REFORMULATING THE FORWARD-FORWARD ALGORITHM WITH LAYER-WISE SIMILARITY-BASED OBJECTIVES

James Gong[1], *Raymond Luo*[1], *Emma Wang*[1], *Leon Ge*[1], *Bruce Li*[1,2], *Felix Marattukalam*[1], and *Waleed Abdulla*[1]

[1]University of Auckland, New Zealand
[2]Imperial College London, United Kingdom
{hgon777, rluo154, sge524}@aucklanduni.ac.nz,
Felix.Marattukalam@auckland.ac.nz, w.abdulla@auckland.ac.nz
wangem@student.ethz.ch, bruce.li25@imperial.ac.uk

ABSTRACT

Backpropagation underpins the success of artificial neural networks, yet it has critical limitations such as backward locking and global error propagation. The Forward-Forward algorithm has been proposed as a more biologically plausible method that replaces the backward pass with an additional forward pass. Nevertheless, the Forward-Forward algorithm significantly trails backpropagation in accuracy, and its optimal form requires multiple forward passes at inference time, resulting in low inference efficiency. In this work, the Forward-Forward algorithm is reformulated within a similarity-learning paradigm. The proposed algorithm is named *Forward-forward Algorithm Unified with Similarity-based Training* (*FAUST*). *FAUST* incorporates similarity-based objectives and, by caching estimates of class centroids, achieves single-pass inference. On matched MLP and CNN architectures, *FAUST* yields improved accuracy over Forward-Forward baselines and narrows the gap to backpropagation across MNIST, Fashion-MNIST, and CIFAR-10. Applying *FAUST* on ResNet-18 yields competitive results on ImageNette.

I. INTRODUCTION

Artificial Neural Networks (ANNs), being simplified models of the brain, have achieved remarkable success across a wide range of tasks. A core algorithm instrumental to this success is backpropagation (BP) [1]. BP uses the chain rule to compute gradients of losses with respect to weights, facilitating gradient-descent-based optimization. However, BP's reliance on global error signals and non-local updates is not biologically plausible [2]. Further, BP requires storing all intermediate activations during training, which can be restrictive for memory-constrained environments and parallelization. These limitations have motivated the development of alternative biologically inspired training algorithms that rely on local or approximate learning signals and better relate to biological neural networks [3], [4], [5], [6], [7], [8], [9]. One proposed alternative is the Forward-Forward (FF) algorithm [10]. FF eliminates the need for the backward pass to update network parameters, and its local update nature aligns more closely with the principles of Hebbian learning [11] and local plasticity [12]. Its predictive accuracy, however, is lower than that of traditional BP. In this research, we reformulate the core framework of FF by incorporating concepts of similarity learning [13], leveraging an anchor-positive-negative structure to guide local learning dynamics. Our main contributions are as follows:

- We propose an FF-based framework that replaces goodness-based learning with a layer-wise similarity-based objective. Our proposed algorithm is named *Forward-Forward Algorithm Unified with Similarity-based Training* (*FAUST*).
- We show that *FAUST*'s layer-wise learning scheme can be combined with an embedding head to improve cross-layer information retention.
- *FAUST* involves extracting images from each class to globally represent the class across training and inference. We show that training with similarity-based margin loss functions on these fixed class representatives can be more cost-efficient and stable than on random samples. Further, caching the embeddings of the representatives improves inference efficiency. At inference time, *FAUST* requires only a single forward pass. In comparison, FF requires C forward passes, where C is the number of classes.
- We show that *FAUST* demonstrates some improvements over existing BP-free benchmarks on fully connected and convolutional architectures.

The code is available at https://github.com/rl116432/FAUST.

II. RELATED WORK

Forward-Forward algorithm. FF, introduced as a biologically inspired alternative to BP, has driven the development of a variety of extensions. A key limitation of FF is its asymmetric treatment of positive and negative activations. To overcome this vulnerability, SymBa-FF [14] introduces a symmetric contrastive loss function that improves convergence speed. Other FF extensions include predictive FF [15], which combines FF with principles of predictive coding, and Collaborative FF (Collab FF) [16], which enhances information flow across layers. Recent work has also extended FF beyond multi-layer perceptrons (MLPs) to convolutional neural networks (CNNs) [17], [18], [19], [20], [21]. A common challenge in CNNs is defining an appropriate layer-wise training signal analogous to the goodness score in FF. Existing approaches address this by redefining goodness over convolutional feature maps. Examples include aggregating activity across channels or spatial locations and introducing lightweight auxiliary prediction heads attached to intermediate convolutional layers to provide local supervision. These modifications enable FF-style training in CNN backbones while preserving the layer-wise learning structure. Closely related to our work are methods that reframe the FF algorithm within a similarity-learning framework. Forward-Forward with Contrastive Marginal Loss (FFCM) [22] adopts a contrastive learning strategy by applying different augmentations to the same input image and training the encoder using a contrastive marginal loss. Self-Contrastive Forward-Forward (SCFF) [23] constructs contrastive pairs by concatenating identical images as positive samples and different images as negative samples, enabling representation learning without external augmentations. Our proposed approach falls within the same family of methods as these works, integrating the Forward-Forward algorithm into a similarity-learning framework.

Similarity learning. Similarity learning involves training models through similarity metrics to produce embeddings, such that semantically similar inputs are closer together, and dissimilar ones are further apart. Siamese networks with contrastive loss [24] and triplet loss [13] have laid the foundation for this paradigm. Triplet-based training, however, often suffers from slow convergence. This limitation has led to the development of full-comparison losses that leverage multiple negatives per anchor (e.g., $(N+1)$-tuplet loss [25]). Additionally, naive implementations of the triplet and $(N+1)$-tuplet loss functions suffer from high computational complexity: training on all valid triplets has a runtime complexity of $\mathcal{O}(M^3/C)$, where M is the size of the training set, and C is the number of classes. To reduce per-batch computation cost, [26], [27], [28] have proposed methods that approximate a representation for each class. This informative representation attempts to eliminate the need to compare all tuplets. We draw upon similar ideas in our method.

III. THE FORWARD-FORWARD ALGORITHM

FF replaces BP's back-and-forth passes with two forward passes through each layer, one on 'positive' data and one on 'negative' data. In FF, the first C dimensions of the input vector are reserved for label encoding, where C is the number of classes. For a sample from class c, the first C elements are set to a one-hot encoded vector with 1 at index c and 0 elsewhere. Positive data are constructed by assigning the correct one-hot encoded label, and negative data are constructed by assigning incorrect one-hot encoded labels.

During training, positive and negative data are passed through the current layer i, producing feature outputs f_i^{pos} and f_i^{neg}. The 'goodness' score, indicating the likelihood of the current encoding being correct, is computed for both outputs:

$$G_i^{\text{pos}} = ||f_i^{\text{pos}}||^2,\ G_i^{\text{neg}} = ||f_i^{\text{neg}}||^2. \tag{1}$$

The objective is to increase the goodness for positive inputs and reduce it for negative ones. The loss function is defined as

$$\mathcal{L}_i^{\text{FF}} = \frac{1}{2}[\log(1 + e^{\theta - G_i^{\text{pos}}}) + \log(1 + e^{G_i^{\text{neg}} - \theta})], \tag{2}$$

where θ is a tunable threshold shifting the margin between positive and negative activations. The weights in the current layer i are updated via gradient descent, while the outputs f_i^{pos} and f_i^{neg} are normalized and forwarded as inputs to the next layer. This learning mechanism is applied throughout the network, allowing the model to train using only local error signals. In the inference phase, the model evaluates all possible one-hot encodings on an unseen input. The encoding that results in the highest total goodness across all layers is selected as the predicted class. A key limitation of FF is that, at inference time, its optimal form requires C separate forward passes, whereas traditional BP-based models require only a single pass. [10] proposes a more efficient FF variant that uses a single forward pass, but this version yields lower accuracy.

IV. METHODOLOGY

In this research, we redefine the objective in the FF algorithm to a similarity-based objective rather than the goodness score. Instead of the traditional FF formulation's approach of feeding one-hot encoded data through the network twice, we present the model with an *anchor* image, $\mathbf{x}$, a corresponding *positive* image belonging to the same class as the anchor image, $\mathbf{x}^+$, and $N-1$ *negative* images belonging to different classes from the

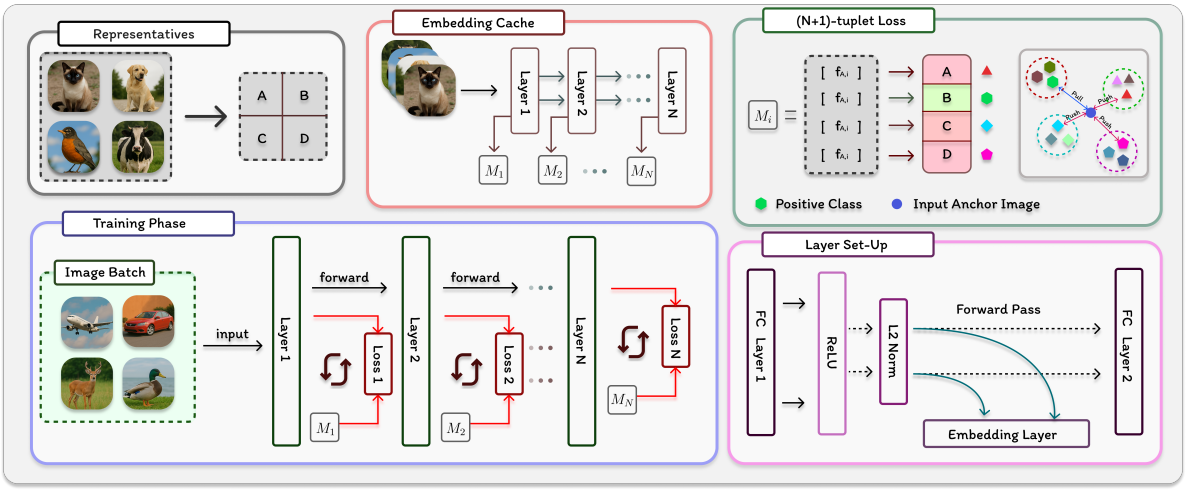


Fig. 1: Overall algorithmic architecture of *FAUST*.

anchor image, $\{\mathbf{x}_i^-\}_{i=1}^{N-1}$. We then apply a $(N+1)$-tuplet loss formulation [25]

$$\mathcal{L}_{tup}(\mathbf{f}, \mathbf{f}^+, \{\mathbf{f}_i^-\}_{i=1}^{N-1}) = \log(1 + \sum_{i=1}^{N-1} \exp(d(\mathbf{f}, \mathbf{f}^+) - d(\mathbf{f}, \mathbf{f}_i^-))), \quad (3)$$

where $d(\mathbf{a}, \mathbf{b})$ is a distance function, and $\mathbf{f} = f(\mathbf{x})$ is a transformation from the input to the embedding space. The Euclidean distance, $d(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}||^2$, is used as the distance metric. We utilize $(N+1)$-tuplet loss rather than traditional triplet loss [13] for its broader scope of comparison, simultaneously maximizing the distance between the anchor and $N-1$ negative samples. This similarity-based loss function incentivizes the network to produce embeddings where the anchor is closer to the positive than the negatives. Subsequently, the loss is calculated based on the embeddings from each layer, and the activations are fed forward to the next layer to be trained similarly (see *Layer Set-Up* in Fig. 1). L2 normalization is applied to ensure stabilized vector magnitudes. Essentially, the layers are trained in a greedy layer-wise fashion without global error propagation.

IV-A. Representatives and cache

For a batch size B, triplet loss [13] with random sampling requires $3B$ forward passes per training step. In comparison, $(N+1)$-tuplet loss incurs a higher computational cost, requiring $(N+1) \times B$ forward passes at each step. To mitigate this overhead, we extract one image from each class to globally represent the class across training and inference (see *Representatives* in Fig. 1). This set of representative images is denoted $\mathcal{R}$, where $|\mathcal{R}| = C$ is the number of classes. Every image within a batch of size B is taken as an anchor. A $(C+1)$-tuplet is formed comprising this anchor image $\mathbf{x}$, the corresponding positive image $\mathbf{x}^+$ from $\mathcal{R}$, and the remaining negative images $\{\mathbf{x}_i^-\}_{i=1}^{C-1} = \mathcal{R} \backslash \{\mathbf{x}^+\}$. In the current approach, class representatives are selected via uniform random sampling as a most rudimentary means of approximating the class centroid. These representatives are fixed throughout training. At the start of each training batch, the embeddings are computed for all the representatives and cached as fixed reference points for the entire batch, eliminating the need for recomputations for each anchor (see *Embedding Cache* and *Training Phase* in Fig. 1). This set-up requires $B+C$ forward passes per training batch, which is strictly less than the $3B$ passes required for triplet loss and maintains a linear runtime complexity.

The use of fixed representatives simplifies the optimization problem from minimizing the loss over all possible tuplet combinations—with variable positives and negatives for a given anchor—to a simpler subproblem involving only fixed positives and negatives. Sec. V-A empirically demonstrates that the use of representatives stabilizes optimization, yielding more consistent gradient directions and reduced training-time variability.

IV-B. Inference

Prior to inference time, each representative image is propagated through the trained network, and its embeddings are cached. The model infers the class of an input by finding the embeddings of the representative image that is closest to the embeddings of the input across all layers. The inferred class is defined as

$$\hat{y}(\mathbf{x}) = \arg\min_{c \in \mathcal{C}} \sum_{i=1}^{L} ||f_i(\mathbf{x}) - \hat{\mathbf{f}}_i^r||_2, \quad (4)$$

where $\hat{\mathbf{f}}_i^r$ denotes the representative image for class c and layer i. This inference mechanism is considerably more efficient than that of traditional FF. FF's optimal form requires C separate forward passes, one for each candidate class. By using cached embeddings, *FAUST* requires only a single forward pass per test input.

IV-C. Embedding layer

In the simplest form of *FAUST* that does not contain embedding layers, each layer is trained by applying a local similarity-based objective to the activations; the same activations are then forwarded to deeper layers. This approach incentivizes each layer to compress away variability that is not immediately favorable to the current local objective, thereby potentially limiting the information available for subsequent feature formations in deeper layers. To overcome this limitation, we introduce a lightweight embedding head that maps the post-activation features into a lower-dimensional embedding space. The local loss function is then applied to the embeddings in the embedding space, while the pre-embedding activations are forwarded to the next layer. This design decouples cross-layer communication from local layer-wise optimization (see *Layer Set-Up* in Fig. 1). Concretely, the activations are computed as

$$\mathbf{g}_{i+1} = \phi(W_1^{i+1}\mathbf{g}_i), \tag{5}$$

whereas the embeddings used by the loss function are

$$\mathbf{f}_{i+1} = W_2^{i+1}\mathbf{g}_{i+1}, \tag{6}$$

where W_1^{i+1} is the layer's weight matrix, and W_2^{i+1} is a trainable linear projection to the embedding space. Importantly, $\mathbf{f}_{i+1}$ is not passed to the next layer. Sec. V-B shows empirically that the embedding layers improve information retention.

IV-D. Biological motivation

FAUST inherits FF's key features for biological plausibility: learning is driven by layer-wise signals and does not require global error backpropagation through the full network. This feature alleviates the backward-locking constraint of BP and makes the learning rule amenable to parallel and asynchronous layer-wise updates. Further, since *FAUST* does not rely on propagating gradients through a global backward pathway, it avoids BP's weight-transport requirement of having precisely matched forward and feedback weights. Nevertheless, FF and *FAUST* remain only biologically *inspired*—both rely on constructing explicit negative examples and supervised sampling mechanisms that do not have a direct biological counterpart.

IV-E. Positioning with respect to prior work

FAUST reinterprets FF within a similarity-learning paradigm; the mechanisms of related approaches, FFCM and SCFF, are presented in Fig. 2. FFCM and SCFF typically learn representations prior to fitting a separate classifier for prediction, whereas *FAUST* performs classification directly in the representation space by matching inputs to cached class representatives. FFCM forms positives via two augmented views of the same image and uses samples from different classes as negatives. In comparison, *FAUST* uses the fixed representative image of the same class as the input as the positive sample and the representatives of different classes as the negative samples. SCFF operates without labels or external augmentations by constructing positives and negatives through the concatenation of identical versus different images.

V. RESULTS

We conduct the following ablation studies on *FAUST* to isolate the effect of each of our design choices.

Effect of triplet vs. $(N+1)$-tuplet loss. To assess the impact of shifting from triplet to tuplet supervision, we compare two ablative variants—*FAUST-random triplet* and *FAUST-random tuplet*—that utilize triplet loss and $(N+1)$-tuplet loss, respectively, with random sampling of the positive and negative images for each anchor.

Effect of random sampling vs. fixed class representatives. We then assess the effectiveness of adopting the representative-based approach by comparing *FAUST-random tuplet* to *FAUST*.

Effect of embedding layer. *FAUST* uses a linear embedding layer on each layer's post-activation features to generate the embeddings used for classifying inputs. To demonstrate the importance of this embedding layer, we compare *FAUST* against a variant—*FAUST-no embedding*—that has the embedding head removed.

V-A. Representatives stabilize optimization

To evaluate whether using representatives yields more stable optimization than random sampling in our local learning set-up, we measure the consistency of gradient directions during training. Specifically, for each layer l and update step t, we compute the cosine similarity between the gradients from two consecutive updates,

$$s_{l,t} = \cos(\mathbf{g}_{l,t}, \mathbf{g}_{l,t+1}).$$

Fig. 3 shows that *FAUST* consistently produces higher cosine similarity scores on average than *FAUST-random tuplet*, indicating that consecutive updates tend to point in more similar directions. This result suggests that fixing positives and negatives via class representatives reduces sampling-induced fluctuations and leads to more uniform gradient updates during training.

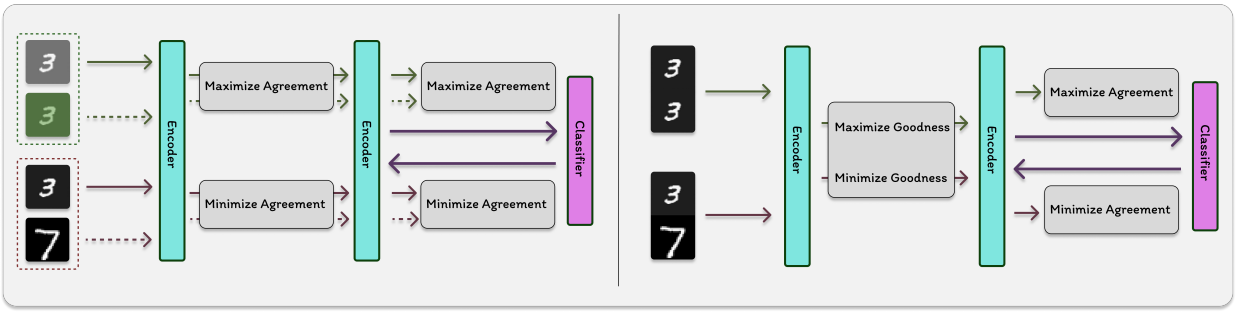


Fig. 2: Visualization of FFCM (Left) and SCFF (Right).

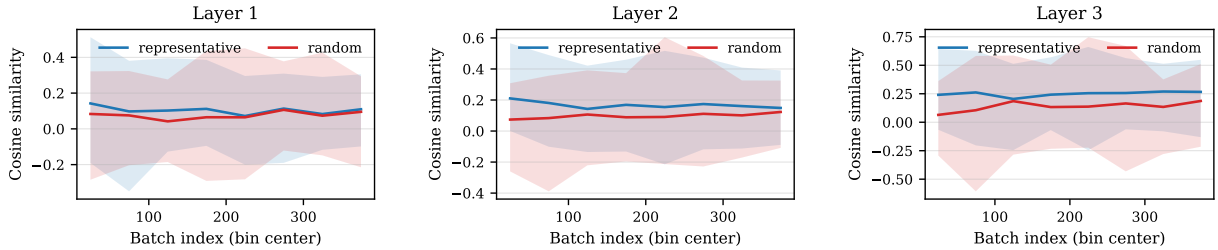


Fig. 3: **Per-layer gradient directions during training**. The solid line shows the mean cosine similarity of each batch-index bin, and the shaded region indicates the minimum–maximum range observed within the same bin. Blue lines (representative) show the results of *FAUST*; red lines (random) show the results of *FAUST-random tuplet*.

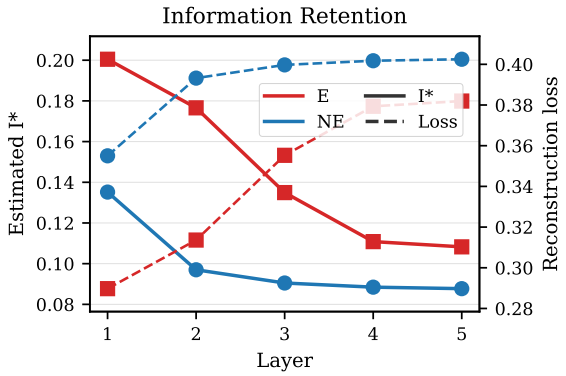


Fig. 4: **Information retention across layers** on an MLP with five layers of 500 neurons and an embedding size of 128 on Fashion-MNIST. E denotes with embedding layers, as in *FAUST*; NE denotes without embedding layers, as in *FAUST-no embedding*.

V-B. Embedding layer reduces information loss

We evaluate whether the incorporation of embedding layers improves information retention in the following manner. Intuitively, the mutual information $I(\mathbf{g}_i;\mathbf{x})$ quantifies how much the hidden representation at layer i, denoted $\mathbf{g}_i$, informs one about the original input $\mathbf{x}$—larger values correspond to more information about $\mathbf{x}$ being recoverable from $\mathbf{g}_i$. Formally,

$$I(\mathbf{g}_i;\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{g}_i),$$

where $H(\mathbf{x})$ is the marginal entropy of the input distribution (a constant for a dataset), and $H(\mathbf{x}|\mathbf{g}_i)$ is the uncertainty about $\mathbf{x}$ after observing $\mathbf{g}_i$. If a sufficiently expressive decoder is trained to reconstruct the original input $\mathbf{x}$ from $\mathbf{g}_i$, the reconstruction loss can serve as a variational upper bound on $H(\mathbf{x}|\mathbf{g}_i)$. Thus, by training a decoder to minimize the reconstruction loss, one can obtain a maximized lower bound on $I(\mathbf{g}_i;\mathbf{x})$ [29].
We apply the following method to *FAUST* and *FAUST-no embedding* to compare the reconstruction losses. We first train the network to convergence and freeze all weights. For each layer i, we then attach a separate decoder and train it to reconstruct the input $\mathbf{x}$ from $\mathbf{g}_i$, with backpropagation applied only to the decoder. We use the same decoder architecture across all layers, namely an MLP with one hidden layer of 500 units and an output layer of size 784 to match the flattened 28×28 images of Fashion-MNIST. Controlling the decoder architecture allows us to attribute the differences in reconstruction losses to the representations $\mathbf{g}_i$ rather than the expressiveness of the decoder. Let θ denote the decoder parameters after optimization; our information retention proxy for layer i is:

$$I^*(\mathbf{g}_i;\mathbf{x}) = H(\mathbf{x}) - \mathcal{R}_\theta(\mathbf{x}|\mathbf{g}_i),$$

Method	MNIST		Fashion-MNIST		CIFAR-10	
MLP-based BP vs. FF models	3/500	4/800	3/500	4/800	3/500	4/800
BP	98.64	98.64	90.11	90.40	57.30	57.63
FF	97.05	97.22	87.10	87.47	46.13	46.44
FFCM [22]	97.01	97.12	87.67	87.64	54.44	54.48
Collab FF [16]	97.90	NR	88.40	NR	48.40	NR
FAUST-random triplet	98.33	97.95	86.61	86.71	47.42	46.76
FAUST-random tuplet	**98.68**	**98.69**	87.69	87.42	54.05	53.79
FAUST-no embedding	91.88	91.54	82.67	82.73	37.96	38.15
FAUST	98.43	98.39	**89.67**	**89.47**	**55.92**	**56.22**
CNN-based BP vs. FF models						
BP (CNN-5)	99.50		93.45		86.22	
CaFo (CNN-3) [20]	99.00		NR		69.50	
Channel-wise FF (CNN-4) [18]	99.42		92.31		78.11	
SCFF (CNN-5) [23]	98.70		NR		80.80	
DeeperForward (CNN-4) [21]	**99.50**		91.83		79.49	
FAUST (CNN-5)	**99.50**		**93.12**		**81.21**	
BP-free, non-FF models						
DTP (CNN-6) [7]	98.93		**90.35**		**89.38**	
PEPITA (CNN-2) [8]	98.01		NR		52.57	
F^3 (MLP-2) [9]	97.16		NR		46.04	
SoftHebb (SoftHebb) [4]	**99.35**		NR		80.31	
DFA (CNN-5) [5]	98.53		87.87		57.53	
FA (CNN-5) [6]	98.99		90.04		62.90	

Table I: **Accuracy across algorithms and datasets.** M/N denotes implementing the algorithm on an MLP with M layers of N neurons. CNN-x denotes implementing the algorithm on a CNN with x convolutional layers. NR indicates not reported.

ResNet-18	No data aug	With data aug
BP	76.45	**81.27**
FAUST	**77.12**	78.06

Table II: **Accuracy of ResNet-18 implementations** on ImageNette with and without data augmentation.

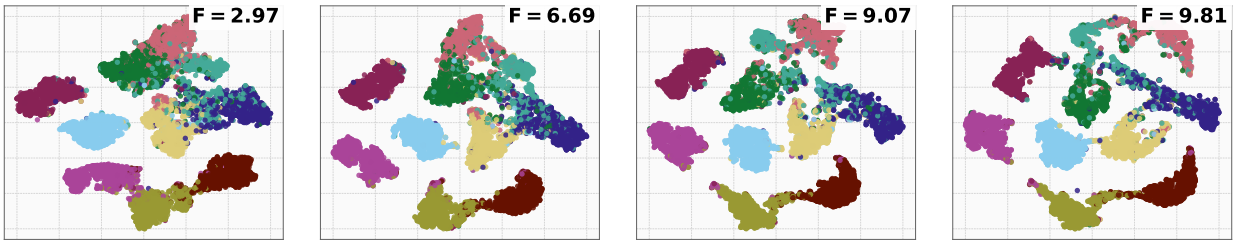
where $\mathcal{R}_\theta(\mathbf{x}|\mathbf{g}_i)$ denotes the expected error for reconstructing $\mathbf{x}$ from $\mathbf{g}_i$ with binary cross-entropy. $I^*(\mathbf{g}_i;\mathbf{x})$ is a lower bound on $I(\mathbf{g}_i;\mathbf{x})$. Fig. 4 shows that introducing embedding layers consistently improves information retention across all layers.

V-C. Classification accuracy

We evaluate *FAUST* on MNIST, Fashion-MNIST, and CIFAR-10 on MLP backbones. We run all MLP-based methods under the same two architectures (3×500 and 4×800); these configurations are tuned not to maximize absolute accuracy but to place competing algorithms on a comparable footing. Table I reports our results alongside BP and prior FF baselines.

Across all datasets, moving from triplet to tuplet loss consistently improves accuracy. On Fashion-MNIST and CIFAR-10, using fixed representatives yields further improvement over random tuplets. Finally, removing the embedding head results in a large degradation across all datasets, indicating that optimizing the local objective in a separate embedding space is valuable for preserving useful representations.

To assess the capability of *FAUST* beyond MLPs, we implement it on a VGG-style CNN containing five convolutional layers. Our results indicate that *FAUST* remains competitive against CNN-based FF models of



(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Fig. 5: ***t*-SNE visualizations** of the representations learned by *FAUST* on an MLP with four layers of 500 neurons and an embedding size of 256 on Fashion-MNIST. The value in the upper-right corner, F, is the Fisher discriminant score. Color-to-class mapping is T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle-boot.

similar depths. We further include results from BP-free, non-FF models to contextualize performance relative to the broader BP-free literature. To assess the scalability of *FAUST*, we implement it along with BP on ResNet-18 and evaluate both networks on ImageNette, a subset of ImageNet [30]. Without data augmentation, *FAUST* achieves a slightly higher peak testing accuracy than BP, as BP experiences more overfitting. With standard ResNet data augmentation [31] enabled, BP benefits substantially and achieves a higher testing accuracy, whereas *FAUST* only shows limited gains.

V-D. Behavior of deeper layers

We analyze how class separability evolves with depth in *FAUST* on Fashion-MNIST. Fig. 5 visualizes layer-wise embeddings using t-SNE, where clusters become progressively more separated as depth increases. We further compute the Fisher discriminant score F, defined as the ratio of between-class scatter to within-class scatter,

$$F = \frac{\sum_c n_c ||\mu_c - \mu||^2}{\sum_c \sum_{x \in c} ||x - \mu_c||^2},$$

where μ_c is the class mean embedding and μ is the global mean embedding. Across the four-layer MLP, F increases from 2.97 to 9.81. This result indicates that, while layers are trained with local objectives, representations become increasingly class-separable in deeper layers.

VI. CONCLUSIONS

In this research, we introduced *FAUST*, a BP-free learning framework that reformulates FF within a layer-wise similarity-learning paradigm. *FAUST* trains each layer with a tuplet-based objective against fixed class representatives and performs inference directly in the representation space. We further show that *FAUST*'s incorporation of embedding layers improves information retention across layers. Empirically, *FAUST* consistently improves over FF baselines on matched MLP and CNN architectures on MNIST, Fashion-MNIST, and CIFAR-10, narrowing the gap to BP. We also tested *FAUST* on ResNet-18, which demonstrates competitive performance on ImageNette. *FAUST* yields convincing empirical results even with the current minimal design of randomly selecting class representatives, suggesting that exploring more informed selection strategies may improve performance and constitute a potentially promising direction for future work.

VII. REFERENCES

[1] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz, "Can the brain do back-propagation? -exact implementation of backpropagation in predictive coding networks.," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22566–22579, Jan. 2020.

[3] Alexander G. Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles, "Backpropagation-free deep learning with recursive local representation alignment," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, pp. 9327–9335, Jun. 2023.

[4] Timoleon Moraitis, Dmitry Toichkin, Adrien Journé, Yansong Chua, and Qinghai Guo, "Soft-hebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, pp. 044017, Dec. 2022.

[5] Arild Nøkland, "Direct feedback alignment provides learning in deep neural networks," 2016.

[6] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman, "Random feedback weights support learning in deep neural networks," 2014.

[7] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio, "Difference target propagation," 2015.

[8] Giorgia Dellaferrera and Gabriel Kreiman, "Error-driven input modulation: Solving the credit assignment problem without a backward pass," 2023.

[9] Katharina Flügel, Daniel Coquelin, Marie Weiel, Charlotte Debus, Achim Streit, and Markus Götz, *Feed-Forward Optimization With Delayed Feedback for Neural Network Training*, p. 74–88, Springer Nature Singapore, 2025.

[10] Geoffrey Hinton, "The forward-forward algorithm: Some preliminary investigations," 2022.

[11] Donald Olding Hebb, *The organization of behavior: A Neuropsychological Theory*, John Wiley & Sons, Jan. 1949.

[12] Natalia Caporale and Yang Dan, "Spike timing–dependent plasticity: A hebbian learning rule," *Annual review of neuroscience*, vol. 31, pp. 25–46, 02 2008.

[13] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 815–823.

[14] Heung-Chang Lee and Jeonggeun Song, "Symba: Symmetric backpropagation-free contrastive learning with forward-forward algorithm for optimizing convergence," 2023.

[15] Alexander Ororbia and Ankur Mali, "The predictive forward-forward algorithm," 2023.

[16] Guy Lorberbom, Itai Gat, Yossi Adi, Alexander Schwing, and Tamir 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.

[17] Thomas Dooms, Ing Jyh Tsang, and Jose Oramas, "The trifecta: Three simple techniques for training deeper forward-forward networks," 2023.

[18] Andreas Papachristodoulou, Christos Kyrkou, Stelios Timotheou, and Theocharis Theocharides, "Convolutional channel-wise competitive learning for the forward-forward algorithm," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, pp. 14536–14544, Mar. 2024.

[19] Riccardo Scodellaro, Ajinkya Kulkarni, Frauke Alves, and Matthias Schröter, "Training convolutional neural networks with the forward-forward algorithm," 2024.

[20] Gongpei Zhao, Tao Wang, Yidong Li, Yi Jin, Congyan Lang, and Haibin Ling, "The cascaded forward algorithm for neural network training," 2023.

[21] Liang Sun, Yang Zhang, Weizhao He, Jiajun Wen, Linlin Shen, and Weicheng Xie, "Deeperforward: Enhanced forward-forward training for deeper and better performance," in *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Hossein Aghagolzadeh and Mehdi Ezoji, "Marginal contrastive loss: A step forward for forward-forward," in *2024 13th Iranian/3rd International Machine Vision and Image Processing Conference (MVIP)*, 2024, pp. 1–6.

[23] Xing Chen, Dongshu Liu, Jeremie Laydevant, and Julie Grollier, "Self-contrastive forward-forward algorithm," 2025.

[24] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 539–546 vol. 1.

[25] Kihyuk Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29, Curran Associates, Inc.

[26] Jake Snell, Kevin Swersky, and Richard S. Zemel, "Prototypical networks for few-shot learning," 2017.

[27] Samantha Guerriero, Barbara Caputo, and Thomas Mensink, "Deepncm: Deep nearest class mean classifiers," in *ICLR 2018 Workshop*, 2018.

[28] Thanh-Toan Do, Toan Tran, Ian Reid, Vijay Kumar, Tuan Hoang, and Gustavo Carneiro, "A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10396–10405.

[29] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," 2022.

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

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.