the models on the cifar-10 dataset. Investigate and compare the performance of the variants to the original model. Write half to one page to 1) summarize your experiment results and discuss 2) the classification performance of the models as well as 3) their size (# of parameters) and 4) the computation time to train them.

Mini-VGG network structure:

Layer	Type(window size) – n filters
1	Conv3 – 64
2	Conv3 – 64
3	Maxpool – 2x2
4	Conv3 – 128
5	Conv3 – 128
6	Maxpool – 2x2
7	Conv3 – 256
8	Conv3 – 256
9	Maxpool – 2x2
10*	fully Connected 512
11**	soft-max

* Note your need a reshape layer before this layer to reshape the data

** Use cross entropy loss (torch.nn.CrossEntropyLoss or tf.nn.softmax_cross_entropy_with_logits()) feed the loss function with the logits before softmax activation but get the prediction for accuracy after softmax activation)

Report the performance of each network in terms of test accuracy and also plot the **validation loss vs train loss**, and **validation accuracy vs train accuracy** for all of the following tasks.

- Implement the mini-vgg model and report its performance using ReLU activation function for all the all conv/fc layers except the last one.
- **Variant 1**: Change the ReLU activation functions to SELU and Swish. Would the performance

improve?

- **Variant 2**: Remove the maxpool layers. Using stride=2 in the conv layer before the maxpool to achieve similar size reduction. Would the performance improve?
- **Variant 3**: Add a few dropout layers in the model. Would the performance improve? Try 2 different ways to add the dropout layers. Describe the ones you tried and their performance.
- **Variant 4**: Remove layers 9 and 10. Add two layers of (1, 1) convolution: conv (1, 1) x 128; conv (1, 1) x10. Then add "GlobalAveragePooling2D" to merge feature maps before pass them to softmax. This is an all-convolutional structure (no fully connected layers).

Clarification

Report the performance of each network by doing the following:

1) Plot training loss vs validation loss

2) Plot training accuracy vs validation accuracy

3) Calculate test accuracy

Hints :

I have given you the details for each layer so just follow the table:

first layer that gets the input is a conv with 64, 3x3 filters and ReLU activation

second layer that gets the output of first layers, is a conv with 64, 3x3 filters and ReLU activation

Third layer that gets the output of the second layer and is maxpool with 2x2 kernels

Fourth layer that gets the output of the third layer is a conv with 128, 3x3 filters and ReLU activation.

...

The layer 10 is a fully connected with 512 neurons make sure that you flatten the input before feeding it to the FC layer.

I expect you to implement the model and define each layer yourself so do not use any predefined code for mini VGG or you will lose points accordingly.

Feel free to use PyTorch or Tensorflow.