MACHINE LEARNING (BITS F464)

ASSIGNMENT-2

SUBMITTED TO- PRATIK NARANG SIR

SUBMITTED BY- SHUBH NEMA
ID-2021A1PS2602P

Date-29/04/2023

INTRODUCTION TO THE REPORT

The purpose of this report is to implement a new learning procedure for neural networks, called the forward-forward algorithm, proposed by Geoffery Hinton, as well as to test its validity and compare its usability to the classical backpropagation approach.

This report is presented as a part of the Assignment for the course BITS-F464 Machine Learning. I would like to thank my professor- Dr. Pratik Narang Sir for the immense support and guidance he provided for the preparation of the report. This is to certify that the report is original, with proper citations wherever required.

NEED FOR THIS APPROACH/ ISSUE WITH BACKPROPAGATION-

The classical method of training artificial neural networks proceeds in two phases. In the forward pass, the input features are traversed through the hidden layers, they get modified by the effect of weights and activation functions, and a prediction is made. The prediction is then compared to the actual label, and hence the error is calculated

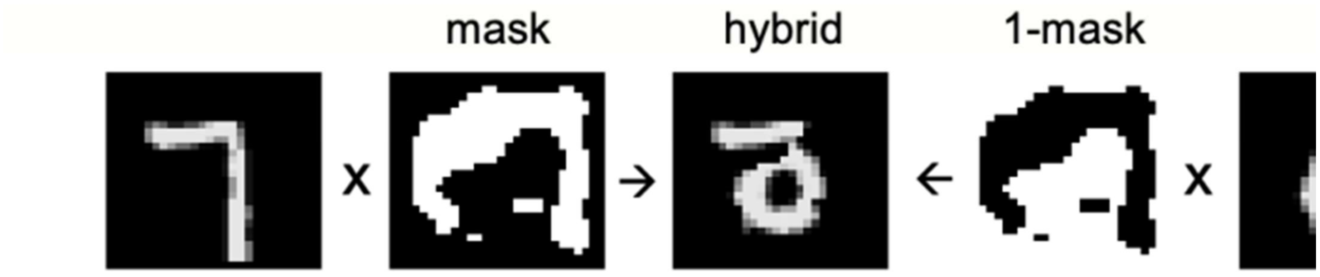
In the second phase, the weights of all the layers have to be modified such that the error minimizes. For that purpose, backpropagation comes into the picture which uses partial derivative and chain rule to calculate gradients and make the necessary adjustment. This approach requires storing the previous weights until the next cycle backpropagates to make changes in them

Although the backpropagation approach is one of the backbones of Deep learning and is much attributed to the success of the same, it is still much different from how the human brain processes information. According to Hinton, One of the main limits of backpropagation is the detachment of learning and inference. To adjust the weights of a neural network, the training algorithm must stop inference to perform backpropagation. In the real world, the brain receives a constant stream of information, and “the perceptual system needs to perform inference and learning in real-time without stopping to perform backpropagation. Moreover, there is no evidence that the cortex explicitly propagates error derivatives or stores neural activities to use in a subsequent backward pass.

The requirement is to mimic the working of the human brain in order to make neural networks much more efficient.

ALGORITHM DESCRIPTION-

The idea of the basic forward-forward algorithm is to replace the forward and backward pass of the Backpropagation algorithm with two forward passes that work in exactly the same way as each other, but on different data and with opposite objectives. This means that we will now have a forward pass with positive data and another with negative data. By positive data, we mean data that has a correct label associated with it and by negative data, we mean data that has an incorrect label associated with it(For the case of supervised learning). Negative data consists of samples that have been intentionally manipulated to resemble non existant features. An image given in the research paper by Prof. Hinton explains this concept.



Since only forward passes are being made, the use of backpropagation is eliminated . The objective function will be defined for each layer as compared to the backpropagation approach which had an objective function at the final output layer. The objective function at each layer will try to increase the amount of goodness for positive data (i.e. as close as possible to 1) and decrease the same for negative data (close to 0). For the same purpose, weights will be modified by calculating the derivative (but at the same layer unlike in backpropagation where a complete backward pass occurs and weights of all the layers get updated simultaneously in that pass).

The amount of goodness can be considered to be the square of the activities in a layer. Goodness is defined as

$$\text{Goodness} = \text{Sigmoid}\left(\sum \mathbf{a}_i^2 - \theta \right)$$

One problem in this approach is that the activations of a former layer may affect the activations of the later layer since the later layer may directly rely on the information contained by the activations of the previous layer and hence effective weight training for all these layers may not be possible. For this purpose, normalization is made to the activations of each layer before them being passed on to the next layer in order to improve the overall fit of the model.

For small datasets such as the MNIST in which we are interested only in one task and we want to use a small dataset that does not have the capacity to model the full distribution of input data supervised learning is a sensible choice. This is done by generating positive and negative data as explained above, for which the label acts as a distinction. The algorithm should ignore all features of the image that do not correlate with the label.

In our report, we will apply the Forward -Forward algorithm as a case of supervised learning, although the algorithm may also be applied to unsupervised learning problems.

We note that the forward-forward algorithm is a greedy algorithm i.e. the information learned in the later layers can not be used to update the information in the previous layers which is a big disadvantage as compared to backpropagation. This problem can be overcome by treating a static image as a video input that is processed by a multi-layer recurrent neural network.

To understand the process, take the MNIST dataset for example. We first encapsulate the label within each image of the training set in order to generate positive data. We then encapsulate some random wrong labels on the images to generate negative data. The original paper suggests that 4 hidden layers should be used, each containing 2000 neurons and having ReLU activation.

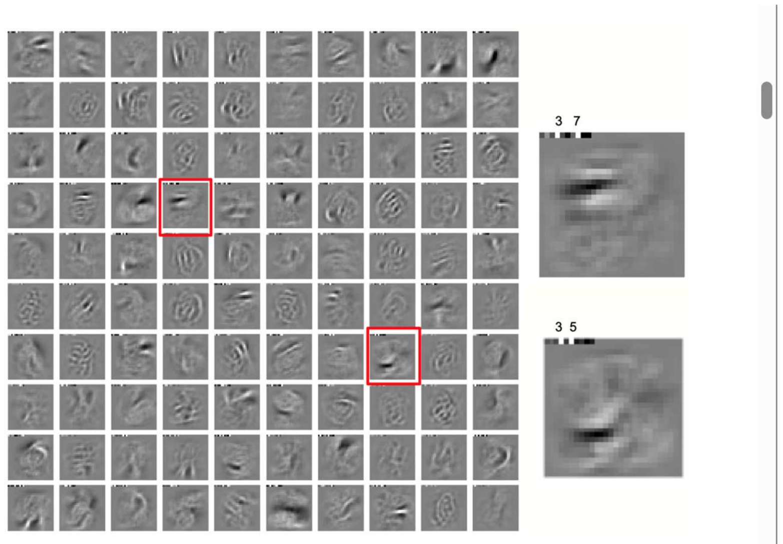


Figure 2: The receptive fields of 100 neurons in the first hidden layer of the network trained on jittered MNIST. The class labels are represented in the first 10 pixels of each image.

Now to assess the model based on the test data, take an unlabeled image and generate its 10 copies, each of them encapsulated with one label between 0-9. Feed each image one by one and whichever of them has the highest value of goodness, is the prediction.

The algorithm implemented can be understood in the following way-

- Import the libraries(Keras, Tensorflow and reticulate in R to interface with Python and TensorFlow)
- Loading the required datasets(MNIST and CIFAR-10)

- Visualize the dataset
- Define an FFDense Layer, which holds an object that acts as a base dense layer. Use Adam as the optimizer with learning rate as 0.03, and define a threshold value and number of epochs.
- Override the call method to perform normalization over the complete input space. Then implement the Forward Forward algorithm which accepts both positive and negative data. Calculate the loss per sample.
- Define an FFNetwork Layer to override train_step, predict and implement 2 custom functions for per-sample prediction and overlaying labels
- Fit the network using the training dataset
- Visualize results
- Predict the accuracy of the network using test dataset.

One of the major advantages of the FF approach as compared to backpropagation is that there is no need to store the activations, something which is much required in backpropagation to compute the gradients and often results in nasty out-of-memory errors.

Moreover, a lot of memory can be saved with this approach, atleast in concept as we do not need to remember anything outside of a layer to update the weights in that layer

DATASET DESCRIPTION-

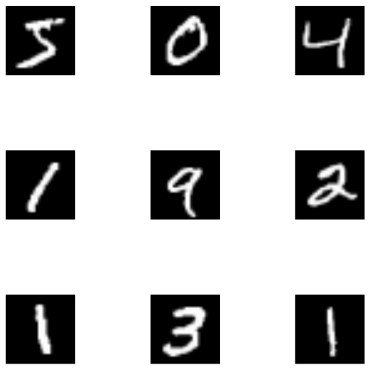
MNIST DATASET

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of 70,000 grayscale images of handwritten digits, each of size 28x28 pixels, with corresponding labels indicating the correct digit from 0 to 9.

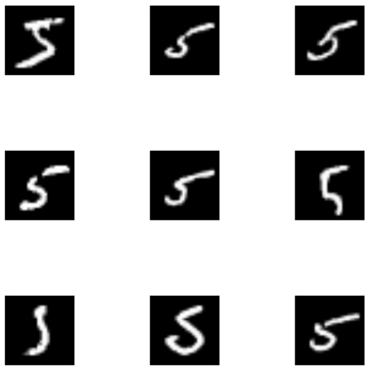
The dataset is divided into two parts: a training set of 60,000 images and a test set of 10,000 images. The images in the training set are used to train machine learning models, while the test set is used to evaluate the performance of these models.

The MNIST dataset is commonly used for image classification tasks, where the goal is to correctly classify the digit in each image. It has become a standard benchmark dataset for comparing the performance of different machine learning algorithms and has been used in a wide variety of applications, including handwriting recognition, OCR (Optical Character Recognition), and computer vision research.

Below given is a matrix taken out from the training sample of the MNIST dataset, which shows some of the handwritten digits.



Another image given above represents the variety in the dataset, i.e. the different number of variations in data associated with a single label.



CIFAR-10 DATASET

The CIFAR-10 dataset is another popular benchmark dataset in the field of machine learning and computer vision. It consists of 60,000 color images in 10 classes, with 6,000 images per class. Each image is 32x32 pixels in size and has a color depth of 3 (representing the RGB channels).

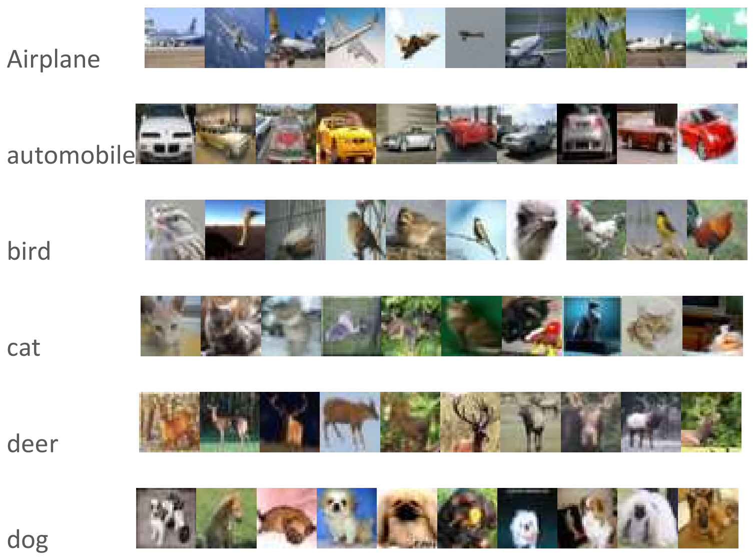
The 10 classes in the CIFAR-10 dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The images were collected from a variety of sources, and the dataset was first introduced in 2009.

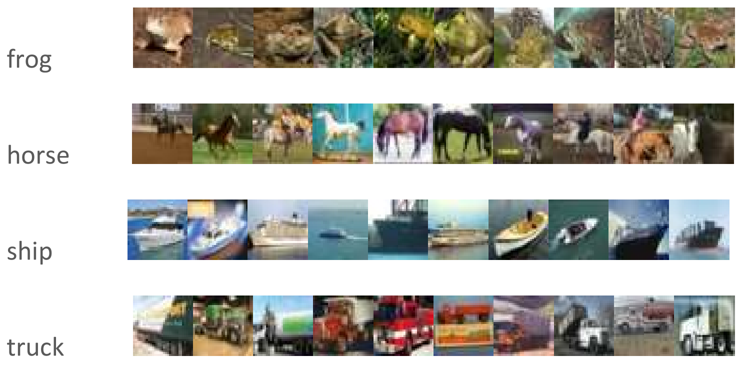
Like the MNIST dataset, the CIFAR-10 dataset is divided into two parts: a training set of 50,000 images and a test set of 10,000 images. The training set is used to train machine learning models, while the test set is used to evaluate the performance of these models.

The CIFAR-10 dataset is commonly used for image classification tasks, where the goal is to correctly classify each image into one of the 10 classes. It is a more challenging dataset than MNIST, as it contains color images with more variability in the visual appearance of objects.

The CIFAR-10 dataset has been used in a wide variety of applications, including object recognition, image segmentation, and visual question answering. It has also been used as a benchmark dataset for testing the performance of deep learning models and has been the basis for several competitions and research projects in the field of computer vision.

The below image shows some random images from the 10 different classes of the dataset.



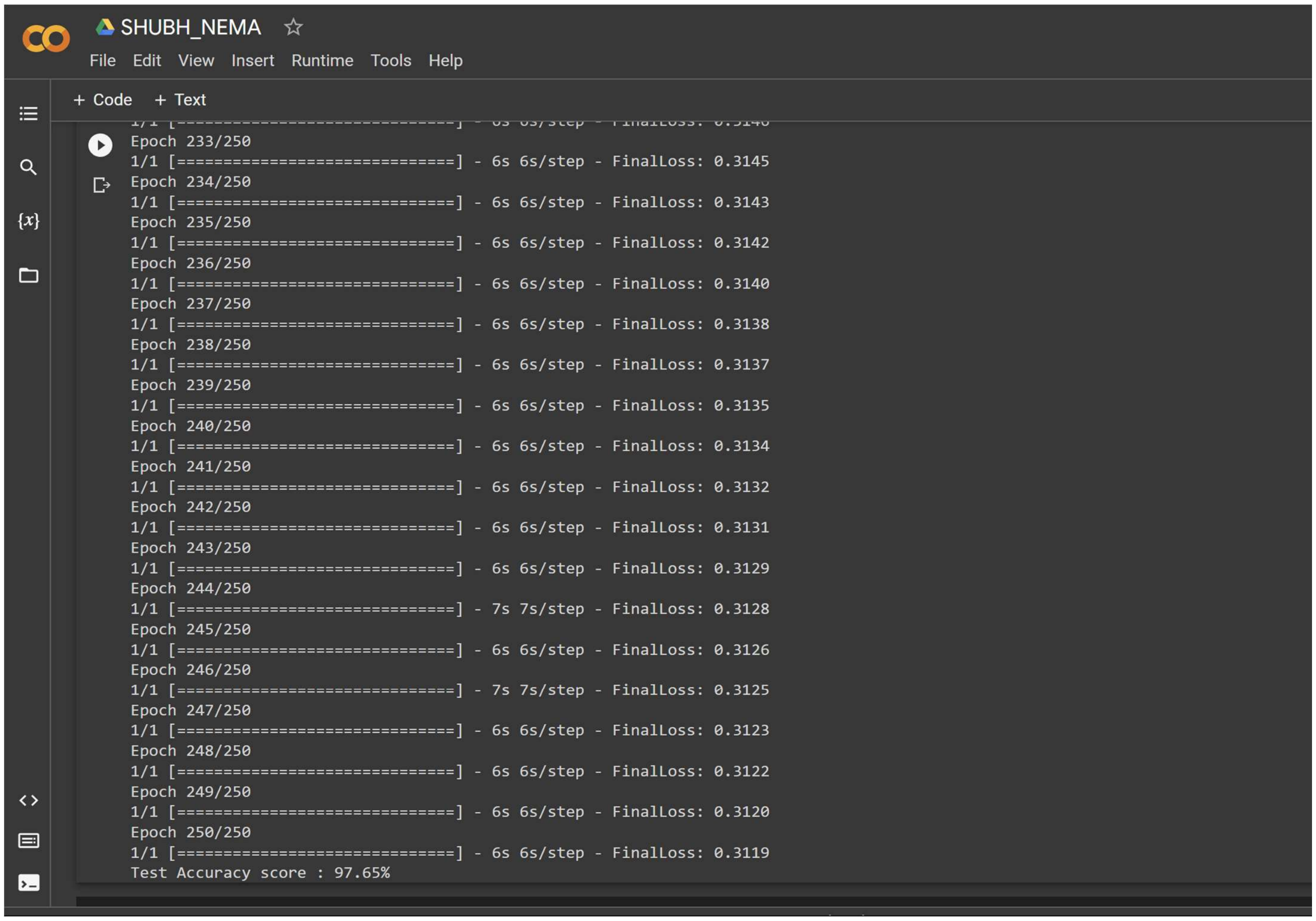


RESULTS-

We ran the model by trying to fit the training samples of the MNIST dataset onto the model. We ran 250 model epochs, which performed 50*250 epochs on each layer.

A picture of the real-time execution of the model is attached.

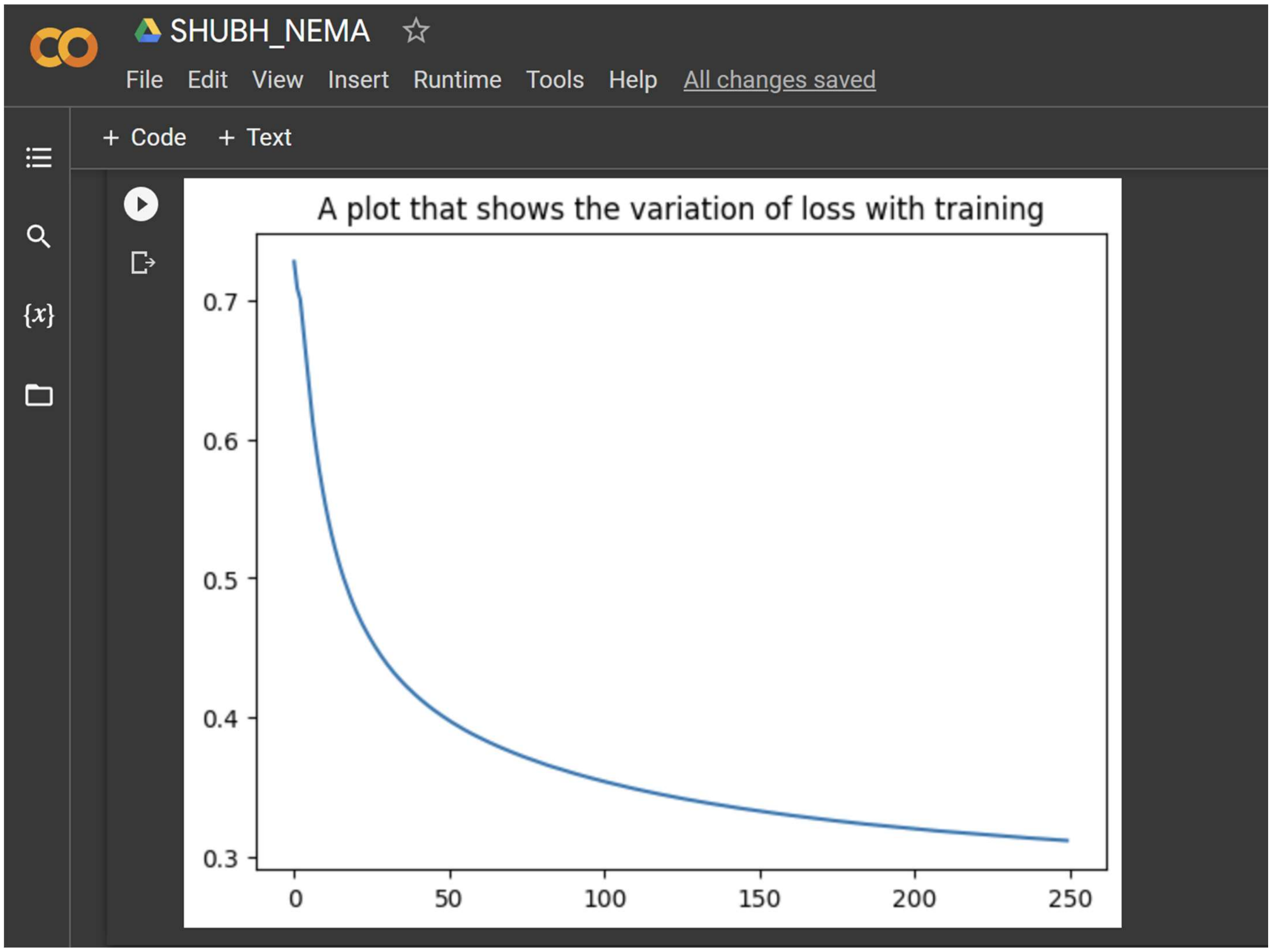
The execution was done on Google Colab because the R studio was facing some compatibility issues to run the TensorFlow and Keras libraries.



```
Epoch 233/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3145
Epoch 234/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3143
Epoch 235/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3142
Epoch 236/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3140
Epoch 237/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3138
Epoch 238/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3137
Epoch 239/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3135
Epoch 240/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3134
Epoch 241/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3132
Epoch 242/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3131
Epoch 243/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3129
Epoch 244/250
1/1 [==============================] - 7s 7s/step - FinalLoss: 0.3128
Epoch 245/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3126
Epoch 246/250
1/1 [==============================] - 7s 7s/step - FinalLoss: 0.3125
Epoch 247/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3123
Epoch 248/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3122
Epoch 249/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3120
Epoch 250/250
1/1 [==============================] - 6s 6s/step - FinalLoss: 0.3119
Test Accuracy score : 97.65%
```

The final loss after 250 epochs was reported to be around 0.3119 and the accuracy score of implementing the MNIST test dataset on the model was found to be 97.65%

The plot given below shows the trend of the total loss with increasing number of epochs.



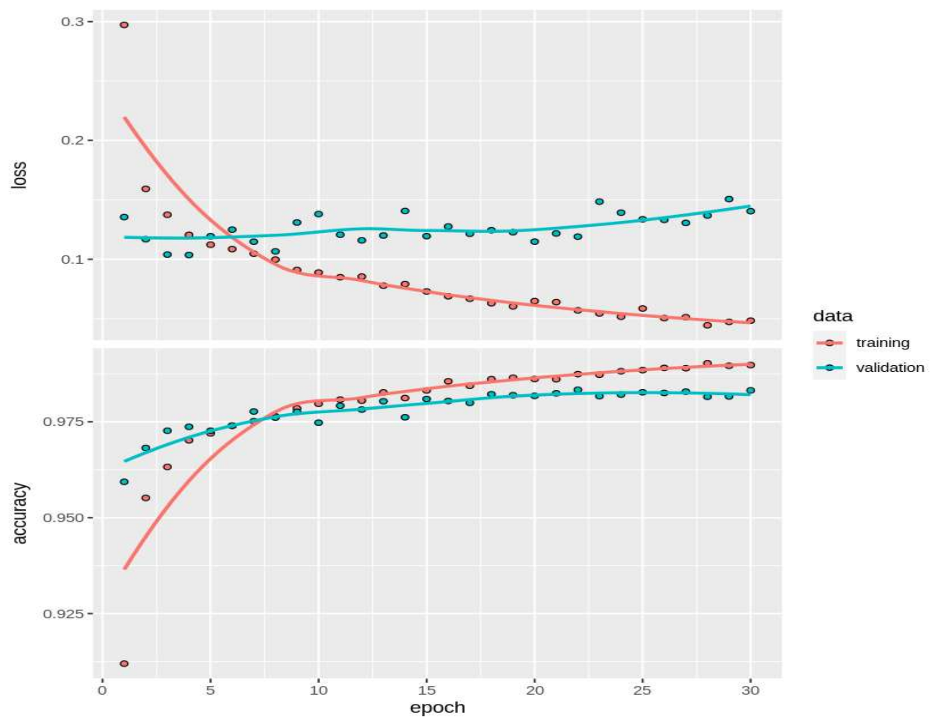
When applied on to the CIFAR-10 dataset, the algorithm produced certain out-of-memory as well as invalid input errors due to which the code was not runnable. This might be due to the inputs of the CIFAR-10 datasets being highly memory intensive(RGB images, dimension 32*32*3 each) as a result of which the neural network so formed exceeded the GPU's capacity to process.

However, with certain modifications to the algorithm and using more powerful processors, the algorithm might be able to run without errors.

DISCUSSION-

- One of the major takeaways is that the algorithm is well applicable to small datasets like the MNIST and CIFAR-10, but it's use on larger datasets has not been validated yet.

- Since the algorithm is still very much in its experimental phase, it does not yield state-of-the-art results. On the MNIST dataset, As compared to the backpropagation approach, with 5 layers and 30 epochs which yielded an accuracy of approximately 98.57%(implemented in the code), the accuracy obtained(around 97%) is still less, although, with proper tuning, it is supposed to come close to the same. The picture attached below shows the accuracy and loss obtained with respect to the number of epochs using the **cross-validation** approach.



- As defined by Prof. Hinton in his paper, a 1.36% test accuracy error with 2000 unit, 4 hidden layer , fully connected network over 60 epochs was obtained using the Forward- Forward algorithm. The implementation of this algorithm which we did in our report resulted in an accuracy of 97.65%, which is again less than what Prof. Hinton described in his paper. So there is still scope that there might be some improvements in the code attached with this report which might increase the accuracy.

- We note that the Forward- Forward algorithm is much slower than backpropagation. The reason for the significant time difference might be the two passes(positive and negative) which are done simultaneously which increases its run-time. This difference might increase further when larger datasets will be used.

- The memory usage for the algorithm may be slightly less than that of the backpropagation approach since it does not need to store any value outside of a single layer to make updates to the weights of that particular layer, although it will depend on the way algorithm is implemented and may vary for different users.

CONCLUSION

In this paper, we tried to implement the Forward- Forward algorithm as mentioned by Prof. Hinton in his paper. We found that the algorithm produces well-acceptable results when applied on small datasets like MNIST, but its application on much larger datasets is not evaluated yet and probably the algorithm requires some more fine tuning to be able to compete with the classical backpropagation approach when applied on the larger datasets.

Backpropagation is certainly a better approach as of now in cases where power is not the limitation. Moreover, its speed is lesser than the former approach. Taking the current situation, A considerable amount of research has to be done in order to generalize the algorithm for a large variety of operations.

The algorithm is the potential successor for backpropagation, but some significant changes are required to be made in order for it to be more productive than other approaches.

REFERENCES-

- Hinton, Geoffrey. "The forward-forward algorithm: Some preliminary investigations." arXiv preprint arXiv:2212.13345 (2022) - 2212.13345.pdf (arxiv.org)
- https://www.youtube.com/watch?v=NWqy_b1OvwQ

- Using the Forward-Forward Algorithm for Image Classification (keras.io)
- https://bdtechtalks.com/2022/12/19/forward-forward-algorithm-geoffrey-hinton/
- https://www.kaggle.com/datasets/swaroopkml/cifar10-pngs-in-folders
- https://medium.com/mlearning-ai/pytorch-implementation-of-forward-forward-algorithm-by-geoffrey-hinton-and-analysis-of-performance-7e4f1a26d70f