Adversarial Training Effects on Forward Forward Network

Shashvat Shah
Tandon School of Engineering,
New York University
sss9772@nyu.edu

Meet Nirav Diwan
Tandon School of Engineering,
New York University
md5517@nyu.edu

Abstract

Deep learning algorithms have been used to construct a model that performs tasks like classification, regression, and now image, speech, and text generation. Such models are very accurate in making predictions. This project aims to implement and evaluate one of such algorithms the Forward-Forward (FF) algorithm proposed by G. Hinton [1], a novel learning procedure for neural networks. The FF algorithm replaces the forward and backward passes of back-propagation with two forward passes using positive and negative data, offering potential simplification in learning and video processing without the need for activity storage or derivative propagation. The focus of this project is on the simple feed-forward supervised method of the FF algorithm, implemented using PyTorch. These algorithms are very crucial for safety-critical applications. Increasing adversarial attack challenges the safety of using such algorithms. In this paper, we have tested vulnerabilities of the proposed neural network with an adversarial attack and how adversarial training affects the performance of the network. This research aims to provide valuable insights into the performance and potential advantages of the FF algorithm. The code for this paper's implementation can be found in this repository https://github.com/shashvatshah9/FFNetworkAdversarial

1 Introduction

The paper by G.Hinton [1] presents a new learning procedure for neural networks called the Forward-Forward (FF) algorithm. Although backpropagation is nearly ubiquitous in modern machine learning and deep learning models, there is little evidence that the brain learns in such a manner [2]. In an attempt to more closely mimic biological brain function, the FF algorithm replaces the forward and backward passes of backpropagation with two forward passes, one with positive data and the other with negative data. The algorithm has shown promising results on small problems and has the potential to simplify learning and enable video processing without storing activities or stopping to propagate derivatives.

In their initial research, Hinton uses three main overarching methods to evaluate the FF algorithm with the MNIST dataset - a simple feed-forward unsupervised method, a simple feed-forward supervised method, and a multi-layer recurrent neural network. In all methods, they use a model with roughly four hidden layers each containing 2000 ReLUs, with slight variations to test the impact of different learning techniques. In the unsupervised method, they test with both fully connected layers and local receptive fields. In the supervised method, they test with different forms of classification - either feeding the network a "neutral label" composed of ten equal entries of 0.1 to represent the ten digits or feeding the network each of the ten labels in separate runs and choosing the label with the best-accumulated goodness. In the multi-layer recurrent method, the activity at any given layer is determined by the activity of the layers on either side.

In this project, we discuss the from-scratch implementation of the supervised version of this algorithm in PyTorch and perform a comparison with vanilla backpropagation by evaluating their performance on the MNIST dataset. And the main objective is to test the adversarial robustness of Forward-Forward networks.

Despite its remarkable successes, deep learning's rise has been shadowed by a disquieting vulnerability: the susceptibility to adversarial examples, and imperceptible modifications capable of inducing egregious errors in model predictions. Pioneered in response to the vulnerability of deep learning models to adversarial examples, adversarial training has emerged as a transformative paradigm for giving robustness to models. This iterative process establishes a dynamic, adversarial exchange between the model and a dedicated adversary. The model receives training not only on its original data but also on carefully crafted adversarial counterparts generated by the adversary to exploit its weaknesses. This continuous feedback loop ensures adaptation and evolution within the model, reinforcing its decision boundaries and fostering generalization beyond the limitations of its initial training data. Consequently, adversarially trained models exhibit enhanced resilience against a variety of perturbations and improved reliability in real-world settings, characterized by inherent uncertainty and potential adversarial manipulation.

Adversarial training can be computationally expensive and susceptible to overfitting on specific adversarial examples (Szegedy et al., 2017 [3]). Additionally, its effectiveness hinges on the choice of training parameters and the employed attack methods. Despite these challenges, the field of adversarial training is rapidly evolving. Researchers are developing novel techniques to improve efficiency and efficacy, focusing on areas such as:

- Early Adversarial Training (2014-2016): Seminal work of Goodfellow et al. (2014) [4], introducing Generative Adversarial Networks (GANs). This paradigm pitted a generator against a discriminator, iteratively refining both to achieve realistic image generation and improved adversarial detection. Madry et al. (2017) [5] subsequently applied this concept to directly train classifiers against crafted adversarial examples, demonstrating significant robustness gains. But, these early methods relied on simple gradient-based attacks, limiting their generalizability and robustness against more sophisticated adversaries. Additionally, computational cost and overfitting to specific attacks proved to be challenges.
- Advanced Adversarial Training (2017-2020): Diversifying Adversaries: To overcome limitations, researchers developed more diverse and effective attack methods. Kurakin et al. (2017) [6] explored adversarial attacks in the physical world, demonstrating their real-world impact.

  - Robust Optimization: Alongside evolving adversaries, robust optimization techniques emerged. Papernot et al. (2016) [7] proposed Jacobian DeepFool, a white-box attack that utilized model gradients to generate adversarial examples.
- The Current Landscape (2020-Present):

  - Adaptive and Dynamic Methods: Tramèr et al. (2019) [8] introduced adversarial distillation, transferring robustness knowledge from adversarially-trained teachers to student models.

  - Beyond Gradient-Based Attacks: The focus has shifted to more diverse and nuanced adversaries. Athalye et al. (2018) [9] presented oblivious black-box attacks, targeting models without access to their internal structure.

  - Towards Unified Frameworks: Fawzi et al. (2019 [10]) introduced adversarial training with meta-learning, adapting defenses to unseen attack methods.

2 Related work

The paper by (Szegedy et al., 2014) "Intriguing Properties of Neural Networks" [11] provides an initial review to explore and understand the vulnerabilities of deep neural networks to small, carefully crafted perturbations in input data. The authors introduced the term "adversarial examples," which are input instances that are slightly modified from correctly classified instances but can lead to misclassification by the neural network. These perturbations are often imperceptible to human observers. The authors proposed a method called the L-BFGS algorithm to find small perturbations that, when added to an input image, result in misclassification. On ImageNet dataset [12], the adversarial examples were indistinguishable to human eyes. The author also found out that the same adversarial example is often misclassified when trained with a different subset of data.

The paper "Explaining and harnessing adversarial examples" by (Goodfellow et al. 2015) [4] is another influential work in the field of adversarial machine learning. The paper introduces the Fast Gradient Sign Method (FGSM), a computationally efficient algorithm for generating adversarial examples. The key idea is to use the gradient of the loss function with respect to the input to iteratively update the input data to maximize the classification error. It discusses the sensitivity of neural networks to small changes in input space, leading to the creation of adversarial examples. This paper also shows a way of Adversarial retraining where models are trained on a mix of clean and adversarial examples, which can improve the robustness of the model against adversarial attacks. We have also tested the robustness of the Forward-Forward algorithm using the Fast Gradient Sign Method (FGSM) proposed by Goodfellow et al.

The paper titled "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images" by Anh Nguyen, Jason Yosinski, and Jeff Clune (2015) [13] investigates the vulnerability of deep neural networks to adversarial examples and explores the phenomenon where networks confidently classify images that are unrecognizable to humans. These images, while nonsensical and unrecognizable to humans, were confidently classified by the neural network. This paper demonstrated that deep neural networks can confidently classify images that are visually unrecognizable and semantically meaningless to humans.

The paper titled "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks" by Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami (2016) [7] investigates the use of knowledge distillation as a defense mechanism against adversarial attacks on deep neural networks. This paper explores the use of knowledge distillation, a training technique that transfers knowledge from a complex model (teacher) to a simpler model (student), as a defense strategy against adversarial attacks. The authors applied the concept of knowledge distillation to train a neural network to mimic the behavior of a larger, more accurate model (teacher). The student model was then evaluated for its resistance to adversarial perturbations. The key finding was that models trained using distillation are more robust to adversarial perturbations than those trained using standard supervised learning.

3 Problem statement and assumptions

The problem statement for this study is to train the Forward-Forward model using the MNIST dataset to achieve competitive results in terms of accuracy and loss compared to other models addressing the same task. The focus is on establishing the model's proficiency in image classification through rigorous training and validation processes.

Upon achieving a high level of accuracy on the test set, the next phase involves subjecting the trained model to adversarial attacks. This critical step aims to assess the model's vulnerability to carefully crafted perturbations in input data. The observations and insights gained from the adversarial attack experiments will be thoroughly discussed in the latter part of this review paper.

To test the robustness of the Forward-Forward model against adversarial attacks, an adversarial retraining approach will be employed. This involves iteratively training the model on both clean and adversarially perturbed data, to enhance its resilience to potential adversarial inputs.

The outcomes of these experiments will contribute to the broader understanding of the model's security and robustness in the face of adversarial threats, providing valuable insights for future advancements in adversarial machine learning.

4 Algorithm review

4.1 Forward Forward Network Training

During initial testing, our implementation consisted of two layers with dimensions $784 \rightarrow 500 \rightarrow 500$, and trained each layer on all MNIST training images at once. In an effort to more closely match the methods used in backpropagation, we restructured the code to accept multiple batches and train the entire model batch by batch before completing one epoch.

One unique aspect of the FF algorithm is the ability to train each layer individually. Because the FF algorithm uses the output goodness of each layer to update the weights of that layer only, we are

able to run the entire set of training data through one layer at a time, either in batches or one large set. This allows for some unconventional training techniques - namely, we can load one layer of the network at a time onto cuda, significantly reducing the GPU RAM footprint. By combining this with batching, we were able to reduce GPU memory usage from 13.1 GB to only 2.0 GB during training. This also accommodates larger networks on hardware with less memory, with the obvious trade-off of training speed, as loading data on and off of cuda has a significant time cost.

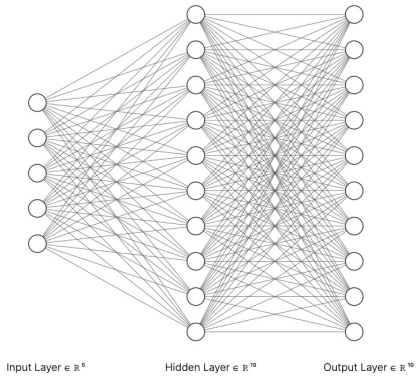

Figure 1: Subsection of Network architecture of Forward-Forward Network

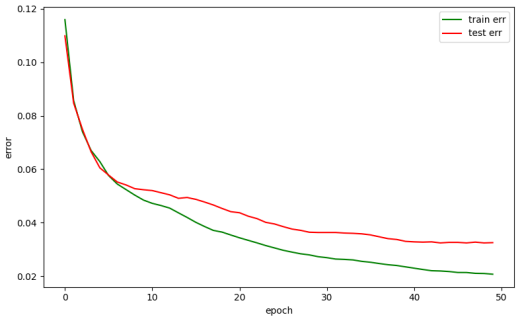

Figure 2: FF training on MNIST

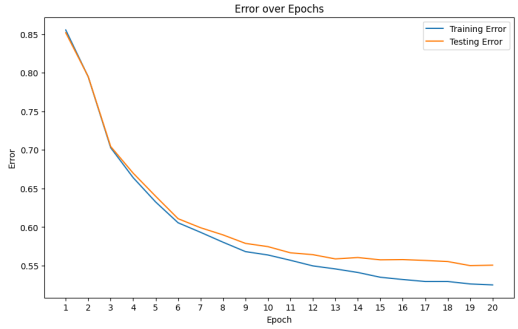

Figure 3: FF training on CIFAR10

Training in this manner requires a few code changes that deviate from regular backpropagation methods using PyTorch. First, the training data tensors must be "detached" from their computational graphs after each layer to prevent backpropagation to previous layers. We accomplish this by calling 'x.detach()', where x is the training data, after each layer is trained. In doing so, calling 'loss.backward()' does not perform true backpropagation because it is only updating the weights of the previous layer.

Another advantage of the FF method is that the output layer does not have to match the number of possible classifications, but can instead have any size. This difference can be seen in Figure 1, where the first displays a simple, downsized representation of the FF network with a large output layer, and the second displays a simple, downsized representation of a similar network using backpropagations with an output layer of size 3 for 3 classes. Following the example of MNIST data, we can use the model to determine the likelihood that an image belongs to one of the ten classes by running the input image through the network ten times, each time with a different classification encoded in the image using the previously described encoding method. For each label, we accumulate the average goodness over every layer, and choose the label with the highest accumulated goodness. However, this requires ten passes through the network for inference on every input image, which significantly slows the evaluation throughput. The training result is shown in Figure 2.

To reproduce similar error or accuracy for CIFAR-10 dataset, we first reused the architecture similar to the training architecture proposed by [1]. The model consisted of 2 layers with dimensions $3072 \rightarrow 500 \rightarrow 500$. The model didn't converge to the expected accuracy levels. So then we used another model with dimensions $3072 \rightarrow 1000 \rightarrow 1000, \rightarrow 500$. This model again converged to a modest error value of 0.55 on the test dataset. The train and test error for this model have been plotted in Figure 3.

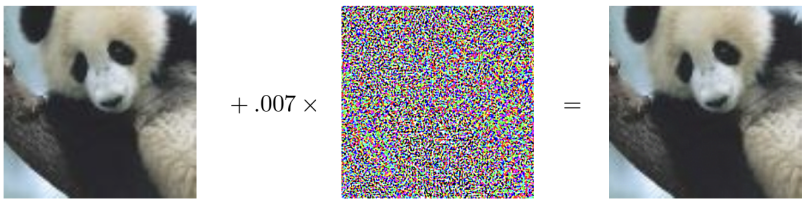

Figure 4: A demonstration of fast adversarial example generation applied to GoogLeNet [14] on ImageNet.

4.2 Linear Perturbation Of Non-linear Models

The algorithm proposed by (Goodfellow et al. 2015) [4] states that neural networks are too linear to resist linear adversarial perturbation. LSTMs, ReLUs, and maxout networks are all intentionally designed to behave in very linear ways so that they are easier to optimize. Let θ be the parameters of a model, x the input to the model, y the targets associated with x, and $J(\theta, x, y)$ be the cost used to train the neural network. The authors go on to state that we can linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained perturbation of

$$\eta = \varepsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{1}$$

Which is referred to as the "Fast Gradient Sign Method" (FGSM) for generating adversarial examples.

Algorithm 1 FGSM Attack on Forward Forward Network

Require: model, device, data_loader, epsilon
Ensure: benign_acc

```
data_iter ← []
for data, target in data_loader do
    encoded_data ← prepare_data(data, target)
    _, output ← eval_loop()
    correct_benign ← correct_benign + (output.argmax(1) == target).sum()
    loss ← CrossEntropyLoss(output, target)
    model.zero_grad()
    loss.backward()
    data_grad ← data.grad.data
    data_denorm ← denorm(data)
    perturbed_data ← fgsm_attack(data_denorm, epsilon, data_grad)
    perturbed_data_normalized ← Normalize(perturbed_data)
    encoded_data_perturbed ← prepare_data(perturbed_data_normalized., target)
    _, output ← eval_loop()
    total ← total + target.size(0)
end for
benign_acc ← correct_benign / total
return benign_acc
```

5 Experimental analysis

To ensure a fair and meaningful comparison, the training of both the ResNet and Forward-Forward networks was halted upon reaching a test accuracy of 89%. This deliberate decision was essential to establish an equal training baseline for both models before subjecting them to an adversarial attack. By aligning the models at a common accuracy threshold, we aimed to eliminate potential disparities arising from differences in training convergence rates. This meticulous approach ensures that any

observed effects during the subsequent adversarial attack phase can be attributed to inherent model characteristics rather than variations in training progress.

Table 1: Table presents a comprehensive comparison of the test accuracy for both the ResNet and Forward-Forward networks across various epsilon values. Initial observations indicate that, for lower epsilon values, the ResNet model exhibits superior performance compared to the Forward-Forward model. However, as epsilon values increase, an interesting trend emerges. The ResNet model experiences a degradation in performance, while the Forward-Forward model not only maintains its accuracy but also surpasses the ResNet model.

	Test accuracy	
Epsilon	ResNet Model	Forward Forward Network
0.20	0.3373	0.2012
0.25	0.1330	0.1812
0.30	0.0396	0.1754
0.35	0.0133	0.1696
0.40	0.0063	0.1615
0.45	0.0049	0.1486
0.50	0.0047	0.1347
0.60	0.0056	0.1193
0.70	0.0076	0.1115
0.80	0.0125	0.1085
0.90	0.0183	0.1053

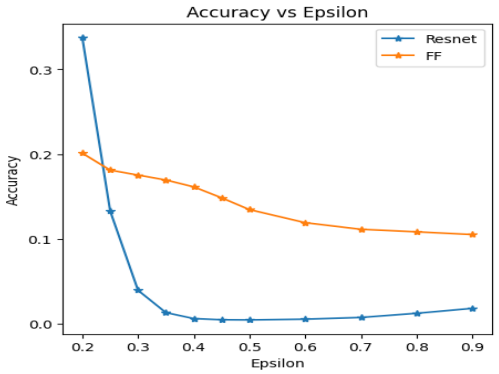

Figure 5: Test Accuracy drop comparison between ResNet and Forward Forward at for different epsilon values

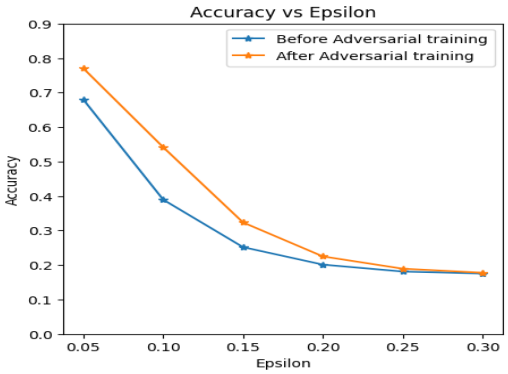

Figure 6: Comparing the adversarial training effect on Forward Forward Network by comparing the before and after test accuracy while attacking the model at different epsilon values

As epsilon values continue to rise, the Forward-Forward model consistently outperforms the ResNet model, showcasing its robustness in the face of adversarial perturbations. It is noteworthy that, while there is a marginal improvement in the ResNet model's performance at higher epsilon values, the significance of this improvement remains considerably lower compared to the Forward-Forward model. These findings underscore the effectiveness of the Forward-Forward model in maintaining resilience against adversarial attacks across a range of perturbation magnitudes.

Table 2: Adversarial Training evaluation

	Test accuracy	
Epsilon	Before Adversarial Training	After Adversarial Training
0.05	0.6796	0.7701
0.10	0.3895	0.5415
0.15	0.2519	0.3235
0.20	0.2012	0.2250
0.25	0.1812	0.1894
0.30	0.1754	0.1777

We conducted Adversarial training on the Forward-Forward model to enhance its robustness against adversarial attacks. Adversarial samples were deliberately introduced during the training phase. The model was trained with both positive and negative encoding of the data. Notably, the Negative encoding was generated with a similar intention as that of an adversarial example. Despite this unconventional methodology, we proceeded with an extended number of training epochs, following a standard approach to effectively address the task.

Upon observation, it is evident that the Adversarial training yielded promising results as can be seen in Figures 5 and 6. The Forward-Forward model demonstrated a slight improvement in performance for low epsilon values, as reflected by a marginal increase in test accuracy. Conversely, for higher epsilon values, the test accuracy remained consistent with the results obtained before the adversarial training, aligning with the anticipated behavior of this model, which is depicted in Table 2.

It is crucial to emphasize that the test accuracy comparison pertains specifically to adversarial attacks before and after adversarial training, rather than the general test accuracy of the model in a non-adversarial scenario. This nuanced evaluation provides insights into the model's adaptability and resilience under adversarial conditions.

6 Conclusion

While Generative Adversarial Networks (GANs) excel at generating visually appealing images, they struggle with mode collapse, where large regions of the image space remain unexplored. Additionally, their reliance on backpropagation for both the generative and discriminative models raises compatibility concerns with biological neurons.

The Forward-Forward (FF) algorithm offers a compelling alternative. It can be seen as a special type of GAN where each hidden layer of the discriminator independently classifies the input as positive or negative, negating the need for backpropagation in learning the generative model. This model simply reuses the representations learned by the discriminator and needs only a linear transformation to generate outputs, again avoiding backpropagation[1].

While initially falling behind the ResNet model at low perturbation levels, the FF displays remarkable resilience as epsilon values climb. Its consistent outperformance at higher levels underscores its potential as a robust defense against increasingly severe adversarial threats.

Furthermore, adversarial training demonstrates its ability to further enhance the FF's resilience. While subtle improvements occur at low epsilon values, its consistent performance at higher levels reinforces its innate aptitude for handling strong perturbations. This aligns with the model's inherent characteristics, as observed in the non-adversarial training experiments.

It's crucial to remember that our analyses focused specifically on adversarial scenarios, not general test accuracy. This targeted evaluation provides valuable insights into the FF's adaptability and resilience under attack, highlighting its potential as a robust and promising alternative for tackling the growing challenge of adversarial threats.

References

[1] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[2] Z. Xu Y. Song, T. Lukasiewicz and R. Bogacz. Can the brain do backpropagation? —exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 2020.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[6] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[7] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2016.

[8] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations, 2019.

[9] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, 2018.

[10] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, Feb 2014.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[13] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.