Part-1)
This part answers the 1st Question of the Summer Induction Assignment

Step-1) Identifying a phenomenon and asking questions about it:

The purpose of this study is to identify the effect of various data augmentations on the performance of deep learning models. While in one direction of thought, it might seem logical that augmenting the data might lead to lowering the accuracy, but using augmentations (to various degrees) while training the model might help the model generalize well on non-training problems and hence, lead to a better overall performance.

Step-2) Literature review:
(While more official reviews may include reading of previous research papers, here I have read a few blogs)

The need to study the behaviour in response to data augmentations may help the bias-variance tradeoff. Training the model on augmented data may lead to a little less accuracy on the training set, this will ultimately help the model generalize well, leading to a higher accuracy on the test set and real-time performance.

Step-3) Ingredients:

The various augmentations may include:

1. DCGANs on various sets
2. Random Crops
3. Random Flipping (Horizontal + Vertical)
4. Adding Random Noise

Steps-4&5) Mathematical Formulations:

For all implementations, we have used the PyTorch framework.

1. For Random Crops, we are using the inbuilt transformations in PyTorch:
   torch.transformations.RandomCrop
2. For Horizontal/ Vertical Flipping, we use the inbuilt transformations in PyTorch:
   torch.transformations.RandomHorizontalFlip(p)
   torch.transformations.RandomVerticalFlip(p)
   Where p = flipping probability
3. To add random noise, a function ‘AddRandomNoise()’ has been implemented

Step-6) Planning the Model:

For networks that analyse images, we are using the **LeNet architecture** on both the MNIST and CIFAR-10 datasets.

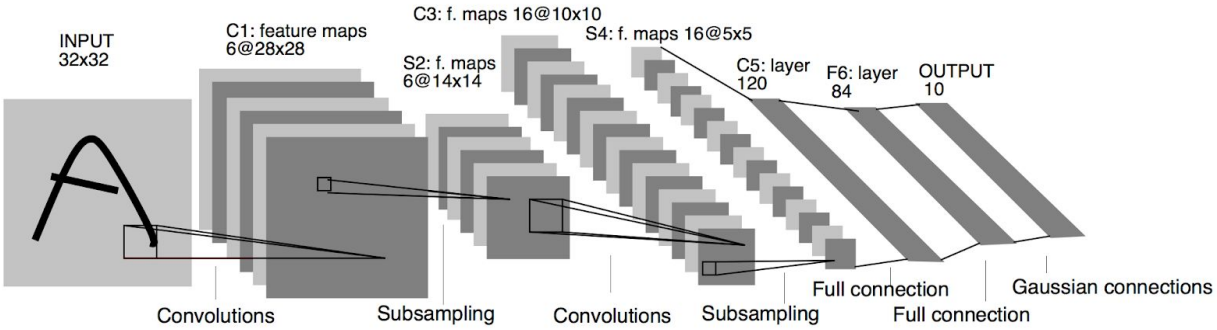

Image taken from : https://bp-gc.in/LeNet_Architecture

DCGANs (Deep Convolutional Generative Adversarial Networks) were implemented using the steps provided at https://bp-gc.in/DCGAN

Part-2)

Observations from the tasks provided and discussion:

1. Trained a Lenet on MNIST Dataset with the following observations:
    a. Accuracy on train set: 99%
    b. Accuracy on test set: 98%
    c. Accuracy on test set applied with random crops after padding with (1,1): 97%
    d. Accuracy on test set applied with random crops after padding with (2,2): 94%
    e. Accuracy on test set with added random noise (tensor + torch.randn(tensor.size))*k -
        i. k = 0.5, accuracy around 76%
        ii. k = 0.25, accuracy around 95%
        iii. k = 0.01, accuracy around 98%
        iv. k = 1, accuracy around 37%
    f. Accuracy on test set with added random noise (tensor + torch.randn(tensor.size))*k combined with random cropping after padding (x,x) -
        i. k = 0.25, x= 1, accuracy around 93%
        ii. k = 0.5, x = 1, accuracy around 72%
        iii. k = 0.5, x = 2, accuracy around 63%
        iv. k = 0.25, x= 2, accuracy around 85%
    g. Accuracy on test set after randomly flipping some images vertically and/or horizontally, with flipping probability p -

(p_v is the probability of flipping an image vertically, and p_h is the probability of flipping an image horizontally)

i. $p_v = 0.1$, $p_h = 0$, accuracy around 93%
ii. $p_H = 0.1$, $p_v = 0.1$, accuracy around 87%
iii. $p_h = 0.25$, $p_v = 0.25$, accuracy around 73%
iv. $p_h = 0.25$, $p_v = 0$ (or $p_h = 0$, $p_v = 0.25$), accuracy around 84%

The fact that the accuracy of the model decreases upon randomly flipping images (0without changing the labels), actually means that our model is detecting some of the flips.

h. Accuracy on test set after randomly flipping images horizontally (p_h =0.25) and vertically ($p_v = 0.25$), adding random noise (k=0.25) and random cropping after padding with (2,2): around 63%

- From the above augmentation techniques tested on our model, we learn that the model is not really robust against major augmentation of the data (as seen in e-iv.). While this may seem evident, as in major data augmentation, we are altering several properties of the data, the model seems to handle minor augmentations pretty well (e-ii. , e-iii., c).
- The model shows some ability to detect flips in the dataset. By flipping about 25% of the images horizontally (or vertically) in e-iv , the accuracy decreases by about 14%. Note that we are still labelling the flipped images as the original ones only. Therefore, by mislabelling 25% of the images, we find that about 14% of the images are no longer labelled as the original labels. Which means that our model has detected those flips. Hence, the model has some ability to detect flips in the dataset.

2. On the CIFAR-10 dataset, the LeNet architecture gave a base accuracy of about 60% on the test set.

3. In the code provided, a DCGAN has also been implemented to generate images similar to the ones in CIFAR-10 dataset.