Ataman, F., Eroğlu, H. (2026). Yeni bir Derin Öğrenme Algoritması: Hibrit KAtman İşbirliği Tabanlı İleri-İleri Ağ (HiLaCob-FF). EMO Bilimsel Dergi, 16(2), 49-63.

Makale Türü: Araştırma Makalesi
Makale Gönderim Tarihi: 26 Kasım 2025
Makale Kabul Tarihi: 11 Mayıs 2026

Yeni Bir Derin Öğrenme Algoritması: Hibrit Katman İşbirliği Tabanlı İleri-İleri Ağ (HiLaCob-FF)

A Novel Deep Learning Algorithm: Hybrid Layer Collaboration-based Forward-Forward Network (HiLaCob-FF)

Fikriye Ataman[1], Halil Eroğlu[2]

0000-0002-0257-7730 0009-0008-8576-2771

[1]Van Yüzüncü Yıl University, Engineering Faculty, Computer Engineering, Van, Türkiye.

[2]Van Yüzüncü Yıl University, Institute of Natural and Applied Sciences, Artificial Intelligence and Robotics, Van, Türkiye.

fataman@yyu.edu.tr, gelecekandroidde@gmail.com

Özet

Bu çalışma, ileri-ileri algoritmasına dayanan "Hibrit Katman İş Birliği Tabanlı İleri-İleri Ağ (HiLaCob-FF)" adlı yeni bir öğrenme algoritması önermektedir. HiLaCob-FF algoritması, yerel bir kayıp yerine global bir kayıp fonksiyonu kullanmaktadır. MNIST, Fashion MNIST ve CIFAR-10 veri kümelerine uygulanarak, modelin genelleme yeteneği kanıtlanmıştır. Bu çalışma, derin öğrenmede alternatif yaklaşımların önemini vurgulamaktadır.

Önerilen yeni kayıp fonksiyonunun model eğitim kayıplarını iyileştirdiği sonucuna varılmıştır. Sonuç olarak, MNIST veri kümesine uygulanan en son kayıp fonksiyonuyla modelin eğitim kaybı %1,42 oranından %0,87'ye düşmüştür. Benzer şekilde, hata oranı CIFAR-10 veri kümesinde %40,91'den %40,36'ya düşmüştür. Önerdiğimiz HiLaCob-FF modeli, MINIST veri setinde %92,32, FashionMINIST veri setinde %78,29 ve CIFAR-10 veri setinde %74,20 başarı oranına ulaşmıştır. Bu performanslar, geri yayılım tabanlı modellerin performansı kadar yüksek olmasa da, sonuçlar ileri-ileri ağlar geliştirmek için önemlidir.

Anahtar Kelimeler: *Derin Öğrenme, İleri-İleri Algoritması, Beyinden Esinlenen Modeller, HiLaCob-FF, Alternatif Öğrenme Algoritmaları*

Abstract

This study proposes a new learning algorithm, the "Hybrid Layer Collaboration-based Forward-Forward Network (HiLaCob-FF)," which is based on the forward-forward algorithm. The HiLaCob-FF algorithm employs a new loss function, a global loss rather than a local loss, and hyperparameter optimization. It was applied to the MNIST, Fashion MNIST, and CIFAR-10 datasets, demonstrating its ability to generalize and recognize patterns. This study underscores the importance of alternative approaches in deep learning.

It is concluded that the proposed loss function improved the model training losses. As a result, with the latest loss function applied to the MNIST dataset, the model's training loss decreased by 1.42% to 0.87%. Similarly, the error rate reduced from 40.91% to 40.36% on the CIFAR-10 dataset. Our proposed HiLaCob-FF model achieves 92.32% success on the MINIST dataset, 78.29% on the FashionMINIST dataset, and 74.20% on the CIFAR-10 dataset. Although these performances are not as high as those of backpropagation-based models, the results are essential for developing forward-forward networks.

Keywords: *Deep Learning, Forward-Forward Algorithm, Brain-Inspired Models, HiLaCob-FF, Alternative Learning Algorithms*

1 Introduction:

Artificial intelligence and deep learning have gained widespread influence through pioneering successes across many fields. These achievements have sparked the interest of computer scientists and neuroscientists in how artificial intelligence systems resemble the complex learning processes of the human brain [1], [2]. Deep learning models learn to perform complex tasks by training on large datasets [3]. Backpropagation is the most widely used algorithm for training these models [4]. The success of deep learning is largely due to the backpropagation algorithm [5]. The backpropagation algorithm trains artificial neural networks by computing error derivatives to optimize weights and other parameters. The success of this algorithm has led to the wide use of deep learning models [3], [6]. However, there are frequent criticisms in the literature about whether the backpropagation algorithm is compatible with brain biology [7], [8], [9], [10]. Questions such as whether the connections between neurons in the brain are updated through backpropagation and how compatible this algorithm is with biological reality are frequently raised by researchers [4], [9], [10], [11], [12], [13], [14], [15]. Brain biology encompasses information-processing mechanisms, primarily mediated by cortical feedforward pathways. Cortex feedforward is a process that occurs through neural networks and progresses from input to output [16]. In this process, nerve cells on the cortex surface process the inputs and produce the final output. Inputs from organs or within the brain are transmitted to different cortical areas via nerve cells on the cortical surface, thanks to the cortex's feedforward mechanism [16]. Questions about whether the backpropagation algorithm is compatible with brain biology stem from the practical and

biophysical issues that underlie it. The fundamental problems of backpropagation include weight symmetry, the freezing of neuron activities, and waiting for the forward pass to complete before updating weights [3]. Despite these criticisms in the literature, research in deep learning and backpropagation algorithms continues. As a result, algorithms more compatible with the brain's biological learning processes have been proposed as alternatives to backpropagation. The most popular of these algorithms are predictive coding [17], Hebbian learning [18], and the forward-forward algorithm [3]. Predictive coding emphasizes the role of active predictions in the brain [17]. It suggests that not only errors but also expected outcomes are important in information processing [17]. Hebbian learning emphasizes the relationship between synchronization and activation in neural networks [18]. It provides a model that more closely reflects synaptic strengthening processes in the brain [18]. The forward-forward algorithm replaces backpropagation's forward and backward transitions with two forward steps [3].

In the literature, studies on the forward-forward algorithm proposed as an alternative to the backpropagation algorithm are rapidly increasing [19], [20], [21], [22], [23], [24]. Training current and popular technologies, such as Large Language Models (LLMs), incurs high costs and energy requirements due to their large parameter counts and large training data sizes. The main reason is that the gradient-based backpropagation algorithm is also behind LLMs. The abundance of trainable parameters and difficulties, such as storing gradients, have increased the search for solutions to the problems associated with the backpropagation algorithm. In this context, researchers are developing various training procedures. There are new studies in the literature related to the forward-forward algorithm [19], [20], [21], [22], [23], [24], [25], [26].

This paper proposes a learning algorithm based on the forward-forward algorithm and a loss function.

The proposed algorithm is called “Hybrid Layer Collaboration based Forward-Forward Network (HiLaCob-FF)”. The proposed HiLaCob-FF algorithm is more motivated to learn by using global loss instead of layer-based normalization, a modified loss function, and local loss. The use of global loss was inspired by the study conducted by Guy Lorberbom et al. (2024) [27]. It provides accuracy values slightly closer to those obtained by backpropagation than the basic forward-forward algorithm proposed by Hinton. The proposed learning algorithm is the first contribution to the literature.

The proposed model experiments are carried out in three stages. In the first experiment, Hinton's loss function, Symba's loss function, and our proposed loss function are applied to Hinton's basic forward-forward algorithm, respectively. The findings from the first experiment show that the proposed loss function yields the best results. The proposed loss function is the second contribution of our study to the literature.

The second experiment evaluates the results obtained by applying the proposed loss function to Hinton's original forward-forward algorithm and the proposed HiLaCob-FF algorithm. According to the results obtained from the second experiment, the HiLaCob-FF learning algorithm exhibited the best performance. At the end of the second experiment, this study introduces a novel learning algorithm, HiLaCob-FF, to the literature. This algorithm is a forward-forward algorithm that combines layer losses and has an improved loss function.

The new hybrid learning algorithm HiLaCob-FF is applied in the third experiment to the MNIST, Fashion MNIST, and CIFAR-10 datasets. The findings demonstrate the ability of our proposed HiLaCob-FF algorithm to generalize to recognize patterns in data and make inferences.

The proposed algorithm can enable deep learning models to be trained in a manner similar to how the human brain learns. This study emphasizes the importance of alternative approaches that can open new horizons in deep learning and inspire future research. This study is important for understanding how artificial intelligence systems can be developed to approach human-like learning capabilities and contribute to the existing literature. The proposed algorithm has the potential to open new horizons in artificial intelligence by enabling deep learning models to be trained more accurately and quickly. This is another important contribution of this study. The proposed algorithm is explained in detail in the materials and methods section.

This study proposes a novel learning framework, namely HiLaCob-FF, that extends the classical Forward-Forward (FF) paradigm in several fundamental respects. The main contributions of this work can be summarized as follows:

First, unlike the original Forward-Forward algorithm, which relies on strictly layer-wise and independent learning, the proposed method introduces a hierarchical coupling mechanism between layers. This is achieved through an interaction term (β), which incorporates the influence of previously learned representations into the current layer’s optimization process. As a result, the learning process is no longer fully local but becomes partially globally aware while still remaining backpropagation-free.

Second, the Extended Forward-Forward approach in the literature typically improves FF by enabling feature propagation between layers through simple activation concatenation. However, this mechanism does not explicitly model inter-layer dependencies in the learning objective. In contrast, HiLaCob-FF goes beyond simple feature sharing by embedding explicit cross-layer energy interactions into the goodness computation, allowing each layer to be influenced not only by its direct input but also by the aggregated representational dynamics of the entire network.

Third, a novel loss function formulation is introduced to stabilize the difference between positive and negative goodness values through a nonlinear sigmoid-exponential structure. This formulation improves gradient sensitivity while preserving the forward-only learning principle.

Overall, the proposed HiLaCob-FF framework bridges the gap between purely local Forward-Forward learning and globally structured representation learning. It preserves the computational simplicity of FF while significantly enhancing its representational capacity through hierarchical coupling and energy-based interaction modeling.

2 Material and Method

2.1 Dataset and Experimental Environment

Within the scope of the study, the widely used datasets, MNIST [28], FashionMNIST [29], and CIFAR10 [30] are used to compare learning algorithms. In the experimental studies, the performance of the current and proposed forward-forward

model is tested with MNIST [28], FashionMNIST [29] and CIFAR10 [30] datasets.

The experiments were written in Python on Anaconda in a laptop environment with an i5 processor. The machine used had 8 GB of RAM. It has an Nvidia GeForce GTX graphics card with CUDA technology. The programs were compiled using the PyTorch library in the Python environment.

2.2 SymBa Algorithm

The SymBa algorithm is presented as an alternative to models based on Backpropagation. It is based on Hinton's Forward-Forward algorithm [31], [32]. Therefore, it has been proposed as an algorithm suitable for the brain's biological structure. Instead, it addresses the problem of asymmetric gradients arising from conflicting convergence directions between positive and negative examples. The focus is on improving the convergence behavior of the Forward-Forward algorithm [31], [32]. The SymBa algorithm balances positive and negative losses to increase performance and convergence speed [31], [32]. It also adds class information during training, allowing the network to lose class information. The loss function used in the SymBa algorithm works with positive and negative input pairs [31], [32]. The loss is calculated by Equation 1.

$$L(P,N) = \log\left(1 + e^{-\alpha(G_P - G_N)}\right) \text{ (Equation 1)}$$

In the Equality, L(P,N) represents the loss function for positive and negative inputs. $G_P - G_N$ represents the difference in the layer's goodness between positive and negative inputs. This value measures how positive and negative inputs are affected by the layer. Moreover, α is a scaling factor that remains constant for all layers and represents a hyperparameter. The purpose of Equality 1 is to take the difference between positive and negative inputs with respect to the layer and apply a sigmoid function to this difference. In this way, training can be completed more quickly than the loss function proposed by Hinton [3]. This formula, unlike previous studies, is proposed as a loss function to accelerate training. SymBa loss uses a function expressed as $\log(1 + e^x)$, which is a smooth approximation of $\max(0, x)$. The term "smooth approximation" refers to a mathematical function that makes a given behavior smoother, more continuous, or more uniform. In this case, the SymBa loss provides a smoother approximation of the ReLU function ($\max(0, x)$) using the function $\log(1 + e^x)$. The $\max(0, x)$ sets negative inputs to 0 while leaving positive inputs unchanged. It returns 0 when x is negative and x when x is positive. The graph of this function contains a vertex constructed perpendicular to the x = 0 plane. Since it abruptly returns zero when x is negative, it has a non-differentiable point at x = 0. However, the $\log(1 + e^x)$ also provides a smooth, continuous increase for negative values of x. It provides a smoother output, especially for negative inputs. For example, the function $\log(1 + e^x)$ starts from zero and increases slowly for negative values. This is a differentiable and continuous function. This generally makes the training process more stable and can reduce the risk of overfitting. Therefore, SymBa Loss tries to optimize the training process using this smooth approach.

2.3 Swish Function and Proposed Swish Variant Function:

The Swish function is a smooth approximation and is expressed as xσ(βx). Here, x denotes the input value, β is a constant typically set to 1, and σ(x) represents the sigmoid activation function. The Swish function has been proposed as an alternative to the widely used ReLU activation function. The Swish function provides linear growth for positive values while approaching zero for negative values. However, the Swish function exhibits a non-monotonic behavior for negative values. This feature can have a regularization effect, increasing the network's generalization ability. This can make learning on the network more regular. A non-monotonic function can help the model learn more diverse features and prevent overfitting. Therefore, the SymBa loss using the Swish function can help the network generalize better and perform better. The SymBa loss using the swish function uses the $x\sigma(x)$ function instead of the $log(1 + e^x)$ function mentioned in the previous explanations. This provides a different approach to calculating the loss and may give better results in certain cases. In this case, the proposed loss function obtained is in the form of Equation 2. In the forward-forward algorithm proposed in this study, the swish variant given in Equation 2 is used as the loss function. The default swish function is given in Equation 3. Swish is a combination of the sigmoid activation function($\sigma(\beta x) = \frac{1}{1+e^{-\beta x}}$) and the ReLU activation function ($f(x) = \max(0, x) = \begin{cases} x_i, if\ x_i \geq 0 \\ 0, if\ x_i < 0 \end{cases}$)[33], [34]

$$L(P,N) = \frac{-\alpha(G_P - G_N)}{1+e^{-\alpha(G_P - G_N)}} \quad \text{(Equation 2)}$$

$$f(x) = x \cdot \sigma(\beta x) = x\left(\frac{1}{1+e^{-\beta x}}\right) = \frac{x}{1+e^{-\beta x}} \text{ (Equation 3)}$$

In deep neural networks, when the training data pair { (x1, y1), (x2, y2) … (xm, ym)} is given, the aim is to create a match between the data sample x and the corresponding label y. This is a learning paradigm used to model and learn the relationship between data samples and their labels. Classical fully connected neural networks have a recursive structure. Each layer processes the output of the previous layer. This recursive structure is effective on large datasets by enabling the network to learn more complex features. Therefore, an n-layer fully connected neural network can be represented as a set of interconnected functions. This means that each layer of the network represents a general feature-learning process that processes the output of the previous layer. Each layer can be thought of as a sub-function that contributes to the network's overall output. Each i-th layer is denoted by the function f_i, as shown in Equation 4.

$$f_{1:n}(x) = \left(f_{layer\ n}\ o\ f_{layer\ n-1}\ o \dots o\ f_{layer\ 1}\right)(x)$$
(Equation 4)

2.4 Backpropagation Algorithm

In backpropagation-based learning, the function parameters in a neural network are the training dataset and a predefined loss function $L(f_{1:n}, x, y)$ [35]. During tuning, one or more data points are transmitted to the network, and a prediction is obtained. The loss function is used to evaluate the network's output. Equation 5 is tried to be optimized with optimization algorithms.

$$\min_{w}(L(f_{1:n}, x, y) \quad \text{(Equation 5)}$$

This optimization problem is usually addressed by stochastic gradient descent (SGD) or similar gradient-based methods. For example, the optimization step for the i-layer in an n-layer neural network can be expressed as in Equation 6.

$$\Delta w_i = -\alpha \frac{\partial L(f_{1:n}, x, y)}{\partial w_i} = -\alpha \frac{\partial L(f_{1:n}, x, y)}{\partial f_{1:n}(x)} \dots \frac{\partial f_{1:i+1}(x)}{\partial f_{1:i}(x)} \frac{\partial f_{1:i}(x)}{\partial w_i}$$

(Equation 6)

In this case, information first propagates through the network. The loss is calculated. The derivative is calculated according to the calculated loss, and the loss is propagated backward in the network [35]. Figure 1 shows the information flow for the backpropagation algorithm along a branch.

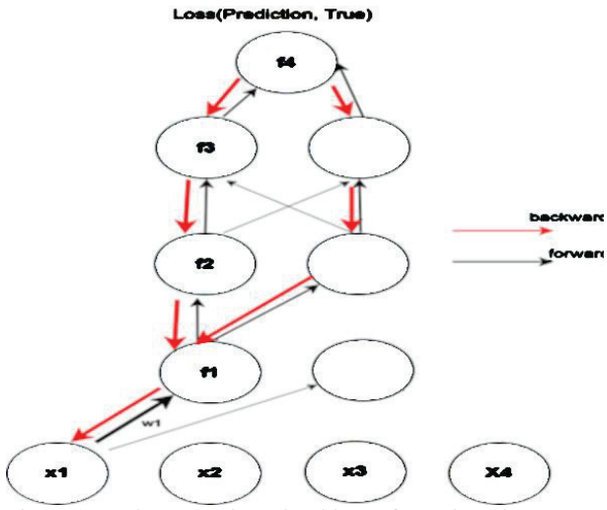


Figure 1: Backpropagation Algorithm Information Flow.

In the backpropagation algorithm, information is propagated in the network with a forward pass. Then, the error is calculated. The network weights are updated backward with the gradient of the error. This update is done with $\frac{\partial loss}{\partial W1} = \frac{\partial loss}{\partial f4} \cdot \frac{\partial f4}{\partial f3} \cdot \frac{\partial f3}{\partial f2} \cdot \frac{\partial f2}{\partial f1} \cdot \frac{\partial f1}{\partial W1}$.

2.5 Hinton's New Forward-Forward Algorithm

The forward-forward algorithm is proposed as an alternative deep neural network model to networks based on a backpropagation algorithm [3]. In the Forward-Forward model, the samples are represented as one-hot encoding vectors associated with the input data points x, and these vectors point to the labels y. Additionally, the training data is divided into two clusters: 'positive' and 'negative'. In the positive samples, the one-hot vector points to the actual label. However, the one-hot vector in negative samples points to a randomly selected false label. Intuitively, it tries to learn whether an unlabeled sample is positive or negative. The forward-forward algorithm aims to make each layer 'more positive' for positive samples and 'more negative' for negative samples. It tries to classify a given data point as positive or negative based on the network's quality. To define goodness, the sum of the squares of the activities of the i-th layer, which have been passed through ReLU, Sigmoid, or any activation function, is used [3]. The square expression here is used in the original article [3]. In this study, we propose a different goodness function for the forward-forward algorithm. The goodness function we propose outperforms that in the original research.

The probability that an (x,y) sample is positive by the i-th layer is calculated by Equation 7. ReLU activation or a different activation function can be used. ReLU activation is used in this study.

$$p_i(\text{pozitif}) = (1 + e^{-(|| fonksiyon_{1:i}(x,y) ||^2 - \Theta)} \quad \text{(Equation 7)}$$

Θ is a hyperparameter. It is called the loss threshold value (Hinton, 2022). This parameter governs the magnitude of the learning. Determining the optimal threshold value for this hyperparameter is crucial. Because it is the main element that ensures the algorithm works well. In each layer of the forward-forward algorithm, the sum of the squares of the activations for each data point in the dataset is taken. This total loss is compared to the threshold value, and the network is trained. The goal of the layer is to maximize this sum so that it exceeds the threshold value for positive examples and falls below the threshold value for negative examples. This difference is important because it is passed to the loss function, which penalizes large values. The goal of the loss function is to maximize the layer activation for positive data and minimize it for negative data.

In this paper, the most appropriate threshold value will be selected from the study by Gandhi et al. (2023) [36]. Gandhi et al. (2023) ask whether different threshold values across layers are meaningful. It is tested using monotonically increasing and decreasing k values across layers. It is concluded that monotonically increasing the threshold significantly outperforms other approaches. The study's finding is that larger threshold values increase performance, as the later layers are responsible for recognizing higher-level features. Meanwhile, lower layers generally act as feature extractors. This monotonically increasing threshold strategy is called the "pyramidal approach" [36].

The study also explored the usage of a threshold timer [36]. The idea behind this approach is that the model can benefit from lower thresholds at the beginning of training, then gradually increase the penalty as training progresses. When the research results are examined, it is shown that a threshold timer yields better results than the start. It is declared that an error reduction of 1.8% to 1.3% was found using this approach [36]. The input is normalized during the forward pass, ensuring that it does not affect the magnitude of the output activation. Normalization is applied between layers to prevent later layers from relying on earlier layers' decisions.

Finally, to perform learning, each layer is optimized in turn. Thus, it is attempted to maximize the goodness of positive examples and minimize the goodness of negative examples. For the i-layer optimization, a positive optimization step using SGD is defined as in Eq. 8.

$$\Delta w_i = \frac{\partial log((1+e^{-(|| fonksiyon_{1:i}(x,y) ||^2 - \Theta)})^{-1})}{\partial w_i} \quad \text{(Equation 8)}$$

For the negative step, the operation is done with the opposite sign. This loss will be referred to as the layer loss throughout the study. Unlike backpropagation, the forward-forward algorithm allows a separate training process for each layer. The network is trained layer by layer. Unlike the training step, there are two reasonable inference methods for the forward-forward network. The first method uses a classification neural network that takes the forward-forward network activations as features. This method is shown in Figure 2. An alternative method is to add a label to the input image and pass it through the network. This process is repeated for each label, and the label with the highest activation is used to classify the image. Among the two methods, the approach using a 1-layer classification network has been reported to yield better experimental results [36].

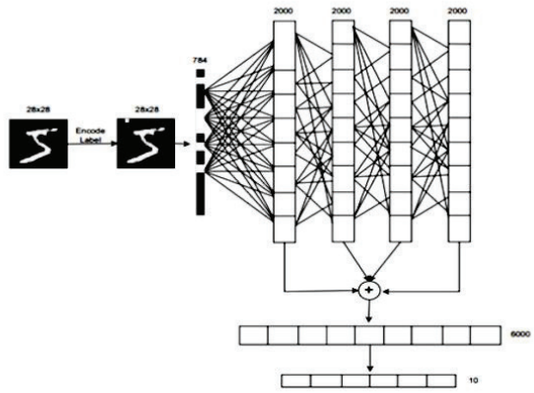


Figure 2: Forward-Forward Algorithm Inference Process [36].

Deep networks have a structure of multiple hierarchical layers. The relationship between layers is critical in deep networks because this hierarchical structure allows the network to extract features efficiently. While the lower layers can learn to recognize 'simple' features, later layers can learn more complex features. The relationship between layers helps the network to make more accurate predictions by creating a robust representation of the data [5].

In the forward-forward algorithm, since each layer only considers the previous layer's output, communication between layers is allowed only through a forward pass. However, forward propagation alone may not help to learn an expected complex representation of the data. As shown in Equation 8, the forward-forward algorithm does not allow information to flow back to previous layers during training. In other words, the first layer is unaware of the existence of the subsequent layers. Therefore, the information they collect during the optimization process cannot be considered. However, as shown in Equation 6, information can flow from one layer to the previous layer during backpropagation. The information flow of the forward-forward algorithm is shown in Figure 3.

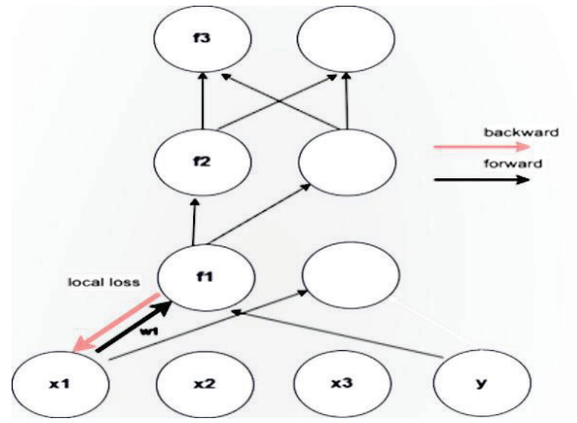


Figure 3: Forward-forward Algorithm Information Flow

Gradient calculation according to the weight in the forward-forward algorithm is done with $\frac{\partial S}{\partial W1} = \frac{\partial S}{\partial ||f1||2} \cdot \frac{\partial ||f1||2}{\partial W1}$. In the forward-forward model, the sum of the goodness of the model's layers is considered. This process can be seen as a vote among the layers on the most appropriate label y for each x. Each layer casts a vote by showing its performance independently of the others. Then it can also evaluate the performance of layer subgroups. It is desirable for each layer to be trained independently of the others, but its contribution to the network can be negative to performance and the ability to learn complex representations. In this case, high errors can be obtained. This is an undesirable situation for deep neural networks. For single-layer networks, the forward-forward algorithm performs well in its current form without any changes.

2.6 Proposed Algorithm: Hybrid Layer Collaboration-based Forward-Forward Network (HiLaCob-FF)

If a feature is added to the forward-forward algorithm that allows the layers to cooperate better during training, better inference and subsequent learning can be encouraged. Concretely, this addition is possible as follows. For the optimization of each layer, the forward-forward algorithm can be developed by considering the network's global goodness, not layer goodness [27]. In order to take into account the global goodness of the network for a data sample (x, y), the goodness of all layers can be added to a specific layer optimization step [27]. The probability calculation for positive examples is formulated as in Equation 9.

$$p_i(\text{pozitive}) = (1 + e^{-(|| \ function\ _{1:i}(x,y) ||^2 + \beta - \Theta)})^{-1}$$ (Equation 9)

Here, the constant β value is the sum of the goodness values from different layers and is shown by Equation 10. Here, F is a function copy with fixed parameters, as shown in Equation 11. In other words, the gradient is not calculated according to F.

$$\beta = \sum_{t \neq i} || \mathcal{F}_{1:t}(x,y)||^2 \quad \text{(Equation 10)}$$
$$\sum_{t<i} || \mathcal{F}_{1:t}(x,y)||^2 \quad \text{(Equation 11)}$$

With this change, the weight updates change as in Equation 12.

$$\Delta w_i = \frac{\partial log((1+e^{-(||\ function_{1:i}(x,y)\ ||^2 + \beta - \theta)})^{-1})}{\partial w_i} \quad \text{(Equation 12)}$$

This update step allows information to flow between layers so that the i-th layer is aware of the subsequent layers and continues training by updating its parameters accordingly. It also does this by preserving the assumptions and requirements of the original forward-forward algorithm. The forward-forward algorithm trains each layer until convergence. We propose an alternative approach to updating the parameters of all layers sequentially. The combination of the combined loss, the modification and development of the loss function as in Equation 2, and the additional training procedure we have modified yields an improved forward-forward algorithm. The weights are updated by calculating the derivative using Equation 13.

$$\frac{\partial Total\ Loss}{\partial w1} = \frac{\partial Loss(||\ function_1\ ||2 + ||\ \mathcal{F}_2\ ||2 + ||\ \mathcal{F}_3\ ||2}{\partial ||\ function_1\ ||2} \cdot \frac{\partial ||\ function_1\ ||2}{\partial w1} \quad \text{(Equation 13)}$$

Algorithm 1 presents Hinton's forward-forward algorithm, and Algorithm 2 presents the layer-connected extended forward-forward algorithm. Algorithm 3 optimizes the loss function in equation 2 by using the loss regularized with the Swish Function [33], [34], [37] in addition to the new hybrid learning algorithm proposed in the study, the Extended Forward-Forward Algorithm with Layer Connection. Experimental results show that the forward-forward algorithm, which is offered as an alternative to the backpropagation algorithm proposed by Hinton, yields better performance. Thus, the forward-forward algorithm, which addresses the basic problems of backpropagation identified by Hinton, such as freezing neuron activities, waiting for the completion of the forward pass before updating weights, and recording gradients, has been extended and modified to obtain better results. This is the main contribution of the study to the literature. In Algorithms 1 and 2, optimization is shown only for positive data. The situation is the same for negative data.

According to the algorithm, the gradient calculation is done as

$$\partial \frac{\partial S}{\partial W1} = \frac{\partial S(||f1||2 + ||F2||2 + ||F3||2}{\partial ||f1||2} \cdot \frac{\partial ||f1||2}{\partial W1}$$

Figure 4 shows the information flow of the HiLaCob-FF algorithm.

Algorithm 1: Forward-Forward Learning Algorithm (Hinton)
Input: Training set S = {(x, y)}, model parameters θ = {θ₁, …, θₙ}, learning rate η, threshold θ₀, number of layers n
Output: Trained parameters θ
1: Initialize all layer parameters θ_i randomly
2: for i = 1 to n do
3: repeat
4: Sample a minibatch (x, y) from S
5: Construct positive input:
6: $x^+ = (x, y)$
7: Construct negative input:
8: $x^- = (x, \tilde{y})$, where $\tilde{y} \neq y$
9: Compute positive activations:
10: $a_i^+ = f_i(x^+; \theta_i)$
11: Compute negative activations:
12: $a_i^- = f_i(x^-; \theta_i)$
13: Compute goodness values:
14: $G_i^+ = \|a_i^+\|^2$
15: $G_i^- = \|a_i^-\|^2$
16: Compute probabilities:
17: $p_i^+ = \sigma(G_i^+ - \theta_0)$
18: $p_i^- = \sigma(G_i^- - \theta_0)$
19: Compute layer-wise loss:
20: $L_i = -\log(p_i^+) - \log(1 - p_i^-)$
21: Update parameters (local learning):
22: $\theta_i \leftarrow \theta_i - \eta\, \partial L_i / \partial \theta_i$
23: until convergence
24: end for

Definition of variables:
θ_i : Parameters of layer i
f_i : Forward function of layer i
S : Training dataset
(x, y) : Input-label pair
$\tilde{y}$: Incorrect label
x^+, x^- : Positive and negative inputs
h_i^+, h_i^- : Layer input with connection
a_i^+, a_i^- : Activations
G_i^+, G_i^- : Goodness values
$\sigma(\cdot)$: Sigmoid function
θ_0 : Threshold
L_i : Loss
η : Learning rate

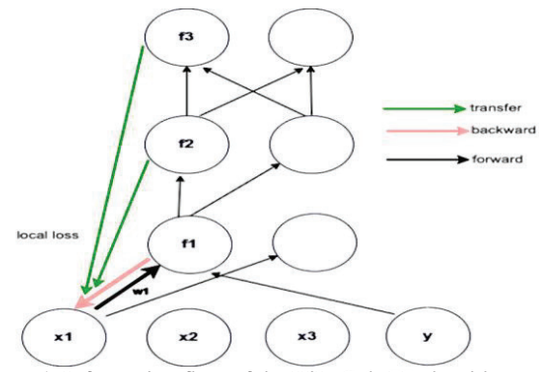


Figure 4: Information flow of the HiLaCob-FF algorithm

Algorithm 2: Extended Forward-Forward Algorithm with Layer Connections

Input: Training set S = {(x, y)}, parameters $\theta = \{\theta_1, \ldots, \theta_n\}$, learning rate η, threshold θ_o, number of layers n
Output: Trained parameters θ

Initialize all parameters θ_i randomly
for i = 1 to n do
  repeat
    Sample a minibatch (x, y) from S
    Construct positive and negative inputs:
      $x^+ = (x, y)$, $x^- = (x, \tilde{y})$, where $\tilde{y} \neq y$
    if i = 1 then
      $h_i^+ = x^+$, $h_i^- = x^-$
    else
      $h_i^+ = \text{concat}(a_{i-1}^+, x^+)$
      $h_i^- = \text{concat}(a_{i-1}^-, x^-)$
    end if
    Compute activations:
      $a_i^+ = f_i(h_i^+; \theta_i)$
      $a_i^- = f_i(h_i^-; \theta_i)$
    Compute goodness values:
      $G_i^+ = \|a_i^+\|^2$
      $G_i^- = \|a_i^-\|^2$
    Compute probabilities:
      $p_i^+ = \sigma(G_i^+ - \theta_o)$
      $p_i^- = \sigma(G_i^- - \theta_o)$
    Compute loss:
      $L_i = -\log(p_i^+) - \log(1 - p_i^-)$
    Update parameters:
      $\theta_i \leftarrow \theta_i - \eta \, \partial L_i / \partial \theta_i$
    Store activations for next layer:
      $a_i^+ \rightarrow$ positive path
      $a_i^- \rightarrow$ negative path
  until convergence
end for

Definition of variables:
θ_i : Parameters of layer i
f_i : Forward function of layer i
S : Training dataset
(x, y) : Input-label pair
$\tilde{y}$: Incorrect label
x^+, x^- : Positive and negative inputs
h_i^+, h_i^- : Layer input with connection
a_i^+, a_i^- : Activations
G_i^+, G_i^- : Goodness values
$\sigma(\cdot)$: Sigmoid function
θ_o : Threshold
L_i : Loss
η : Learning rate

Algorithm 3: Proposed HiLaCob-FF

Input: Training set S = {(x, y)}, number of samples n, threshold θ, coefficient α, learning rate lr
Output: Updated weights W

Initialize θ, α, lr and model weights W randomly
Define loss function:
  loss(G_P, G_N, α) = [$-\alpha$ (G_P − G_N)] / [1 + exp($-\alpha$ (G_P − G_N))]
for i = 1 to n do
  Sample (x, y) from S[i]
  Compute hybrit contribution term:
    $\beta = \Sigma \, |F_{1 \rightarrow t}(x, y)|^2$ for all $t \neq i$
  Compute layer-wise positive goodness:
    $G\_P = \sigma(\|F_{1 \rightarrow i}(x, y)\|^2 + \beta - \theta)$
  Compute layer-wise negative goodness:
    $G\_N = \sigma(\|F_{1 \rightarrow i}(x, y)\|^2 + \beta - \theta)$
  Compute loss:
    L = loss(G_P, G_N, α)
  Compute gradient term:
    $g\_wi = -(1 / (1 + \exp(L))) \cdot \exp(-L) \cdot (x \cdot \sigma(x))$
  Update weights:
    $W\_i \leftarrow W\_i - lr \cdot g\_wi$
end for
Repeat until convergence

Definition of Variables
θ : Threshold value
n : Number of samples in dataset S
α : Scaling coefficient
(x, y) : Input-label pair
β : Cross-layer interaction term (energy from other layers)
G_P : Positive goodness score
G_N : Negative goodness score
L : Loss value
g_wi : Gradient for weight update
W_i : Weight of layer i
lr : Learning rate
$F_{1 \rightarrow i}$: Feature transformation from layer 1 to i
$\sigma(\cdot)$: Sigmoid functio

2.7 Performance Metrics

Many criteria can be used to evaluate the performance of the results. These include F-score, recall, precision, and accuracy. The F-score is the harmonic mean of precision and recall. It is used to measure the accuracy of the classification model. The closer the F-score value is to 1, the better the classification performance of the model [38]. Recall is used to measure the sensitivity of the classification model. Recall is the actual positive rate [38]. Precision is used to measure the accuracy of the classification model. Precision is the true positive rate [38]. Accuracy measures the classification model's performance. Accuracy is the ratio of correctly predicted data points to the total data points [38]. Table 1 shows the error matrix.

Given the error matrix, Recall, Precision, Accuracy, and F-score are calculated using equations 14, 15, 16, and 17 respectively.

Table 1: Confusion Matrix

		Predicted Data		
		Negative	Pozitive	
Real Data	Negative	TN	FP	
	Pozitive	FN	TP	
				Total

$$Recall = \frac{Recall = \mathrm{TP}}{TP+FN} \quad \text{(Equation 14)}$$

$$\mathrm{Precision} = \frac{TP}{TP+FP} \quad \text{(Equation 15)}$$

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \quad \text{(Equation 16)}$$

$$F - Score = 2x\frac{\mathrm{PrecisionxRecall}}{\mathrm{Precision+Recall}} \quad \text{(Equation 17)}$$

3 Findings and Discussion

3.1 Network Parameters

Since the backpropagation algorithm yields effective results in the early stages, the epoch count was kept low during network training in experimental studies. The parameter values used during training are given in Table 2. These values are the same in all algorithm tests.

Table 2: Neural Network Training Parameters

Parameters	Value
Number of Neurons	1000
Layer	12
Learning Rate	0.1
Epoch	20
Batch size	4096
Optimizer	Adam

3.2 Experimental Results with Novel Loss Functions on Forward-Forward Network

Under this heading, experimental results are presented by running different loss functions on the forward-forward algorithm using the MNIST dataset. Considering the parameters in Table 2, the training and test losses obtained with the forward-forward algorithm proposed by Hinton are given in Table 3. In the first experiment, the forward-forward algorithm was trained on the loss values used by Hinton. When Figures 5 and 6, given below, are examined, it is observed that the training and test losses decrease to 11%-12% by the end of 20 epochs.

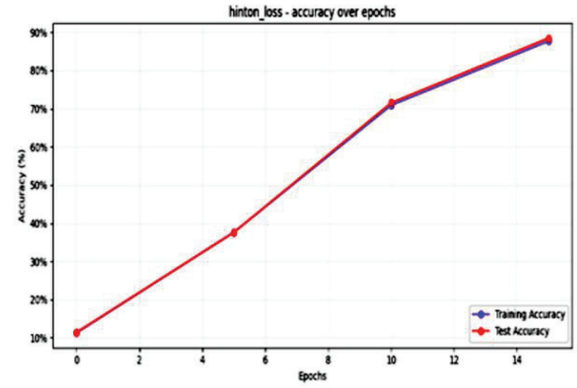


Figure 5: Forward-Forward Algorithm Epoch/Accuracy Graph

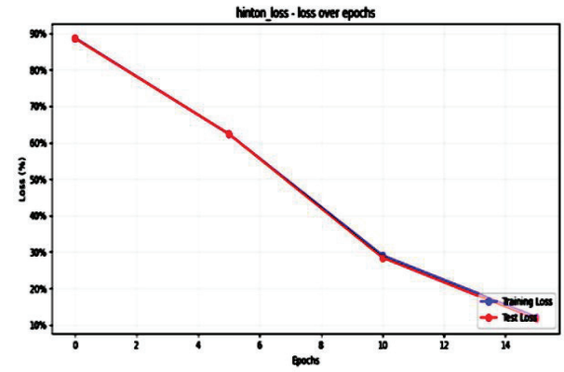


Figure 6: Epoch/Loss Graph of Forward-Forward Algorithm

Table 3: Three different loss function results of the Forward-Forward Algorithm

		0	5	10	15
Training Loss	HL	88.73	62.39	29.07	12.29
	SL	16.11	5.36	3.85	2.41
	Our	**12.47**	**2.82**	**1.49**	**1.01**
Testing Loss	HL	88.61	62.40	28.50	11.55
	SL	15.83	5.57	4.35	3.10
	Our	**12.12**	**3.02**	**2.24**	**1.85**
HL: Hinton Loss SL: SymBa Loss Our: Proposed SymBa Loss					

In the second experiment, the loss function developed with the SymBa Algorithm was applied to the Forward-Forward algorithm. Table 5 presents the results obtained by applying the Symba loss function to the Forward-Forward algorithm. When the results in Table 3, the training loss graph in Figure 7, and the accuracy graph in Figure 8 are examined, it is observed that the training and test losses decreased significantly at the end of the 20th Epoch in the Forward-Forward algorithm trained using the SymBa Loss.

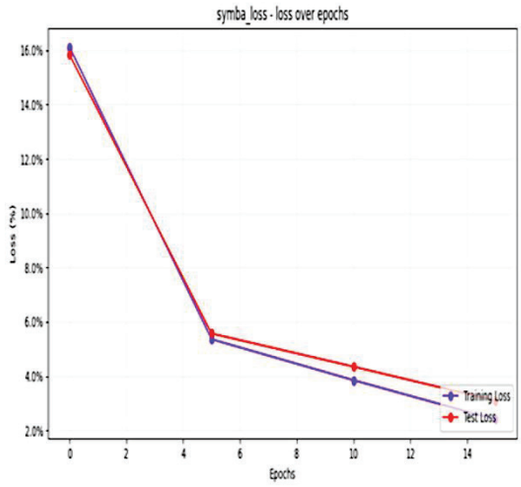


Figure 7: Epoch/Loss Chart of Forward Algorithm Using Symba Loss

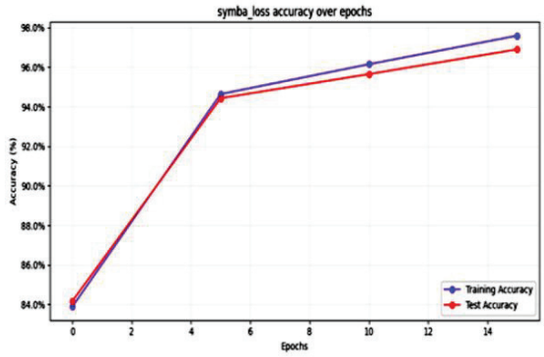


Figure 8: Epoch/Accuracy Chart of Forward Algorithm Using Symba Loss

In the third experiment, the Symba loss function is used as a variant of the swish function, as given in Equation 2. This arrangement is suggested as a novel loss function. The network is retrained using the suggested swish variant loss function. The findings obtained are given in Table 3. In addition, the training accuracy graph is shown in Figure 9, and the loss values are shown in Figure 10 below. When the results in Table 3 and Figures 9 and 10 are examined, it is observed that the training and test losses decreased by 1-1.5% at the end of the 20th Epoch for the forward algorithm trained with the swish variant added to the Symba loss function via Equation 2.

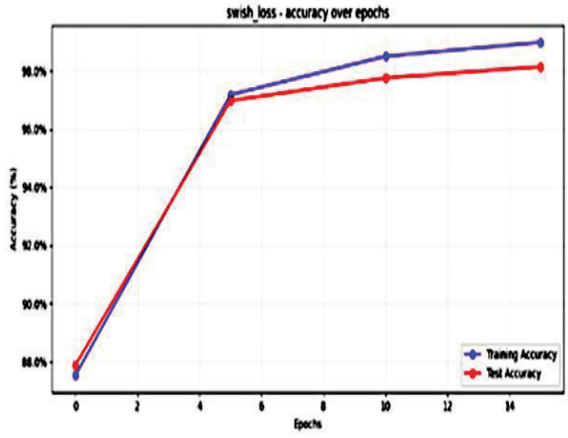


Figure 9: Epoch/Accuracy Graph of Forward Algorithm Using Symba Loss+Swish Variant

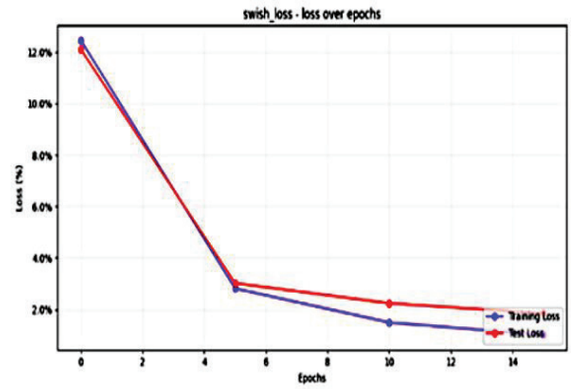


Figure 10: Epoch/Loss Chart of Forward Algorithm Using Symba Loss+Swish Variant

3.3 Experimental results with proposed HiLaCob-FF Algorithm

As an alternative to Hinton's backpropagation algorithm, the forward-forward algorithm was trained on the MNIST dataset using a new loss function that adds layer losses. The findings prove the success of our proposed algorithm in our tests.

Table 4 and Figure 11, presented below, show Hinton's forward-forward algorithm and loss function, the swish loss function proposed in this study, and experimental data obtained with HiLaCob-FF.

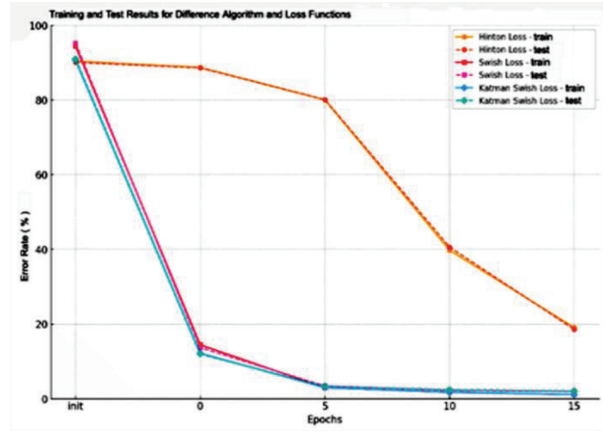


Figure 11: Different Loss Functions in Forward-Forward Algorithm

Table 4: Various Loss Functions Result in Forward-Forward Algorithm

Epoch	HLT	HLTe	SLT	SLTe	OT	OTe
init	90,42	90,1	94,52	95,26	90,69	91
0	88,73	88,62	14,44	13,69	12,2	11,99
5	80,06	80,12	2,98	3,44	2,98	3,33
10	39,78	40,62	1,76	2,28	1,79	2,4
15	19,09	18,59	1,14	1,98	1,16	2,03
HLT: Hinton Loss - Training (%) HLTe: Hinton Loss - Test (%)						

Epoch	HLT	HLTe	SLT	SLTe	OT	OTe
SLT: Swish Loss - Training (%) LSTe: Swish Loss - Test (%) OT: Our Layered Swish Loss - Training (%) OTe: Our Layered Swish Loss - Test (%)						

When Figure 11 and Table 4 are examined, the change in error percentages across different architectures of the forward-forward algorithm and loss functions in the training and testing stages is clearly evident. The neural network that started with Hinton loss had a very high error rate (around 90%) at the beginning, but reduced it to 19% by the end of the 20th epoch. In experiments with Swish loss, the initial error rate was lower, but as the model trained, it decreased to 1-2%. However, what is striking is that, despite the rapid decrease in the model trained with the Swish loss at the beginning, the error rate reached very low levels in subsequent epochs. A similar trend was observed in the HiLaCob-FF algorithm with the Swish loss combination, which combines layer losses, but the error rates decreased more evenly throughout training. These results show that Swish loss initially yielded lower errors, and the algorithm that combines layer losses exhibited more balanced performance with Swish loss, successfully minimizing error rates in both the training and testing stages.

When the epoch times are examined in Figure 12, the layer-connected Hinton forward algorithm and the model with the Swish loss function generally have the longest epoch times among the configurations. In particular, a significant increase in time is observed after the sixth epoch. The HiLaCob-FF model with layer-connected and Swish loss function exhibited more consistent epoch times and was trained in slightly less time than the Hinton-FF model overall. The layer-connected model using the Hinton loss function initially achieved a fast training time, but training time increased as the epochs progressed.

When the training performance graph in Figure 12 is examined, the model with layer-connected and a Swish loss function remained more consistent as the epochs progressed. When Figure 13 is examined in terms of total training time, the model trained with Hinton-Loss took the longest, and compared to the layer-connected and Swish loss function models that use less memory, a minor difference was observed between the training time and memory usage of the Hinton-FF model. In terms of memory usage, the Hinton-Loss model has the lowest.

These results comprehensively analyze the effects of model configuration and loss function on training time and performance and suggest a balanced choice among various configurations.

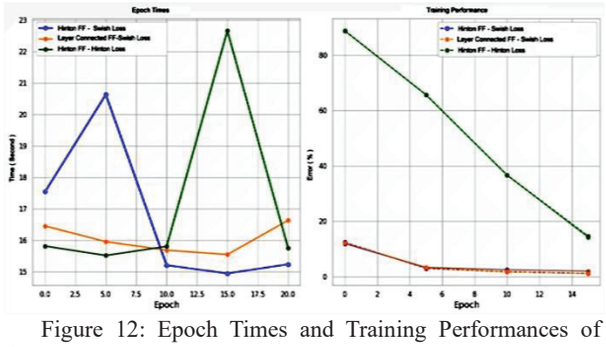


Figure 12: Epoch Times and Training Performances of Different Forward-Forward Algorithms and Loss Functions

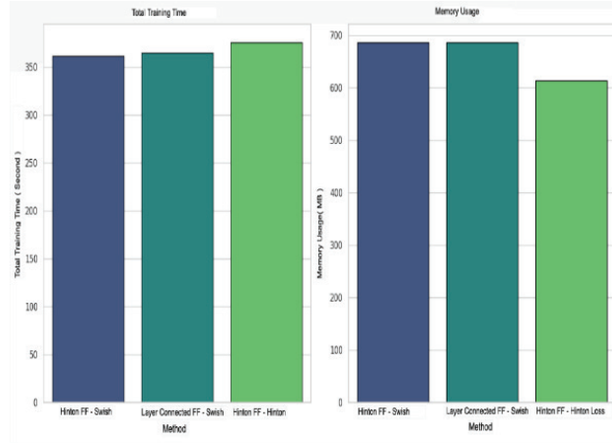


Figure 13: Epoch Times and Training Performances of Different Forward Algorithms and Loss Functions

To demonstrate our algorithm's ability to generalize across datasets, training was performed on the Fashion MNIST and CIFAR-10 datasets, in addition to the MNIST dataset, using the parameters listed in Table 4. The results obtained are presented in Table 5 below.

Table 5: Performance Values of HiLaCob-FF Algorithm on Different Datasets

Dataset	Algorithm	Precision	Recall	F1-Score
MNIST	HiLaCob-FF	92.32	92.32	92.31
Fashion MNIST	HiLaCob-FF	78.29	79.12	78.12
CIFAR-10	HiLaCob-FF	74.20	75.44	73.66

Upon examining Table 5, the performance results indicate that the proposed HiLaCob-FF algorithm is generally robust. In particular, the high Precision, Recall, and F1-score values on the MNIST dataset indicate that the model exhibits highly effective, balanced performance. Although the algorithm's performance is slightly lower on Fashion MNIST and CIFAR-10, it still achieves remarkable results on more difficult datasets. These results confirm the algorithm's ability to adapt to different datasets and demonstrate its overall successful performance. The model's performance across various datasets indicates broad potential for success in the field of application.

Table 6: Comparison of HiLaCob-FF Algorithm on Different Datasets and Different Loss Functions.

Dataset	Symba Error Rate	HiLaCob-FF Error Rate
MNIST	%1.42	%0.87
CIFAR-10	%40.91	%40.36

Table 6 shows the performance of the Symba algorithm on the MNIST and CIFAR-10 datasets. In the MNIST dataset, the Symba algorithm reduced the error rate to 1.42% [31], [32]. The proposed algorithm reduced this error rate to 0.87%. In the CIFAR-10 dataset, the Symba algorithm determined the error rate as 40.91% [31], [32], while the proposed algorithm improved this success by 0.55%, bringing it to 40.36%. These results demonstrate that the proposed algorithm achieves higher

accuracy than the Symba algorithm in both datasets. In addition, these data were obtained by considering the experimental setup in the Symba algorithm [31], [32]. Our aim is not only to achieve better performance than the Symba algorithm, but also to reveal a new training and weight-update procedure, which is one of the fundamental problems of deep learning.

In this study, the lower performance on CIFAR-10 compared to MNIST is due to the dataset's greater complexity. The CIFAR-10 dataset is much more difficult because the images are colored, the backgrounds are complex, and the objects are at different angles and positions. Therefore, when applying deep neural networks to the CIFAR datasets, generalization performance is generally lower than on other benchmark datasets. Thus, it is normal to have similar results in this study. Furthermore, it should be considered that the algorithm and the new loss function are being tested.

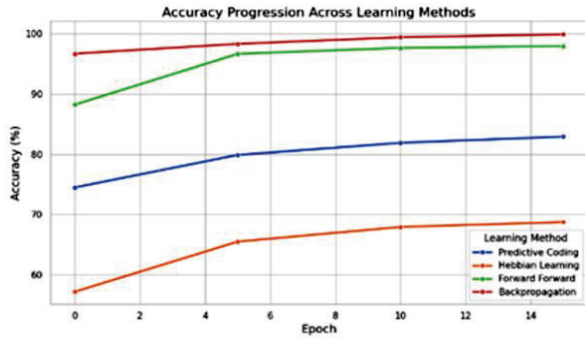


Figure 14: Accuracy Values of Different Learning Methods on the MNIST Dataset

In Figure 14, the training accuracy of four learning methods on the MNIST dataset is compared across the 0th, 5th, 10th, and 15th epochs. Because backpropagation differs from the learning mechanism in our brain, brain-compatible models such as predictive coding and Hebbian learning, which are considered alternative learning methods, have the potential to serve as alternatives to backpropagation. They can reach higher accuracy levels with fine-tuning. The Forward-Forward algorithm is another innovative approach that has gained popularity in recent years. The form developed in this study is illustrated in Figure 14, which shows it to be a strong alternative to backpropagation. While the graph in Figure 14 shows that the Forward-Forward algorithm achieved remarkable success, reaching 97.95% accuracy after the 15th epoch, it also highlights an area where predictive coding and Hebbian learning methods may yield better results with fine-tuning. These results demonstrate the potential of alternative learning methods that do not rely on backpropagation and are highly brain-compatible in neural network studies.

4 Discussion and Conclusion

The Forward-Forward (FF) algorithm is fundamentally based on layer-wise local learning, where each layer is trained using its own objective without relying on backpropagation or global error signals propagated through the network. In this study, we introduce a global "goodness" derivative to enhance the discriminative capability of the model. While this modification improves performance, it also raises important theoretical questions regarding locality, its relation to backpropagation, and biological plausibility.

First, it is important to clarify that the proposed global "goodness" derivative does not violate the core principle of local learning in FF. Unlike backpropagation, where gradients are computed via the chain rule and propagated across layers, the proposed approach does not transmit error derivatives layer-by-layer. Instead, the global goodness term acts as a scalar evaluative signal, guiding learning without explicitly coupling parameter updates across layers through gradient dependencies. Each layer continues to update its parameters based on its own activations and locally computed goodness measures. Therefore, the learning rule remains local in terms of parameter updates, even though it is informed by a globally defined metric.

Second, the proposed method should not be interpreted as an approximation of backpropagation. Backpropagation requires precise gradient computation across all layers and depends on weight symmetry and full differentiability of the network. In contrast, the introduced global goodness derivative does not enforce gradient consistency across layers, nor does it require symmetric weights or backward signal transport. The absence of chained derivatives distinguishes our method fundamentally from gradient-based optimization. Thus, the method occupies an intermediate position between purely local learning and globally guided optimization, without collapsing into backpropagation.

From a biological perspective, the proposed formulation preserves key aspects of biological plausibility associated with the FF algorithm. Specifically, it avoids the need for backward weight transport and maintains forward-only signal processing during learning. The global goodness signal can be interpreted as analogous to neuromodulatory signals observed in biological neural systems, such as dopamine-based reward signals, which provide global feedback without specifying exact synaptic credit assignment. In this sense, the proposed approach remains consistent with biologically inspired learning paradigms, while introducing a weak form of global coordination.

Nevertheless, the introduction of a global component inevitably introduces a degree of indirect dependency between layers. This represents a trade-off between strict locality and improved learning performance. Rather than viewing this as a limitation, we position the proposed method as a hybrid framework that extends the FF algorithm by incorporating global guidance while preserving local update rules.

In summary, the proposed approach maintains the essence of layer-wise local learning, does not approximate backpropagation, and remains compatible with biologically plausible learning principles. At the same time, it introduces a controlled form of global information that enhances learning effectiveness. This positions the method within a broader class of hybrid learning algorithms that balance local computation with global coordination.

In this study, a novel training approach based on the Forward-Forward (FF) algorithm was proposed by introducing a layered loss formulation, referred to as the HiLaCob-FF algorithm. In addition, a modified loss design incorporating a SymBa-based structure and a swish variant was investigated to improve the learning dynamics of FF-based models.

Experimental results on the MNIST dataset demonstrate that the proposed approach significantly improves training

stability and convergence behavior compared to the original Hinton loss. In particular, the proposed layered loss formulation achieves lower error rates and more balanced performance throughout training. While the standard Hinton loss exhibits high initial error rates and slower convergence, the proposed method yields a smoother, more consistent error reduction. Furthermore, integrating the swish-based variant further improves loss minimization.

The generalization capability of the proposed method was further evaluated on Fashion-MNIST and CIFAR-10 datasets. The results indicate that the HiLaCob-FF algorithm maintains competitive performance across different datasets, achieving strong results on MNIST and reasonable performance on more complex datasets. However, the relatively low performance observed on CIFAR-10 highlights the limitations of the current approach in handling high variability, complex backgrounds, and low-resolution color images. This behavior is consistent with the known difficulty of CIFAR-10 compared to simpler benchmark datasets.

Comparative analysis with the SymBa algorithm shows that the proposed method achieves modest but consistent improvements in error rates across datasets. These improvements, although limited in magnitude for complex datasets, suggest that the proposed loss formulation contributes positively to the learning process. It should be noted that the primary contribution of this study is not only performance improvement but also the introduction of an alternative training and weight-update mechanism within the Forward-Forward framework.

In terms of computational characteristics, the proposed method demonstrates stable training behavior with slightly improved time efficiency compared to the standard FF implementation, while maintaining comparable memory usage. These findings indicate that the proposed approach provides a balanced trade-off between performance and computational cost.

Despite these promising results, several limitations remain. The experimental evaluation is limited to relatively small-scale datasets, and the absence of extensive comparisons with backpropagation-based deep learning models restricts the ability to fully position the proposed method within the broader literature. In addition, further analysis is required to better understand the theoretical implications of introducing a global goodness-related derivative in a framework originally designed for local learning.

Comparisons with backpropagation are limited to the specified experimental conditions and selected architectures. The method's relative performance across different network depths, activation functions, and optimization methods has not been comprehensively analyzed in this study.

The presented results show that the proposed method is feasible within the FF algorithm-based learning paradigm using global goodness derivatives, but it does not make a definitive claim that it can compete with backpropagation under all conditions.

Future work will focus on extending the proposed method to larger and more complex datasets, conducting comprehensive ablation studies, and performing detailed comparisons with state-of-the-art backpropagation-based models. Moreover, investigating the theoretical properties and biological plausibility of the proposed formulation remains an important direction for further research.

In conclusion, the HiLaCob-FF algorithm represents a promising step toward improving Forward-Forward-based learning by enhancing loss design and training stability, while also highlighting key challenges that must be addressed for broader applicability.

5 References

[1] W. Zhang, S. Ma, X. Ji, X. Liu, Y. Cong, and L. Shi, "The development of general-purpose brain-inspired computing," *Nature Electronics 2024 7:11*, vol. 7, no. 11, pp. 954–965, Nov. 2024, doi: 10.1038/s41928-024-01277-y.

[2] Song, Y., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2020). Can the brain do backpropagation?---exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, *33*, 22566-22579.

[3] Hinton, G. (2022). The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, *2*(3), 5. https://www.cs.toronto.edu/~hinton/FFA13.pdf (03.05.2025)

[4] Y. Song, T. Lukasiewicz, Z. Xu, and R. Bogacz, "Can the brain do backpropagation? — Exact implementation of backpropagation in predictive coding networks," *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, 2020.

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[6] Werbos, P. J. (2005, September). Applications of advances in nonlinear sensitivity analysis. In *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31–September 4, 1981* (pp. 762-770). Berlin, Heidelberg: Springer Berlin Heidelberg.

[7] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cogn. Sci.*, vol. 11, no. 1, pp. 23–63, 1987, doi: 10.1016/S0364-0213(87)80025-3.

[8] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an integration of deep learning and neuroscience," *Front. Comput. Neurosci.*, vol. 10, no. SEP, p. 94, 2016, doi: 10.3389/fncom.2016.00094.

[9] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nat. Rev. Neurosci.*, vol. 21, no. 6, pp. 335–346, 2020, doi: 10.1038/s41583-020-0277-3.

[10] Pozzi, I., Bohte, S., & Roelfsema, P. (2020). Attention-Gated Brain Propagation: How the brain can implement reward-based error

backpropagation. *Advances in neural information processing systems*, *33*, 2516-2526.

[11] P. Mazzoni, R. A. Andersen, and M. I. Jordan, "A more biologically plausible learning rule for neural networks," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 88, no. 10, pp. 4433–4437, 1991, doi: 10.1073/pnas.88.10.4433.

[12] Kohan, A. A., Rietman, E. A., & Siegelmann, H. T. (2018). Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. *arXiv preprint arXiv:1808.03357*.

[13] Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*.

[14] Xie, X., & Seung, H. S. (2003). Equivalence of backpropagation and contrastive Hebbian learning in a layered network. *Neural computation*, *15*(2), 441-454.

[15] Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2024). Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature neuroscience*, *27*(2), 348-358.

[16] K. S. Rockland, "Notes on Visual Cortical Feedback and Feedforward Connections," *Front. Syst. Neurosci.*, vol. 16, 2022, doi: 10.3389/fnsys.2022.784310.

[17] Millidge, B., Seth, A., & Buckley, C. L. (2021). Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*.

[18] R. L. Sumner, M. J. Spriggs, S. D. Muthukumaraswamy, and I. J. Kirk, "The role of Hebbian learning in human perception: a methodological and theoretical review of the human Visual Long-Term Potentiation paradigm," *Neurosci. Biobehav. Rev.*, vol. 115, pp. 220–237, 2020, doi: 10.1016/j.neubiorev.2020.03.013.

[19] A. Bewtra, "A Further Investigation of the Forward-Forward Algorithm," 2023.

[20] I. Oguz *et al.*, "Forward–forward training of an optical neural network," *Opt. Lett.*, vol. 48, no. 20, p. 5249, 2023, doi: 10.1364/ol.496884.

[21] Tang, D. Y. (2023). The integrated forward-forward algorithm: Integrating forward-forward and shallow backpropagation with local losses. *arXiv preprint arXiv:2305.12960*.

[22] Ahamed, M. A., Chen, J., & Imran, A. A. Z. (2023). FFCL: Forward-Forward contrastive learning for improved medical image classification. In *Medical Imaging with Deep Learning, short paper track*.

[23] A. Reyes-Angulo and S. Paheding, "The Forward-Forward Algorithm as a feature extractor for skin lesion classification: A preliminary study.," 2023. doi: 10.52591/lxai202307235.

[24] Zhao, G., Wang, T., Jin, Y., Lang, C., Li, Y., & Ling, H. (2025). The cascaded forward algorithm for neural network training. *Pattern Recognition*, *161*, 111292.

[25] A. Mehonic and A. J. Kenyon, "Brain-inspired computing needs a master plan," *Nature 2022 604:7905*, vol. 604, no. 7905, pp. 255–260, Apr. 2022, doi: 10.1038/s41586-021-04362-w.

[26] Y. Zhang *et al.*, "A system hierarchy for brain-inspired computing," *Nature 2020 586:7829*, vol. 586, no. 7829, pp. 378–384, Oct. 2020, doi: 10.1038/s41586-020-2782-y.

[27] G. Lorberbom, I. Gat, Y. Adi, A. Schwing, and T. Hazan, "Layer Collaboration in the Forward-Forward Algorithm," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, pp. 14141–14148, Mar. 2024, doi: 10.1609/aaai.v38i13.29324.

[28] Y. LeCun, (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[29] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

[30] A. Krizhevsky, V. Nair, & G.Hinton. (2014, September). *The CIFAR-10 dataset*.

[31] H.-C. Lee and J. Song, "SymBa: Symmetric Backpropagation-Free Contrastive Learning with Forward-Forward Algorithm for Optimizing Convergence," Mar. 2023. doi: https://doi.org/10.48550/arXiv.2303.08418.

[32] M. Ghader, S. R. Kheradpisheh, B. Farahani, and M. Fazlali, "Backpropagation-free spiking neural networks with the forward–forward algorithm," *Scientific Reports 2026 16:1*, vol. 16, no. 1, pp. 14294-, Mar. 2026, doi: 10.1038/s41598-026-41671-4.

[33] S. Sunkari *et al.*, "A refined ResNet18 architecture with Swish activation function for Diabetic Retinopathy classification," *Biomed. Signal Process. Control*, vol. 88, p. 105630, Feb. 2024, doi: 10.1016/j.bspc.2023.105630.

[34] M. A. Mercioni and S. Holban, “P-Swish: Activation Function with Learnable Parameters Based on Swish Activation Function in Deep Learning,” *2020 14th International Symposium on Electronics and Telecommunications, ISETC 2020 - Conference Proceedings*, Nov. 2020, doi: 10.1109/ISETC50328.2020.9301059.

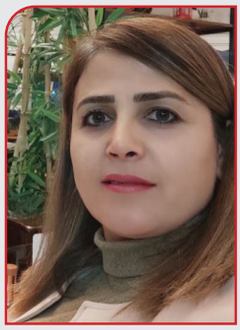
[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[36] Gandhi, S., Gala, R., Kornberg, J., & Sridhar, A. (2023). Extending the forward forward algorithm. *arXiv preprint arXiv:2307.04205*.

[37] M. A. Mercioni and S. Holban, “Soft-Clipping Swish: A Novel Activation Function for Deep Learning,” in *SACI 2021 - IEEE 15th International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, 2021, pp. 225–230. doi: 10.1109/SACI51354.2021.9465622.

[38] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation,” in *AAAI Workshop - Technical Report*, Springer Berlin Heidelberg, 2006, pp. 24–29. doi: 10.1007/11941439_114.

Özgeçmişler



Dr. Fikriye ATAMAN graduated from Mersin University, Department of Computer Engineering in 2006. She completed her master's degree in Computer Engineering at Atatürk University and her doctorate in Statistics at Van Yüzüncü Yıl University. She continues to work as a faculty member in the Department of Computer Engineering, Faculty of Engineering, Van Yüzüncü Yıl University. She is married and has two children. Her main research areas are artificial intelligence, big data, and machine learning.

Halil EROĞLU graduated from the Software Engineering Department of Fırat University. He continues to work actively as a software engineer. His areas of interest are artificial intelligence and deep neural networks. He is pursuing a master's degree in Artificial Intelligence and Robotics at the Institute of Science, Van Yüzüncü Yıl University. He is married and has one child.