# This Project was completed as part of MSc coursework

## Task 1: Read Dataset and Create Data-loaders: 5%

* Code taken from myutils.py and Edited for relevant Dataset, Normalisation was applied to both Train and Test Datasets, with Data augmentation applied to just the Training Dataset.

## Task 2: Create the Model (K=16, N=4)

* Architecture of 1 Block:
  + MLP layer (2 layers) predicting a vector a = [a1, ..., a16] with 16 elements from input tensor X:
    - AdaptiveAvgPool (X) calculates the spatial average per channel returning a vector of d channels
    - AdaptiveAvgPool (X) is passed through 2 Linear Layers, in-channels=3 to 20, 20 to 16
    - This output is run through a RELU function
  + K Conv layers which are combined using a to produce a single output: O = a1\*Conv1(X) + ... + a16\*Conv16(X)
  + Output passed through a Batchnorm then a Maxpool.
* 4 Blocks used (Convolutional Layers Kernel-size: 3, Stride: 1, Padding: 1)
  + Out-channels for each block
    - Block 1: 3 to 64
    - Block 2: 64 to 128
    - Block 3: 128 to 256
    - Block 4: 256 to 512
* Classifier:
  + AdaptiveAvgPool Layer
  + Linear Layer to 10 outputs

## Task 3: loss and optimiser: 5%

* Applied in Training Script
* Optimiser is torch.optim.Adam, lr=0.001
* Loss is Crossentopyloss()

## Task 4: write the training script to train the model

* Script taken from myutils.py with altered Optimiser
* Epochs = 50
* Lr = 0.001
* Device = CUDA (Colab GPU)
* Code for Evaluation and Plotting (Animator) included in Notebook

## Task 5: Final Model Accuracy on CIFAR-10 Validation Set:

* Train Accuracy = 0.970
* **Test Accuracy = 0.869**