A Further Investigation of the Forward-Forward Algorithm

Avishan Bewtra
Penn State University
State College, PA
aqbewtra@gmail.com

***Abstract*—The aim of this paper is to further investigate the Forward-Forward Algorithm (FF) and its effects on machine learning literature. FF was proposed by Geoffrey Hinton as an alternative to backpropagation. Neuroscientists confirm that the brain does not backpropagate according to a finite set of loss functions. As a result, in the search for general artificial intelligence, we need to begin research that challenges backpropagation. The main problem with backpropagation are it's two major limitations: 1) It requires large timeouts in between predictions to make weight updates; and 2) It requires perfect knowledge of the network architecture to calculate partial derivatives and the gradient. FF addresses these two limitations. But still, we conclude it is not yet a viable substitute for backpropagation. In this paper, we provide the following contributions: 1) A Survey of FF's Impact on Subsequent Literature; 2) A more comprehensive comparison between backpropagation and FF, including classification problems like MNIST, CIFRAR-10, and CIFAR-100; and 3) A discussion of FF's strengths and weaknesses, it's addressal of backpropagation's limitations, and it's best applications in the future. We also provide some areas for future work related to FF.**

I. Introduction

Earlier this year, 2023, Geoffrey Hinton released a preprint of some preliminary investigations of the Forward-Forward Algorithm. [1]

The human brain is a complex neural network, each with unique connections between neurons. Neurons fire to other connected neurons, and en masse, the billions of neurons in a brain can do a huge variety of things: hold representations of thoughts and our world; control the body; maintain homeostasis—among many other functions outside the scope of this survey. Machine learning techniques have been modeled loosely based on this idea of a biological neural network as a model for mathematical functions. Still, they have been designed to be more basic than the brain, as most current machine learning tasks are much simpler and more confined problems than that of the human condition. As a result, deep neural networks, often used for computer visions tasks, have a similar—although more narrow—function compared to that of the human brain. This field of machine learning that utilizes deep neural networks is called deep learning, or representation learning. However, from neuroscience, we know that the brain does not optimize the same way as most deep neural networks.

The most popular and successful optimization techniques for deep neural networks have utilized some form of gradient descent. Consider a neural network, F, with weights, θ. Given a dataset of tuples, $(x_i, y_i) \in D$: the goal is to select θ such that $F_\theta(x_i)$ is (on average) the best model of y_i for all i. In other words, we want to select θ to minimize the average measured error between $F_\theta(x_i)$ and y_i. Given some cost function ℓ, we might consider the following minimization problem:

$$\theta^* := \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=0}^{n} \ell(F_\theta(x_i), y_i)$$

However, when F_θ is a deep neural network, there is no closed form solution for θ^*, so gradient descent techniques are used to make incremental updates to θ based on the gradient of the cost function, $\nabla_\theta \ell(F_\theta(x_i), y_i)$. The success of these gradient descent techniques have been enabled by the backpropagation algorithm, popularized by Hinton et al. in 1986. Backpropogation, as described by Hinton, is the backward pass of through the neural network, using the chain rule to calculate partial derivatives of a layer, for example l_2, with respect to the layer that precedes it, for example l_1. Starting at the cost function, ℓ, these partial derivatives can be used to calculate the total gradient of ℓ with respect to θ: $\nabla_\theta \ell$.

However, neuroscientists understand the brain does not backpropagate, at least not in the way machine learning models do [2]. The brain does not have a single cost function like ℓ, from where the entirety of the brains weights can be updated based on the gradient of that single cost function. While comprehensive, backpropagation is difficult to scale to larger, more general neural networks. It is hypothesized that the brain, instead, makes sets of local updates based on lots of different proverbial cost functions, avoiding the ineffability of backpropagation for general intelligence in humans.

FF was not only designed to optimize with intuition more like the brain, but it aims to address these key limitations of backpropagation:

- Requires large timeouts in between predictions to update weights.
- Requires perfect knowledge of the architecture to calculate partial derivatives.

FF was not designed to beat backpropagation on benchmark tasks; rather, it was intended to open a field of study to new optimization techniques. This sentiment is critical. In this investigation, we'll explore FF's applications and compare

its performance to backpropagation. We'll also propose fields of research to further investigate the intuition proposed by Hinton.

Instead of calculating all partial derivatives based on one error function, optimize the neural activity of a single layer based on the layer's own cost function. Given sets of positive data, P, and negative data, N, FF optimizes an individual layer's weights to maximize neural activity for positive data and minimize neural activity for negative data, irrespective of other layers. In FF, given a data point x_i, we optimize a network such that the neural activity—or the total measured output of the network for a given x_i, y_i—is analogous to the probability $x_i \in P$. As notated by Hinton, this is represented by the function of, as we'll call it, the measure of neural activity:

$$p(x_i \in P) = \sigma\Big(\sum_j y_j^2 - thresh\Big)$$

The curation of these positive and negative datasets, P and N, will be explored in this investigation, as it holds critical potential to improve the results of FF.

In summary, we aim to make the following contributions:

A. A Survey of FF's Impact on Subsequent Literature
B. A comparison of FF (and variations) with backpropagation across multiple benchmark datasets.
C. Further investigation of the FF intuition: Why does FF work? Where is it's value? What followup questions beg to be asked?

It is important to note that, for the sake of simplicity, this paper is limited to fully-connected feed-forward neural neural networks. However, as we highlight in *Future Work*, it is critical to do more testing on networks with recurrences and convolutions in the future.

This paper is organized with a literature review of techniques inspiring the FF algorithm; followed by our contributions; and finally our discussion of conclusions and future work.

II. Literature Review

A. The Forward Forward Algorithm

The Forward-Forward algorithm is a follow-up to the backpropagation techniques most popularly used to optimize deep neural networks. Backpropagation is used to calculate the partial derivatives of each weight in the network with respect to the loss function. Using the chain rule, the backpropagation graph is created, then used to calculate partial derivatives of a layers weights based on those that came before it in the network. However, this is slow, and prevents training in real time. FF attempts to remedy these issues by only training one layer at a time, replacing the traditional forward and backward pass with two forward passes.

Consider a dataset $(x_i, y_i) \in D$, where x_i is a datapoint and y_i its corresponding label. Hinton describes two sub-datasets, a positive dataset, P, and a negative dataset, N. P is full of real data; in other words P is a copy of D: $P := D$.

N's negative datapoints are considered to be relevant, nonsense features. The purpose of these negative features are to teach a network which features are irrelevant, as the network will be trained to have minimal neural activity for these points in N. Hinton's initial proposal is to create hybrid images from the original dataset. See Figure 1 for the example from Hinton's paper:

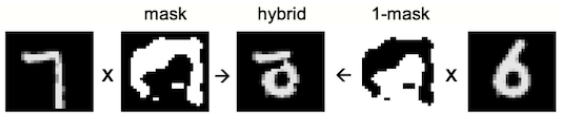

Fig. 1. A hybrid image as an example of negative data.

To optimize, we want to maximize neural activity for $x_i \in P$ and minimize neural activity for $x_i \in N$.

Each layer is trained to completion in sequence. Consider a neural network from its first layer through layer k, F_{θ_k}. For positive data, $x_i \in P$ this yields the optimization problem to solve for the weights of layer k, θ_k:

$$\max_{\theta_k} \sum_{x_{pos}} \sigma\Big(\sum_j F_{\theta_k}(x_{pos})_j^2 - thresh\Big) - \sum_{x_{neg}} \sigma\Big(\sum_j F_{\theta_k}(x_{neg})_j^2 - thresh\Big)$$

In optimizing this function, we are only concerned with inputs and outputs with respect to layer k. As a result, our proverbial backpropagation graph is only one layer deep; thus we don't need to propagate anything backwards at all. Avoiding a comprehensive calculation of partial derivatives has the potential to save significant time and resources in model training. Only one layer's derivates must be calculated; there is no need to propagate anything backwards beyond layer k.

Optimization will still proceed with stochastic gradient descent. However, we must only calculate the gradient of our optimization function with respect to θ_k—instead of the entirety of θ. In other words, while backpropagation weight updates look like this:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \nabla_\theta$$

Our update, instead, looks like this:

$$\theta_k^{(t+1)} \leftarrow \theta_k^{(t)} - \alpha \nabla_{\theta_k}$$

B. The Self-supervised and Contrastive Paradigm

SimCLR was a breakthrough paper in the increasingly practical field of self-supervised learning [4]. SimCLR aims to maximize agreement between images with similar features and maximize disagreement between those with different features. The goal is to train a well-performing feature extractor that can embed datapoints into feature-space, similar to the practice of transfer learning. This philosophy inspired Hinton's proposed FF, and was studied in background work for this research paper.

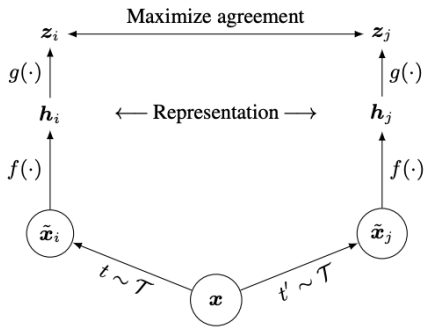

Fig. 2. Simple framework for contrastive learning of visual representations.

In Figure 2, the optimization problem lies in training a network to generate similar representations—h_i and h_j—for similar data points—$\tilde{x}_i$ and $\tilde{x}_j$.

The FF algorithm borrows the notion of maximizing agreement between similar datapoints from SimCLR; for positive data, maximizing activity, and for negative data, minimizing activity. However, by optimizing for both positive and negative data, Hinton's FF also maximizes disagreement between dissimilar datapoints.

C. Generative Adversarial Networks

GANs were another research publications that inspired the forward-forward algorithm [3]. Revolutionizing the field of generative AI, GANs proposed a pair of competing networks in which a discriminator, D, was pitted against a generator, G, where D was optimized to classify images as real or fake, and G was optimized to generate fake images that D would classify as real.

This yields the adversarial optimization problem:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

This discrimination concept is modeled in the way FF maximizes neural activity for positive data and minimizes it for negative data.

Due to the use of a sigmoid in the measure of neural activity for FF, it can be thought of as a classification of a datapoint as positive or negative. This notion in the FF algorithm can be thought of as discrimination in the context of GANs. While we are not training two neural networks, the optimization problem is similar to GANs in that one must balance a networks responses to positive and negative—or real and synthetic—data.

III. Implementation

All code for implementing FF and backpropagation was done in PyTorch, and code can be found on the author's GitHub.[1]

Notable contributions found in the linked GitHub repository include:

- An modified implementation of FF, with many major components borrowed from @mohammadpz's public implementation on GitHub.
- A function generating matching networks: one prepared for FF optimization and another for backpropagation. This was critical. The last layer of FF is architected to be different from the traditional classification model, so it was important to be able to generate identical networks with identical parameter counts for adequate comparison of results.
- A training loop where the results of backpropagation and FF can easily be compared for equivalent networks.

IV. Contributions

A. A Survey of FF's Impact on Subsequent Literature

FF, as an optimization technique alone, has more knobs to be tuned than backpropagation. Those especially unique to FF include, but are not limited to: the way negative data, N, is generated; and the way a threshold, $thresh$, is selected. We will include a survey of the most recent literature based upon Hinton's original proposal, including a discussion of the way these critical parameters affect performance, especially relative to backpropagation, as discussed in the following section.

It is worth noting that the field of FF is still nascent; Hinton's FF paper is only a few months old, so there hasn't been much time for the literature to develop. As a result, we felt it was necessary to provide a brief survey of the most significant literature released in the last four months. These three have made the most significant impact in the field since January, 2023, when Hinton's FF paper was released; they are representative of the impact FF has already made in academic research.

1) "The Cascaded Forward Algorithm for Neural Network Training": The Cascaded Forward (CaFo) Algorithm, proposed by Zhao et al., proposes another alternative to backpropagation using the FF's notion of local loss, instead of a singular global loss [5]. Instead of utilizing negative data, CaFo enforces that the output of each layer—at each hidden state, h_i, as they refer to them in the paper—h_i becomes closer and closer to the label y_i. In this way, local-layer weights are optimized based on the local loss, irrespective of any other previous manipulation to h_i. Therefore no backpropagation graph is generated. There is no backward pass; only, like FF, two forward passes are needed.

2) "Building Artificial Neural Circuits for Domain-general Cognition: [...]": In this publication, Achterberg et al. describe the way we can build neural networks based on the brain [6]. The authors emphasize the importance of local loss functions, citing Hinton's FF algorithm, and the combination with widespread recurrence loops in order to step closer to "flexible domain-general cognition in artificial neural networks." They cite these techniques as not only crucial to the creation of more flexible NN techniques, but to the understanding of cognition—both artificial and biological.

This is monumental for machine learning research. Neuroscience and studies of biological cognition have heavily

[1]Code found at: https://github.com/aqbewtra/pytorch_forward_forward

influenced machine learning research. Achterberg et al. now suggest that machine learning research can also help models for neuroscience and biological study. They continue to suggest as we develop systems that behave more closely to the brain—in the pursuit of artificial general intellgence (AGI)—challenges to traditional optimization techniques like FF are a significant portion of what might get us there.

3) "The Predictive Forward-Forward Algorithm": The Predictive Forward-Forward (PFF) Algorithm is a generalization of Hinton's FF algorithm [7]. Based on FF's forward-only philosophy, PFF, the authors, "design a novel, dynamic recurrent neural system that learns a directed generative circuit jointly and simultaneously with a representation circuit." It's most significant contribution is the introduction of representation reconstructions, which make the potential model of biological neurons more helpful by generating reconstructions of the original datapoint in order to validate meaningful learned representations.

B. Comparison: FF vs. Backpropagation

Here we provide a direct performance comparison with networks trained using backpropagation. We will compare classification accuracy and F1 score across the MNIST, CIFAR-10, CIFAR-100 datasets. We will pay special attention to test error and F1 score.

Note: In the charts below, the Models column contains the number of features at each hidden layer in the given network.

MNIST Test Metrics				
Models	**FF Acc**	**FF F1**	**BP Acc**	**BP F1**
500, 500	.9294	.9286	.9519	.9514
1000, 500	.9326	.9318	.9521	.9516
1500, 500	.9365	.9359	.9514	.9508
1000, 500, 100	.9235	.9562	.9566	.9562
1500, 500, 100	.9331	.9351	**.9567**	**.9563**

CIFAR-10 Test Metrics				
Models	**FF Acc**	**FF F1**	**BP Acc**	**BP F1**
3000, 500	.8504	.8403	.8916	.8891
3000, 1000	.8503	.8380	.8925	.8911
3000, 2000, 1000	.8462	.8298	.8918	.8884
5000, 1000, 500	.8581	.8527	.8950	.8931
5000, 2000, 1000	.8598	.8491	**.8959**	**.8954**

CIFAR-100 Test Metrics				
Models	**FF Acc**	**FF F1**	**BP Acc**	**BP F1**
3000, 500	.6801	.6567	.7229	.7002
3000, 1000	.6375	.6328	.7472	.7409
3000, 2000, 1000	.6946	.6763	.7481	.7374
5000, 1000, 500	.6924	.6583	.7493	.7241
5000, 2000, 1000	.7217	.7070	**.7633**	**.7412**

Backpropagation outperformed FF on all of the tested datasets: MNIST, CIFAR-10, and CIFAR-100. As illustrated in the charts above, across five selected fully-connected, backpropogation had higher accuracy and F1 metrics for all of the datasets.

This is significant in reinforcing our hypothesis: in a controlled environment, with training-timeout time available and complete knowledge of the neural network architecture, backpropogation is still far better than Hinton's vanilla FF algorithm.

This may be for multiple reasons. First, the cost function for backpropogation is more specific than FF's measure of neural activity. It is able to better capture label-specific error. This is also due to the fact backpropagation's gradient is more comprehensive, so all layers are updates to work together to reach the maximum-likelihood estimator. The notion of maximum-likelihood estimation backs up the intuition behind backpropagation and traditional stochastic gradient descent with the cross-entropy loss function. There is no such formulation for FF.

Simply, for simple classification problems, FF will not outperform backpropagation in it's current state. This reinforces our understanding of backpropagation's strengths: it's comprehensive optimization and statistical formulation. Still, this does not diminish the strengths of FF. In the next section we will discuss circumstances for selecting FF over backpropagation, and how FF addresses the weaknesses of backpropagation to best fit certain applications.

C. FF Discussion: Strengths, Weaknesses, and Applications

In ideal conditions, backpropagation with stochastic gradient descent converges to the maximum likelihood estimator. We will investigate, in a dialogue with relevant literature, the lack of a statistical formulation of the FF algorithm.

FF works. We have shown relatively successful performance metrics on multiple vision tasks. Still, the measure of neural activity selected by Hinton has no statistical formulation. It is largely derived through intuition. Optimizing the entire network based on the gradient of the Cross Entropy loss function, for example, is proven to approach the Maximum Likelihood Estimator. The MLE formulation suggests (but does not necessarily guarantee) some properties of consistency, efficiency, and error according to the normal distribution. There is no such formulation for the FF measure of neural activity. This may be one of its strongest limitations, and one of its brightest hopes as the research develops: that a more rigorously developed measure of neural activity may be developed.

However, while this investigation focused on it's weaknesses, FF has multiple strengths to be further explored—namely, an ability to work without complete knowledge of the NN graph; fast convergence times relative to backpropogation; and an ability to find success in real-time training, without large overhead amounts of time computing gradients before inference time.

Because FF relies on local loss, there is no need to know the whole graph of a NN. If there is a portion of the graph where the partial derivatives can't be calculated, backpropagation falls apart; without partial derivatives for parts of the NN, the

gradient cannot be calculated for any preceding weights, and those preceding weights can't be properly updated. This means that while backpropogation outperformed FF in very controlled environments, backpropagation fails in certain environments where FF, as explained by Hinton, succeeds.

FF also generates predictions for positive data in real time, so no large time-out is required for training. As a result, training can be done more practically in production.

These two strengths suggest FF may be a potential technique for on-board training for edge devices. We suggest more research should be conducted to reap potential value this area.

These intuitions are also critical for the development of artificial general intelligence (AGI).

Humans don't have to sleep in order to update neurons and "learn". And neuroscientists, such as in [2], know that humans also don't backpropagate with respect to every neuron. As a result, there must be a lot of local updates in the brain, with lots of local losses. We, humans, are multi-modal to the utmost degree; and yet, our brain is not perfectly compartmentalized into totally separate neural networks; there are lots of overlaps and conjunctions between different portions. If AGI research is to continue to be inspired by the human brain and biological cognition (which even this is not certain, and is widely debated; this topic would require another paper in itself), we should be challenging the default approach of using backpropagation to optimize NNs. With this in mind, Hinton's paper widens a narrow field of research that has the potential change the field of machine learning.

It is crucial that the reader understand the following: the value of the FF algorithm is not in its ability to replace backpropagation; rather, its key strength is in its ability to open other avenues where backpropagation fails, or where it is impractical, or intractable. Therefore, in summary—as we'll expand in the *Future Work* section—we emphasize the following as the best future applications of FF:

- When portions of an NN are unknown.
- When large-timeouts for training or re-training are impractical; or when training in real-time, in production, is valuable.
- As a foundation for future algorithms to utilize local loss functions in the pursuit of AGI.

V. Conclusions

In summary, we've discussed the following conclusions pertaining to FF, its impact in academic research, its performance relative to mainstream optimization techniques, and its most valuable applications:

- As discussed in [2], FF has already significantly affected research in the intersection of neuroscience and modelling with artificial neural networks, inspiring changes in philosophy as researchers pursue AGI.
- Backpropogation is consistently better than FF for traditional classification problems. Upon further investigation, backpropogation is almost perfectly tailored for learning in environments where: 1) ample resources (time, compute, etc.) are available for training, ahead of production; and 2) where complete knowledge of the NN is available. These resources, along with ample data, are among the most critical requirements for high performing machine learning models. Intuitively, as discussed, backpropagation is hard to beat.
- FF is best suited for applications in environments where having training timeouts are impractical and where there isn't complete knowledge of the NN architecture.
- For it's comparable results, FF may have applications in real-time learning and learning on edge (IoT) devices.
- A lack of specificity in the measure of neural activity function and a lack of a statistical formulation hinder FF. These are the two largest obstacles for FF in competition with backpropagation.

VI. Future Work

Based on our investigation, we suggest the following areas for future work in order to better understand and improve FF:

- Investigation of a statistical formulation of FF, perhaps in relation to Maximum Likelihood Estimation (MLE): Cost function is critical to a model's success, so it is critical to do more work in order to better measure the local performance of a layer. Selecting a cost function based on theoretical derivation, as opposed to intuition, may bear better performance for FF.
- Development of negative data generation: the creation of Hinton's negative dataset, N, is also critical to solid model performance. The technique described is based on intuition. A more rigorous investigation of negative data generation techniques will greatly affect performance. Perhaps using a model to generate negative samples, as in the GAN / Adversarial paradigm.
- Selection of threshold, $thresh$.
- Applications of FF in artificial general intelligence (AGI). Considering recent literature [2], FF is a significant jumping off point for creating artificial neural networks that function more like the brain.
- Applications in real-time learning. Hinton poses some unanswered questions about applications of FF in systems that need to learn and make predictions in near-real-time.
- Academic applications in learning without complete knowledge of the NN graph. In adversarial machine learning, for example, an attacker may not have complete knowledge of the NN. Implementing FF may be a new way for an attacker to optimize for an augmentation, δ, such that a datapoint, $x + \delta$, breaks the model.
- Selection of non-uniform layer cost functions. There seems to be no need for each layer to have the same cost function, and experimenting this may yield interesting results.
- Further comparison of backpropagation and FF, specifically looking at rate of convergence and training time.
- Further comparison of backpropagation and FF in training convolutional and recurrent neural networks.

REFERENCES

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

[2] T.P. Lillicrap, et al. "Backpropagation and the brain". Nature Reviews Neuroscience 21, 335–346, 2020.

[3] I. Goodfellow et al.,"Generative adversarial networks" in Advances in Neural Information Processing Systems, Red Hook, NY, USA:Curran Associates, pp. 2672-2680, 2014.

[4] Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In International conference on machine learning, pp. 1597-1607. PMLR, 2020.

[5] Zhao, Gongpei, et al. "The cascaded forward algorithm for neural network training." arXiv preprint arXiv:2303.09728, 2023.

[6] Achterberg, Jascha, et al. "Building artificial neural circuits for domain-general cognition: a primer on brain-inspired systems-level architecture." arXiv preprint arXiv:2303.13651, 2023.

[7] Ororbia, Alexander, and Ankur Mali. "The Predictive Forward-Forward Algorithm." arXiv preprint arXiv:2301.01452, 2023.