ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240682

LightFF: Lightweight Inference for Forward-Forward Algorithm

Amin Aminifar[a,1], **Baichuan Huang**[b,1,*], **Azra Abtahi**[b] **and Amir Aminifar**[b]

[a]Heidelberg University, Germany
[b]Lund University, Sweden

Abstract. The human brain performs tasks with an outstanding energy efficiency, i.e., with approximately 20 Watts. The state-of-the-art Artificial/Deep Neural Networks (ANN/DNN), on the other hand, have recently been shown to consume massive amounts of energy. The training of these ANNs/DNNs is done almost exclusively based on the back-propagation algorithm, which is known to be biologically implausible. This has led to a new generation of forward-only techniques, including the Forward-Forward algorithm. In this paper, we propose a lightweight inference scheme specifically designed for DNNs trained using the Forward-Forward algorithm. We have evaluated our proposed lightweight inference scheme in the case of the MNIST and CIFAR datasets, as well as two real-world applications, namely, epileptic seizure detection and cardiac arrhythmia classification using wearable technologies, where complexity overheads/energy consumption is a major constraint, and demonstrate its relevance. Our code is available at https://github.com/AminAminifar/LightFF.

1 Introduction

The state-of-the-art Artificial Neural Networks (ANNs)/Deep Neural Networks (DNNs) consume massive amounts of energy and pose a threat to the environment [46]. A prime example is GPT-3, a Large Language Model (LLM), that consumes over 1000 megawatt-hour for training alone, which is equivalent to a small town's power consumption for a day [38]. The training of these ANNs/DNNs is done almost exclusively based on the back-propagation algorithm, which is known to be biologically implausible [26]. This has led to a wide range of biologically plausible alternatives, e.g., the Forward-Forward algorithm by [15], focusing on training DNNs without resorting to the biologically-implausible back-propagation scheme, to bridge the existing performance–efficiency gap between the ANNs/DNNs and the cortex.

The majority of the state-of-the-art studies based on the Forward-Forward algorithm have mainly focused on the training of neural networks. However, the inference over already-trained models also consumes a massive amount of energy. Indeed, inference accounts for approximately 60% of the total machine learning energy used at Google [39]. This is because the training overheads are per model and often only incurred once, while the inference overheads are per input/usage and, in the long run, the inference overheads may even dominate the overall energy overheads of ANN/DNN, owing to the many-billion-user services that incorporate machine learning.

In this paper, for the first time, we propose a lightweight inference scheme specifically designed for DNNs trained using the Forward-Forward algorithm by [15]. The key insight is that the local energy-based techniques, such as the Forward-Forward algorithm, provide a strong intermediate measure to decide whether the local energy or the goodness in the case of the Forward-Forward algorithm is sufficient to make a confident decision, without the need to complete the entire forward pass. Our proposed scheme is inspired by the human nervous system [45], where the reflexes do not pass directly into the brain, but synapse in the spinal cord, hence without the delay of routing signals through the brain. This is while the complex inputs that require detailed analysis are processed by the brain.

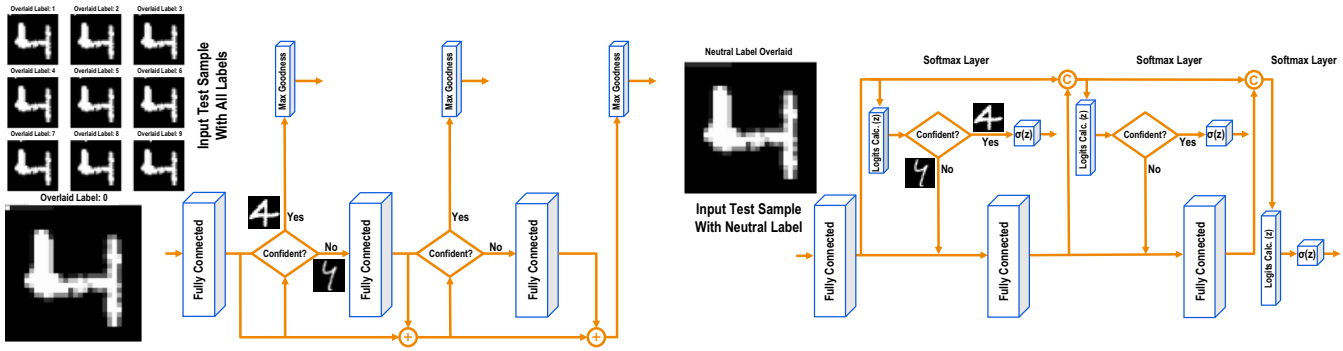
We have evaluated our proposed lightweight inference scheme in the case of the MNIST [23] and CIFAR [21] datasets and shown that the inference overheads can be reduced by up to 10.4 and 2.2 times, respectively. To demonstrate the relevance of our proposed scheme, we have also evaluated our proposed scheme in the context of two real-world medical applications in the Internet of Things (IoT) domain, namely, epileptic seizure detection using wearable devices [50] and cardiac arrhythmia classification using wearable devices [28]. Wearable technologies, and IoT systems in general, are extremely limited in terms of resources, i.e., computing power and energy/battery, and present an excellent application for our proposed lightweight inference to enable real-time and long-term monitoring of patients in ambulatory settings.

2 Lightweight Inference

The intrinsic characteristic of the Forward-Forward algorithm [15] makes it possible to perform inference without completing a forward pass through all the layers of the network. This can be used to save resources when performing inference operations. In this section, we present an approach for lightweight inference based on the Forward-Forward algorithm.

Inference for a model trained based on the Forward-Forward algorithm can be performed using two different procedures: In the first procedure, for a test sample, we repeat the forward pass operation for all possible labels and calculate the accumulative goodness. We select the label with the highest accumulative goodness. We refer to this procedure as *multi-pass*, as it requires multiple forward passes. In the second procedure, a softmax layer at the head of the network, which is learned in the training phase, is used to infer based on the activity of layers for a test sample with a neutral label. This is referred to as *one-pass*, as it requires a single forward pass. Hinton's one-pass

* Corresponding Author. Email: baichuan.huang@eit.lth.se.
[1] Equal contribution.



(a) Light Multi-Pass Inference

(b) Light One-Pass Inference

Figure 1: Lightweight Procedures for Inference

inference process is, in essence, similar to that of the PEPITA algorithm proposed by [7], hence our proposed lightweight inference for the one-pass procedure may also be applied to PEPITA, as we show in Section 3.

2.1 *Inference Based on Multi-Pass Procedure*

In this part, we focus on the multi-pass inference procedure. In our approach, the network's training algorithm and architecture are the same as in [15]. For our lightweight inference, however, instead of performing the forward pass through all layers, once the operations for each layer are completed, we inspect the confidence level of the result, and based on that, we decide whether to continue the forward pass. The accumulated goodness up to that layer is used to determine the label.

Figure 1(a) shows our lightweight inference scheme. As illustrated in the figure, after each layer, we check the confidence. The difficulty of the classification task varies from one test sample to another. As shown, the first layer(s) is sufficient for straightforward test samples, while for other samples, more layers may be required.

In the Forward-Forward algorithm, we train the model to have high goodness for the positive samples and low goodness for negative ones. Considering this, we use the magnitude of goodness to decide how confident we are about our inference result. The goodness in one layer is the sum of the squared activities, $\sum_{i=1}^{n} {a_i}^2$, where a_i is the output of ith neuron and n is the number of neurons in that layer. For the confidence, we only need to add one single neuron with a sigmoid activation function in each layer whose weights are the activities of that layer and the layers before, and its bias is our confidence threshold.

The confidence threshold for each layer (i.e., the bias of the neuron introduced in each layer) is learned based on the validation set. Considering the fully connected network that is already trained by the Forward-Forward algorithm, the input of validation set $\boldsymbol{X}^{(l)}$ and the constructed binary label $\bar{\boldsymbol{y}}^{(l)}$ for layer l are leveraged for training the confidence threshold (i.e., bias) for layer l. We define the ground-truth label $\bar{\boldsymbol{y}}_i^{(l)} = 1$ if the input sample $\boldsymbol{X}_i^{(l)}$ is correctly classified by the layer and $\bar{\boldsymbol{y}}_i^{(l)} = 0$ otherwise. For the confidence threshold $b^{(l)}$ of hidden layer l, one single neuron, denoted as $\sigma(\boldsymbol{w}^{(l)} \cdot \boldsymbol{a}^{(l)} + b^{(l)})$, is connected to the corresponding activations $\boldsymbol{a}^{(l)}$, where $\boldsymbol{w}^{(l)}$ is the weight between hidden layers l and the single neuron. σ is the sigmoid function for the single neuron connected to hidden layers l. We set $\boldsymbol{w}^{(l)} = \boldsymbol{a}^{(l)}$ to capture the goodness value, hence only the value of $b^{(l)}$ needs to be trained. We consider the Binary Cross-Entropy loss function, with $\bar{\boldsymbol{y}}_i^{(l)}$ as the label for the ith sample. Alternatively, we can calculate and set the bias values based on the mean and standard deviation of the goodness for the validation set. As shown in Fig. 2, the distance between the mean values of the negative and positive samples/distributions increases as we consider more layers.

Next, let us discuss the computational overhead for utilizing one layer of a Forward-Forward network that directly impacts the energy consumption of the platform on which the inference is performed. Suppose we pass a test sample through a fully connected hidden layer of Rectified Linear Units (ReLUs) with n neurons. We show the required computations for calculating the activity vector and the goodness for layer i with C_i^a and C_i^g, respectively. The overall computation required for a forward pass on layer i is $C_i^a + C_i^g$. In this case, if our network has N hidden layers, the computation required for calculating the goodness for all layers of the network would be $\sum_{i=1}^{N}(C_i^a + C_i^g)$. In the case of inference based on the multi-pass procedure, if the data has M class labels, then, for each test sample, the computational overhead would be $C_{MP} = M \cdot \sum_{i=1}^{N}(C_i^a + C_i^g)$.

In our lightweight inference scheme, we make a trade-off between classification accuracy and computational efficiency (that leads to energy efficiency). As discussed, we define a confidence threshold for every layer determined by the amplitude of accumulated goodness up to that layer based on a validation dataset. The objective would be to increase the classification performance, which could require using more hidden layers, while decreasing the number of layers used for choosing a label, for the sake of computational efficiency.

We show the computation required for the calculation of goodness for one test sample in our lightweight multi-pass inference by $C_{Light-MP}$. Using our lightweight inference scheme, if we present the probability of completing the inference operation at layer i by p_i, then we can show the expected computational overhead for the calculation of goodness for one test sample as follows:

$$\begin{aligned}\mathbb{E}(C_{Light-MP}) &= M \cdot \sum_{i=1}^{N}\left(\sum_{j=1}^{i}(C_j^a + C_j^g)\right) \cdot p_i \\ &= M \cdot \sum_{j=1}^{N}(C_j^a + C_j^g) \cdot \left(1 - \sum_{i=1}^{j-1} p_i\right),\end{aligned} \tag{1}$$

where $\sum_{i=1}^{N} p_i = 1$.[1] Therefore, we have $(1 - \sum_{i=1}^{j-1} p_i) \leq 1$. As a result, in the worst-case scenario, i.e., when for $i < N$, $p_i = 0$, and for $i = N$, $p_i = 1$, the computational overhead is the same as the inference procedure in [15], i.e., $C_{MP} = M \cdot \sum_{i=1}^{N}(C_i^a + C_i^g)$.

[1] For $j = 1$, we consider $\sum_{i=1}^{j-1} p_i = 0$.

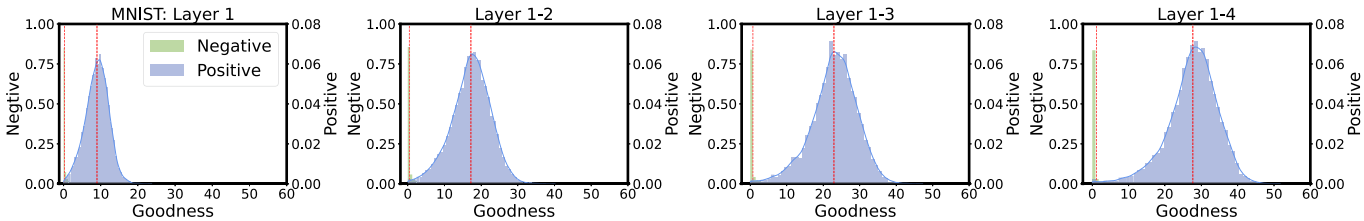


Figure 2: The distribution of MNIST validation data (Green: Negative data; Blue: Positive data) and the corresponding mean values (red vertical lines) in the Forward-Forward algorithm. The distance between the mean values of the negative and positive samples/distributions increases as we consider more layers.

2.2 *Inference Based on One-Pass Procedure*

In this part, we focus on the one-pass inference procedure. In general, one-pass inference is more efficient compared to the previous procedure, as instead of passing the test sample with all labels through the network, a test sample is passed through the network once with a neutral label. For our lightweight one-pass inference procedure, instead of only one softmax layer at the head of the network, we train one softmax layer for each hidden layer. The input of each softmax layer is the concatenated activity vectors of all the previous hidden layers. The softmax layers are trained based on train samples with neutral labels (and after all the hidden layers are trained based on positive and negative training samples).

Figure 1(b) shows our lightweight one-pass inference scheme. The typical configuration of a softmax layer includes calculating the logits and applying the softmax activation function. As the size of the input to the softmax layer increases in each layer by concatenation of activity vectors, the size of the Logit Calculation block correspondingly increases at each layer. These details are visually represented in the figure. After each layer, the activity is passed to its respective softmax layer. We inspect the confidence based on the calculated logits for that softmax layer. As illustrated in the figure, for test samples with less complexity, the first layer(s) is sufficient, but for more complex samples, we continue the forward pass.

To capture confidence, we incorporate one neuron with a sigmoid activation function in each softmax layer, which is trained based on the validation dataset. The input to this neuron is the maximum logit value for that softmax layer. Similarly, considering the fully connected network that is already trained by the Forward-Forward algorithm, the input of validation set $\boldsymbol{X}^{(l)}$ and the constructed binary label $\bar{\boldsymbol{y}}^{(l)}$ are leveraged for training and capturing the confidence for layer l. We define the ground-truth label $\bar{\boldsymbol{y}}_i^{(l)} = 1$ if the input sample $\boldsymbol{X}_i^{(l)}$ is correctly classified by the layer and $\bar{\boldsymbol{y}}_i^{(l)} = 0$ otherwise. As discussed, the input to the neuron $\sigma(\boldsymbol{w}^{(l)} \cdot \boldsymbol{z}^{(l)} + b^{(l)})$ is the logit value $\boldsymbol{z}^{(l)}$, where $\boldsymbol{w}^{(l)}$ and $b^{(l)}$ are the weight and the bias of the single neuron to capture confidence. We consider the Binary Cross-Entropy loss function, with $\bar{\boldsymbol{y}}_i^{(l)}$ as the label for the ith sample. Alternatively, we can consider the maximum logit value and calculate and set the bias values based on the mean and standard deviation of the input values (of confidence neurons) for the validation set.

The computational overhead for the one-pass procedure is slightly different from the previous part. First, we do not pass the test sample M times through the network. Second, instead of calculating the goodness of the network's layers for inference, we pass the output of the layers (activity vector) to the softmax layer. However, calculating the activity of each layer is similar to the previous procedure (shown as C_i^a for layer i). We show the required computations for the ith softmax layer with C_i^s. The overall computation required for a forward pass on layer i is $C_i^a + C_i^s$. Assuming our network has N hidden layers, making an inference operation by the approach in [15], requires $C_{OP} = \sum_{i=1}^{N} C_i^a + C_N^s$ computational overhead.

We show the computation required for one lightweight one-pass inference operation by $C_{Light-OP}$. Using our lightweight inference scheme, if we present the probability of completing the inference operation at layer i by p_i, then we can show the expected computational overhead for making an inference for one test sample as follows:

$$\begin{aligned}\mathbb{E}(C_{Light-OP}) &= \sum_{i=1}^{N}\left(\sum_{j=1}^{i}(C_j^a + C_j^s)\right)\cdot p_i \\ &= \sum_{j=1}^{N}(C_j^a + C_j^s)\cdot\left(1 - \sum_{i=1}^{j-1} p_i\right),\end{aligned} \tag{2}$$

where $\sum_{i=1}^{N} p_i = 1$. In the worst-case scenario, i.e., when $i < N$, p_i is equal to zero, and for $i = N$, p_i is equal to one, the computational overhead is $\sum_{i=1}^{N}(C_i^a + C_i^s)$, which is slightly $(\sum_{i=1}^{N-1} C_i^s)$ worse than the approach in [15]. However, as in most datasets, we have samples with variant complexities, the worst-case scenario is not very likely. The first few layers of the model are sufficient for the majority of samples. We experimentally demonstrate this in Section 3. On the other hand, C_i^a is usually significantly larger than C_i^s because, in most cases, the number of neurons n in each layer is much larger than the number of classes M. Therefore, even if a small proportion of samples use fewer layers, C_i^s overheads can be compensated.

3 Evaluation

3.1 *Experimental Setup*

3.1.1 *Datasets*

To evaluate our lightweight inference scheme, we consider the MNIST dataset of handwritten digits [23] and the CIFAR-10 dataset of object recognition [21]. In addition, we also evaluate our proposed scheme in the context of two real-world medical applications, namely, epilepsy monitoring and seizure detection using wearable technologies based on the CHB-MIT Scalp electroencephalogram (EEG) Dataset [50] and cardiac arrhythmia classification using wearable technologies based on the MIT-BIH Arrhythmia Electrocardiogram (ECG) Dataset [28], where complexity overhead/energy consumption is a major constraint to ensure real-time and long-term monitoring in ambulatory settings.

MNIST [23]: The MNIST dataset contains handwritten digits, which are 28×28 grayscale images in 10 different classes (one for each of the 10 digits). We used 50,000 images for training, 10,000 images for validation, and 10,000 images for testing in the inference.

CIFAR-10 [21]: The CIFAR-10 dataset is a Computer Vision dataset for object recognition, which consists of 32×32 color images in 10 different classes. We used 45,000 images for training, 10,000 images for validation, and 5,000 images for testing in the inference.

Table 1: Error (%) Evaluation of Our Lightweight Inference Scheme

Dataset	Layers	Forward-Forward [MP] [15]		Forward-Forward [OP] [15]		PEPITA [PT] [7]	
		MP	Light-MP	OP	Light-OP	PT	Light-PT
MNIST [23]	2	1.45±0.05	**1.43**±0.04	1.45±0.05	**1.23**±0.03	2.19±0.04	**1.90**±0.07
	3	**1.40**±0.01	**1.40**±0.02	1.45±0.09	**1.06**±0.02	4.98±0.28	**4.85**±0.09
	4	**1.51**±0.01	**1.51**±0.02	1.53±0.10	**1.02**±0.07	-	-
	5	1.60±0.05	**1.53**±0.01	1.57±0.03	**1.09**±0.09	-	-
	6	1.62±0.01	**1.55**±0.02	1.57±0.05	**0.93**±0.04	-	-
CIFAR-10 [21]	2	50.99±0.09	**46.17**±0.27	47.42±0.26	**46.56**±0.25	49.52±0.36	**48.94**±0.30
	3	50.59±0.09	**46.12**±0.36	47.42±0.12	**46.41**±0.32	49.30±0.29	**48.54**±0.25
	4	50.65±0.11	**46.05**±0.30	47.78±0.37	**46.25**±0.37	-	-
	5	50.37±0.46	**45.43**±0.03	48.01±0.17	**46.30**±0.23	-	-
	6	50.75±0.28	**45.83**±0.34	47.67±0.23	**46.06**±0.40	-	-
CHB-MIT [50]	2	37.54±0.53	**36.22**±0.40	36.79±0.26	**31.38**±0.54	36.08±1.44	**31.91**±1.60
	3	37.54±0.55	**34.65**±0.93	36.22±0.55	**28.86**±0.80	34.98±1.88	**31.20**±0.77
	4	39.62±0.55	**34.84**±1.45	37.98±0.98	**28.23**±0.08	-	-
	5	38.74±0.18	**32.29**±0.82	37.29±1.64	**25.15**±0.17	-	-
	6	39.18±0.35	**32.07**±0.53	37.29±0.72	**24.02**±0.23	-	-
MIT-BIH [28]	2	10.82±0.63	**10.06**±0.44	10.44±0.08	**10.13**±0.19	16.05±0.41	**13.98**±0.50
	3	10.29±0.16	**9.68**±0.34	10.82±0.66	**10.68**±0.29	15.19±0.40	**14.31**±0.21
	4	10.74±0.97	**10.07**±0.21	10.86±0.13	**10.54**±0.14	-	-
	5	10.87±0.18	**9.84**±0.25	10.93±0.61	**10.11**±0.21	-	-
	6	10.66±0.04	**9.54**±0.41	10.59±0.39	**9.99**±0.12	-	-

CHB-MIT Scalp EEG Dataset [50]: This dataset contains EEG recordings, collected from 22 patients with epilepsy (5 males and 17 females). The recordings are grouped into 23 cases and are divided into seizure and non-seizure classes. To be consistent with wearable IoT devices for real-time seizure monitoring [52], only the data corresponding to two channels, i.e., T7F7 and T8F8, are considered for the classification. The data corresponding to patients 6, 14, and 16 is eliminated due to their very short-lasting seizures. We use 70%, 15%, and 15% of the dataset for training, validation, and test process, respectively.

MIT-BIH Arrhythmia Dataset [28]: This dataset encompasses ECG recordings from 47 different patients with cardiovascular problems. Five different types of arrhythmias are categorized by beat annotations [2]. We use the pre-processed data provided in [20]. We consider 70% of data for training, 15% for validation, and 15% to test the model.

3.1.2 Baselines

As discussed previously, we apply our proposed lightweight inference in the context of three state-of-the-art techniques, namely:

- Forward-Forward, Multi-Pass (MP), [15]: This is the implementation of the original proposal in [15], where the inference process requires several (as many as the number of classes) forward passes, hence referred to as Multi-Pass (MP).[2]
- Forward-Forward, One-Pass (OP), [15]: This is the implementation of the efficient inference proposal in [15], where the inference process requires only one forward pass, hence referred to as One-Pass (OP), which is built on top of the MP implementation.
- PEPITA (PT), [7]: This is the implementation of the algorithm proposed in [7], which is referred to as PEPITA (PT), where the inference process requires one forward pass. The PEPITA implementation by [7][3] only supports up to 3 hidden layers [40].

[2] https://github.com/loeweX/Forward-Forward
[3] https://github.com/GiorgiaD/PEPITA

3.1.3 Implementation Details

Our proposed lightweight inference scheme is implemented in PyTorch [37]. We have trained all three state-of-the-art algorithms and tested our lightweight inference scheme on the server of 2×16-core Intel(R) Xeon(R) Gold 6226R (Skylake) Central Processing Units (CPUs) and 1 NVIDIA Tesla T4 Graphics Processing Cards (GPUs).

For training MP and OP, we consider several scenarios for fully connected networks with a maximum of 6 hidden layers and a maximum of 2,000 neurons for each layer. We also consider the model used by [15] (4 hidden layers and 2000 neurons for each layer) as the default model in MP and OP. We use the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001. We set the batch size to 100 for the MNIST, CIFAR-10, and MIT-BIH datasets. For the CHB-MIT dataset, the personalized model was trained with batch gradient descent. As for other training parameters, we have considered the values according to the Forward-Forward algorithm.

For training PT, we consider the model used by [53] (3 hidden layers and 1024 neurons for each layer) as the default model in PT. We use a momentum optimizer. We set the batch size to 64 for MNIST, CIFAR-10, and MIT-BIH datasets. We set the number of epochs to 100 for all four datasets in MP, OP, and PT. After training, we extract the confidence threshold for each layer based on the validation data.

The datasets in our experiments are balanced, and error, i.e., the total number of incorrectly classified inputs divided by the total number of inputs, is used as the metric of classification performance. We use Multiply–Accumulate Operations (MACs) to evaluate the complexity of inference processes. In addition, each test sample exploits the corresponding number of layers used in lightweight inference. We use the "Mean Layers" to capture the average/mean number of layers used for all the test samples.

3.2 Results

In this section, we evaluate our lightweight inference scheme in terms of prediction performance/error and computational complexity.

Table 2: Complexity (MACs) Improvement of Our Lightweight Inference Scheme (Mean Layers is shown in parentheses)

Dataset	Layers	Forward-Forward [MP] [15]		Forward-Forward [OP] [15]		PEPITA [PT] [7]	
		MP	Light-MP	OP	Light-OP	PT	Light-PT
MNIST [23]	2	5.31M	**2.18M** (1.18)	5.35M	**1.57M** (1.01)	1.76M	**0.95M** (1.19)
	3	9.12M	**2.56M** (1.28)	9.18M	**1.61M** (1.03)	2.76M	**1.14M** (1.37)
	4	12.94M	**2.95M** (1.38)	13.01M	**1.65M** (1.04)	-	-
	5	16.75M	**3.29M** (1.47)	16.85M	**1.69M** (1.05)	-	-
	6	20.57M	**3.67M** (1.57)	20.68M	**1.73M** (1.06)	-	-
CIFAR-10 [21]	2	9.67M	**7.76M** (1.49)	9.71M	**7.47M** (1.42)	4.00M	**3.25M** (1.25)
	3	13.49M	**9.10M** (1.85)	13.54M	**8.90M** (1.79)	5.00M	**3.43M** (1.43)
	4	17.30M	**10.31M** (2.16)	17.38M	**10.34M** (2.16)	-	-
	5	21.12M	**11.55M** (2.49)	21.21M	**11.80M** (2.54)	-	-
	6	24.93M	**12.62M** (2.72)	25.05M	**13.13M** (2.89)	-	-
CHB-MIT [50]	2	5.77M	**4.54M** (1.67)	5.80M	**3.26M** (1.33)	2.00M	**1.17M** (1.16)
	3	9.58M	**6.26M** (2.12)	9.64M	**4.52M** (1.66)	3.00M	**1.25M** (1.25)
	4	13.39M	**8.39M** (2.68)	13.47M	**5.62M** (1.95)	-	-
	5	17.21M	**10.19M** (3.16)	17.31M	**6.62M** (2.21)	-	-
	6	21.02M	**11.02M** (3.38)	21.14M	**7.43M** (2.42)	-	-
MIT-BIH [28]	2	4.17M	**1.07M** (1.18)	4.21M	**0.71M** (1.08)	1.18M	**0.44M** (1.25)
	3	7.98M	**1.49M** (1.29)	8.04M	**1.02M** (1.16)	2.18M	**0.73M** (1.54)
	4	11.80M	**1.99M** (1.43)	11.88M	**1.31M** (1.24)	-	-
	5	15.61M	**2.01M** (1.43)	15.71M	**1.59M** (1.30)	-	-
	6	19.43M	**3.06M** (1.71)	19.53M	**1.84M** (1.38)	-	-

Table 3: Execution Time (in millisecond) for Our Scheme Against the Forward-Forward (FF) Algorithm

Dataset	Layers	FF [MP]		FF [OP]	
		MP	Light-MP	OP	Light-OP
MNIST	2	6.58	**2.24**	0.72	**0.30**
	3	14.77	**3.25**	1.40	**0.31**
	4	22.81	**3.39**	2.30	**0.32**
	5	29.70	**4.22**	3.06	**0.34**
	6	36.73	**5.52**	3.75	**0.36**
CIFAR-10	2	14.85	**10.38**	1.39	**1.15**
	3	22.32	**13.19**	2.20	**1.47**
	4	29.57	**16.38**	2.98	**1.76**
	5	35.83	**17.45**	3.36	**2.01**
	6	42.38	**19.50**	4.31	**2.24**
CHB-MIT	2	1.55	**1.19**	0.77	**0.53**
	3	3.42	**1.94**	1.42	**0.76**
	4	4.73	**2.85**	2.28	**1.01**
	5	6.02	**3.51**	3.44	**1.21**
	6	7.33	**3.74**	3.72	**1.36**
MIT-BIH	2	2.56	**0.88**	0.58	**0.23**
	3	6.83	**1.36**	1.18	**0.28**
	4	11.26	**1.83**	1.95	**0.34**
	5	14.46	**1.87**	2.88	**0.41**
	6	18.03	**2.90**	3.49	**0.44**

3.2.1 *Performance Evaluation*

In this section, we investigate the performance of our lightweight inference scheme based on MP, OP, and PT. We consider the networks with the same number of neurons (2000 neurons for MP and OP; 1024 neurons for PT) in each hidden layer. We adjust the number of hidden layers from 2 to 6 for MP and OP. We adjust the number of hidden layers from 2 to 3 for PT because PT only supports up to 3 hidden layers [40].

Table 1 shows the error comparison between these three state-of-the-art techniques and our lightweight inference scheme. For MP, the Light-MP error is on par with the MP error for MNIST, CIFAR-10, CHB-MIT, and MIT-BIH datasets in different layer settings. Similarly, for OP, the Light-OP error is on par with the OP error for all four datasets. Finally, for PT, the Light-PT error is also comparable to the PT error for all four datasets.[4] In summary, our lightweight inference achieves a comparable classification error and even a smaller error in some cases.

[4] Note that, as shown in [53], the overall error marginally increases with the number of layers.

3.2.2 *Complexity Evaluation*

In this section, we evaluate the complexity of our lightweight inference scheme based on MP, OP, and PT. First, we refer to the same experimental settings in Table 1 and report the MACs and Mean Layers (shown in parentheses) used by our lightweight inference scheme in Table 2. For Light-MP, Light-OP, and Light-PT evaluated on MNIST, CIFAR-10, CHB-MIT, and MIT-BIH datasets, the value of Mean Layers used in our lightweight inference scheme is always significantly lower than the number of the network layers. Similarly, the MACs of Light-MP, Light-OP, and Light-PT are always significantly lower than the MACs of MP, OP, and PT, respectively, for all four datasets in different layer settings. However, as the number of network layers increases from 2 to 6 for MP and OP, and from 2 to 3 for PT, the Mean Layers used tend to increase but the relative percentage of layers used decreases. Moreover, we have conducted experiments to show the actual execution time reduction of our lightweight inference scheme, on a MacBook Pro with the Apple M1 pro CPU and 32 GB of RAM. Table 3 presents the improvement of our scheme against the Forward-Forward algorithm. Taking MNIST with 4 layers as an example, the execution time of FF [MP], i.e., 22.81 ms, is reduced by a factor of 6.7× to 3.39 ms, and the execution time of FF [OP], i.e., 2.30 ms, by a factor of 7.2× to 0.32 ms.

Hence, considering Table 1, Table 2, and Table 3, we conclude that our lightweight inference scheme improves computational efficiency by reducing the number of layers used in inference, with a comparable classification error in all cases.

3.2.3 *Detailed Analysis and Discussion*

To show the relevance of our lightweight inference scheme for the new generation of forward-only techniques, we further analyze our proposed lightweight inference scheme for neural networks trained based on Backpropagation (BP), MP, OP, and PT.

First, we investigate the distance between the negative data and the positive data for the four datasets considered in this work. Here, we calculate the mean values of the negative data and positive data for

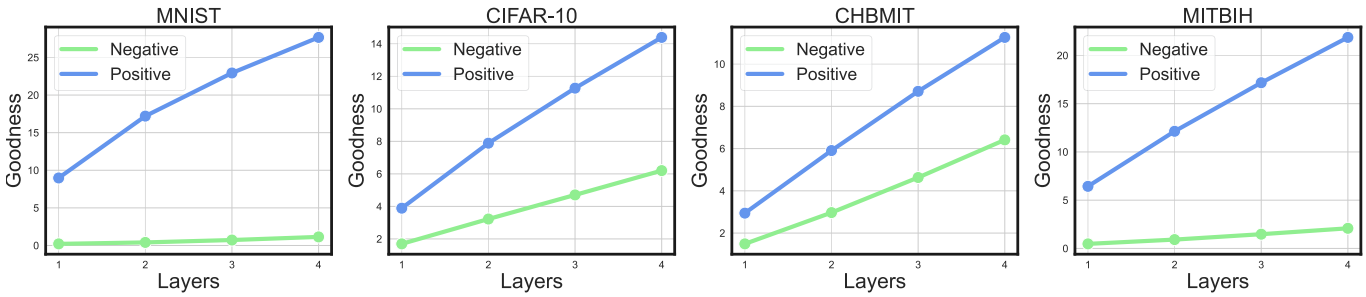


Figure 3: The mean values of negative data and positive data in the Forward-Forward algorithm (MP).

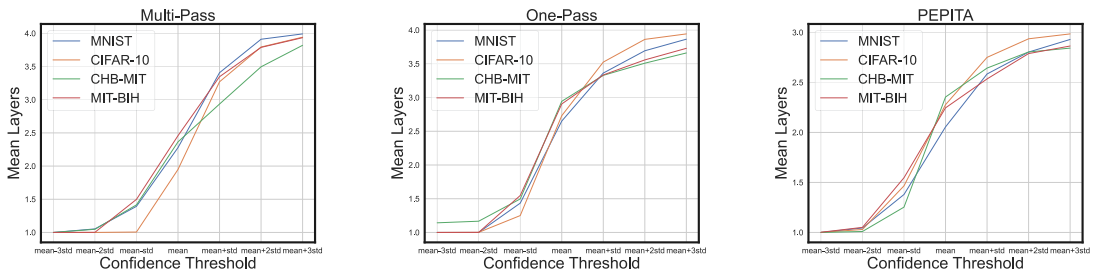


Figure 4: The average/mean number of layers used by our lightweight inference schemes versus confidence threshold.

MNIST, CIFAR-10, CHB-MIT, and MIT-BIH datasets. These mean values are in consecutive layers based on MP. As shown in Fig. 3, the distance between the mean values of the negative and positive data increases as we consider more layers. The figure shows that a higher confidence level is attained as more layers are taken into account.

Then, we investigate the relationship between the Mean Layers and the confidence threshold to show the computational efficiency of our proposed lightweight inference scheme. We conduct seven experiments separately with different confidence thresholds for the lightweight inference schemes based on MP, OP, and PT for their default model. At each layer, we extract the *mean* and *std* from validation data and then use the threshold for confidence measurement. The different confidence thresholds range from $mean - 3 \cdot std$ to $mean + 3 \cdot std$, where $mean - std$ is the default setting. As shown in Fig 4, if the confidence threshold increases, the mean number of layers used increases accordingly. The reason is that a higher confidence threshold means that fewer test samples in the inference process are regarded as confident and need to be passed to the next layers. Hence, our lightweight inference scheme provides the possibility to achieve different levels of computational efficiency for different confidence thresholds.

We also investigated the probability of the number of layers used for Light-MP, Light-OP, and Light-PT. Towards this, we extract the probability of each layer used for all the test data based on confidence. We refer to the same experimental settings in Table 1. Fig 5 shows the probability of the number of layers used in the proposed schemes. We use the lightest color to represent one layer and the darkest color to represent six layers. For the MNIST dataset, more than 77% of the test samples are confident just at the first layer, for Light-MP, Light-OP, and Light-PT considering different number of layers. For CIFAR-10 datasets, more than 50% of the test samples are confident at the first layer for Light-MP, Light-OP, and Light-PT with different numbers of layers. For CHB-MIT and MIT-BIH datasets, respectively, more than 32% and 69% of the test samples are confident at the first layer for our proposed lightweight inference scheme based on MP, OP, and PT. Fig 5 shows that by considering confidence, our lightweight inference scheme decreases the mean number of layers used in inference.

Finally, let us apply the proposed lightweight inference scheme for a network trained based on BP and compare the error against our lightweight inference scheme for networks trained based on FF. Although early-exit strategies at the intermediary layers has been studied in deep neural networks trained based on BP, such strategies generally degrade the classification performance, according to our experiments shown in Table 4. Our lightweight inference scheme, on the other hand, makes use of the inherent nature of forward-only algorithms, such as FF, that extract relevant features (for the final prediction outcome) early on in deep neural networks, e.g., local energy-based methods. Therefore, our lightweight inference scheme reduces the inference time and complexity of networks trained based on forward-only algorithms, without any major degradation in the classification performance, while such strategies may deteriorate the performance of networks trained using BP.

Table 4: Our Lightweight Inference Scheme for BP

Dataset	BP[42]		FF[15]	
	BP	Light-BP	FF	Light-FF
MNIST [23]	1.33±0.04	5.21±0.69	1.53±0.10	1.02±0.07
CIFAR-10 [21]	43.62±0.33	54.22±0.11	47.78±0.37	46.25±0.37
CHB-MIT [50]	25.63±0.40	40.69±0.76	37.98±0.98	28.23±0.08
MIT-BIH[28]	8.25±0.46	11.55±0.09	10.86±0.13	10.54±0.14

4 Related Work

Over the past years, various biologically plausible alternatives have been proposed to address the inherent biologically-implausible nature of back-propagation, e.g., [26, 32, 25, 49, 48, 56, 13, 44, 33, 30, 31, 15, 19, 24, 7, 5, 14].

In [15], Hinton proposes one of the recent alternatives for back-propagation, called the Forward-Forward algorithm, which is suitable to be implemented on low-power analog hardware. In the Forward-Forward algorithm, the forward and backward passes of back-propagation are replaced by two forward passes. This algorithm

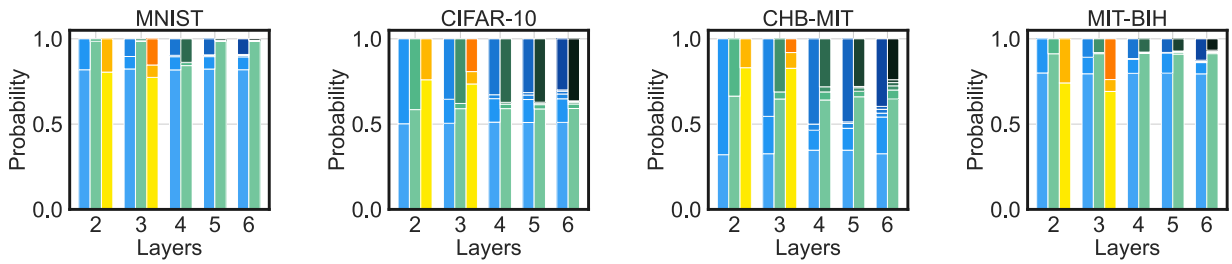


Figure 5: Probability of the number of layers used, for 2–6 layer networks, four datasets, and three algorithms (MP, OP, and PT). Blue: MP; Green: OP; Yellow: PT. The lightest color represents 1 layer and the darkest color represents 6 layers.

addresses, at least partially, four well-known problems in learning with backpropagation, i.e., weight transport [12], non-local weight update [57], frozen activity [27], and update locking [6, 18].

In [7], Dellaferrera and Kreiman propose a similar scheme to the Forward-Forward learning, called PEPITA, executing the forward pass twice. In the second forward pass, the input data is modified by the output error of the first forward pass, while the Forward-Forward scheme does not use any feedback. In [19], Journe et. al also consider two forward passes for the learning procedure and proposes an unsupervised Hebbian-based learning algorithm. Moreover, several very recent studies propose to use the modified versions of the Forward-Forward algorithm for the learning procedure, e.g., [34, 59, 41, 3, 8].

Lightweight inference has also been considered in several state-of-the-art studies. Several lightweight inference schemes have been developed both for classical DNNs [36, 55, 35, 11, 4, 17, 1] and for classical feature-based machine-learning techniques [51, 9, 58, 10, 16]. For instance, early exit and cascade networks, which incorporate mechanisms for making predictions at intermediate stages of processing, have been proposed to reduce the inference time and energy [54, 29, 47, 22, 43]. However, lightweight inference for the forward-only techniques has not been addressed to date. In this work, we bridge this gap and develop the first lightweight inference scheme for neural networks trained based on forward-only techniques.

5 Conclusions

In this paper, we propose a lightweight inference scheme based on the new generation of forward-only techniques, measuring the confidence at each layer. This lightweight inference scheme aims for computational efficiency in the inference of these forward-only DNNs. We have evaluated our lightweight inference scheme on the Forward-Forward Multi-Pass [15], One-Pass [15], and PEPITA [7], based on the MNIST, CIFAR-10, CHB-MIT, and MIT-BIH datasets. The results show that our lightweight inference scheme achieves computational efficiency by decreasing the number of layers used in inference, with a comparable classification error.

Our work aims at developing resource/energy-efficient inference mechanisms for modern AI/ML, towards a new generation of sustainable AI/ML techniques. Moreover, our proposed inference scheme enables the adoption of machine learning techniques by resource-constrained wearable devices and enables real-time and long-term health monitoring, on a personalized basis. As such, our work also contributes to the realization of the "precision medicine" paradigm. To demonstrate this, we have considered two health applications to showcase our proposed inference scheme even in the context of real-world medical applications.

Limitations. Despite their recent success, as we also show for two real-world health applications, the state-of-the-art forward-only techniques are only in their infancy and are yet to be developed to be able to tackle the most complex/challenging learning tasks in the domain. As a result, our proposed inference scheme is also limited to the extent the state-of-the-art forward-only training techniques apply.

Broader Ethical Impact. There are no potential ethical impacts and future societal implications/consequences to be highlighted here.

Acknowledgements

This research has been partially supported by the Swedish Wallenberg AI, Autonomous Systems and Software Program (WASP), the Swedish Research Council (VR), Swedish Foundation for Strategic Research (SSF), the ELLIIT Strategic Research Environment, and the European Union (EU) Interreg Program. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[1] J. Antorán, J. Allingham, and J. M. Hernández-Lobato. Depth uncertainty in neural networks. *Advances in neural information processing systems*, 33:10620–10634, 2020.

[2] V. Arlington. Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms. *ANSI-AAMI EC57*, 1998.

[3] S. Baghersalimi, A. Amirshahi, T. Teijeiro, A. Aminifar, and D. Atienza Alonso. Layer-wise learning framework for efficient dnn deployment in biomedical wearable systems. *[Proceedings of IEEE BSN 2023]*, 2023.

[4] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017.

[5] K. H. R. Chan, Y. Yu, C. You, H. Qi, J. Wright, and Y. Ma. Redunet: A white-box deep network from the principle of maximizing rate reduction. *The Journal of Machine Learning Research*, 23(1):4907–5009, 2022.

[6] W. M. Czarnecki, G. Świrszcz, M. Jaderberg, S. Osindero, O. Vinyals, and K. Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *International Conference on Machine Learning*, pages 904–912. PMLR, 2017.

[7] G. Dellaferrera and G. Kreiman. Error-driven input modulation: solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pages 4937–4955. PMLR, 2022.

[8] K. Flügel, D. Coquelin, M. Weiel, C. Debus, A. Streit, and M. Götz. Feed-forward optimization with delayed feedback for neural networks. *arXiv preprint arXiv:2304.13372*, 2023.

[9] F. Forooghifar, A. Aminifar, L. Cammoun, I. Wisniewski, C. Ciumas, P. Ryvlin, and D. Atienza. A self-aware epilepsy monitoring system for real-time epileptic seizure detection. In *ACM/Springer Mobile Networks and Applications (MONET)*. ACM/Springer, 2019.

[10] F. Forooghifar, A. Aminifar, T. Teijeiro, A. Aminifar, J. Jeppesen, S. Beniczky, and D. Atienza. Self-aware anomaly-detection for epilepsy monitoring on low-power wearable electrocardiographic devices. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–4. IEEE, 2021.

[11] P. Georgiev, S. Bhattacharya, N. D. Lane, and C. Mascolo. Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–19, 2017.

[12] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[13] J. Guerguiev, T. P. Lillicrap, and B. A. Richards. Towards deep learning with segregated dendrites. *Elife*, 6:e22901, 2017.

[14] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

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

[16] B. Huang, A. Abtahi, and A. Aminifar. Lightweight machine learning for seizure detection on wearable devices. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–2. IEEE, 2023.

[17] B. Huang, R. Zanetti, A. Abtahi, D. Atienza, and A. Aminifar. Epilepsynet: Interpretable self-supervised seizure detection for low-power wearable systems. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–5. IEEE, 2023.

[18] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017.

[19] A. Journé, H. G. Rodriguez, Q. Guo, and T. Moraitis. Hebbian deep learning without feedback. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8gd4M-_Rj1.

[20] M. Kachuee, S. Fazeli, and M. Sarrafzadeh. Ecg heartbeat classification: A deep transferable representation. In *2018 IEEE international conference on healthcare informatics (ICHI)*, pages 443–444. IEEE, 2018.

[21] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5 (4):1, 2010.

[22] S. Laskaridis, A. Kouris, and N. D. Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 1–6, 2021.

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

[24] J. H. Lee, S. Haghighatshoar, and A. Karbasi. Exact gradient computation for spiking neural networks via forward propagation. In *International Conference on Artificial Intelligence and Statistics*, pages 1812–1831. PMLR, 2023.

[25] Q. Liao, J. Leibo, and T. Poggio. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[26] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.

[27] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6): 335–346, 2020.

[28] R. Mark, P. Schluter, G. Moody, P. Devlin, and D. Chernoff. An annotated ecg database for evaluating arrhythmia detectors. In *IEEE Transactions on Biomedical Engineering*, volume 29, pages 600–600, 1982.

[29] Y. Matsubara, M. Levorato, and F. Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.

[30] A. Meulemans, M. Tristany Farinha, J. García Ordóñez, P. Vilimelis Aceituno, J. Sacramento, and B. F. Grewe. Credit assignment in neural networks through deep feedback control. *Advances in Neural Information Processing Systems*, 34:4674–4687, 2021.

[31] B. Millidge, A. Tschantz, and C. L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.

[32] A. Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[33] A. Nøkland and L. H. Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pages 4839–4850. PMLR, 2019.

[34] A. Ororbia and A. A. Mali. The predictive forward-forward algorithm. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 45, 2023.

[35] P. Panda, A. Sengupta, and K. Roy. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 475–480. IEEE, 2016.

[36] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 124–132. IEEE, 2015.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[38] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training, 2021.

[39] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7): 18–28, 2022. doi: 10.1109/MC.2022.3148714.

[40] D. P. Pau and F. M. Aymone. Suitability of forward-forward and pepita learning to mlcommons-tiny benchmarks. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2023.

[41] M. Psenka, D. Pai, V. Raman, S. Sastry, and Y. Ma. Representation learning via manifold flattening and reconstruction. *Journal of Machine Learning Research*, 25(132):1–47, 2024.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[43] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, 26(4): 1992–2004, 2017.

[44] J. Sacramento, R. Ponte Costa, Y. Bengio, and W. Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

[45] K. Saladin. *Anatomy & Physiology: The Unity of Form and Function*. McGraw-Hill Higher Education, 2009. ISBN 9780071283410. URL https://books.google.se/books?id=pByBGQAACAAJ.

[46] S. Savazzi, V. Rampa, S. Kianoush, and M. Bennis. An energy and carbon footprint analysis of distributed and federated learning. *IEEE Transactions on Green Communications and Networking*, 2022.

[47] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5): 954–966, 2020.

[48] B. Scellier and Y. Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[49] M. Schiess, R. Urbanczik, and W. Senn. Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites. *PLoS computational biology*, 12(2):e1004638, 2016.

[50] A. H. Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.

[51] D. Sopic, A. Aminifar, A. Aminifar, and D. Atienza. Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems. In *IEEE Transactions on Biomedical Circuits and Systems*, 2018.

[52] D. Sopic, A. Aminifar, and D. Atienza. e-glass: A wearable system for real-time detection of epileptic seizures. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

[53] R. F. Srinivasan, F. Mignacco, M. Sorbaro, M. Refinetti, A. Cooper, G. Kreiman, and G. Dellaferrera. Forward learning with top-down feedback: Empirical and analytical characterization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=My7lkRNnL9.

[54] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[55] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[56] J. C. Whittington and R. Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

[57] J. C. Whittington and R. Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

[58] R. Zanetti, A. Arza, A. Aminifar, and D. Atienza. Real-time eeg-based cognitive workload monitoring on wearable devices. *IEEE Transactions on Biomedical Engineering*, 69(1):265–277, 2021.

[59] G. Zhao, T. Wang, Y. Li, Y. Jin, C. Lang, and H. Ling. The cascaded forward algorithm for neural network training. *arXiv preprint arXiv:2303.09728*, 2023.