Visual Recognition
Mini-Project Intermediate Report

Team Members:

- Vyom Sharma (IMT2020026)
- Rachit Agrawal (IMT2020018)
- Sathvik I Bhat (IMT2020009)

GitHub Link (Code):

https://github.com/tilasmipathar/VR-Mini-Project

Question 3a:

The CIFAR10 dataset was used for the object recognition task, which contains 60000 32x32 color images divided into 10 classes, with 6000 images per class.

We used 2 different architectures for training the model which have been described later on. For each architecture, we trained using 4 different optimizers (Adam, Adagrad, SGD without momentum, and SGD with momentum of 0.8) and cross-entropy loss. We kept the batch size at 256 and trained each model for 50 epochs. The results include training time and classification accuracy on test data.

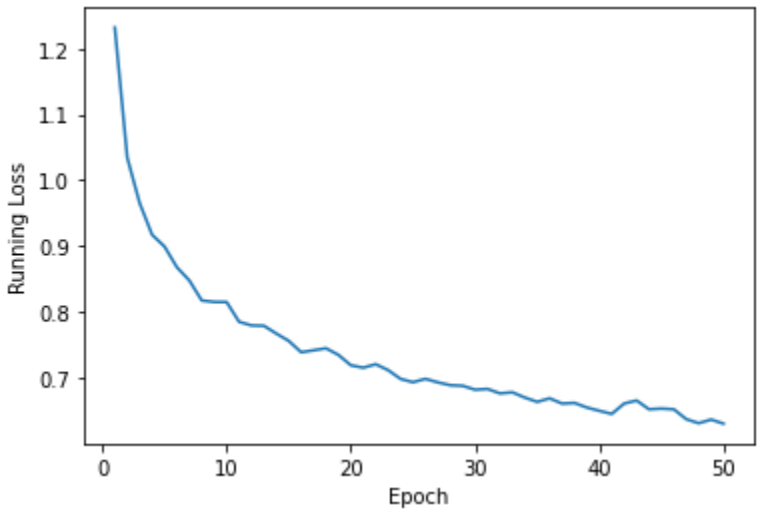
2 Convolutional Layers + 2 Fully Connected Layers -

We used 2 convolution layers and 2 fully connected layers with the following architecture -

```
Cnn(
  (network): Sequential(
    (0): Conv2d(3, 8, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(8, 16, kernel_size=(5, 5), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Flatten(start_dim=1, end_dim=-1)
    (7): Linear(in_features=400, out_features=128, bias=True)
    (8): ReLU()
    (9): Linear(in_features=128, out_features=10, bias=True)
  )
)
```

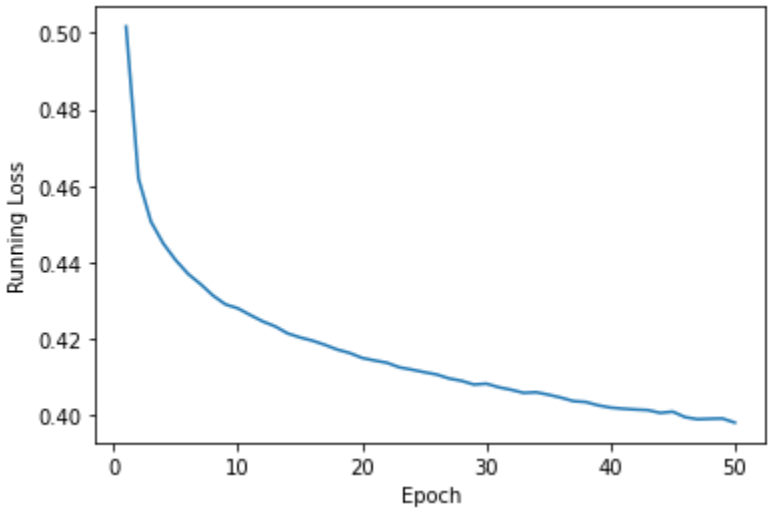
We trained the model for Relu/Tanh/Sigmoid activation functions and the results were as follows -

- **Relu -**
  - **Adam -**



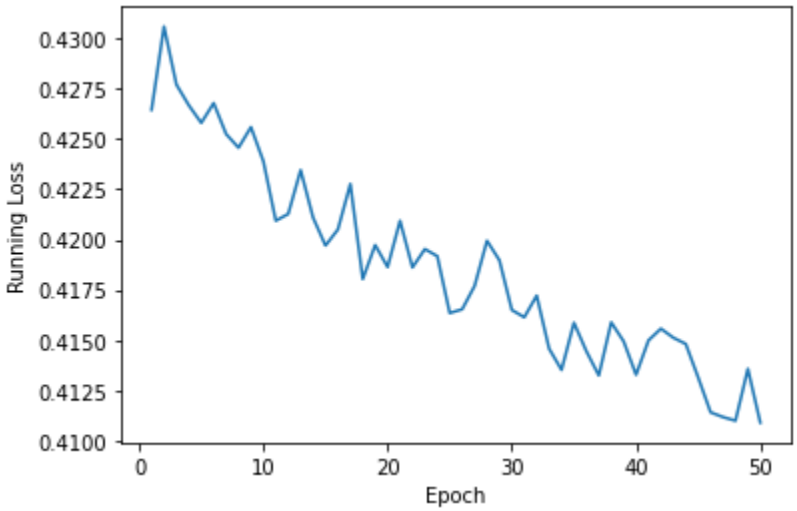
Training Time - 626.47 s
Classification Accuracy - 73.06%

  - **Adagrad -**



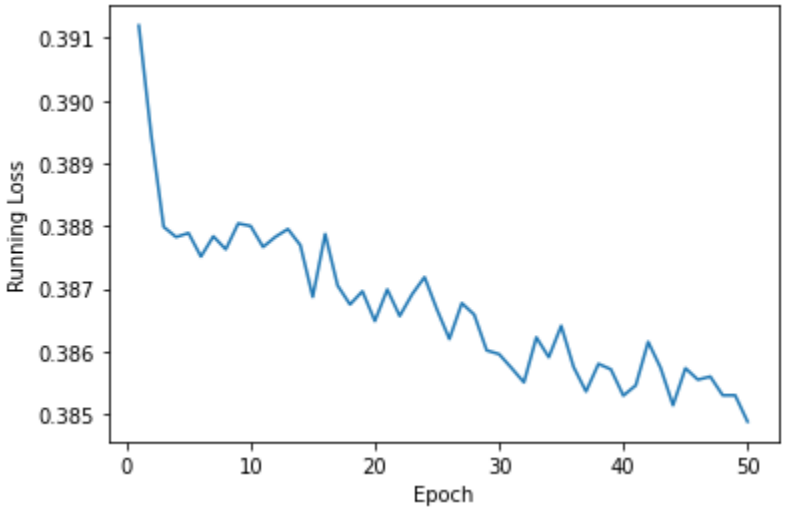
Training Time - 619.37 s
Classification Accuracy - 81.69%

- **SGD with momentum of 0.8 -**



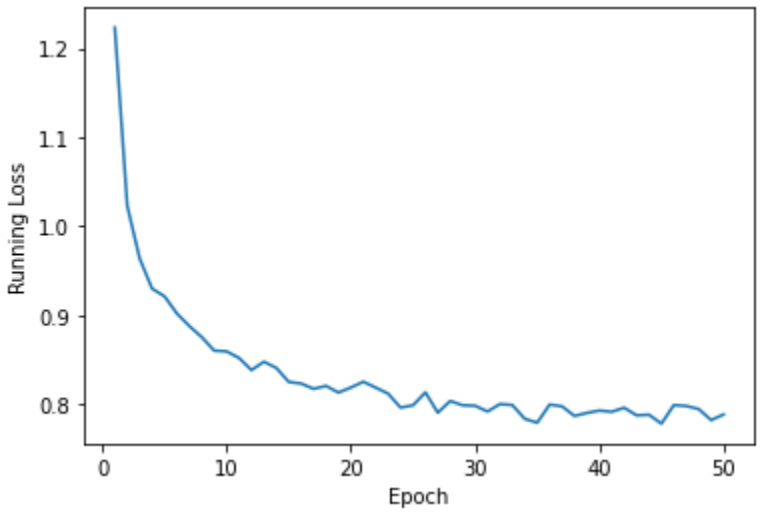
Training Time - 607.17 s
Classification Accuracy - 80.44%

- **SGD without momentum -**



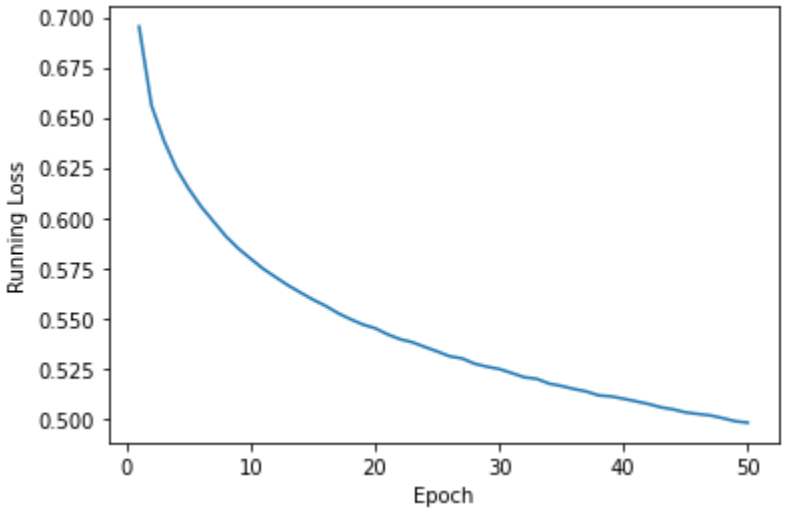
Training Time - 596.25 s
Classification Accuracy - 81.92%

- **Tanh -**
  - **Adam -**



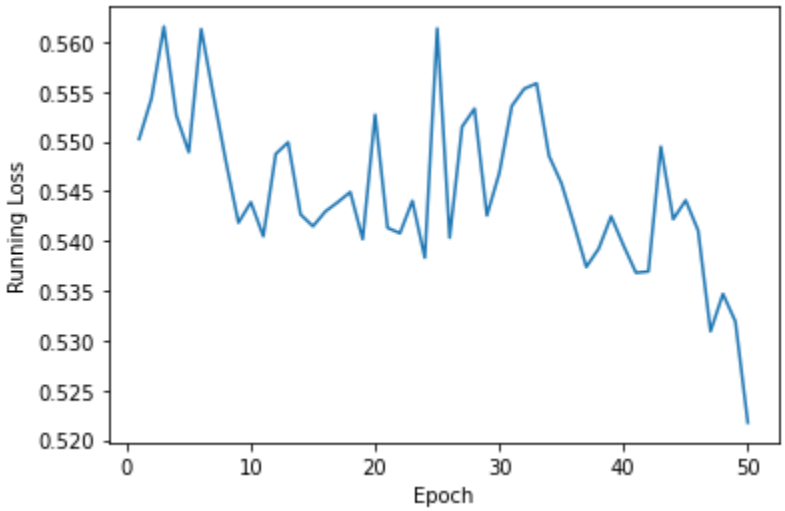
Training Time - 614.72 s
Classification Accuracy - 62.02%

  - **Adagrad -**



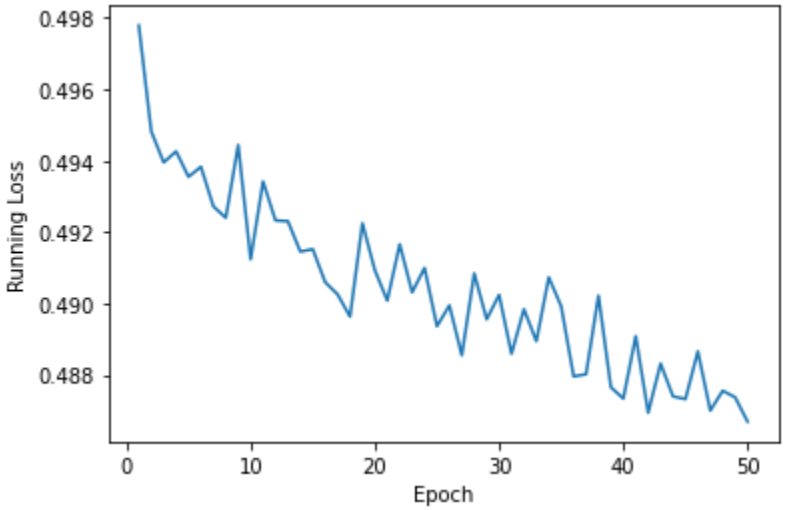
Training Time - 608.42 s
Classification Accuracy - 77.92%

- **SGD with momentum of 0.8 -**



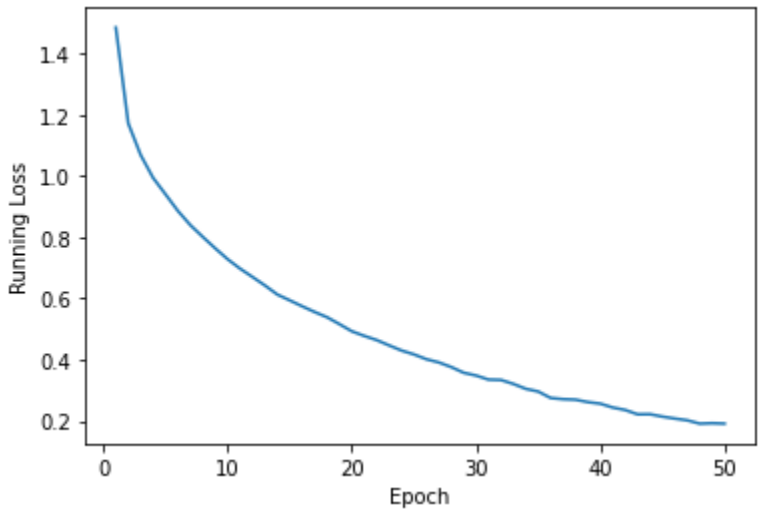
Training Time - 602.87 s
Classification Accuracy - 76.29%

- **SGD without momentum -**



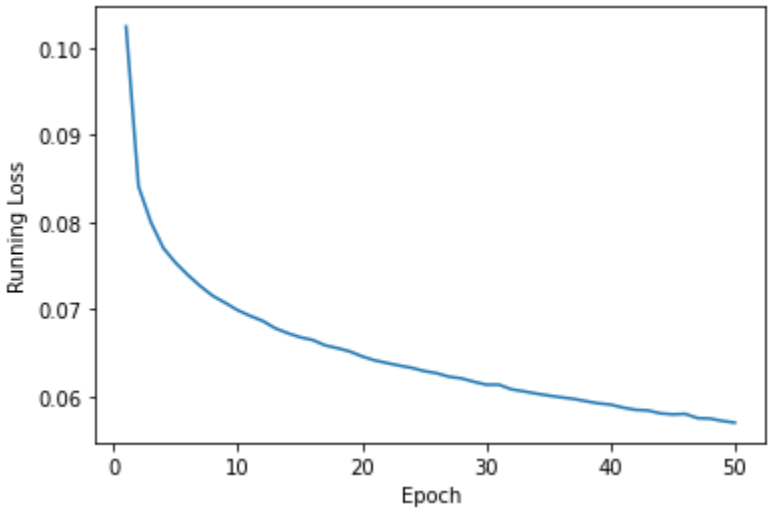
Training Time - 613.77 s
Classification Accuracy - 77.5%

- **Sigmoid -**
  - **Adam -**



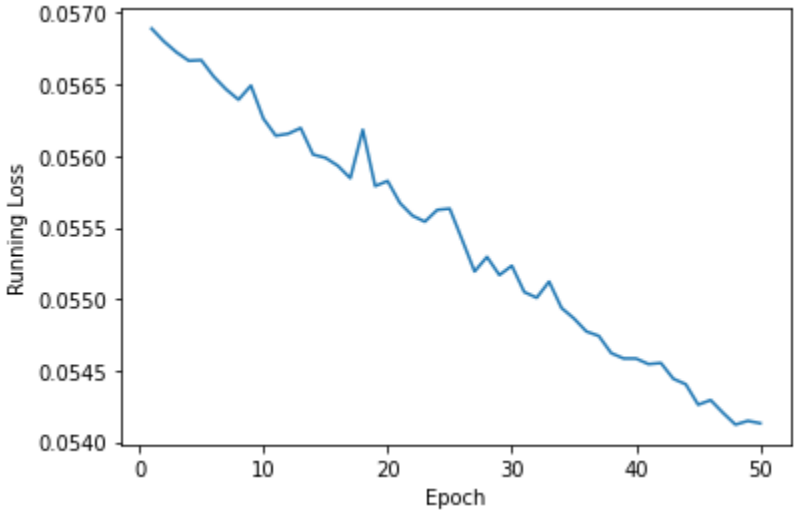
Training Time - 612.04 s
Classification Accuracy - 93.59%

  - **Adagrad -**



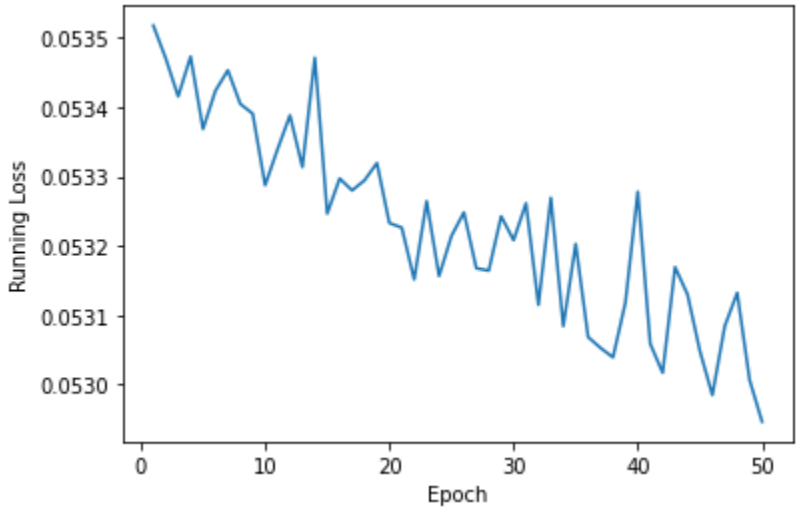
Training Time - 643.25 s
Classification Accuracy - 99.16%

- **SGD with momentum of 0.8 -**



Training Time - 640.37 s
Classification Accuracy - 99.22%

- **SGD without momentum -**



Training Time - 619.59 s
Classification Accuracy - 99.23%

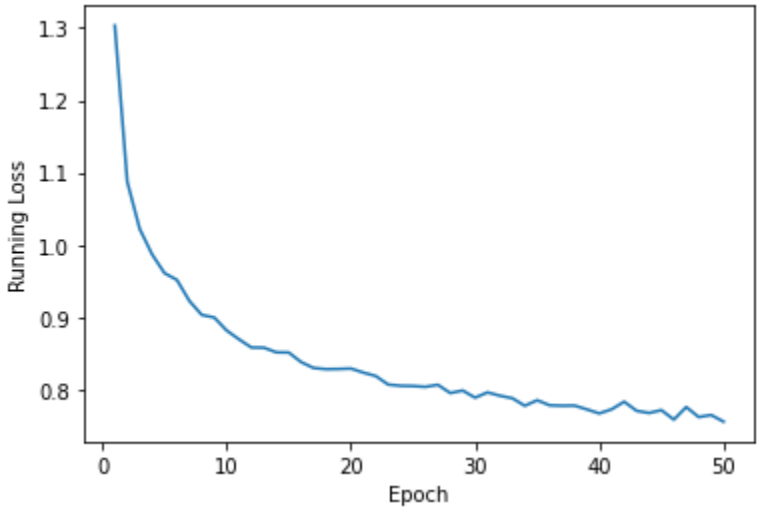
3 Convolutional Layers + 2 Fully Connected Layers -

We used 3 convolution layers and 2 fully connected layers with the following architecture -

```
Cnn2_tanh(
  (network): Sequential(
    (0): Conv2d(3, 8, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (1): Tanh()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(8, 16, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (4): Tanh()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (7): Tanh()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (9): Flatten(start_dim=1, end_dim=-1)
    (10): Linear(in_features=128, out_features=64, bias=True)
    (11): Tanh()
    (12): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

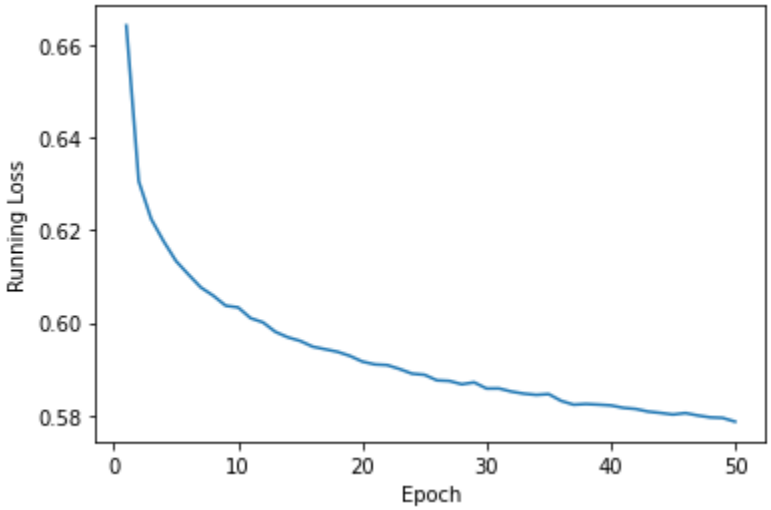
We trained the model for Relu/Tanh/Sigmoid activation functions and the results were as follows -

- **Relu -**
  - **Adam -**



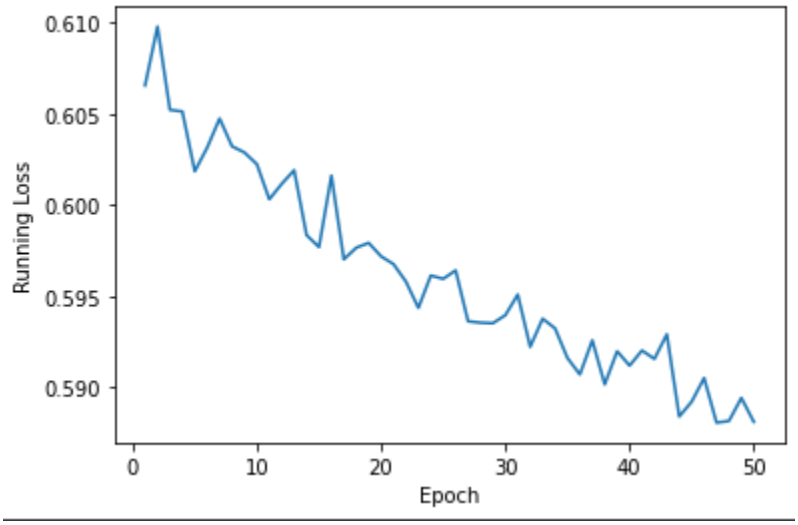
Training Time - 610.03 s
Classification Accuracy - 65.62%

  - **Adagrad -**



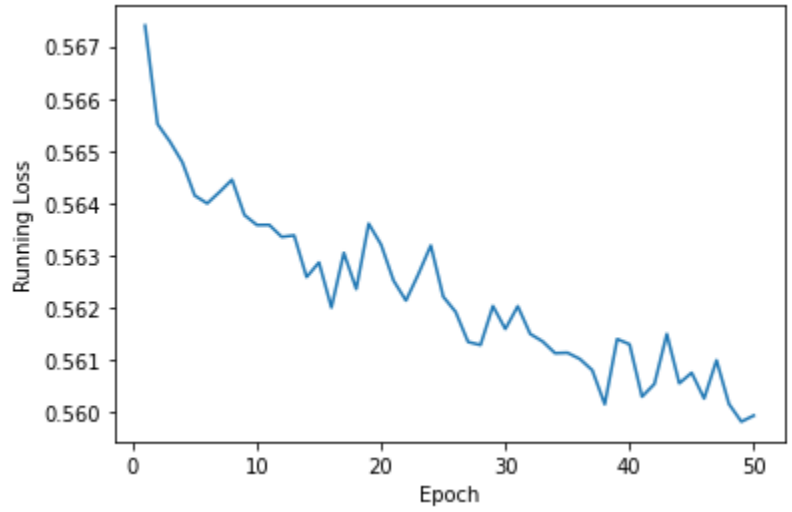
Training Time - 608.49 s
Classification Accuracy - 73.38%

- **SGD with momentum of 0.8 -**



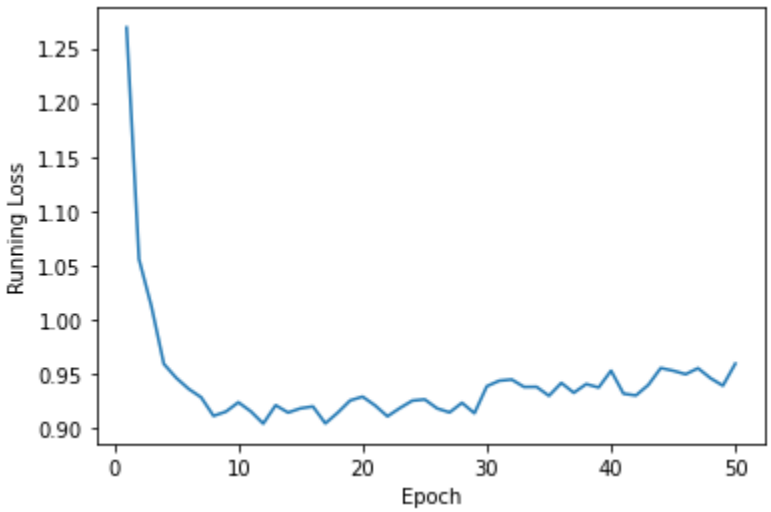
Training Time - 619.23 s
Classification Accuracy - 73.42%

- **SGD without momentum -**



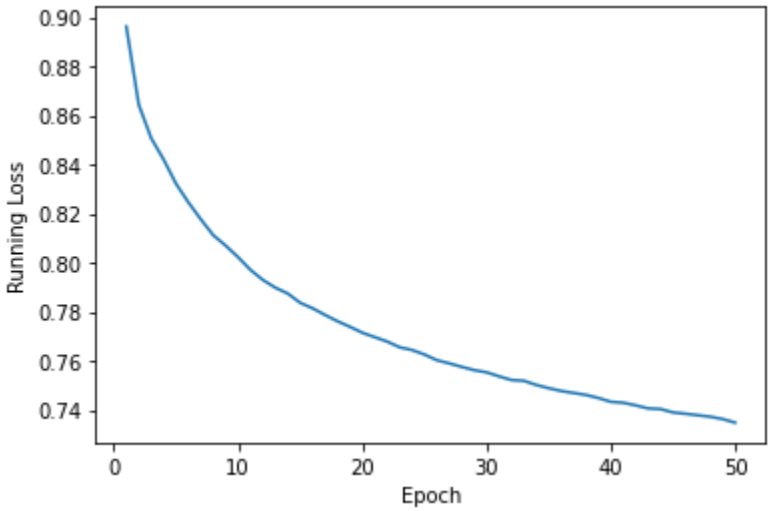
Training Time - 596.16 s
Classification Accuracy - 74.19%

- **Tanh -**
  - **Adam -**



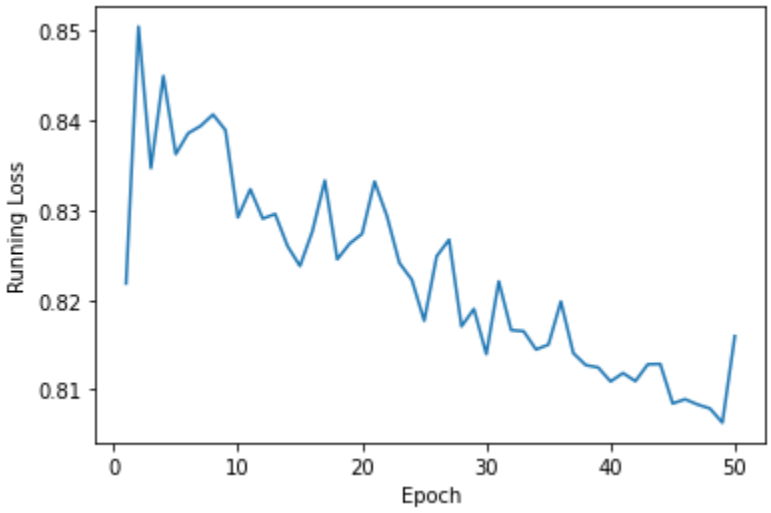
Training Time - 608.81 s
Classification Accuracy - 57.17%

  - **Adagrad -**



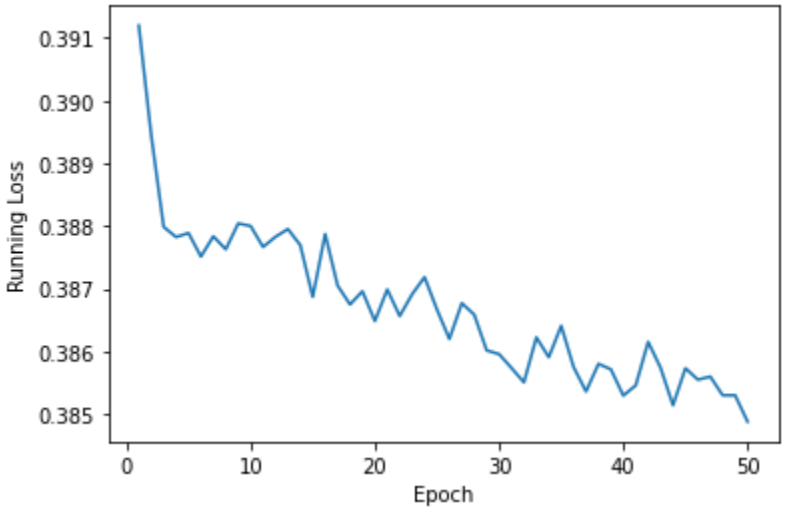
Training Time - 601.13 s
Classification Accuracy - 66.54%

- **SGD with momentum of 0.8 -**



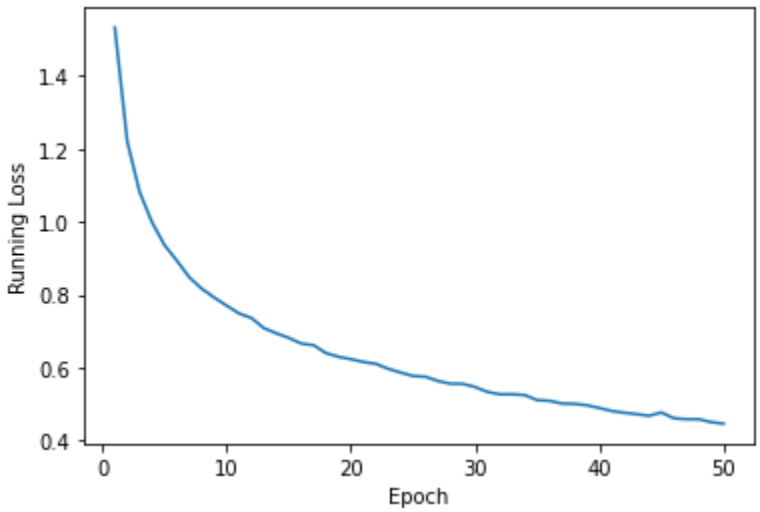
Training Time - 602.33 s
Classification Accuracy - 62.96%

- **SGD without momentum -**



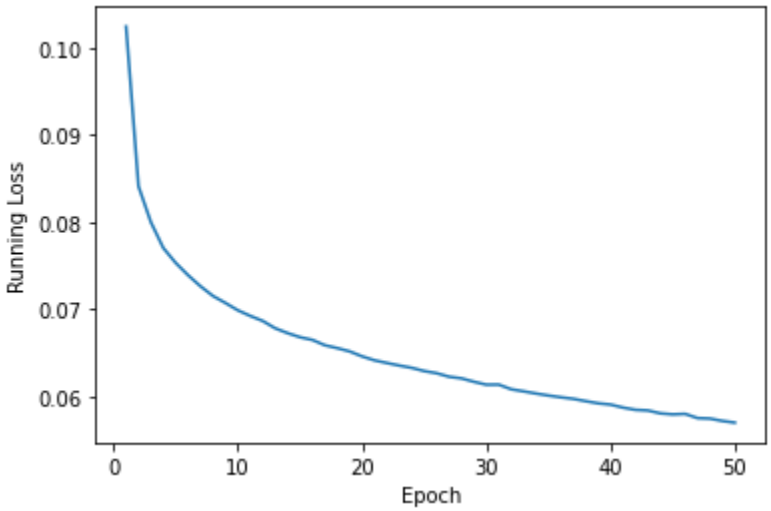
Training Time - 634.82 s
Classification Accuracy - 64.99%

- **Sigmoid -**
  - **Adam -**



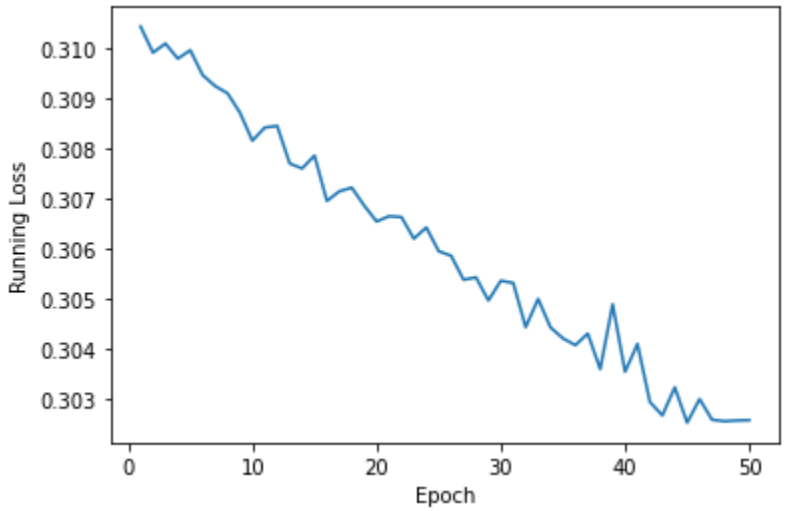
Training Time - 601.06 s
Classification Accuracy - 79.83%

  - **Adagrad -**



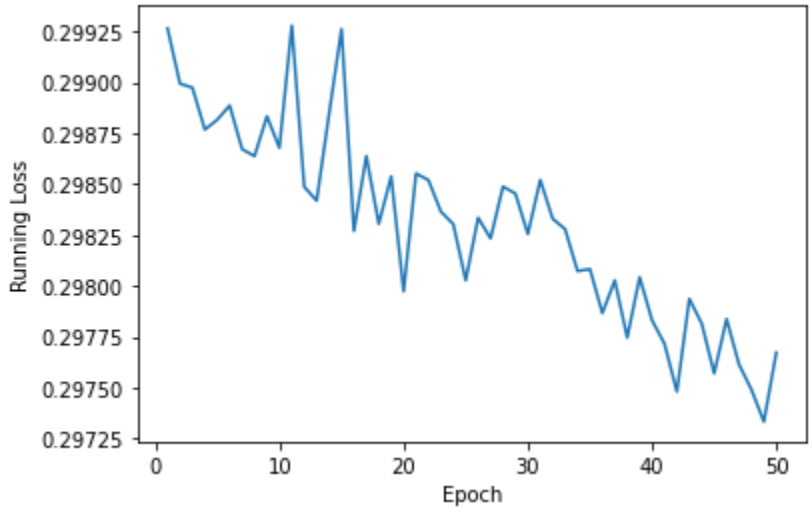
Training Time - 598.53s
Classification Accuracy - 86.88%

- **SGD with momentum of 0.8 -**



Training Time - 596.22 s
Classification Accuracy - 87.14%

- **SGD without momentum -**



Training Time - 604.55 s
Classification Accuracy - 87.36%

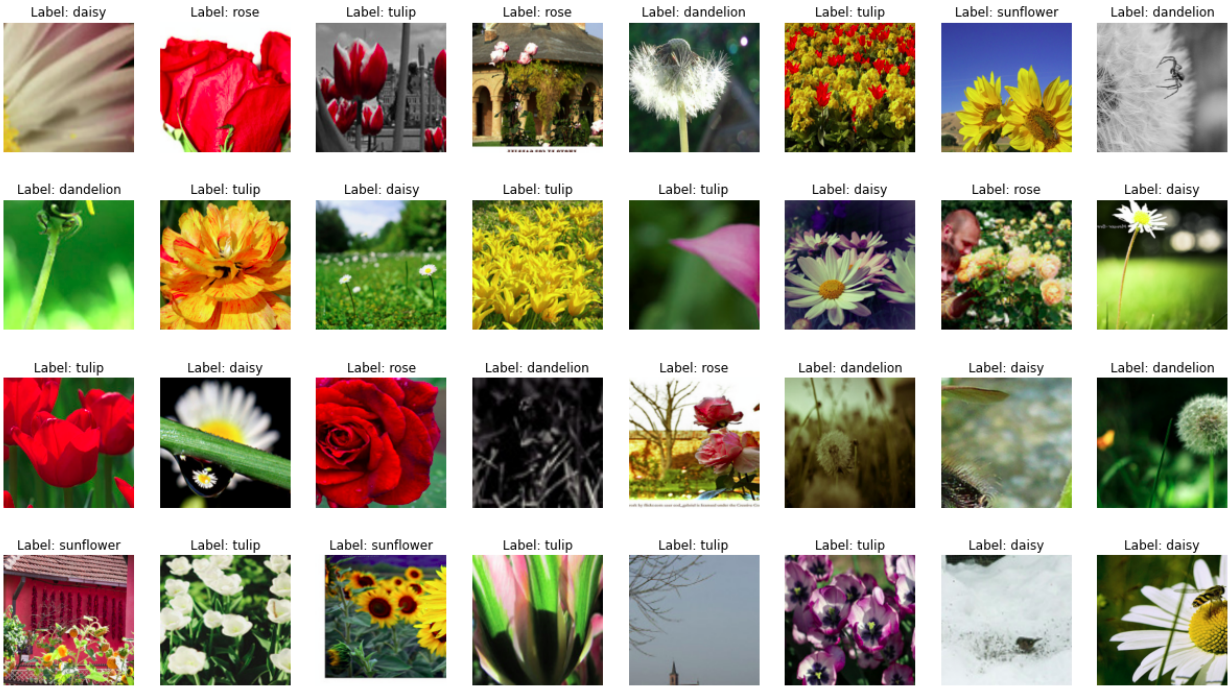
So upon observing the results, we can clearly see that the 2 convolutional layers + 2 fully connected layers network with Sigmoid activation outperforms the rest of the architectures. Adagrad, SGD with momentum, and SGD without momentum all give classification accuracy of greater than 99% on the test data. So our recommended architecture would be 2 convolutional layers + 2 fully connected layers network with cross-entropy loss and any optimizer from Adagrad, SGD with the momentum of 0.8, and SGD without momentum.

Question 3b:

The dataset was chosen: Flower Recognition Dataset (link)
Dataset Description:

- This dataset contains 4242 images of flowers.
- The data collection is based on data from Flickr, google images, and Yandex images.
- The pictures are divided into five classes: chamomile, tulip, rose, sunflower, and dandelion.
- For each class, there are about 800 photos. Photos are not high resolution, about 320x240 pixels.

Here are a few sample images from this dataset.



1. We used the **ResNet model** to create the base model. ResNet is pre-trained on the ImageNet dataset, a large dataset consisting of 1.4M images and 1000 classes.
2. Firstly, we freeze the convolutional base and use it as a feature extractor. Also, we added a classifier on top of it and train the top-level classifier.
3. Freezing the weights by setting “requires_grad == False” prevents the weights in a given layer from being updated during training.
4. We freeze the weights for all of the networks except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

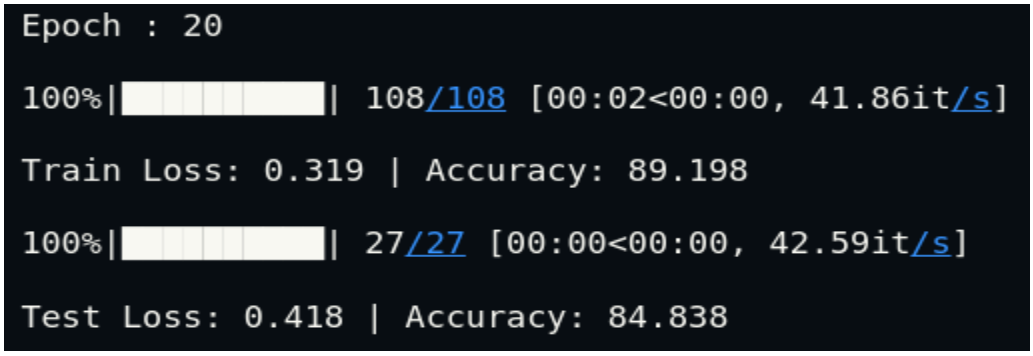
5. We reshaped the final layer to have the same number of outputs as the number of classes in the new dataset(Flower Recognition).
6. Finally, we created an optimizer that only updates the desired parameters(i.e “requires_grad == True”). Next, we make a list of such parameters and input this list into the SGD algorithm constructor.

Loss Function	Cross-Entropy Loss
Epochs	20
Batch Size	32
Learning Rate	0.0001
Momentum	0.9

Result:

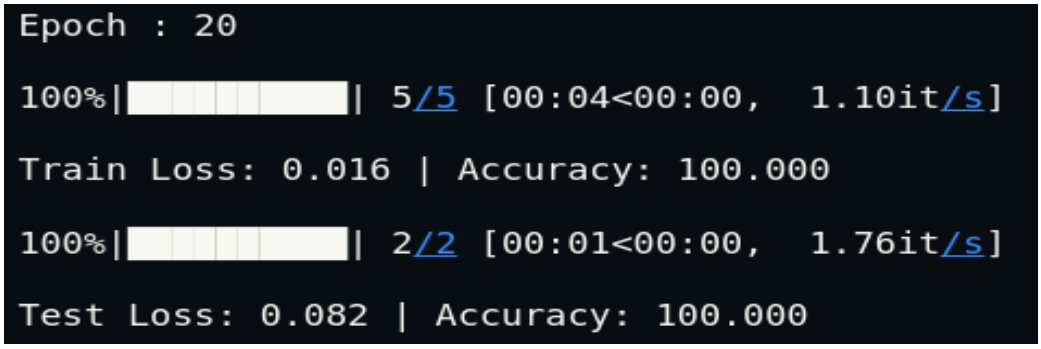
- **Flower Recognition Dataset:**

Train Accuracy = 89.198 %
Test Accuracy = 84.838 %



- **Bike Vs Horse Dataset:**

Train Accuracy = 100 %
Test Accuracy = 100 %



Question 3c:

5 additional features of YOLO V2 are:

1. YOLO V2 employs a new neural network architecture known as Darknet-19, which is more powerful and has fewer layers than the previous version of YOLO.

2. **Anchor boxes:** Anchor boxes are used by YOLO V2 to improve object detection accuracy. These boxes are pre-defined, fixed shapes used to represent objects of various sizes and aspect ratios.

3. **Batch normalization:** YOLO V2 employs batch normalization to improve model stability during training. This technique normalizes each layer's activations to have a zero mean and unit variance.

4. **High-resolution classifier:** YOLO V2 employs a high-resolution classifier to improve object detection accuracy. To better capture the details of small objects, this classifier was trained on a dataset of high-resolution images.

5. **Multi-scale training:** To improve the detection of small objects, YOLO V2 employs a multi-scale training approach. To better capture the features of small things, the model is trained on images of various sizes and scales.

Question 3d:

Sort and DeepSORT are both object-tracking algorithms that are used to track objects in video streams. Differences between the two algorithms are as follows:

- Sort (Simple Online and Realtime Tracking) is a simpler algorithm that only uses a Kalman filter for object tracking, whereas DeepSORT (Deep Learning for Multi-Object Tracking) is a more complex algorithm that uses deep neural networks for object detection and feature extraction, in addition to a Kalman filter for object tracking.

- Sort assumes that the objects being tracked have already been detected and localized in the frame, whereas DeepSORT includes a detection network that can be trained to detect and track objects.

- DeepSORT uses deep neural networks to extract more detailed and informative features such as appearance and motion information. In contrast, Sort only uses the position and velocity of the objects for tracking.

- DeepSORT is generally thought to be more robust and accurate in tracking objects than Sort, especially in scenarios where objects may occlude or overlap.

- DeepSORT requires more computational resources than Sort due to its use of deep neural networks, potentially making it slower and less efficient for real-time applications. Sort, on the other hand, is faster and less computationally intensive.

Approach:

- Loaded the respective models i.e. YOLO v5/ FasterRCNN_ResNet50_FPN_V2

- Processed the video frame by frame

- For each frame, Obtain the bounding boxes in the form of (x_min,x_max,y_min,y_max) and also the id associated with the item from the tracker(SORT / DEEPSORT). Update the trackers according to their format.

- Maintain two lists, one for total cars and one for cars in the frame, and add every unique id encountered to this list and increase respective counts and display this on the image as a new id indicates the addition of a new car.

- Since the cars were clustered in particular regions in some videos, we filter out the detection range by specifying the range of x or y coordinates. The counter only counts cars present within the region

- Write out the obtained frames to a video

Outputs:

- YOLO+SORT
  https://drive.google.com/file/d/1n_704d3agAlNABqm0rPMNJvf0O3K8HJm/view?usp=sharing
- YOLO+DEEPSORT
- https://drive.google.com/file/d/1GpiGyqz57j2vyDk1hZybgJ7Iv-LYx4QO/view?usp=sharing

- FRCNN+SORT https://drive.google.com/file/d/1MwPxOkGWDnCT509MsAxVmkCwJrLAmv_S/view?usp=sharing
- FRCNN+DEEPSORT https://drive.google.com/file/d/1BciRmVVIm1LOytO7LQBYdvPrHI6mTsBt/view?usp=sharing

Notable Differences:

- Yolo is much faster to process the videos compared to frcnn. For our run, YOLO took around 1 min whereas frcnn took around 20 mins to complete with the SORT tracker.
- Deepsort is much more accurate in tracking object. Sometimes, Sort retracks the same object with a new id and thereby increasing the count of cars, but deepsort does not make this error.