FORWARD ITERATIVE OPTIMIZATION NETWORK FOR IMAGE RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks trained with backpropagation (BP) have achieved remarkable success across a wide range of domains. However, BP contains several limitations like weight transport, backward locking, computational and memory overhead. Although recent BP-free learning methods have explored alternative training mechanisms, they often compromise predictive performance or computational efficiency. To overcome these limitations, we propose Forward Iterative Optimization (FIO) framework for network learning. Different from existing BP-free methods, FIO derives analytical solutions for parameter updates and incorporates them into an iterative optimization strategy, thereby reducing training-time memory overhead and enabling efficient parameter updates. Based on this strategy, we first develop a parameter optimization method for a single-layer network and then extend it to multi-layer and multi-head architectures. Extensive experiments demonstrate that FIO achieves competitive or even superior performance compared with both BP and representative BP-free methods, while requiring fewer parameters and lower computational cost.

1 INTRODUCTION

Artificial neural networks trained with backpropagation (BP) have achieved remarkable success across a broad range of learning tasks (Nalisnick et al., 2023). Despite its effectiveness, BP has several well-known limitations. From the perspective of biological plausibility, BP relies on non-local error signals, precise symmetry between forward and backward synaptic weights, and coordinated credit assignment based on a global objective, all of which are difficult to reconcile with biological learning mechanisms (Lillicrap et al., 2020; Whittington & Bogacz, 2019). From a computational perspective, BP requires both forward and backward passes and the storage of intermediate activations for gradient computation, leading to substantial memory consumption and computational overhead, particularly in deep neural networks and resource-constrained environments (Choudhary et al., 2020).

Existing alternatives to backpropagation can be broadly categorized into three directions: *(i)* biologically inspired methods (Whittington & Bogacz, 2017), which typically formulate learning through iterative neural inference and local error minimization; *(ii)* asymmetric-weight methods (Lillicrap et al., 2016), which relax the requirement of symmetric forward and backward weights through alternative feedback pathways; and *(iii)* forward-only algorithms (Hinton, 2022), which avoid the conventional backward pass and update parameters primarily through forward computations.

Although these approaches relax different constraints of standard BP, they still exhibit important limitations in optimization efficiency and representation learning. Biologically inspired methods can incur substantial optimization overhead and face scalability challenges, as iterative inference or relaxation may require many computational steps, while training dynamics can become increasingly unstable with network depth (Scellier & Bengio, 2017; Zahid et al., 2023; Ha et al., 2026). Asymmetric-weight methods can suffer from feedback-gradient misalignment, resulting in degraded performance and limited scalability to deep and structured architectures (Bartunov et al., 2018; Refinetti et al., 2021; Caillon et al., 2026). Forward-only algorithms eliminate explicit backward propagation, but many rely on local objectives, auxiliary signals, or layer-wise optimization, which may weaken inter-layer coordination and limit hierarchical representation learning (Dellaferrera & Kreiman, 2022; Salvatori et al., 2024; Yao et al., 2026). Consequently, existing BP-free methods can suffer from inefficient local optimization, limited representational capacity, or reduced scalability

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

as network depth increases (Hinton, 2022; Qi et al., 2025; Caillon et al., 2025; Gong et al., 2026). Developing a learning mechanism that more fundamentally eliminates dependence on backward gradient information while preserving effective representation learning, computational efficiency, and scalability therefore remains an important research challenge.

In this paper, we propose a Forward Iterative Optimization framework, namely **FIO**, for network learning as an alternative to gradient-based backpropagation. Unlike existing BP-free methods, FIO employs an iterative optimization strategy in which model parameters are updated through analytical solutions rather than gradient-based updates, thereby avoiding the storage of intermediate states required for backward computation and enabling efficient parameter optimization. Furthermore, the multi-head design promotes diverse feature projections, while the multi-layer extension enhances the flexibility of representation learning.

In summary, our main contributions are listed as follows:

- We propose a multi-layer and multi-head projection learning framework that performs representation learning without conventional backward gradient propagation.
- We develop a forward iterative optimization strategy in which network parameters are updated through analytical solutions and constrained optimization, providing a deterministic and mathematically interpretable learning procedure.
- Extensive experiments demonstrate that the proposed method consistently outperforms existing BP-free baselines and achieves competitive classification performance with standard BP, while requiring fewer parameters and lower computational and memory costs.

2 Related Works

Biologically Inspired Learning Biologically inspired learning seeks to replace biologically implausible components of BP, such as global error propagation and weight symmetry, with learning rules based on local neural activity and biologically motivated computation. Hebbian learning established the fundamental principle of local synaptic plasticity, in which synaptic updates depend primarily on the activities of pre- and post-synaptic neurons (Hebb, 2005). Predictive Coding (PC) extends this principle toward error-driven learning by iteratively minimizing prediction errors between adjacent layers, and supervised PC has been shown to approximate BP under specific conditions (Whittington & Bogacz, 2017). More recently, Dendritic Localized Learning (DLL) introduces compartmentalized pyramidal neurons and dendritic feedback to enable layer-local error computation and parameter updates (Lv et al., 2025).

These approaches represent a progression from activity-dependent synaptic plasticity to structured local error minimization and biologically motivated credit assignment. DLL, in particular, combines localized learning with dendritic computation to further reduce reliance on globally propagated errors. However, biologically inspired methods often require iterative neural-state inference, auxiliary feedback dynamics, or locally generated error signals, which can increase training complexity and retain dependence on error-based optimization. Their performance may also remain below that of standard BP on challenging learning tasks, illustrating the difficulty of simultaneously achieving biological plausibility, optimization efficiency, and strong representation learning.

Asymmetric Feedback Feedback Alignment (FA) replaces the transpose of the forward weights used in BP with fixed random feedback weights, thereby removing the requirement for exact weight symmetry (Lillicrap et al., 2016). Representative extensions include Direct Feedback Alignment (DFA), which directly propagates output errors to individual hidden layers through fixed random feedback pathways, further reducing reliance on sequential backward error propagation (Nøkland, 2016).

Subsequent studies have focused on improving the efficiency, theoretical understanding, and scalability of asymmetric feedback. Sparse feedback connections have been introduced to reduce the computational cost of error transmission while preserving effective learning (Crafton et al., 2019). More recent theoretical analyses show that effective asymmetric feedback only requires sufficient alignment between the projected error and the true gradient direction, a principle further leveraged by Gradient-Aligned Projected Error (GrAPE) for fine-tuning BP-pretrained models (Caillon et al., 2026).

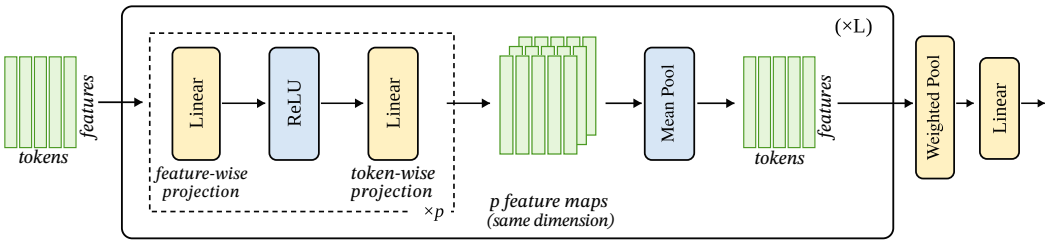


Figure 1: The overall architecture of the proposed FIO. The feature-wise and token-wise projections operate directly along the feature and token dimensions of each feature map, respectively, without requiring explicit transposition.

Forward-Only Algorithms Forward-only learning seeks to train neural networks without explicitly propagating gradients through the conventional backward pass, thereby reducing reliance on global credit assignment and backward error transmission. The Forward-Forward (FF) algorithm replaces the standard forward-backward procedure with positive and negative forward passes, allowing each layer to optimize a local goodness objective independently (Hinton, 2022). PEPITA instead projects the output error to the input through a fixed random matrix and performs an additional forward pass on the resulting perturbed input, establishing a connection between forward-only learning and asymmetric feedback (Dellaferrera & Kreiman, 2022). LightFF improves the efficiency of FF through a lightweight formulation that reduces its computational overhead while preserving the forward-only training paradigm (Aminifar et al., 2024). More recently, FOTON introduces orthogonality constraints into forward-only optimization, establishes a theoretical connection between its updates and BP in linear orthogonal networks, and extends the applicability of forward-only learning to deeper architectures (Caillon et al., 2025).

Collectively, these developments have advanced forward-only learning toward greater efficiency and stronger theoretical grounding. They also exhibit close connections with the preceding research directions: FF inherits the emphasis on local learning from biologically inspired approaches, while PEPITA retains the random feedback mechanism characteristic of asymmetric-feedback methods. Recent theoretical analyses have further identified shared learning principles between forward-only learning and feedback alignment (Srinivasan et al., 2024). Nevertheless, existing forward-only methods often rely on specially designed local objectives, externally constructed error perturbations, or additional structural constraints. Moreover, FF-based methods that independently optimize layer-local objectives may provide limited coordination across network layers, potentially restricting hierarchical representation learning as network depth increases.

3 METHOD

The overall framework is illustrated in Figure 1, where the proposed model consists of L cascaded blocks, each of which contains p parallel projection heads. Within each block, every head first applies a linear transformation along the feature dimension, followed by a ReLU activation and a linear transformation along the token dimension. The resulting p feature maps share the same dimensionality and are aggregated by mean pooling to form the output representation of the current block, which is then passed to the subsequent block. After the final block, weighted pooling is applied along the token dimension to aggregate token-level features into a single representation, followed by a final feature-wise linear transformation that produces the classification logits.

We present the proposed method progressively from its basic formulation to the complete architecture. We first introduce the optimization procedure for a single block with a single projection head, i.e., $L = 1$ and $p = 1$. We then extend the formulation to multiple cascaded blocks with arbitrary depth L while retaining the same final pooling and classification layers. Finally, we incorporate p parallel projection heads within each block to learn multiple intermediate representations, yielding the complete multi-layer and multi-head architecture.

Throughout this paper, we utilize symbols such as s, $\boldsymbol{v}$, $\boldsymbol{M}$, and $\mathbf{T}$ to denote scalars, vectors, matrices, and tensors, respectively. Unless otherwise specified, tensor multiplications are performed along

compatible adjacent dimensions, with vectors and matrices treated as tensors of the corresponding order. For example, if $\boldsymbol{a} \in \mathbb{R}^m$ and $\mathbf{B} \in \mathbb{R}^{m \times n \times d}$, then $\boldsymbol{a}\mathbf{B} \in \mathbb{R}^{n \times d}$.

3.1 Single-Hidden-Layer Model

3.1.1 Reformulation of Image Classification

Following the common practice of representing images as sequences of patches, we reformulate each image as a collection of tokens, where each token is represented by a feature vector. The proposed model performs feature transformations along the feature dimension while simultaneously modeling interactions across tokens.

Given n images, let $\mathbf{X}_i^{img} \in \mathbb{R}^{h \times w \times c}$ denote the i-th image, where h, w and c denote the image height, width, and number of channels, respectively. Each image is reshaped into $\mathbf{X}_i \in \mathbb{R}^{m \times 1 \times d}$, where m denotes the number of tokens and d denotes the feature dimension of each token. The corresponding one-hot label is denoted by $\boldsymbol{Y}_i \in \mathbb{R}^{1 \times k}$, where k is the number of classes.

Accordingly, the complete training set is represented by $\mathbf{X} \in \mathbb{R}^{m \times n \times d}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$. For the single-hidden-layer model with a single projection head, we formulate training as the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{a}, \boldsymbol{A}_0, \boldsymbol{W}, \boldsymbol{W}_0} \ & \mathcal{L} = \left\| \left(\boldsymbol{a} \left(\boldsymbol{A}_0 \, \sigma(\mathbf{X}\boldsymbol{W}_0) \right) \right) \boldsymbol{W} - \boldsymbol{Y} \right\|_F^2, \\ \text{s.t.} \quad & \sum_{r=1}^{m} a_r = 1, \quad a_r \geq 0, \ r = 1, \ldots, m, \end{aligned} \tag{1}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d \times e}$ projects each token from the d-dimensional input space into an e-dimensional latent space, σ applies an element-wise ReLU activation to the projected features, and $\boldsymbol{A}_0 \in \mathbb{R}^{m \times m}$ subsequently models interactions among the m tokens. The aggregation vector $\boldsymbol{a} \in \mathbb{R}^m$ is constrained to the $(m-1)$-dimensional simplex, yielding a normalized non-negative aggregation of token-level representations. Finally, $\boldsymbol{W} \in \mathbb{R}^{e \times k}$ maps the aggregated latent representation to the k-dimensional classification logits.

Accordingly, in Figure 1, the yellow parameterized modules correspond to the learnable parameters in the formulation: $\boldsymbol{W}_0$ and $\boldsymbol{A}_0$ represent the feature-wise and token-wise projections, respectively, $\boldsymbol{a}$ provides the weights for the weighted pooling operation, and $\boldsymbol{W}$ serves as the final linear classification head. The overall objective is defined as the squared Frobenius norm of the difference between the predictions and the ground-truth labels, equivalently the sum of squared ℓ_2 losses over all samples.

3.1.2 Alternating Optimization

We solve the optimization problem using an alternating optimization strategy over four parameter groups, updating each group while keeping the others fixed.

Optimization of $\boldsymbol{a}$. Fixing $\boldsymbol{A}_0$, $\boldsymbol{W}_0$ and $\boldsymbol{W}$, Eq. (1) reduces to the following constrained least-squares problem:

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & \|\boldsymbol{a}\boldsymbol{B} - \boldsymbol{y}\|_2^2, \\ \text{s.t.} \quad & \sum_{r=1}^{m} a_r = 1, \quad a_r \geq 0, \ r = 1, \ldots, m, \end{aligned} \tag{2}$$

where $\boldsymbol{B} = \mathrm{reshape}((\boldsymbol{A}_0 \, \sigma(\mathbf{X}\boldsymbol{W}_0)) \, \boldsymbol{W}) \in \mathbb{R}^{m \times (nk)}$ is obtained by merging the sample and output dimensions while preserving the token dimension, and $\boldsymbol{y} \in \mathbb{R}^{(nk)}$ is obtained by reshaping $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$.

We solve this simplex-constrained least-squares problem using an active-set method (Forsgren et al., 2016). Since no explicit closed-form solution is available, $\boldsymbol{a}$ is updated iteratively until convergence.

Optimization of $\boldsymbol{W}$. Fixing $\boldsymbol{a}$, $\boldsymbol{A}_0$ and $\boldsymbol{W}_0$, we define the aggregated latent representation as

$$\bar{\boldsymbol{Z}} = \boldsymbol{a}\left(\boldsymbol{A}_0\,\sigma(\mathbf{X}\boldsymbol{W}_0)\right), \tag{3}$$

where $\bar{\boldsymbol{Z}} \in \mathbb{R}^{n\times e}$ and e denotes the latent feature dimension. Based on Eq. (1), by adding an ℓ_2-regularization term, we formulate the optimization of $\boldsymbol{W}$ as

$$\min_{\boldsymbol{W}} \left\|\bar{\boldsymbol{Z}}\boldsymbol{W} - \boldsymbol{Y}\right\|_F^2 + \lambda_W \left\|\boldsymbol{W}\right\|_F^2, \tag{4}$$

where $\lambda_W > 0$ is a sufficiently small regularization coefficient. The same convention applies to λ_{A_0} and λ_{W_0} in the following subproblems. Their positivity ensures that the corresponding coefficient matrices are positive definite and therefore invertible, while their small values minimize the influence of the additional regularization terms. The corresponding closed-form solution is

$$\boldsymbol{W} = \left(\bar{\boldsymbol{Z}}^\top\bar{\boldsymbol{Z}} + \lambda_W\boldsymbol{I}_e\right)^{-1}\bar{\boldsymbol{Z}}^\top\boldsymbol{Y}, \tag{5}$$

where $\boldsymbol{I}_e \in \mathbb{R}^{e\times e}$ denotes the identity matrix. Since $\lambda_W > 0$, the matrix $\bar{\boldsymbol{Z}}^\top\bar{\boldsymbol{Z}} + \lambda_W\boldsymbol{I}_e$ is positive definite and therefore invertible. The detailed derivation of this closed-form solution is provided in Appendix A.

Optimization of $\boldsymbol{A}_0$. Fixing $\boldsymbol{a}$, $\boldsymbol{W}_0$ and $\boldsymbol{W}$, we define

$$\boldsymbol{B}_0 = \mathrm{reshape}\left(\sigma(\mathbf{X}\boldsymbol{W}_0)\boldsymbol{W}\right) \in \mathbb{R}^{m\times(nk)}, \tag{6}$$

where the sample and output dimensions are merged while preserving the token dimension.

By adding an ℓ_2-regularization term, the optimization of $\boldsymbol{A}_0$ is formulated as the following regularized least-squares problem:

$$\min_{\boldsymbol{A}_0} \left\|\boldsymbol{a}\boldsymbol{A}_0\boldsymbol{B}_0 - \boldsymbol{y}\right\|_2^2 + \lambda_{A_0}\left\|\boldsymbol{A}_0\right\|_F^2, \tag{7}$$

where $\boldsymbol{y} \in \mathbb{R}^{(nk)}$ denotes the reshaped label vector and $\lambda_{A_0} > 0$ is the regularization coefficient.

The corresponding closed-form solution is

$$\boldsymbol{A}_0 = \mathrm{vec}(\boldsymbol{a})\,\mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top\left(\left\|\boldsymbol{a}\right\|_2^2\boldsymbol{B}_0\boldsymbol{B}_0^\top + \lambda_{A_0}\boldsymbol{I}_m\right)^{-1}, \tag{8}$$

where $\mathrm{vec}(\boldsymbol{a}) \in \mathbb{R}^{m\times 1}$ and $\mathrm{vec}^\top(\boldsymbol{y}) \in \mathbb{R}^{1\times(nk)}$ denote the column and row forms of $\boldsymbol{a}$ and $\boldsymbol{y}$, and $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ is the identity matrix. The detailed derivation of this closed-form solution is provided in Appendix B.

Optimization of $\boldsymbol{W}_0$. The optimization of $\boldsymbol{W}_0$ is non-convex due to the ReLU activation. We therefore update its columns sequentially while fixing the remaining parameters and the current ReLU activation pattern. Let $\boldsymbol{u}_i \in \mathbb{R}^{d\times 1}$ denote the i-th column of $\boldsymbol{W}_0$ and $\boldsymbol{v}_i^\top \in \mathbb{R}^{1\times k}$ the i-th row of $\boldsymbol{W}$. We define the residual excluding this contribution as

$$\boldsymbol{T} = \boldsymbol{Y} - \sum_{j\neq i}\boldsymbol{a}\boldsymbol{A}_0\,\sigma(\mathbf{X}\boldsymbol{u}_j)\boldsymbol{v}_j^\top \in \mathbb{R}^{n\times k}. \tag{9}$$

For the current ReLU activation pattern, let $\mathbf{M}_i \in \mathbb{R}^{m\times n\times d}$ denote the binary mask obtained by broadcasting $\mathbb{I}(\mathbf{X}\boldsymbol{u}_i > 0) \in \mathbb{R}^{m\times n\times 1}$ along the last dimension, and define $\boldsymbol{G}_i = \boldsymbol{a}\boldsymbol{A}_0(\mathbf{M}_i \odot \mathbf{X}) \in \mathbb{R}^{n\times d}$, where $\odot$ denotes the Hadamard product, i.e., element-wise multiplication. Using vectorization, the contribution of $\boldsymbol{u}_i$ can be written as $\boldsymbol{H}_i\boldsymbol{u}_i$, where $\boldsymbol{H}_i = \boldsymbol{v}_i \otimes \boldsymbol{G}_i \in \mathbb{R}^{(nk)\times d}$ and $\otimes$ denotes the Kronecker product. By adding an ℓ_2-regularization term, the resulting subproblem is

$$\min_{\boldsymbol{u}_i} \left\|\boldsymbol{H}_i\boldsymbol{u}_i - \widehat{\boldsymbol{T}}\right\|_F^2 + \lambda_{W_0}\left\|\boldsymbol{u}_i\right\|_F^2, \tag{10}$$

where $\widehat{\boldsymbol{T}} = \mathrm{vec}(\boldsymbol{T}) \in \mathbb{R}^{(nk)\times 1}$ and $\lambda_{W_0} > 0$ is the regularization coefficient.

The corresponding closed-form update is

$$\boldsymbol{u}_i = \left(\boldsymbol{H}_i^\top\boldsymbol{H}_i + \lambda_{W_0}\boldsymbol{I}_d\right)^{-1}\boldsymbol{H}_i^\top\widehat{\boldsymbol{T}}, \tag{11}$$

where $\boldsymbol{I}_d \in \mathbb{R}^{d\times d}$ denotes the identity matrix. Sequentially applying this update for $i = 1, \ldots, e$ yields the updated matrix $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$, and the detailed derivation of this closed-form update is provided in Appendix C.

For clarity, the complete optimization procedure for the single-hidden-layer model is summarized in Algorithm 1.

Algorithm 1 Alternating Optimization for the Single-Hidden-Layer Model

1: Initialize $\boldsymbol{a}$, $\boldsymbol{A}_0$, $\boldsymbol{W}$, and $\boldsymbol{W}_0$.
2: **repeat**
3: Update $\boldsymbol{a}$ using the active-set method.
4: Update $\boldsymbol{W}$ using the analytical solution in Eq. (5).
5: Update $\boldsymbol{A}_0$ using the analytical solution in Eq. (8).
6: Update $\boldsymbol{W}_0$ by sequentially applying Eq. (11) to each column.
7: **until** convergence or maximum iterations
8: **return** $\boldsymbol{a}$, $\boldsymbol{A}_0$, $\boldsymbol{W}$, and $\boldsymbol{W}_0$.

3.2 Multi-Hidden-Layer Model

Although the single-layer model is computationally efficient, its representation capacity remains limited. To improve model expressiveness, we extend this model by stacking multiple hidden layers to learn hierarchical representations. The optimization objective for the ℓ-th layer ($\ell \neq 0$) is formulated as

$$\begin{aligned} \min_{\boldsymbol{a},\boldsymbol{A}_\ell,\boldsymbol{W},\boldsymbol{W}_\ell} \quad & \mathcal{L}_\ell = \left\| \left(\boldsymbol{a} \left(\boldsymbol{A}_\ell \, \sigma \left(\mathbf{Z}_\ell \boldsymbol{W}_\ell \right) \right) \right) \boldsymbol{W} - \boldsymbol{Y} \right\|_F^2, \\ \text{s.t.} \quad & \sum_{r=1}^{m} a_r = 1, \quad a_r \geq 0, \; r = 1, \ldots, m, \end{aligned} \tag{12}$$

where $\mathbf{Z}_\ell = \boldsymbol{A}_{\ell-1} \, \sigma(\mathbf{Z}_{\ell-1} \boldsymbol{W}_{\ell-1}) \in \mathbb{R}^{m \times n \times e}$ denotes the input representation to the ℓ-th layer, with $\mathbf{Z}_1 = \boldsymbol{A}_0 \, \sigma(\mathbf{X} \boldsymbol{W}_0)$. $\boldsymbol{W}_\ell \in \mathbb{R}^{e \times e}$ denotes the corresponding feature-wise projection matrix, and $\boldsymbol{A}_\ell \in \mathbb{R}^{m \times m}$ models token interactions within that layer.

Since Eq. (12) has the same form as Eq. (1), each layer can be optimized using the single-hidden-layer procedure in Algorithm 1. The layers are trained sequentially in a feed-forward manner, with previously optimized projection parameters kept fixed, while the shared $\boldsymbol{a}$ and $\boldsymbol{W}$ are re-optimized for the newly added layer. This formulation enables flexible depth extension, with the model depth selected according to validation performance. The complete optimization procedure for the multi-hidden-layer model is summarized in Algorithm 2 in Appendix E.

3.3 Multi-Head Projection Model

To further enhance the feature extraction and intermediate representation capabilities of the model, we increase the model width by introducing multiple parallel projection heads. The optimization objective for the ℓ-th block is formulated as

$$\begin{aligned} \min_{\boldsymbol{a},\{\boldsymbol{A}_{\ell,i},\boldsymbol{W}_{\ell,i}\}_{i=1}^{p},\boldsymbol{W}} \quad & \mathcal{L}_\ell = \left\| \boldsymbol{a} \left(\frac{1}{p} \sum_{i=1}^{p} \boldsymbol{A}_{\ell,i} \, \sigma \left(\mathbf{Z}_\ell \, \boldsymbol{W}_{\ell,i} \right) \right) \boldsymbol{W} - \boldsymbol{Y} \right\|_F^2, \\ \text{s.t.} \quad & \sum_{r=1}^{m} a_r = 1, \quad a_r \geq 0, \; r = 1, \ldots, m, \end{aligned} \tag{13}$$

where p denotes the number of projection heads and $\ell = 0, \ldots, L-1$ indexes the current block. The block representations are recursively defined by averaging the p head representations as $\mathbf{Z}_{\ell+1} = p^{-1} \sum_{i=1}^{p} \boldsymbol{A}_{\ell,i} \sigma \left(\mathbf{Z}_\ell \boldsymbol{W}_{\ell,i} \right) \in \mathbb{R}^{m \times n \times e}$, with $\mathbf{Z}_0 = \mathbf{X}$, where $\boldsymbol{A}_{\ell,i} \in \mathbb{R}^{m \times m}$ models token interactions, while the feature-wise projection matrix satisfies $\boldsymbol{W}_{\ell,i} \in \mathbb{R}^{e \times e}$ for $\ell \geq 1$, with $\boldsymbol{W}_{0,i} \in \mathbb{R}^{d \times e}$ for the first block.

The optimization of each projection head can be reformulated into the same form as Eq. (1), as detailed in Appendix D. Therefore, each head can be optimized using the single-head optimization procedure while fixing the remaining heads. The complete optimization procedure for the multi-head projection model is summarized in Algorithm 3 in Appendix F.

Table 1: Main comparison of classification accuracy (%) on MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. The best result in each column is highlighted in bold while the second is underlined.

Method	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
BP	**98.89**	89.48	54.90	**28.01**
iPC (Salvatori et al., 2024)	98.45	<u>89.90</u>	<u>57.70</u>	26.64
DLL (Lv et al., 2025)	97.57	87.50	45.87	23.60
FA (Lillicrap et al., 2016)	98.34	89.50	53.00	24.70
DFA (Nøkland, 2016)	98.30	89.15	52.60	24.10
GrAPE (Caillon et al., 2026)	98.53	89.30	53.40	26.22
FF (Hinton, 2022)	97.12	87.88	50.35	27.07
LightFF (Aminifar et al., 2024)	98.57	89.60	53.83	27.15
FOTON (Caillon et al., 2025)	98.61	87.84	55.70	23.84
FIO	<u>98.65</u>	**90.69**	**60.41**	<u>27.88</u>

Table 2: Comparison of model size and single-sample inference efficiency. Time denotes the average inference time for one sample over five repeated runs.

	MNIST & Fashion-MNIST		CIFAR-10 & CIFAR-100	
Method	Params	Time	Params	Time
Baseline Methods	0.67M	36.91 ms	4.21M	42.74 ms
FIO	0.47M	1.26 ms	2.09M	2.36 ms

4 EXPERIMENTS

4.1 SETTINGS

We evaluate the proposed method on four widely used image classification benchmarks: MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), CIFAR-10, and CIFAR-100 (Krizhevsky & Hinton, 2009). Each image is partitioned into non-overlapping 4×4 patches, which are treated as input tokens. This results in $m = 49$ tokens with a feature dimension of $d = 16$ for MNIST and Fashion-MNIST, and $m = 64$ tokens with a feature dimension of $d = 48$ for CIFAR-10 and CIFAR-100. To ensure a controlled comparison, we do not employ data augmentation or additional training techniques, including random cropping, random flipping, dropout, batch normalization, or mix-up. Consequently, performance differences primarily reflect the underlying learning and optimization mechanisms rather than dataset-specific training heuristics.

All models use two hidden layers, as most methods achieve their best convergence performance under this configuration. For comparison baselines, the hidden dimension is set to 512 for MNIST and Fashion-MNIST and to 1024 for CIFAR-10 and CIFAR-100, with the batch size fixed to 256. By contrast, our method utilizes a hidden dimension of 384 for all four datasets and operates in a batch-free manner without mini-batch optimization.

The proposed model employs 6 projection heads on MNIST and Fashion-MNIST and 12 projection heads on CIFAR-10 and CIFAR-100. All experiments are conducted on an NVIDIA GeForce RTX 4090 GPU and repeated over five independent runs, with the mean classification accuracy reported in Table 1.

4.2 RESULTS

Table 1 reports the classification accuracy of different methods using two hidden layers. As shown, our method outperforms standard BP by 1.21% and 5.51% on Fashion-MNIST and CIFAR-10. On MNIST and CIFAR-100, although our method performs 0.24% and 0.13% below BP, it still outperforms all other comparison methods.

Table 3: Ablation study on network depth, projection heads, and hidden dimension. Classification accuracy is reported in percentage (%). For each group, only the hyperparameter under study is varied, while the remaining model configurations are kept fixed. The best result in each column of each group is highlighted in bold.

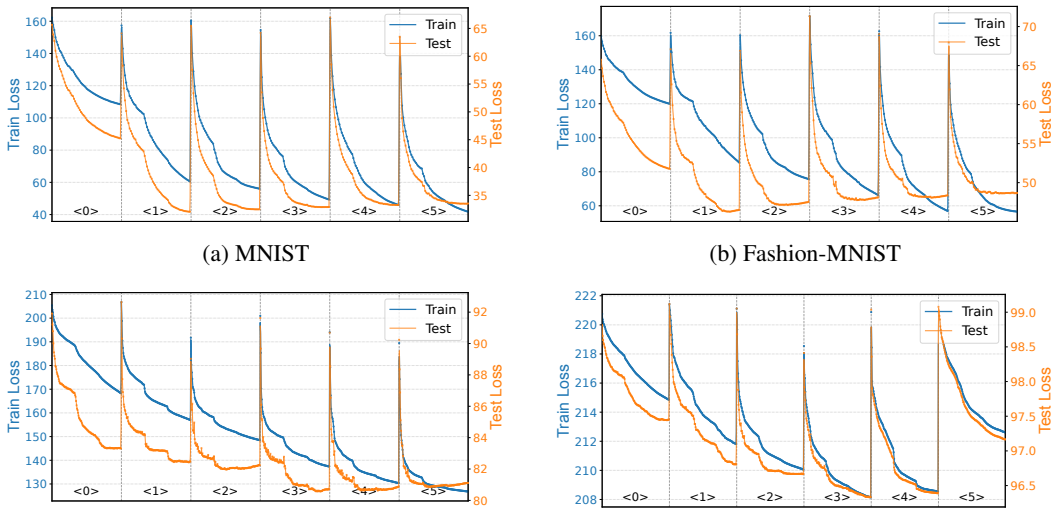
Settings	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
Hidden Layers (Single Head)				
1 Layer	78.46	79.26	51.52	18.93
2 Layers	96.61	88.07	52.69	23.56
4 Layers	97.29	**88.47**	**53.72**	24.55
8 Layers	**97.62**	88.14	53.33	**25.54**
Projection Heads (2 Hidden Layers)				
1 Head	96.61	88.07	52.69	23.56
2 Heads	96.05	86.53	56.59	21.06
4 Heads	96.97	89.01	58.76	24.03
8 Heads	**98.35**	**90.16**	**59.86**	**26.75**
Hidden Dimension (2 Hidden Layers and 8 Heads)				
128	98.01	89.71	59.72	25.28
256	98.35	90.16	**59.86**	26.75
384	**98.65**	**90.69**	59.76	**27.58**

To further demonstrate the parameter and computational efficiency of our method, we conduct a cost comparison experiment and report the results in Table 2. Note that all baseline methods share the same inference architecture and thus have identical model sizes and inference costs. As we can see from Table 2, our method uses only about 70% and 50% of the parameters required by the comparison methods on the MNIST-family and CIFAR-family datasets. In terms of inference efficiency, our method achieves approximately $29\times$ and $18\times$ speedups over the baselines on the two dataset groups, respectively. We do not report wall-clock training time, as implementation differences in optimization steps and parameter-update granularity make direct comparisons less meaningful. Instead, Appendix G provides a detailed analysis of the training-time complexity. These computational advantages make our method particularly suitable for resource-constrained scenarios, while its multi-head design also offers potential for further acceleration through parallel computation.

4.3 Ablation Study

We conduct ablation studies on all four datasets to evaluate the effectiveness of the single-layer optimization framework in Section 3.1 and the contributions of the multi-layer and multi-head extensions in Sections 3.2 and 3.3, respectively. We consider three groups of ablation experiments: (1) varying the number of hidden layers with a single projection head; (2) varying the number of projection heads with two hidden layers; and (3) varying the hidden dimension with two hidden layers and eight projection heads. The specific model configurations and corresponding results for each ablation group are summarized in Table 3.

As shown in the first group, increasing the number of hidden layers from one to two substantially improves performance, with gains of 18.15% on MNIST and 8.81% on Fashion-MNIST. Further increasing the depth has only a marginal effect. The second group shows that increasing the number of projection heads generally improves performance, highlighting the benefit of richer parallel representations. The third group indicates that enlarging the hidden dimension can also improve performance, although the gains become limited once the dimension is sufficiently large and may even slightly decrease. Overall, increasing model depth, the number of projection heads, or the hidden dimension can improve performance, but scaling any single architectural factor alone does not guarantee consistent gains.



(a) MNIST (b) Fashion-MNIST

(c) CIFAR-10 (d) CIFAR-100

Figure 2: Training and test loss curves on different datasets. Each partition corresponds to the optimization of one block, and the blue and orange curves denote the training and test losses, respectively.

4.4 Convergence Analysis

According to the optimization procedure described in Section 3, the training loss is expected to decrease within each block as the block-wise optimization proceeds. To empirically verify this convergence behavior, we construct a network with a hidden dimension of 128, 4 projection heads, and 6 cascaded blocks, and report the training and test losses throughout the optimization process.

As shown in Figure 2, within each block, both the training and test losses consistently decrease as optimization proceeds, until the test loss approaches a stable level, indicating that the proposed optimization procedure can effectively drive each block toward convergence.

When a new block is introduced, the newly initialized random projection parameters cause a sharp increase in the overall loss. As the new block is subsequently optimized, the loss decreases rapidly again, and the training loss typically reaches a lower level than that obtained in the preceding block. Although the convergence curves indicate that the model can already converge with a shallow architecture, a lower training loss does not necessarily correspond to a lower test loss. This is possibly due to overfitting the model.

Note that the current model does not employ residual connections. Although residual connections produce smoother convergence curves, they do not improve the best classification accuracy and may lead to overfitting as depth increases. For clarity, their effects on the proposed framework are further discussed in Appendix H.

5 Conclusion

In this paper, we propose a Forward Iterative Optimization framework for network learning, aiming to eliminate the reliance on gradient computation and backpropagation. We first introduce our single-hidden-layer model and provide a deterministic and mathematically interpretable learning procedure, and then extend it to multi-layer and multi-head architectures to enhance the model's representation capability. Experiments demonstrate that our model achieves higher accuracy than existing BP-free methods while reaching convergence with fewer parameters and lower computational cost. Despite these advantages, our method does not consistently yield further performance gains when increasing model depth or parameter count. Therefore, in future work, we will investigate how to improve the scalability and generalization of the proposed framework as the depth and capacity of the model increase.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and improving the readability of the manuscript, as well as to support the retrieval and discovery of potentially relevant literature. All retrieved references were independently checked and verified by the authors before being included in the manuscript. AI tools were not used to generate research ideas, methodological developments, experimental results, or scientific conclusions. The authors take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

We provide sufficient details to facilitate the reproducibility of our results. The formulation and optimization procedure of FIO are described in Section 3, with detailed derivations of the analytical parameter updates, complete optimization algorithms, and training-time complexity analysis provided in the appendix. Section 4.1 specifies the datasets, input tokenization, model and baseline configurations, hardware environment, and evaluation procedure used in our experiments. All experiments are repeated over five independent runs, and the reported classification results are averaged across these runs. In addition, the anonymized source code and corresponding configuration files are provided in the supplementary material.

REFERENCES

Amin Aminifar, Baichuan Huang, Azra Abtahi, and Amir Aminifar. LightFF: Lightweight inference for the Forward-Forward algorithm. In *European Conference on Artificial Intelligence*, pp. 1728–1735, 2024.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

John Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25(9):772–781, 1978.

Paul Caillon, Alex Colagrande, Erwan Fagnou, Blaise Delattre, and Alexandre Allauzen. Forward only learning for orthogonal neural networks of any depth. *arXiv preprint arXiv:2512.20668*, 2025.

Paul Caillon, Erwan Fagnou, Blaise Delattre, and Alexandre Allauzen. Scaling direct feedback learning with Jacobian alignment guarantees. In *International Conference on Learning Representations*, 2026.

Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53:5113–5155, 2020.

Brian Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct feedback alignment with sparse connections for local learning. *Frontiers in Neuroscience*, 13:525, 2019.

Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pp. 4937–4955, 2022.

Anders Forsgren, Philip E. Gill, and Elizabeth Wong. Primal and dual active-set methods for convex quadratic programming. *Mathematical Programming*, 159:469–508, 2016.

James Gong, Bruce Li, and Waleed Abdulla. Mono-Forward: Revisiting Forward-Forward through objective-locality decomposition. *arXiv preprint arXiv:2501.09238*, 2026.

Myoung Hoon Ha, Hyunjun Kim, Yoondo Sung, Youngha Jo, Min Kang, and Sang Wan Lee. Stable and scalable deep predictive coding networks with meta-prediction errors. In *International Conference on Learning Representations*, 2026.

Donald Olding Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, 2005.

Geoffrey Hinton. The Forward-Forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7: 13276, 2016.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21:335–346, 2020.

Changze Lv, Jingwen Xu, Yiyang Lu, Xiaohua Wang, Zhenghua Wang, Zhibo Xu, Di Yu, Xin Du, Xiaoqing Zheng, and Xuanjing Huang. Dendritic localized learning: Toward biologically plausible algorithm. In *International Conference on Machine Learning*, pp. 41682–41700, 2025.

Eric Nalisnick, Padhraic Smyth, and Dustin Tran. A brief tour of deep learning from a statistical perspective. *Annual Review of Statistics and Its Application*, 10:219–246, 2023.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks. *arXiv preprint arXiv:2506.23800*, 2025.

Maria Refinetti, Stéphane d'Ascoli, Ruben Ohana, and Sebastian Goldt. Align, then memorise: The dynamics of learning with feedback alignment. In *International Conference on Machine Learning*, pp. 8925–8935, 2021.

Tommaso Salvatori, Yuhang Song, Yordan Yordanov, Beren Millidge, Lei Sha, Cornelius Emde, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. A stable, fast, and fully automatic learning algorithm for predictive coding networks. In *International Conference on Learning Representations*, 2024.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

Ravi Francesco Srinivasan, Francesca Mignacco, Martino Sorbaro, Maria Refinetti, Avi Cooper, Gabriel Kreiman, and Giorgia Dellaferrera. Forward learning with top-down feedback: Empirical and analytical characterization. In *International Conference on Learning Representations*, 2024.

James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29: 1229–1262, 2017.

James C. R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23:235–250, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jie-En Yao, Hong-En Chen, and C.-C. Jay Kuo. HCL-FF: Hierarchical and contrastive learning for Forward-Forward algorithm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27429–27438, 2026.

Umais Zahid, Qinghai Guo, and Zafeirios Fountas. Predictive coding as a neuromorphic alternative to backpropagation: A critical evaluation. *Neural Computation*, 35(12):1881–1909, 2023.

A Derivation of the Optimization of $\boldsymbol{W}$

Lemma 1. *Let $\boldsymbol{Q}$ be any real matrix, $\alpha \geq 0$, and $\beta > 0$. Then both*

$$\alpha \boldsymbol{Q}^\top \boldsymbol{Q} + \beta \boldsymbol{I} \quad \text{and} \quad \alpha \boldsymbol{Q}\boldsymbol{Q}^\top + \beta \boldsymbol{I} \tag{A.1}$$

are positive definite and therefore invertible, where the dimension of $\boldsymbol{I}$ is determined by the corresponding matrix.

Proof. For any nonzero vector $\boldsymbol{x}_1$ of compatible dimension,

$$\boldsymbol{x}_1^\top \left(\alpha \boldsymbol{Q}^\top \boldsymbol{Q} + \beta \boldsymbol{I}\right) \boldsymbol{x}_1 = \alpha \|\boldsymbol{Q}\boldsymbol{x}_1\|_2^2 + \beta \|\boldsymbol{x}_1\|_2^2 > 0. \tag{A.2}$$

Similarly, for any nonzero vector $\boldsymbol{x}_2$ of compatible dimension,

$$\boldsymbol{x}_2^\top \left(\alpha \boldsymbol{Q}\boldsymbol{Q}^\top + \beta \boldsymbol{I}\right) \boldsymbol{x}_2 = \alpha \|\boldsymbol{Q}^\top \boldsymbol{x}_2\|_2^2 + \beta \|\boldsymbol{x}_2\|_2^2 > 0. \tag{A.3}$$

□

With $\boldsymbol{a}$, $\boldsymbol{A}_0$, and $\boldsymbol{W}_0$ fixed, let $\bar{\boldsymbol{Z}} = \boldsymbol{a}\left(\boldsymbol{A}_0 \sigma(\boldsymbol{X}\boldsymbol{W}_0)\right) \in \mathbb{R}^{n \times e}$. The optimization of $\boldsymbol{W} \in \mathbb{R}^{e \times k}$ is formulated as

$$\min_{\boldsymbol{W}} \left\|\bar{\boldsymbol{Z}}\boldsymbol{W} - \boldsymbol{Y}\right\|_F^2 + \lambda_W \|\boldsymbol{W}\|_F^2, \tag{A.4}$$

where $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$ denotes the one-hot label matrix and $\lambda_W > 0$ is a sufficiently small regularization coefficient.

Setting the gradient of Eq. (A.4) with respect to $\boldsymbol{W}$ to zero and omitting the common scalar factor 2 yields

$$\bar{\boldsymbol{Z}}^\top \left(\bar{\boldsymbol{Z}}\boldsymbol{W} - \boldsymbol{Y}\right) + \lambda_W \boldsymbol{W} = 0. \tag{A.5}$$

Rearranging Eq. (A.5) yields

$$\left(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}} + \lambda_W \boldsymbol{I}_e\right) \boldsymbol{W} = \bar{\boldsymbol{Z}}^\top \boldsymbol{Y}, \tag{A.6}$$

where $\boldsymbol{I}_e \in \mathbb{R}^{e \times e}$ denotes the identity matrix.

By Lemma 1, with $\boldsymbol{Q} = \bar{\boldsymbol{Z}}$, $\alpha = 1$, and $\beta = \lambda_W > 0$, the matrix $\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}} + \lambda_W \boldsymbol{I}_e$ is positive definite and therefore invertible. Thus, the analytical solution of $\boldsymbol{W}$ is

$$\boxed{\boldsymbol{W} = \left(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}} + \lambda_W \boldsymbol{I}_e\right)^{-1} \bar{\boldsymbol{Z}}^\top \boldsymbol{Y}}. \tag{A.7}$$

B Derivation of the Optimization of $\boldsymbol{A}_0$

The optimization of $\boldsymbol{A}_0$ is formulated as

$$\min_{\boldsymbol{A}_0} \|\boldsymbol{a}\boldsymbol{A}_0\boldsymbol{B}_0 - \boldsymbol{y}\|_2^2 + \lambda_{A_0} \|\boldsymbol{A}_0\|_F^2, \tag{B.1}$$

where $\boldsymbol{a} \in \mathbb{R}^m$, $\boldsymbol{A}_0 \in \mathbb{R}^{m \times m}$, $\boldsymbol{B}_0 \in \mathbb{R}^{m \times (nk)}$, $\boldsymbol{y} \in \mathbb{R}^{(nk)}$, and $\lambda_{A_0} > 0$.

Let $\boldsymbol{\xi} = \boldsymbol{a}\boldsymbol{A}_0 \in \mathbb{R}^m$. For a fixed $\boldsymbol{\xi}$, the minimum-Frobenius-norm solution satisfying $\boldsymbol{a}\boldsymbol{A}_0 = \boldsymbol{\xi}$ is

$$\boldsymbol{A}_0 = \frac{\operatorname{vec}(\boldsymbol{a}) \operatorname{vec}^\top(\boldsymbol{\xi})}{\|\boldsymbol{a}\|_2^2}, \qquad \|\boldsymbol{A}_0\|_F^2 = \frac{\|\boldsymbol{\xi}\|_2^2}{\|\boldsymbol{a}\|_2^2}. \tag{B.2}$$

Since $\boldsymbol{a}$ is constrained to the simplex, $\|\boldsymbol{a}\|_2^2 > 0$.

Therefore, Eq. (B.1) reduces to

$$\min_{\boldsymbol{\xi}} \|\boldsymbol{\xi}\boldsymbol{B}_0 - \boldsymbol{y}\|_2^2 + \frac{\lambda_{A_0}}{\|\boldsymbol{a}\|_2^2} \|\boldsymbol{\xi}\|_2^2. \tag{B.3}$$

Setting the derivative with respect to $\boldsymbol{\xi}$ to zero and omitting the common scalar factor 2 yields

$$\boldsymbol{\xi}\left(\boldsymbol{B}_0\boldsymbol{B}_0^\top + \frac{\lambda_{A_0}}{\|\boldsymbol{a}\|_2^2}\boldsymbol{I}_m\right) = \boldsymbol{y}\boldsymbol{B}_0^\top. \tag{B.4}$$

By Lemma 1, the coefficient matrix is positive definite and therefore invertible. Thus,

$$\boldsymbol{\xi} = \boldsymbol{y}\boldsymbol{B}_0^\top \left(\boldsymbol{B}_0\boldsymbol{B}_0^\top + \frac{\lambda_{A_0}}{\|\boldsymbol{a}\|_2^2}\boldsymbol{I}_m \right)^{-1}. \tag{B.5}$$

Expressing both sides of Eq. (B.5) in row-matrix form gives

$$\mathrm{vec}^\top(\boldsymbol{\xi}) = \mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top \left(\boldsymbol{B}_0\boldsymbol{B}_0^\top + \frac{\lambda_{A_0}}{\|\boldsymbol{a}\|_2^2}\boldsymbol{I}_m \right)^{-1}. \tag{B.6}$$

Substituting $\boldsymbol{\xi}$ into Eq. (B.2) gives the analytical solution

$$\boxed{\boldsymbol{A}_0 = \mathrm{vec}(\boldsymbol{a})\,\mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top \left(\|\boldsymbol{a}\|_2^2\boldsymbol{B}_0\boldsymbol{B}_0^\top + \lambda_{A_0}\boldsymbol{I}_m \right)^{-1}}. \tag{B.7}$$

C Derivation of the Optimization of $\boldsymbol{W}_0$

Fix all columns of $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$ except its i-th column $\boldsymbol{u}_i \in \mathbb{R}^{d\times 1}$, and let $\boldsymbol{v}_i^\top \in \mathbb{R}^{1\times k}$ denote the i-th row of $\boldsymbol{W} \in \mathbb{R}^{e\times k}$. The residual excluding the contribution of $\boldsymbol{u}_i$ is

$$\boldsymbol{T} = \boldsymbol{Y} - \sum_{j\neq i} \boldsymbol{a}\boldsymbol{A}_0\sigma(\mathbf{X}\boldsymbol{u}_j)\boldsymbol{v}_j^\top, \tag{C.1}$$

where $\boldsymbol{T} \in \mathbb{R}^{n\times k}$.

For the current ReLU activation pattern, let $\mathbf{M}_i \in \mathbb{R}^{m\times n\times d}$ denote the binary mask obtained by broadcasting $\mathbb{I}(\mathbf{X}\boldsymbol{u}_i > 0) \in \mathbb{R}^{m\times n\times 1}$ along the last dimension. Keeping the activation pattern fixed gives

$$\sigma(\mathbf{X}\boldsymbol{u}_i) = (\mathbf{M}_i \odot \mathbf{X})\boldsymbol{u}_i, \tag{C.2}$$

where $\odot$ denotes the Hadamard product.

Define $\boldsymbol{G}_i = \boldsymbol{a}\boldsymbol{A}_0(\mathbf{M}_i \odot \mathbf{X})$, where $\boldsymbol{G}_i \in \mathbb{R}^{n\times d}$. Using the vectorization identity, the contribution of $\boldsymbol{u}_i$ can be written as

$$\mathrm{vec}\left((\boldsymbol{G}_i\boldsymbol{u}_i)\boldsymbol{v}_i^\top\right) = \boldsymbol{H}_i\boldsymbol{u}_i, \tag{C.3}$$

where $\boldsymbol{H}_i = \boldsymbol{v}_i \otimes \boldsymbol{G}_i \in \mathbb{R}^{(nk)\times d}$, and $\otimes$ denotes the Kronecker product.

Let

$$\widehat{\boldsymbol{T}} = \mathrm{vec}(\boldsymbol{T}) \in \mathbb{R}^{(nk)\times 1}. \tag{C.4}$$

The optimization of $\boldsymbol{u}_i$ therefore reduces to

$$\min_{\boldsymbol{u}_i} \left\| \boldsymbol{H}_i\boldsymbol{u}_i - \widehat{\boldsymbol{T}} \right\|_F^2 + \lambda_{W_0} \|\boldsymbol{u}_i\|_F^2, \tag{C.5}$$

where $\lambda_{W_0} > 0$ is the regularization coefficient.

Setting the derivative with respect to $\boldsymbol{u}_i$ to zero and omitting the common scalar factor 2 yields

$$\left(\boldsymbol{H}_i^\top\boldsymbol{H}_i + \lambda_{W_0}\boldsymbol{I}_d\right)\boldsymbol{u}_i = \boldsymbol{H}_i^\top\widehat{\boldsymbol{T}}, \tag{C.6}$$

where $\boldsymbol{I}_d \in \mathbb{R}^{d\times d}$ denotes the identity matrix.

By Lemma 1, the coefficient matrix is positive definite and therefore invertible. Thus, the analytical update of $\boldsymbol{u}_i$ is

$$\boxed{\boldsymbol{u}_i = \left(\boldsymbol{H}_i^\top\boldsymbol{H}_i + \lambda_{W_0}\boldsymbol{I}_d\right)^{-1}\boldsymbol{H}_i^\top\widehat{\boldsymbol{T}}}. \tag{C.7}$$

Sequentially applying this update for $i = 1, \ldots, e$ yields the updated $\boldsymbol{W}_0$, with the ReLU activation pattern recomputed after each column-wise update.

D Derivation of the Multi-Head Optimization

For the ℓ-th block, the multi-head optimization objective is

$$\min_{\boldsymbol{a},\{\boldsymbol{A}_{\ell,i},\boldsymbol{W}_{\ell,i}\}_{i=1}^{p},\boldsymbol{W}} \mathcal{L}_\ell = \left\| \boldsymbol{a} \left(\frac{1}{p} \sum_{i=1}^{p} \boldsymbol{A}_{\ell,i} \sigma(\mathbf{Z}_\ell \boldsymbol{W}_{\ell,i}) \right) \boldsymbol{W} - \boldsymbol{Y} \right\|_F^2, \quad \text{s.t.} \quad a_r \geq 0, \quad r = 1, \ldots, m, \qquad \sum_{r=1}^{m} a_r = 1, \tag{D.1}$$

where p denotes the number of projection heads, $\ell = 0, \ldots, L-1$ indexes the current block, and $i = 1, \ldots, p$ indexes the projection head.

To optimize the i-th projection head, we fix $\boldsymbol{a}$, $\boldsymbol{W}$, and the remaining heads. Multiplying the residual by p and collecting the fixed terms gives the effective target $\boldsymbol{T}_{\ell,i} \in \mathbb{R}^{n \times k}$, which has the same dimensionality as $\boldsymbol{Y}$:

$$\boldsymbol{T}_{\ell,i} = p\boldsymbol{Y} - \boldsymbol{a} \sum_{j \neq i} \boldsymbol{A}_{\ell,j} \sigma(\mathbf{Z}_\ell \boldsymbol{W}_{\ell,j}) \boldsymbol{W}. \tag{D.2}$$

For the fixed $\boldsymbol{a}$, $\boldsymbol{W}$, and remaining heads, Eq. (D.1) can then be rewritten as

$$p^2 \mathcal{L}_\ell = \| \boldsymbol{a} \boldsymbol{A}_{\ell,i} \sigma(\mathbf{Z}_\ell \boldsymbol{W}_{\ell,i}) \boldsymbol{W} - \boldsymbol{T}_{\ell,i} \|_F^2 . \tag{D.3}$$

Since p^2 is a positive constant, it does not affect the minimizer. Therefore, optimizing the i-th projection head is equivalent to

$$\min_{\boldsymbol{A}_{\ell,i},\boldsymbol{W}_{\ell,i}} \| \boldsymbol{a} \boldsymbol{A}_{\ell,i} \sigma(\mathbf{Z}_\ell \boldsymbol{W}_{\ell,i}) \boldsymbol{W} - \boldsymbol{T}_{\ell,i} \|_F^2 . \tag{D.4}$$

Eq. (D.4) has the same projection-parameter optimization form as Eq. (1), with $\boldsymbol{T}_{\ell,i}$ serving as the effective target for the current head. Thus, $\boldsymbol{A}_{\ell,i}$ and $\boldsymbol{W}_{\ell,i}$ can be optimized using the same analytical updates as in the single-head case while keeping $\boldsymbol{a}$, $\boldsymbol{W}$, and the remaining heads fixed.

E Optimization Procedure for the Multi-Hidden-Layer Model

Algorithm 2 Sequential Optimization for the Multi-Hidden-Layer Model

1: Optimize $\boldsymbol{a}$, $\boldsymbol{A}_0$, $\boldsymbol{W}$, and $\boldsymbol{W}_0$ using the single-hidden-layer optimization procedure.
2: Compute $\mathbf{Z}_1 = \boldsymbol{A}_0 \sigma(\mathbf{X}\boldsymbol{W}_0)$.
3: **for** $\ell = 1, \ldots, L-1$ **do**
4: Initialize $\boldsymbol{A}_\ell$ and $\boldsymbol{W}_\ell$.
5: Update $\boldsymbol{a}$, $\boldsymbol{A}_\ell$, $\boldsymbol{W}$, and $\boldsymbol{W}_\ell$ using the single-hidden-layer optimization procedure on $\mathbf{Z}_\ell$.
6: Fix the optimized parameters $\boldsymbol{A}_\ell$ and $\boldsymbol{W}_\ell$.
7: Compute $\mathbf{Z}_{\ell+1} = \boldsymbol{A}_\ell \sigma(\mathbf{Z}_\ell \boldsymbol{W}_\ell)$.
8: **end for**
9: Select the model depth based on validation performance.
10: **return** the optimized model parameters.

F Optimization Procedure for the Multi-Head Projection Model

For the ℓ-th block, the multi-head parameters are optimized iteratively as follows.

Algorithm 3 Iterative Optimization for the Multi-Head Projection Model

```
Initialize a, W, and {A_{ℓ,i}, W_{ℓ,i}}_{i=1}^p.
repeat
    Fix {A_{ℓ,i}, W_{ℓ,i}}_{i=1}^p and update a and W.
    for i = 1, ..., p do
        Fix a, W, and the remaining heads.
        Compute T_{ℓ,i} using Eq. (D.2).
        Optimize A_{ℓ,i} and W_{ℓ,i} using the projection-parameter updates of the single-head model.
    end for
until convergence of the current block
return a, W, and {A_{ℓ,i}, W_{ℓ,i}}_{i=1}^p.
```

G Training-Time Complexity Analysis

We analyze the training-time complexity of the proposed optimization procedure. For the single-hidden-layer model, let $\mathbf{X} \in \mathbb{R}^{m\times n\times d}$, $\boldsymbol{W}_0 \in \mathbb{R}^{d\times e}$, $\boldsymbol{A}_0 \in \mathbb{R}^{m\times m}$, and $\boldsymbol{W} \in \mathbb{R}^{e\times k}$. The analysis assumes standard dense matrix operations, with reusable intermediate results computed once and shared across subsequent updates. For simplicity, the computational cost of element-wise additions and subtractions is omitted.

Optimization of $\boldsymbol{a}$. The optimization of $\boldsymbol{a}$ is the following simplex-constrained least-squares problem:

$$\begin{aligned} &\min_{\boldsymbol{a}} \|\boldsymbol{a}\boldsymbol{B} - \boldsymbol{y}\|_2^2, \\ &\text{s.t.} \quad \sum_{r=1}^{m} a_r = 1, \qquad a_r \geq 0, \quad r = 1, \ldots, m, \end{aligned} \tag{G.1}$$

where $\boldsymbol{B} \in \mathbb{R}^{m\times(nk)}$ and $\boldsymbol{y} \in \mathbb{R}^{(nk)}$. Since $\boldsymbol{B}$ and $\boldsymbol{y}$ are already available when this subproblem is solved, their construction costs are not included in the complexity of this parameter update. The same convention applies to the complexity analyses of $\boldsymbol{W}$, $\boldsymbol{A}_0$, and $\boldsymbol{W}_0$.

Let $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{a}) \in \mathbb{R}^{m\times 1}$. Expanding the objective in explicit column form and omitting the constant term $\|\boldsymbol{y}\|_2^2$, the problem can be written in the standard quadratic programming form

$$\begin{aligned} &\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}, \\ &\text{s.t.} \quad -\boldsymbol{I}_m \boldsymbol{x} \leq \boldsymbol{0}, \qquad \boldsymbol{1}^\top \boldsymbol{x} = 1, \end{aligned} \tag{G.2}$$

where $\boldsymbol{P} = 2\boldsymbol{B}\boldsymbol{B}^\top \in \mathbb{R}^{m\times m}$, $\boldsymbol{q} = -2\boldsymbol{B}\,\mathrm{vec}(\boldsymbol{y}) \in \mathbb{R}^{m\times 1}$, and $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ and $\boldsymbol{1}^\top \in \mathbb{R}^{1\times m}$ denote the identity matrix and the all-ones vector.

Constructing $\boldsymbol{P}$ requires $\mathcal{O}(m^2nk)$ operations, while constructing $\boldsymbol{q}$ requires $\mathcal{O}(mnk)$ operations. Therefore, constructing the quadratic program is dominated by

$$\mathcal{O}(m^2nk). \tag{G.3}$$

Let τ_A denote the number of active-set iterations. At each iteration, the search direction is obtained by solving the KKT system associated with an equality-constrained quadratic subproblem over the current free variables (Forsgren et al., 2016). Let $s \leq m$ denote the number of free variables at a given iteration. Since our simplex constraint contains a single equality constraint, the corresponding KKT system has dimension $(s+1)\times(s+1)$. In the worst case, all m variables are free, yielding a system of dimension $(m+1)\times(m+1)$. A dense direct solve therefore requires

$$\mathcal{O}\left((m+1)^3\right) = \mathcal{O}\left(m^3\right) \tag{G.4}$$

operations per iteration. Therefore, including the construction of the quadratic program, the overall complexity of updating $\boldsymbol{a}$ is

$$\mathcal{O}\left(m^2nk + \tau_A m^3\right). \tag{G.5}$$

In our implementation, the active-set iterations are terminated when the improvement becomes negligible rather than requiring convergence to an exact optimum. Empirically, τ_A is typically on the order of 10 across our experiments. Under the considered settings, we therefore treat τ_A as an empirical constant with respect to m and n. Since the number of classes k is also small and fixed for each dataset, the practical complexity of updating $\boldsymbol{a}$ can be summarized as

$$\mathcal{O}\left(m^2 n + m^3\right). \tag{G.6}$$

Optimization of $\boldsymbol{W}$. The analytical solution of $\boldsymbol{W}$ is

$$\boldsymbol{W} = \left(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}} + \lambda_W \boldsymbol{I}_e\right)^{-1} \bar{\boldsymbol{Z}}^\top \boldsymbol{Y}, \tag{G.7}$$

where $\bar{\boldsymbol{Z}} \in \mathbb{R}^{n \times e}$. Computing $\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}}$ requires $\mathcal{O}(ne^2)$ operations, while computing $\bar{\boldsymbol{Z}}^\top \boldsymbol{Y}$ requires $\mathcal{O}(nek)$. The inversion of the resulting $e \times e$ matrix requires $\mathcal{O}(e^3)$, and multiplying the inverse by $\bar{\boldsymbol{Z}}^\top \boldsymbol{Y}$ requires $\mathcal{O}(e^2 k)$. Therefore, the overall complexity of updating $\boldsymbol{W}$ is

$$\mathcal{O}\left(ne^2 + nek + e^3 + e^2 k\right). \tag{G.8}$$

Since the number of classes k is typically a small constant, the terms involving k are dominated by the other terms. Thus, the complexity can be simplified as

$$\mathcal{O}\left(ne^2 + e^3\right). \tag{G.9}$$

Optimization of $\boldsymbol{A}_0$. The analytical solution of $\boldsymbol{A}_0$ is

$$\boldsymbol{A}_0 = \mathrm{vec}(\boldsymbol{a})\,\mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top \left(\|\boldsymbol{a}\|_2^2 \boldsymbol{B}_0 \boldsymbol{B}_0^\top + \lambda_{A_0} \boldsymbol{I}_m\right)^{-1}, \tag{G.10}$$

where $\boldsymbol{B}_0 \in \mathbb{R}^{m \times (nk)}$. Computing $\mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top$ requires $\mathcal{O}(mnk)$ operations. Computing $\|\boldsymbol{a}\|_2^2$ requires $\mathcal{O}(m)$ operations, while computing $\boldsymbol{B}_0 \boldsymbol{B}_0^\top$ requires $\mathcal{O}(m^2 nk)$ operations. The inversion of the resulting $m \times m$ matrix requires $\mathcal{O}(m^3)$ operations. After $\mathrm{vec}^\top(\boldsymbol{y})\boldsymbol{B}_0^\top$ and the inverse matrix are available, multiplying them and forming the outer product with $\mathrm{vec}(\boldsymbol{a})$ require $\mathcal{O}(m^2)$ operations. Therefore, the overall complexity of updating $\boldsymbol{A}_0$ is

$$\mathcal{O}\left(m^2 nk + mnk + m + m^2 + m^3\right). \tag{G.11}$$

Since the number of classes k is typically a small constant and the lower-order terms are dominated, the complexity can be simplified as

$$\mathcal{O}\left(m^2 n + m^3\right). \tag{G.12}$$

Optimization of $\boldsymbol{W}_0$. For the i-th column $\boldsymbol{u}_i$ of $\boldsymbol{W}_0$, we first compute

$$\boldsymbol{G}_i = \boldsymbol{a}\boldsymbol{A}_0(\mathbf{M}_i \odot \mathbf{X}), \tag{G.13}$$

where $\boldsymbol{G}_i \in \mathbb{R}^{n \times d}$. Since $\mathbf{M}_i$ is obtained directly from the indicator function, its construction does not involve matrix multiplication and is not included in the multiplication complexity. As $\boldsymbol{a}$ and $\boldsymbol{A}_0$ are fixed during the update of $\boldsymbol{W}_0$, $\boldsymbol{a}\boldsymbol{A}_0$ can be computed once with $\mathcal{O}(m^2)$ operations and reused for all columns.

For each $\boldsymbol{u}_i$, computing $\boldsymbol{G}_i$ then requires $\mathcal{O}(mnd)$ operations. The corresponding matrix $\boldsymbol{H}_i$ is

$$\boldsymbol{H}_i = \boldsymbol{v}_i \otimes \boldsymbol{G}_i, \tag{G.14}$$

where $\boldsymbol{H}_i \in \mathbb{R}^{(nk) \times d}$. Since $\boldsymbol{v}_i \in \mathbb{R}^{k \times 1}$ and $\boldsymbol{G}_i \in \mathbb{R}^{n \times d}$, the Kronecker product contains knd entries (Brewer, 1978). Therefore, explicitly constructing $\boldsymbol{H}_i$ requires $\mathcal{O}(knd)$ operations.

The analytical update of $\boldsymbol{u}_i$ is

$$\boldsymbol{u}_i = \left(\boldsymbol{H}_i^\top \boldsymbol{H}_i + \lambda_{W_0} \boldsymbol{I}_d\right)^{-1} \boldsymbol{H}_i^\top \widehat{\boldsymbol{T}}. \tag{G.15}$$

Computing $\boldsymbol{H}_i^\top \boldsymbol{H}_i$ requires $\mathcal{O}(nkd^2)$ operations, while computing $\boldsymbol{H}_i^\top \widehat{\boldsymbol{T}}$ requires $\mathcal{O}(nkd)$. The inversion of the resulting $d \times d$ matrix requires $\mathcal{O}(d^3)$ operations, and the final multiplication requires $\mathcal{O}(d^2)$. Therefore, excluding the shared computation of $\boldsymbol{a}\boldsymbol{A}_0$, the complexity of updating a single column $\boldsymbol{u}_i$ is

$$\mathcal{O}\left(mnd + nkd + nkd^2 + d^3 + d^2\right). \tag{G.16}$$

Since k is typically a small constant and the lower-order terms are dominated, this can be simplified as

$$\mathcal{O}\left(mnd + nd^2 + d^3\right). \tag{G.17}$$

Since $\boldsymbol{W}_0 \in \mathbb{R}^{d \times e}$ contains e columns that are updated sequentially, the overall complexity of updating $\boldsymbol{W}_0$ is

$$\mathcal{O}\left(m^2 + emnd + end^2 + ed^3\right). \tag{G.18}$$

Single-Hidden-Layer Model. Combining the above parameter updates, the computational complexity of one alternating optimization iteration for the single-hidden-layer model is

$$\mathcal{O}\left(m^2n + m^3 + ne^2 + e^3 + emnd + end^2 + ed^3\right). \tag{G.19}$$

Let τ denote the number of alternating optimization iterations required for convergence. The overall training-time complexity of the single-hidden-layer model is therefore

$$\mathcal{O}\left(\tau\left(m^2n + m^3 + ne^2 + e^3 + emnd + end^2 + ed^3\right)\right). \tag{G.20}$$

In practice, the optimization typically converges within approximately 10 iterations, so τ can be regarded as a small constant and absorbed into the Big-$\mathcal{O}$ notation. Therefore, the overall complexity can be simplified as

$$\boxed{\mathcal{O}\left(m^2n + m^3 + ne^2 + e^3 + emnd + end^2 + ed^3\right)}. \tag{G.21}$$

Multi-Hidden-Layer Model. For each subsequently added hidden layer, the input representation satisfies $\mathbf{Z}_\ell \in \mathbb{R}^{m\times n\times e}$ and $\boldsymbol{W}_\ell \in \mathbb{R}^{e\times e}$. Substituting $d = e$ into the single-hidden-layer complexity and omitting the dominated terms, the complexity of optimizing one subsequently added hidden layer is

$$\mathcal{C}_{\text{hidden}} = \mathcal{O}\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right). \tag{G.22}$$

For the first hidden layer, the input feature dimension is d rather than e. Since $d \le e$ in our settings, its complexity is upper-bounded by $\mathcal{C}_{\text{hidden}}$. Let τ denote the maximum number of alternating optimization iterations across all layers. Therefore, for a model with L hidden layers, the overall training-time complexity is upper-bounded by

$$\mathcal{O}\left(L\tau\,\mathcal{C}_{\text{hidden}}\right). \tag{G.23}$$

In practice, τ is typically on the order of 10 and can therefore be regarded as a small constant. The overall complexity can thus be simplified as

$$\boxed{\mathcal{O}\left(L\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right)\right)}. \tag{G.24}$$

Multi-Head Projection Model. For the multi-head model, each optimization iteration first fixes $\{\boldsymbol{A}_{\ell,i}, \boldsymbol{W}_{\ell,i}\}_{i=1}^{p}$ and updates the shared parameters $\boldsymbol{a}$ and $\boldsymbol{W}$. According to the previous analysis, their combined complexity is

$$\mathcal{C}_{\text{shared}} = \mathcal{O}\left(m^2n + m^3 + ne^2 + e^3\right). \tag{G.25}$$

The p projection heads are then optimized sequentially. For the i-th head, $\boldsymbol{A}_{\ell,i}$ and $\boldsymbol{W}_{\ell,i}$ are optimized using the projection-parameter updates of the single-head model. Since $\mathbf{Z}_\ell \in \mathbb{R}^{m\times n\times e}$ and $\boldsymbol{W}_{\ell,i} \in \mathbb{R}^{e\times e}$, the complexity of optimizing one head is

$$\mathcal{C}_{\text{proj}} = \mathcal{O}\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right). \tag{G.26}$$

Under the reuse assumption stated above, the effective targets $\boldsymbol{T}_{\ell,i}$ are formed from cached projection contributions and do not change the leading-order complexity. Since $\mathcal{C}_{\text{shared}} = \mathcal{O}(\mathcal{C}_{\text{proj}})$, the complexity of one complete optimization iteration for a multi-head block is

$$\mathcal{O}\left(p\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right)\right). \tag{G.27}$$

Let τ denote the number of optimization iterations required for convergence of the current block. The overall training-time complexity of optimizing the current multi-head block is therefore

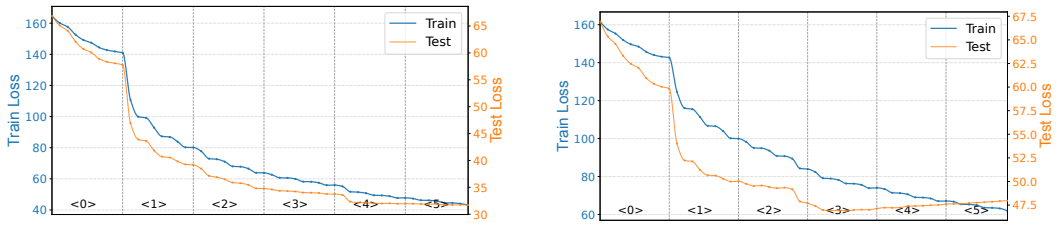
$$\mathcal{O}\left(\tau p\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right)\right). \tag{G.28}$$

In practice, τ is typically on the order of 10 and can be regarded as a small constant. Therefore, the training-time complexity of one multi-head block can be written as

$$\boxed{\mathcal{O}\left(p\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right)\right)}. \tag{G.29}$$

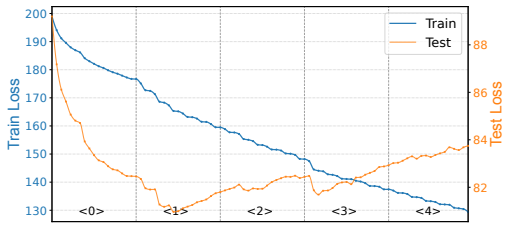
Using the same upper-bound argument for the first block, the overall training-time complexity of an L-block multi-head model is

$$\boxed{\mathcal{O}\left(Lp\left(m^2n + m^3 + mne^2 + ne^3 + e^4\right)\right)}. \tag{G.30}$$

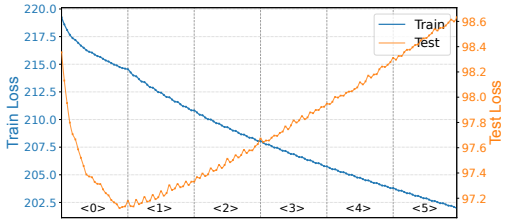


(a) MNIST

(b) Fashion-MNIST



(c) CIFAR-10



(d) CIFAR-100

Figure H.1: Training and test loss curves of Residual FIO on different datasets. Each partition corresponds to the optimization of one block, and the blue and orange curves represent the training and test losses.

Table H.1: Classification accuracy (%) of FIO with and without residual connections. The number in parentheses indicates the model depth corresponding to the best accuracy.

Method	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
Original FIO	98.65 (2)	90.69 (2)	60.91 (4)	28.05 (4)
Residual FIO	96.55 (5)	90.23 (4)	59.64 (2)	26.56 (1)

H Residual Connections

In this section, we investigate the effect of residual connections on the proposed FIO framework. We keep the initial multi-head feature transformation unchanged, such that $\mathbf{Z}_1 = p^{-1}\sum_{i=1}^{p} \boldsymbol{A}_{0,i}\sigma\left(\mathbf{X}\boldsymbol{W}_{0,i}\right) \in \mathbb{R}^{m\times n\times e}$ denotes the input representation to the first residual block. For the subsequent blocks indexed by $\ell = 1, \ldots, L-1$, residual connections are introduced while the multi-head projection structure remains unchanged. Given $\mathbf{Z}_\ell \in \mathbb{R}^{m\times n\times e}$, the output representation of the ℓ-th block is

$$\mathbf{Z}_{\ell+1} = \mathbf{Z}_\ell + \frac{1}{p}\sum_{i=1}^{p} \boldsymbol{A}_{\ell,i}\sigma\left(\mathbf{Z}_\ell\boldsymbol{W}_{\ell,i}\right), \tag{H.1}$$

where $\mathbf{Z}_{\ell+1} \in \mathbb{R}^{m\times n\times e}$ shares the same dimensionality as $\mathbf{Z}_\ell$, and p denotes the number of projection heads. Thus, the projected representation of the current block is added to its input before being passed to the subsequent block.

Accordingly, the optimization objective for the ℓ-th block is

$$\begin{aligned} \min_{\boldsymbol{a},\{\boldsymbol{A}_{\ell,i},\boldsymbol{W}_{\ell,i}\}_{i=1}^{p},\boldsymbol{W}} \mathcal{L}_\ell &= \left\| \boldsymbol{a}\left(\mathbf{Z}_\ell + \frac{1}{p}\sum_{i=1}^{p} \boldsymbol{A}_{\ell,i}\sigma\left(\mathbf{Z}_\ell\boldsymbol{W}_{\ell,i}\right)\right)\boldsymbol{W} - \boldsymbol{Y}\right\|_F^2, \\ \text{s.t.} \quad a_r \geq 0, \quad r = 1,\ldots,m, &\qquad \sum_{r=1}^{m} a_r = 1. \end{aligned} \tag{H.2}$$

Eq. (H.2) has the same multi-head structure as Eq. (D.1), with an additional residual term $\mathbf{Z}_\ell$. When optimizing the i-th projection head, this residual term is fixed and can be absorbed into the effective

target as

$$\boldsymbol{T}_{\ell,i}^{\mathrm{res}} = p\boldsymbol{Y} - p\boldsymbol{a}\boldsymbol{Z}_\ell\boldsymbol{W} - \boldsymbol{a}\sum_{j\neq i}\boldsymbol{A}_{\ell,j}\sigma\left(\boldsymbol{Z}_\ell\boldsymbol{W}_{\ell,j}\right)\boldsymbol{W}. \tag{H.3}$$

The resulting subproblem has the same form as Eq. (D.4). Therefore, the residual model can be optimized using the same multi-head optimization procedure in Algorithm 3, with $\boldsymbol{T}_{\ell,i}^{\mathrm{res}}$ used as the effective target.

We use the same dataset, tokenization, hidden-dimension, and head configurations as in Section 4.1, while varying the model depth. Table H.1 reports the best classification accuracy with and without residual connections, while Fig. H.1 presents the corresponding training and test loss curves.

As shown in Fig. H.1, the effect of residual connections varies across the two dataset families. For MNIST and Fashion-MNIST, the test loss continues to decrease as the first several residual blocks are optimized, reaching its minimum around the third or fourth added block and remaining relatively stable thereafter. In contrast, for CIFAR-10 and CIFAR-100, the minimum test loss is reached at a much shallower depth, after which introducing additional residual blocks generally increases the test loss. This indicates that the smoother optimization behavior induced by residual connections does not translate into better generalization. Table H.1 further shows that Residual FIO achieves lower best accuracy than the original FIO on all four datasets.