Assignment 3

You are free to use Tensorflow, Pytorch or any other DL library. For beginners, we recommend using Pytorch, or Keras. As deep learning models often require higher computation and memory, you can try using an online platform like Google Colaboratory or Kaggle Notebooks.

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

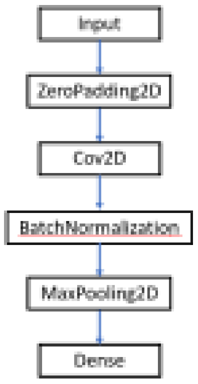

Figure 1. CNN architecture

(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().

Question 2: (NOTE: Module names are for Pytorch, you are free to use Keras,Tensorflow or any other library, the steps will be same)

Use the MNIST dataset for all the experiments. In this question, you will implement a fully functioning under complete autoencoder for feature learning. Use a 70:30 data split.

Perform the following tasks:

(a) Load the dataset using " torchvision.datasets" library, preprocess the dataset using "torchvision.transforms" library and finally create a train loader and test loader of some batch size using "torch.utils.data" library.

(b) Create an autoencoder class with decoder and encoder architecture and a hidden neuron representation.

(c) Initialise an optimizer using "torch.optim" library and you can also use a learning rate scheduler using "torch.optim.lr_scheduler" library.

(d) Write a training loop over epochs and over the batches of the train set and calculate the average loss between the reconstructed image and the input image.

(e) Write a testing loop over the batches of the test set and calculate the loss between the reconstructed image and the input image.

(f) You can save the best performing model using “torch.save”.

Report the test reconstruction loss and plot the train loss vs epochs for the following cases:

1) Perform the experiment using different optimizers like Adam, RMSProp, SGD with momentum, SGD without momentum.
2) Plot the test reconstruction loss vs hidden neurons for all the above optimizers.
3) Perform PCA reconstruction and compare with the Autoencoder reconstruction using test reconstruction loss (you can use inbuilt PCA).

Question 3: (NOTE: Module names are for Pytorch, you are free to use Keras,Tensorflow or any other library, the steps will be same)

Use the MNIST dataset for all the experiments. In this question, you will implement a fully functioning MLP classifier for label classification. Use a 70:30 data split. Perform the following tasks:

(a) Load the dataset using “torchvision.datasets” library as train and test, preprocess the dataset using “torchvision.transforms” library and finally create a train loader and test loader of some batch size using “torch.utils.data” library.

(b) Create an MLP class with fully connected layers with relevant activation functions.

(c) Initialise an optimizer using “torch.optim” library and you can also use a learning rate scheduler using “torch.optim.lr_scheduler” library.

(d) Write a training loop over epochs and over the batches of the train set and calculate the average train cross entropy loss.

(e) Write a testing loop over the batches of the test set and calculate the test accuracy.

(f) You can save the best performing model using “torch.save”.

Part 1:

Report the test accuracy and plot the train loss vs epochs for the following cases:

1) Perform the experiment using different optimizers like Adam, RMSProp, SGD with momentum, SGD without momentum.
2) Try different weight initialisations of the neural network like Xavier, uniform and normal.
3) Try different learning rates and also perform exponential annealing on the learning rate.

Part 2:

Perform the following experiment using the autoencoder code written in the previous question:

1) Train a MLP classifier using the feature representation from under complete autoencoder trained on the train set and test the MLP classifier on the feature representation from under complete autoencoder tested on the test set.
2) Use the MLP classifier from Part 1 which shows the best accuracy score.
3) Compare both the accuracy scores.

Question 4: (NOTE: Module names are for Pytorch, you are free to use Keras,Tensorflow or any other library, the steps will be same)

Siamese network - MNIST Dataset

1) Implement a Siamese network on the given dataset for classification by splitting the data into train and test datasets
2) Use different loss functions like triplet loss function , contrastive loss, regularized cross entropy and compare the accuracies.
3) compare accuracies for various optimisers like RMSprop, Mini Batch Gradient Descent , Adam Optimizer and select the best one. Give reasons.
4) Now try Hyper Parameter Optimization to improve the accuracy and report accuracies before and after.
5) Mention Pros and Cons of Siamese Networks.