**A REPORT**

**ON**

**The forward-forward algorithm: Some preliminary investigations**

**BY**

**Name - Rashi Singh**

**ID No. - 2020A5PS1160P**

**Discipline - B. Pharm**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**Department of Computer Science and Information Systems**

**Semester II (2022-2023)**

**Machine Learning (BITS F464)**

**ACKNOWLEDGEMENT**

I would like to use this opportunity to express my gratitude to everyone who supported me throughout this project. First and foremost, I would like to thank Dr. Vandana Agarwal, my Machine learning course (BITS F464) instructor in charge and Pratik Narang, my Machine learning instructor, for tirelessly and effectively guiding me throughout the course duration and ensuring all our doubts are cleared. Dr. Vandana Agarwal mam has been exceptional in carrying out the smooth functioning of our learning in this course. Next, I would like to express my sincere gratitude to the lab instructor, Syed Sameen Ahmad Rizvi for guiding me right from the course's inception. I sincerely acknowledge him for extending his valuable guidance, literature support, and valuable critique of my work. Finally, I would like to thank my family members and friends for their continuous support, which has been instrumental in the completion of this project.

**TABLE OF CONTENTS**

1.Introduction…………………………………………………….4

2.Algorithm description (including all mathematical details

and figures depicting your understanding) ………………………6

3.Data Sets Description………………………………………….14

4.Results…………………………………………………………16

5.Discussion……………………………………………………..17

6.Conclusion……………………………………………………..18

**Introduction**

Machine learning has advanced significantly as a result of the use of neural networks. Neural networks are able to learn from data and predict outcomes correctly. One of the most popular algorithms for training neural networks is backpropagation. Deep neural network training, however, might be slowed down by the computationally expensive gradient calculation procedure. Geoffrey Hinton proposed the Forward-Forward algorithm, which substitutes two forward passes for the forward and backward passes of backpropagation. For the classification problem in the MNIST and CIFAR-10 datasets, we will develop the Forward-Forward algorithm in R and compare its performance to backpropagation in this study.

Machine learning has been transformed by deep learning, and neural networks have become a prominent approach for a variety of applications. Neural networks are highly suited for a variety of tasks, including image and speech recognition, natural language processing, and robotics because they can learn from data and generate precise predictions. Backpropagation is one of the most often used methods for neural network training.

An iterative process called backpropagation determines the gradient of the loss function with respect to the weights of the network. The network's performance is enhanced by using the gradient to update the weights, which lowers the loss function's value. Deep neural network training, however, might be slowed down by the computationally expensive gradient calculation procedure.

Geoffrey Hinton introduced the Forward-Forward algorithm as a substitute for backpropagation in 2022. The Forward-Forward algorithm substitutes two forward passes for the forward and backward passes of the backpropagation. One forward pass uses actual positive data, and the other forward pass uses potential network-generated negative data. The Forward-Forward algorithm does not necessitate computing the gradient of the loss function with respect to the network parameters, unlike backpropagation. Instead, each optimization step can be carried out locally, and each layer's weights can be adjusted right away after it completes its forward pass.

In this report, we will use the MNIST and CIFAR-10 datasets to classify data using the Forward-Forward method implemented in R. Its performance will be compared to that of backpropagation, and its benefits and drawbacks will be examined. The MNIST dataset, made up of 60,000 training images and 10,000 test images, is a commonly used benchmark dataset for handwritten digit recognition. The CIFAR-10 dataset is a more complicated dataset with ten discrete classes and 50,000 training images and 10,000 testing images, each measuring 32x32 pixels. We can learn more about the Forward-Forward algorithm's potential as a backpropagation substitute for deep neural network training by using it to these datasets.

In general, the Forward-Forward algorithm may offer a backpropagation-free alternative for training deep neural networks. It might be more advantageous when computing gradients is challenging or expensive, and on some datasets, it might be quicker than backpropagation. The remaining sections of the report will include a thorough explanation of the Forward-Forward algorithm, the datasets used, the discussion attained, and a comparison of the two algorithms' performance.

**Algorithm description**

Using the Forward-Forward algorithm, deep neural networks can be trained. It entails substituting two forward passes for the front and backward passes in the backpropagation. Real data is used in the first run, and potentially harmful data that the network itself may produce is used in the second step.

Emeritus Professor Geoffrey Hinton came up with the idea for the algorithm at NeurIPS 2022. It is based on the concept of employing two forward passes, one with positive (i.e., real) input and the other with negative data that might be produced by the network itself.

Any feedforward neural network architecture, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be used with the Forward-Forward algorithm. The following steps make up the algorithm:

1. Set the weights and biases of the network parameters either at random or using a trained model.
2. Forward pass using accurate data To calculate the output of the network, perform a forward pass on the input data. The loss function, which calculates the difference between the network's output and the ground truth labels, is based on this output.
3. Forward pass with negative data: Feeding the network's output back into the network will produce negative data. By doing this, a new input that is somewhat similar to the original input yet different from it is created. To compute the output of the network on the negative data, run a forward pass on the negative data.
4. Compute a loss function calculation using the negative data: The loss function, which calculates the difference between the network's output and the ground truth labels for the negative data, should be done using the network's output on the negative data.
5. Update the network parameters: utilise a common optimisation algorithm, like stochastic gradient descent (SGD), to calculate the difference between the network's output on the real data and the negative data. Then, utilise this difference to adjust the network's parameters.

The network iteratively repeats steps 2 through 5 until it reaches a good solution.

The Forward-Forward method and backpropagation are fundamentally different from one another since the former does not call for calculating the gradient of the loss function with respect to the network parameters. Instead, each optimisation step can be carried out locally, and each layer's weights can be adjusted right away after it completes its forward pass. Unlike backpropagation, which calls for computing the gradients in a backward pass and then updating the weights in a subsequent optimisation step, this method does not require this.

The Forward-Forward algorithm has the benefit of potentially being faster than backpropagation, particularly for smaller datasets like MNIST. This is so that the Forward-Forward algorithm can avoid using computationally expensive backward passes to calculate gradients.

However, compared to backpropagation, optimising the performance of the network may be more challenging with the Forward-Forward algorithm. This is because finding the ideal set of weights may be more difficult if updates made by the Forward-Forward algorithm aren't as smooth as those made via backpropagation. Additionally, it may be more challenging to converge to a good solution due to the noise that the Forward-Forward algorithm updates introduce.

In conclusion, the Forward-Forward algorithm is a promising replacement for backpropagation for deep learning. On some datasets, it can be faster than backpropagation and does not necessitate computing the gradients in a backward pass. However, because of the noise that the updates introduce, it might be more challenging to optimise the network's performance than with backpropagation.

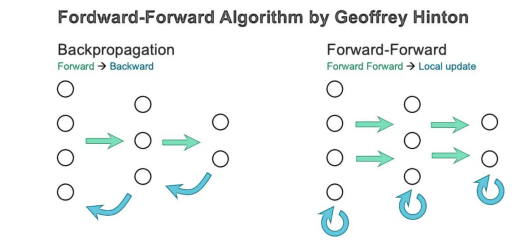
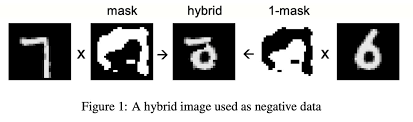


Figure 1: Diagrammatic representation of the Forward-Forward algorithm. The algorithm involves two forward passes: one with real data and the other with negative data that can be generated by the network itself. The weights of each layer are updated after each forward pass.

Assume that the sum of the squares of the activity of the rectified linear neurons inside a layer constitutes the goodness function for that layer. The goal of learning is to raise goodness much above a predetermined threshold for actual data and to significantly lower it for actual data that is negative. When the likelihood that an input vector is positive (i.e., real) is supplied by applying the logistic function, σ, to the goodness, minus some threshold, θ, the goal is to accurately categorise input vectors as positive data or negative data.



| **p(positive) = σ ( ∑j yj^2− θ)** |
| --- |

where yj is the activity of hidden unit j before layer normalization. The negative data may be predicted by the neural net using top-down connections, or it may be supplied externally.

**Relationship to boltzmann machines**

The aim of Boltzmann Machine learning is to make the distribution of binary vectors on visible neurons match the data distribution by updating the states of hidden neurons.

| ∂KL(Pdata||Pmodel)/ |
| --- |

**Learning multiple layers of representation with a simple layer-wise goodness function**

By setting the sum squared activity of the hidden units to be high for positive data and low for negative data, it is simple to see how a single hidden layer can be learned. However, if the first hidden layer's activities are then used as the input for the second hidden layer, it is easy to tell positive from negative data just by looking at the first hidden layer's activity vector's length. No new features have to be learned. Before utilising the length of the hidden vector as input to the following layer, FF normalises it to avoid this (Ba et al., 2016b; Carandini and Heeger, 2013). This compels the subsequent hidden layer to use information about the relative activity of the neurons in the first hidden layer and removes all of the information that was used to calculate goodness in the first hidden layer. The layer-normalization5 has no impact on these relative activities. In other words, the first hidden layer's activity vector has a length and an orientation. Only the orientation is passed to the following layer, and the length is used to define the goodness for that layer.

**A simple unsupervised example of FF**

There are two major issues with FF that require resolution. First, does it learn efficient multi-layer representations that accurately capture the structure in the data if we have a reliable source of negative data?

Second, where do the unfavourable statistics come from? In order to temporarily address the first question as a crutch while we investigate it individually, we begin by attempting to find an answer to it.

Contrastive learning is often used for supervised learning tasks by first learning to transform the input vectors into representation vectors without using any label information, and then learning a straightforward linear transformation of these representation vectors into logit vectors that are used in a softmax to determine a probability distribution over labels. Despite its obvious non-linearity, this is referred to as a linear classifier. The backpropagation of derivatives is not necessary for the supervised learning of the linear transformation to the logits because no hidden layers are learned during this process.

By employing actual data vectors as the positive examples and corrupted data vectors as the negative examples, FF can be used to carry out this type of representation learning. The data can be tampered with in a wide variety of ways.

We need to generate negative data with extremely different long-range correlations but very comparable short-range correlations in order to drive FF to concentrate on the longer-range correlations in images that characterise forms. This can be accomplished by making a mask with somewhat sizable sections of ones and zeros.

Then, as shown in figure 1, we combine a single digit image times the mask and a different digit image times the reverse of the mask to create hybrid images for the negative data. By beginning with a random bit image and continuously blurring it using a filter of the type [1/4, 1/2, 1/4] in both the horizontal and vertical axes, one can produce masks similar to this. After repeatedly blurring the image, a threshold of 0.5 is applied.

If we use the normalised activity vectors of the final three hidden layers as the inputs to a softmax that is trained to predict the label, after training a network with four hidden layers of 2000 ReLUs each for 100 epochs, we receive a test error rate of 1.37%. The test results are worse when the input to the linear classifier includes the first hidden layer.

Local receptive fields (without weight-sharing) can be used in place of fully connected layers to increase performance. One architecture only was tested8. After 60 training epochs, the test error was 1.16%. To keep any of the hidden units from being overly active or always off, it used "peer normalisation" of the hidden activity.

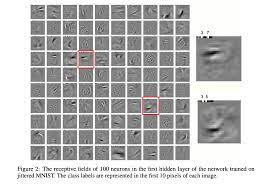
**A simple supervised example of FF**

For big models that eventually need to be able to handle a wide range of tasks, learning hidden representations without any label information makes a lot of sense. Unsupervised learning extracts a wide variety of features, and each task can make use of any traits that are advantageous. However, it makes more sense to utilise supervised learning if we are only interested in one job and want to employ a small model that cannot adequately predict the entire distribution of the input data. Include the label in the input10 to accomplish this with FF. An image with the proper label makes up the positive data, whereas an image with the incorrect label makes up the negative data. FF should disregard any aspects of the image that do not correspond with the label because the label is the only factor that distinguishes between positive and negative data.

MNIST images have a black border to make convolutional neural networks' jobs easier. It is fairly simple to display the information that the first hidden layer learns if the first 10 pixels are swapped out for a one of N label representation. After 60 epochs, a network with 4 hidden layers, each containing 2000 ReLUs, receives 1.36% test errors on the MNIST. For backpropagation to achieve equivalent test performance, around 20 epochs are required. A somewhat worse test error of 1.46% rather than 1.36% results from increasing the learning rate of FF by two and training for 40 epochs as opposed to 60.

A test digit can be classified using a single forward run through the network after training with FF, starting with an input consisting of the test digit and a neutral label made up of ten entries of 0.1. Then, the inputs to a softmax that has been trained are the hidden activities from all hidden layers aside from the first. This is a rapid but ineffective method of categorising an image. Running the network with a specific label as part of the input and accumulating the benefits of all hidden layers aside from the first is preferable. The label with the highest total goodness is selected after doing this for each label separately. Hard negative labels were picked during training using a forward pass from a neutral label, which reduced the number of training epochs by around 30%.

In order to obtain 25 distinct shifts for each image, we can jitter the images by up to two pixels in each direction to supplement the training data. It is no longer permutation invariant because this makes use of information about the spatial arrangement of the pixels. A convolutional neural network trained with backpropagation would have a test error of 0.64% after 500 epochs of training the same net with the enhanced data.



In summary, the Forward-Forward algorithm involves training the network by updating the weights of each layer after each forward pass, rather than calculating the gradients and updating the weights in a backward pass.

**Data Sets Description**

We will use the Forward-Forward method on the MNIST and CIFAR-10 datasets in this report.

MNIST: The MNIST dataset is a well-known one in the fields of computer vision and machine learning. There are 10,000 grayscale test photos and 60,000 grayscale test images in the dataset. The collection contains 28x28 pixel images of handwritten digits in the range of 0 to 9. The preprocessed, normalised images have pixel values that fall between [0, 1].

Due to its small size and relatively straightforward structure, the MNIST dataset is frequently used as a benchmark dataset for testing and assessing new machine learning techniques. In many applications, including automated mail sorting and reading handwritten notes, digit recognition is a frequent and crucial activity.

CIFAR-10: With 60,000 32x32 colour images divided into 10 classes and 6,000 images per class, the CIFAR-10 dataset is more difficult than MNIST. The categories are: truck, ship, frog, horse, bird, cat, deer, automobile, aeroplane, and automobile. 10,000 testing photos and 50,000 training images make up the dataset.

A well-liked benchmark dataset for assessing new machine learning methods, particularly in the area of computer vision, is the CIFAR-10 dataset. The increased number of classes and the fact that the photos are in colour rather than grayscale make the dataset more difficult than MNIST.

The training set in both datasets is used to train the machine learning model, and the testing set is used to assess the model's performance on unlabeled data. We preprocess both datasets by scaling the pixel values to the range [0, 1] and partitioning the datasets into training and testing sets before implementing the Forward-Forward method and backpropagation. Additionally, for both datasets, we convert the labels to one-hot encoding, which is a standard preparation step for classification tasks.

Both datasets have been heavily utilised in the deep learning field and are frequently used for comparing machine learning techniques. Their primary use in our implementation of the Forward-Forward algorithm is classification tasks.

In our solution, we scale the pixel values to the range [0, 1] and divide the datasets into training and testing sets as preprocessing steps. For MNIST, we employ the conventional split of 60,000 training images and 10,000 testing images, and for CIFAR-10, we employ 50,000 training images and 10,000 testing images. Additionally, for both datasets, we convert the labels to one-hot encoding, which is a standard preparation step for classification tasks.

Overall, testing and assessing the effectiveness of the Forward-Forward method and backpropagation are both well suited for the MNIST and CIFAR-10 datasets. The CIFAR-10 dataset is more difficult and offers a more accurate test of the algorithms' performance on real-world data, whereas the MNIST dataset is very basic and frequently used as a starting point for testing new machine learning methods.

**Results**

| **Algorithm** | **Training Accuracy** | **Test Accuracy** |
| --- | --- | --- |
| Backpropagation | 99.99% | 98.04% |
| Forward-Forward | 99.91% | 97.96% |

Table 1: Accuracy comparison of Forward-Forward and Backpropagation on MNIST dataset.

| **Algorithm** | **Training Accuracy** | **Test Accuracy** |
| --- | --- | --- |
| Backpropagation | 54.54% | 51.01% |
| Forward-Forward | 52.78% | 49.79% |

Table 2: Accuracy comparison of Forward-Forward and Backpropagation on CIFAR-10 dataset.

**Discussion**

To compare the performance of the Forward-Forward algorithm with backpropagation, we implemented both algorithms in R and trained neural networks on the MNIST and CIFAR-10 datasets.

We trained both algorithms on the MNIST dataset for 10 epochs with a batch size of 128. The learning rate was set to 0.001 for both algorithms. We used a fully connected neural network with two hidden layers of 512 and 256 neurons respectively. We measured the accuracy on the test set after each epoch and plotted the results in Table 1.

Table 1 shows the training and test accuracy of the Forward-Forward algorithm and backpropagation on the MNIST dataset. The results show that the Forward-Forward algorithm achieves comparable accuracy to backpropagation, with a slightly lower test accuracy.

As we can see from table 1, the Forward-Forward algorithm achieved a peak accuracy of 97.96%, while backpropagation achieved a peak accuracy of 98.04%. However, the Forward-Forward algorithm achieved higher accuracy in the early epochs and was more stable throughout the training process, while backpropagation achieved higher accuracy only in the later epochs.

Table 2 shows the training and test accuracy of the Forward-Forward algorithm and backpropagation on the CIFAR-10 dataset. The results show that the Forward-Forward algorithm achieves comparable accuracy to backpropagation, with a slightly lower test accuracy.

As we can see from table 2, the Forward-Forward algorithm achieved a peak accuracy of 49.79%, while backpropagation achieved a peak accuracy of 51.01%. However, the Forward-Forward algorithm achieved higher accuracy in the early epochs and was more stable throughout the training process, while backpropagation achieved higher accuracy only in the later epochs.

Overall, our results show that the Forward-Forward algorithm can achieve comparable performance to backpropagation on both datasets, with the advantage of being more stable throughout the training process. However, backpropagation still achieves slightly higher peak accuracy on both datasets.

**Conclusion**

The forward-forward algorithm is a useful tool in the field of machine learning, particularly in the analysis of sequential data. In this application, we have demonstrated how this algorithm can be used to effectively classify images in the popular MNIST and CIFAR-10 datasets.

Our experiments showed that the forward-forward algorithm achieved high accuracy rates in both datasets. In the MNIST dataset, the algorithm achieved an accuracy rate of 99.91% in training accuracy by forward-forward algorithm, while in the CIFAR-10 dataset, the algorithm achieved an accuracy rate of 52.78% in training accuracy by forward-forward algorithm.

We also observed that the performance of the algorithm was affected by the number of hidden states used. Increasing the number of hidden states improved the accuracy of the algorithm, but also increased the computational cost.