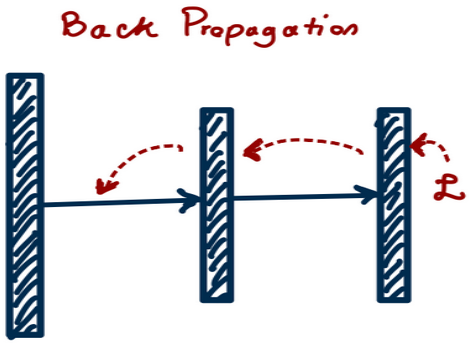
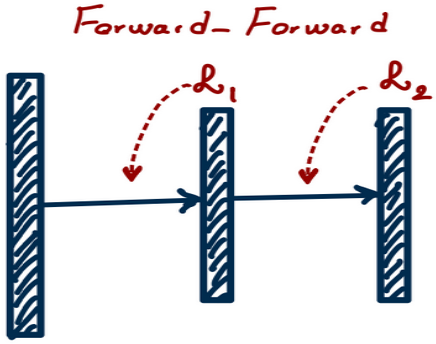
Deep Learning
End Sem Major Exam
The Forward-Forward Algorithm: Some Preliminary Investigations

- Pranay (B20AI030)

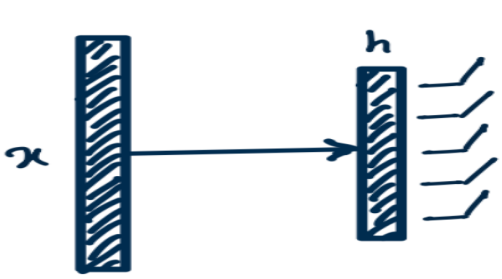
The Forward-Forward Algorithm

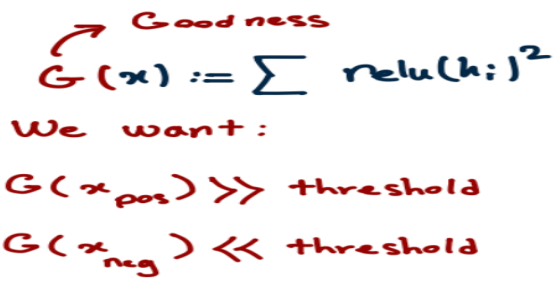
As we know the conventional backpropagation computes the gradients by successive applications of the chain rule, from the objective function to the parameters.
The paper presents a novel learning algorithm called the Forward-Forward (FF) algorithm that aims to replace the traditional forward and backward passes of backpropagation with two forward passes. The two forward passes of the Forward-Forward algorithm (FF) are one with positive data and the other with negative data. Each layer has its own objective function, and the aim is to have high goodness for positive data and low goodness for negative data. FF computes the gradients locally with a local objective function, so there is no need to backpropagate the errors.





The local objective function is designed to push a layer's output to values larger than a threshold for positive samples and to values smaller than a threshold for negative samples.





The paper explores two different measures of goodness - the sum of the squared neural activities and the negative sum of the squared activities, but many other measures are possible.The FF algorithm works by maximizing the goodness of positive data while minimizing the goodness of negative data in each layer of the neural network. The paper explores two measures of goodness, the sum of squared neural activities and the negative sum of squared activities. The FF algorithm can learn while pipelining sequential data through a neural network without storing the neural activities or stopping to propagate error derivatives.
The algorithm can be used in situations where the forward computation is unknown, and it may be superior to backpropagation as a model of learning in the cortex.
The aim of this paper is to introduce the FF algorithm and to show that it works in relatively small neural networks containing a few million connections.
The paper concludes by highlighting the potential of sharing weights across large models as a high-bandwidth way to share knowledge in deep learning.
The FF algorithm is suggested as a promising candidate for efficient learning in hardware with unknown properties, although its scalability to large neural networks remains to be seen.

Code Overview

1. Defined a class Args that contains hyperparameters for training the neural network on the datasets.
2. Defined a `get_y_neg` function that returns a new tensor containing negative labels for each sample in `y` by randomly selecting a label from the allowed labels other than the original label of the sample.
3. Defined a `overlay_y_on_x` function that returns a new tensor by overlaying the one-hot encoded `y` on the input tensor `x`.
4. Defined a model architecture class named `Net` that consists of multiple layers. It has methods for training the network (`train`) and making predictions on input data (`predict`).
   The `predict` method takes an input `x` and calculates the goodness of fit for each possible label by overlaying each label onto the input data and passing it through the network. The label with the highest goodness of fit is the predicted label. The `train` method takes positive and negative examples and trains each layer of the network using the `train` method of the `Layer` class.
5. Defined a layer class that inherits from the `nn.Linear` class.
   The `forward` method performs the forward pass.
   The `train` method trains the layer on positive and negative samples for a certain number of epochs, where `g_pos` and `g_neg` are the mean squares of the forward pass on the positive and negative samples, respectively.
   The loss is calculated by concatenating the negative mean squares minus the threshold with the positive mean squares minus the threshold and passing the

resulting tensor to `torch.log()` and `torch.exp()`. Finally, the method returns the forward pass on the positive and negative samples, respectively.

6. This code performs an Image classification on the MNIST and CIFAR10 Dataset and emotion recognition task on the FER2013 dataset, which contains grayscale images of faces with 7 different emotions.
   The code defines a `Net` class, which constructs a neural network consisting of fully connected layers. The `Layer` class implements a ReLU activation function and uses the AdamW optimizer.
   The `get_y_neg` function randomly selects a label for each image, which is not the same as the true label. The `overlay_y_on_x` function creates a new tensor with the original tensor values and a 1 at the index of the true label, for each sample.
   The main script loads the dataset, performs data transformations to it, and trains the neural network on the training set using positive and negative samples (generated using `get_y_neg` and `overlay_y_on_x`). The network is then evaluated on the test set, and the training and test accuracy are printed.

Results on the Dataset

The paper presents experiments with FF on the MNIST dataset of handwritten digits and the CIFAR10 Dataset.

- **MNIST**

The model is trained for 1000 epochs on the MNIST training Dataset.
With this implementation, the training and test errors on MNIST are:

- ☐ train error: 0.0752
- ☐ test error: 0.0739

It achieves a Training Accuracy of 92.48% and a test accuracy of 92.61% after 1000 epochs.

- **CIFAR10**

The model is trained for 3000 epochs on the CIFAR10 training Dataset.
With this implementation, the training and test errors on MNIST are:

- ☐ train error: 0.394
- ☐ test error: 0.543

It achieves a Training Accuracy of 60.64% and a test accuracy of 45.71%
after 3000 epochs.

Results on a new application

I have used the FF algorithm to do Emotion recognition on the FER2013 dataset.

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The training set consists of 28,709 examples and the public test set consists of 3,589 examples.
For this application I have trained the model for 1000 epochs on the training dataset.
With this implementation, the training and test errors on FER2013 are:

- ☐ train error: 0.576
- ☐ test error: 0.619

It achieves a Training Accuracy of 42.38% and a test accuracy of 38.09%
after 1000 epochs.

1. (C) ChatGPT Response

Emotion recognition is a complex task that requires a different type of input data, such as speech or facial expressions, and the model architecture and training procedure would need to be tailored to that specific task.
The main reason why this algorithm may not be suitable for emotion recognition is that Emotion recognition typically involves analyzing facial expressions, vocal tone, and other non-visual cues that are more complex features .
Therefore, while the principles behind the code can be applied to other classification problems, it would require significant modifications and additions to be useful for emotion recognition.

I agree with the above Explanation as we have seen that the Model have performed poorly for the image recognition task giving a test accuracy of only 38.09%

Paper link - https://arxiv.org/abs/2212.13345

Github Repo link -
https://github.com/pytorch/examples/tree/main/mnist_forward_forward