Question 1:

Use the **CIFAR-10** dataset for all the experiments. In this question, you will implement a fully functioning CNN for classification. Use a 70:30 data split.
Compile the model by calling model.compile(optimizer = "...", loss = "categorical cross entropy metrics= ["accuracy"]). (Free to use your library).
Train the model on train data by calling model.fit(x = ..., y = ..., epochs = ...).
Note that if you run fit() again, the model will continue to train with the parameters it has already learnt instead of reinitializing them.
Test the model on test data by calling model.evaluate(x = ..., y = ...).

The model architecture is provided in the figure. Perform the following tasks:

(1) Use all three channels for the classification.
  Use input shape (32 x 32 x 3).
(2) Use SGD optimizer with suitable learning rate, and suitable activations in the required layers.
(3) Try out all the following variations in the architecture of Fig.1. Report the performance of all models.

  (a) No BatchNormalization

  (b) Two Dense Layers. (*Note: The last Dense layer should have 10 nodes as there are 10 classes. For the Dense layer before that, use 64 nodes.)*

  (c) 2 blocks of Conv2D -> BatchNorm2D->MaxPooling2D

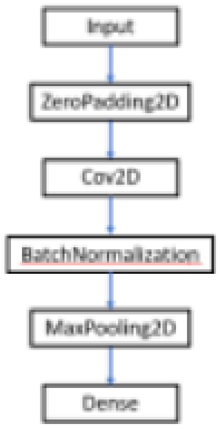

Figure 1. CNN architecture

(d) 3 blocks of Conv2D -> BatchNorm2D->MaxPooling2D
Add a table contrasting the performance of the given architecture with all above variations. State your analysis.

(4) Save the best model and plot the accuracy vs epoch. Show the model architecture using model.summary().

Image Classification using CNN [8 Marks]

Problem 1. Use CIFAR10 dataset for this question. It contains 60,000 color images in 10 classes, with 6,000 images in each class.The dataset is available in both Pytorch and Tensorflow.

In this problem we will use CNN to classify these images.

1. Train the CNN network for image classification. **[2 Marks]**
2. Experiment by changing learning rate, batch size ,number of filters at each layer, number of pooling layers etc. Show the effect of changing some of these parameters. Plot the epoch - MSE, epoch - Accuracy curves during training for every setting you use. **[3 Marks]**
3. Show the calculations for sizes after each convolution layer and each pooling layer for one particular setting. **[3 Marks]**

Transfer Learning for Image Classification [6 Marks]

Problem 2. We will use VGG16, a convolutional neural network in this question.The pre-trained model for VGG-16 trained on ImageNet dataset is available in both pytorch and tensorflow. A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task.

In this problem we will use transfer learning to classify images form the hymenoptera-data datset. You can download the dataset here. Use the images in train folder for training and images in validation folder for validation.

The base convolutional network already contains features that are generically useful for classifying pictures. However, the final, classification part of the pretrained model is specific to the classification task being used.

1. Read about VGG16 and briefly explain it's architecture**[1.5 Marks]**
2. Freeze the convolutional base of pre-trained VGG16 and add the fully connected component for classification task in this question. Train the model.**[1.5 Marks]**
3. Now, Unfreeze some of the top layers in the pre-trained VGG16. Train the model again. Do you see any improvement?**[1.5 marks]**
4. Plot the [epochs (vs) Accuracy] and [epochs (vs) loss curves] for both (2) and (3). Also,calculate the validation accuracy for both cases.**[1.5 Marks]**