

FROST: Forward-Forward Round-Robin Optical-Compatible Softmax-Free Transformer

MSc Thesis
Denis Krylov

Delft University of Technology

FROST: Forward-Forward Round-Robin Optical-Compatible Softmax-Free Transformer

by

Denis Krylov

to obtain the degree of Master of Science at the Delft University of Technology.

Student number: 6277470
Project duration: 28/10/2025 - 09/06/2026
Thesis committee: David Tax, TU Delft, Chair
Stephanie Tan, TU Delft, Daily supervisor
Richard Norte, TU Delft, External examiner

Preface

This thesis is the result of my MSc graduation project in Data Science and Artificial Intelligence Technology at Delft University of Technology, on training a Transformer without backpropagation.

When I started, I underestimated how hard the central goal would be. Getting a Transformer to learn anything at all from forward passes alone - no backward pass, no automatic differentiation, and no softmax - took far more than swapping one component for another. Many configurations simply refused to train, and it was only through a long sequence of small design decisions, several of which began as failures, that the model eventually reached a useful operating point. That process taught me as much about why ordinary backpropagation works as about the alternative I was trying to build.

I could not have done it without the constant guidance and support of my supervisors, Dr. Stephanie Tan and Dr. David Tax. Their feedback during our weekly meetings kept the project on course through its dead ends, and their patience gave me the room to find my own way to the result. I am also grateful to the photonics research team, in particular Dr. Richard Norte and Dr. Sid Kumar, for the discussions that kept the machine-learning choices honest about what a real chip can and cannot do.

Last but not least, I thank my family and friends. Their encouragement carried me through the long stretches when nothing seemed to work, and they reminded me, again and again, that there was a world beyond the loss curves. This work is as much a product of their patience as of my own, and I am deeply grateful for it.

*Denis Krylov
Delft, June 2026*

Abstract

Training state-of-the-art AI models is growing far more costly than digital hardware can sustain, motivating analog photonic substrates that perform matrix multiplication directly in the propagation of light. Inference on such hardware is established, but training is not: backpropagation requires a backward pass and exact gradients that an analog chip cannot natively provide. Forward-only, backpropagation-free learning offers a way around this, yet it has been demonstrated only for multilayer perceptrons and convolutional networks, never for the Transformer, the architecture that now dominates AI and whose self-attention couples every position across the sequence, resisting purely local objectives.

This thesis asks whether a Transformer can be trained using only the forward-pass operations a photonic substrate provides, and characterizes what such training costs. The proposed method combines a layer-wise Forward-Forward prototype-based objective, directional-derivative gradient estimation (with no backward pass and no automatic differentiation), and a softmax-free Spherical attention adapted from the Kramers-Kronig kernel, with the four attention projections trained one at a time in a round-robin schedule; six training variants are compared across seven vision and sequence tasks to isolate the gradient estimator and the update schedule. The answer is affirmative: the fully forward-only variant trains a Transformer to a useful operating point on six of the seven tasks. Locality, not the forward-only gradient alone, is what makes this possible, and the depth-resilience of local learning is shown to extend, conditionally, to self-attention. The remaining gap to backpropagation has three sources, two inherent to forward-only learning (local credit assignment and gradient-estimation variance) and one architectural (the softmax-free attention cannot form the sharp selection that content-addressed retrieval requires). Because every operation reduces to a forward pass and a measured scalar loss, the method is a candidate for in-situ training on a photonic chip, the validation step this work points to.

Contents

Preface	i
Abstract	ii
Nomenclature	iv
1 Introduction	1
2 Background	4
2.1 Neural Network Training	4
2.2 Transformer Architecture	5
2.3 Physical and photonic neural networks	7
2.4 Limitations of Backpropagation and Backpropagation-Free Alternatives	8
2.5 The Forward-Forward Algorithm and its Variants	10
3 Methodology	17
3.1 Overview and design rationale	17
3.2 Transformer Architecture	17
3.3 Training Methods	20
3.4 Experimental Protocol	25
4 Results	26
4.1 Vision Tasks	26
4.2 Sequence Tasks	30
4.3 Summary	36
5 Discussion & Conclusion	39
5.1 Principal findings	39
5.2 Limitations	39
5.3 Addressable Bottlenecks and Future Work	40
5.4 Implications and real-world translation	40
5.5 Conclusion	40
References	42
A Model Configurations	45
B Kramers-Kronig Attention	47
C Declaration of Generative AI Usage	49

Nomenclature

Abbreviations

Abbreviation	Definition
AD	Automatic Differentiation
AdamW	Adam optimizer with decoupled weight decay
BP	Backpropagation
CaFo	Cascaded Forward algorithm
CLS	Classification token
CNN	Convolutional Neural Network
CwC	Channel-wise Competitive Learning
DD	Directional Derivative
DFA	Direct Feedback Alignment
DF	Distance-Forward learning
EP	Equilibrium Propagation
FA	Feedback Alignment
FF	Forward-Forward
FF-AD	Forward-Forward with Automatic Differentiation
FF-DD	Forward-Forward with Directional Derivatives
FFN	Feed-Forward Network
FF-Joint	Forward-Forward with joint update of all attention projections
FFzero	Forward-only learning framework
GELU	Gaussian Error Linear Unit
GHL	Global-guided Hebbian Learning
GRU	Gated Recurrent Unit
HPN	Hyperspherical Prototype Network
ILT	Interleaved Layer Training
KKA	Kramers-Kronig Attention
KKR	Kramers-Kronig Relationship
LN	Layer Normalization
LSTM	Long Short-Term Memory
MAC	Multiply-Accumulate (operation)
MeZO	Memory-efficient Zeroth-Order optimizer
MHA	Multi-Head Attention
MHSA	Multi-Head Self-Attention
MLP	Multi-Layer Perceptron
MZI	Mach-Zehnder Interferometer
NLP	Natural Language Processing
Pre-LN	Pre-Layer Normalization
RNG	Random Number Generator
RNN	Recurrent Neural Network
RoPE	Rotary Position Embedding
RR	Round-Robin
SCFF	Self-Contrastive Forward-Forward
SFF	Scalable Forward-Forward
SGD	Stochastic Gradient Descent
SNN	Spiking Neural Network
SPSA	Simultaneous Perturbation Stochastic Approximation
STDP	Spike-Timing-Dependent Plasticity

Abbreviation	Definition
S2TDPT	Spiking STDP Transformer
ViT	Vision Transformer
ZO	Zeroth-Order

Introduction

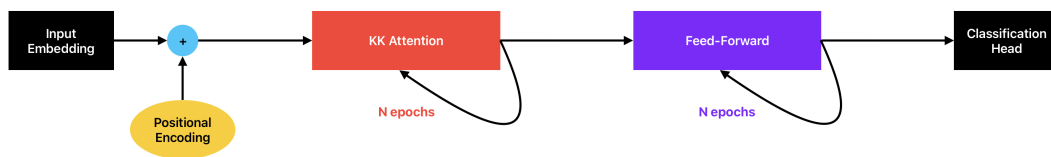


Figure 1.1: Schematic of FF-Transformer training paradigm.

The computational and energy required to train and deploy state-of-the-art AI models is growing considerably faster than digital hardware capacity. The main issue is in the processing of large matrices, which needs to be done step by step on digital hardware. By contrast, optical physical neural networks realize these operations directly in the propagation of light, where the per-operation energy of a dot product scales with the inverse of the vector length rather than remaining fixed as in digital multiply-accumulate hardware; at sufficiently large model scale this yields a favorable energy-scaling regime [1]. One such example is an integrated photonic deep neural network capable of performing end-to-end image classification directly on a single silicon chip [2]. Another example is the photonic transformer chip, which implements self-attention purely through optical interference [3]. Morsali et al. [4] further demonstrate that a hybrid electronic-photonic ViT accelerator can cut energy consumption by up to 84% while incurring under 1.6% degradation relative to the full-precision model.

The remaining bottleneck in these photonic learning approaches is training. Most photonic networks are trained in silico, with both the forward and backward passes executed on a differentiable digital twin of the chip. The accuracy of such training is limited by how faithfully the twin captures the hardware: fabrication tolerances, thermal drift and detection noise open a simulation-reality gap that degrades performance once parameters are transferred to the physical device [1]. Training directly on the chip removes the simulation-reality gap, but backpropagation is hard to implement in photonics: the principal obstacle is realizing the nonlinear activation and its derivative on-chip rather than in software. Ashtiani et al. [5] addressed this with the first fully integrated demonstration of on-chip gradient-descent training, in which a separate backward path mirrors the forward path using nominally identical photonic devices - an approach that, by construction, calls for a second set of weight and activation blocks. This problem can be solved by the hardware reuse, though physical backpropagation still imposes a trade-off between hardware scalability and existence of simulation-reality gap.

A second route of work avoids the backward pass completely. Bandyopadhyay et al [6] introduce an end-to-end coherent optical deep neural network, three-layer perceptron with six neurons per layer, that performs both inference and in-situ training on a single silicon photonic chip. Rather than probing one parameter at a time, the system perturbs all weights simultaneously along a single random direction Δ , measures the loss twice on-chip, and updates the parameters using the directional derivative. The chip reaches 92.5% test accuracy on a six-class vowel classification task, matching a digital model of the same size. The update rule, however, is global rather than layer-wise: a single scalar directional derivative

drives every parameter at once. Experimentally, this scheme has so far been demonstrated only on a shallow feedforward network; its behavior on deeper models, or on architectures with attention-style non-local coupling, remains unexplored.

There are also software-approaches reaching for the same forward-only premise. Hinton’s Forward-Forward algorithm [7] trains each layer with a local objective computed from two forward passes, one on positive and one on negative data, removing global error propagation. The FFzero framework [8] extends this idea in two directions that matter for physical learning. First, the positive/negative-data scheme of the original Forward-Forward method is discarded entirely: each class is assigned a fixed reference vector, drawn once at random subject to a maximal-separation constraint, and goodness is redefined as the alignment between a layer’s normalized output and the reference of the correct class. Every layer is trained locally to make the normalized output align with the prototype of the correct class. This principle generalizes for both classification and regression problems. Second, both automatic differentiation and the backward pass are removed: each layer-wise objective is optimized by perturbing all of its weights along a random direction and estimating a single directional derivative from two forward evaluations. The main finding is that backpropagation trained with the same directional-derivative estimator collapses as depth growth, because of estimation error compound. FFzero remains stable on the other hand, because each layer is optimized independently and this errors don’t propagate across layers. The framework was demonstrated on multilayer perceptrons (MLP) and convolutional neural networks (CNN) across classification and regression tasks, and on simulated photonic substrate as a proof of concept for in-situ training.

Forward-Forward has very recently been applied to Transformers, but only at block or sub-layer granularity and with automatic differentiation retained inside each block; the prototype-based, autodiff-free variant (FFzero) has been demonstrated only on MLPs and CNNs (Chapter 2). A fully backpropagation-free Transformer, the architecture that now dominates modern AI, therefore remains unaddressed, which makes its physical implementation a critical milestone for optical computing. Furthermore, Transformers present a unique algorithmic challenge for local learning rules: their self-attention mechanism fundamentally couples positions across the entire sequence, which inherently resists purely local, layer-wise objectives. Therefore, determining whether these forward-only principles can successfully optimize a Transformer is essential to bridging the gap between photonic hardware capabilities and state-of-the-art AI architectures. The implemented architecture replaces standard softmax-based attention with a softmax-free mechanism, termed Spherical attention, inspired by Kramers-Kronig attention (KKA) [3]; the training procedure is built around a layer-wise Forward-Forward objective with directional-derivative updates [8]. The central contribution is to show that a Transformer can be trained by a forward-only, local rule at all, and to identify where this succeeds and where it breaks down. Enabling this required two design choices: the softmax-free Spherical attention above, and a Round-Robin schedule that trains the four attention projections one at a time within each block rather than jointly, staying faithful to the one-component-at-a-time locality of Forward-Forward and giving each projection its own dedicated local gradient signal. A systematic ablation within the photonic-compatible setup isolates two design choices: the gradient estimator (autograd vs. directional derivative) and the attention-projection training schedule (round-robin vs. joint). The full photonic-compatible system is then compared end-to-end against a standard Transformer trained with backpropagation, so that the headline accuracy gap reports the total cost of moving to a forward-only, photonic-compatible architecture rather than an isolated optimizer or architecture effect. The illustration of the transformer training paradigm idea is shown in Figure 1.1.

Across vision-classification and sequence-to-sequence benchmarks, the Forward-Forward Transformer reaches a useful operating regime rather than matching backpropagation outright. With exact local gradients it equals the backpropagation baseline on the near-linearly-separable MNIST task, where a single block suffices, and trains well on the algorithmic transduction tasks, but it trails on problems that require composing information across positions and fails on content-addressed retrieval (Recall task), where the softmax-free attention cannot form the sharp selection the task needs. The fully forward-only variant with directional-derivative updates, preserves this ranking at lower absolute accuracy, staying above the random baseline on six of the seven tasks; its one systematic weakness is the tasks that demand precise per-position decisions (Argsort), where the noise of the single-direction estimate is most damaging.

The remainder of this thesis is structured as follows. Chapter 2 provides the theoretical background, covering traditional neural network and Transformer training, basics of photonic neural networks, the physical limitations of standard backpropagation, and existing backpropagation-free methods such as the Forward-Forward algorithm and its variations. Chapter 3 describes the methodology, introducing a hardware-compatible Transformer architecture that replaces standard attention with softmax-free Spherical attention. This chapter also outlines the proposed layer-wise forward-forward training pipeline utilizing directional-derivative updates rather than automatic differentiation. Chapter 4 presents the experimental results, evaluating the proposed methods against standard backpropagation across various vision and sequence tasks, alongside an analysis of the computational costs. Finally, chapter 5 discusses the main findings, evaluates the feasibility of translating these methods to real-world photonic hardware, acknowledges the limitations of the current approach, and provides recommendations for future work.

2

Background

2.1. Neural Network Training

Neural networks represent data through compositions of non-linear transformations across multiple layers [9]. A standard Multi-Layer Perceptron (MLP) consists of an input layer, one or more hidden layers, and an output layer. In the forward pass, each layer l computes an activation vector $\mathbf{a}^{(l)}$ by applying a non-linear activation function ϕ element-wise to an affine transformation of the previous layer's output:

$$\mathbf{a}^{(l)} = \phi\left(W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad (2.1)$$

where $W^{(l)}$ is the weight matrix, $\mathbf{b}^{(l)}$ is the bias vector, $\mathbf{a}^{(0)}$ is the input. The pre-activation $\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ is passed through ϕ to introduce the non-linearity needed to learn complex mappings.

Training minimizes a scalar loss $\mathcal{L}$ between the network's predictions and the ground-truth labels. Gradients are computed via *backpropagation* (BP), which applies the chain rule recursively from the output layer back to the input [10]. The error signal $\delta^{(l)}$ at layer l is obtained by propagating the downstream error backward through the transposed weight matrix:

$$\delta^{(l)} = \left(W^{(l+1)\top} \delta^{(l+1)}\right) \odot \phi'\left(\mathbf{z}^{(l)}\right), \quad (2.2)$$

where $\odot$ denotes element-wise multiplication, ϕ' is the derivative of the activation function applied element-wise, and $\delta^{(l)} := \partial\mathcal{L}/\partial\mathbf{z}^{(l)}$ is the error signal at layer l .

The recursion is initialized at the output layer L by

$$\delta^{(L)} = \nabla_{\mathbf{a}^{(L)}} \mathcal{L} \odot \phi'\left(\mathbf{z}^{(L)}\right), \quad (2.3)$$

where $\nabla_{\mathbf{a}^{(L)}} \mathcal{L}$ is the gradient of the loss with respect to the network's output.

Each layer's parameters are then updated by gradient descent:

$$W^{(l)} \leftarrow W^{(l)} - \eta \delta^{(l)} \left(\mathbf{a}^{(l-1)}\right)^\top, \quad \mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \delta^{(l)}, \quad (2.4)$$

where η is the learning rate.

Evaluating these gradients by hand is impractical for deep architectures. Modern frameworks such as PyTorch [11] instead rely on *automatic differentiation* (AD): the forward pass records a computational graph, and the chain rule is applied to this graph in reverse. Reverse-mode AD recovers every parameter

gradient in a single backward traversal whose cost is proportional to the forward pass, which is what makes BP usable at depth. The procedure is, however, inherently global. The backward sweep assumes that every operation in the forward pass admits a transpose or inverse that can be applied in the opposite direction, and the intermediate activations $\mathbf{a}^{(l)}$ produced during the forward pass have to be retained until their gradients are consumed.

2.2. Transformer Architecture

Historically, dominant models for sequence transduction tasks, such as machine translation and language modeling [12], [13], were based on recurrent neural networks (RNNs), particularly long short-term memory (LSTM) networks [14] and gated recurrent units (GRUs) [15]. These architectures typically followed an encoder-decoder structure in which an encoder maps an input sequence to a continuous representation, and a decoder generates the output sequence token by token. Although attention mechanisms were later integrated to relate signals across long distances, they were usually combined with recurrence.

A primary drawback of recurrent models (such as LSTMs and GRUs) is their inherently sequential computation. They compute hidden states h_t as a function of the previous state h_{t-1} and the current input. This design makes long-range dependencies harder to learn because the path length between distant positions grows with sequence distance. It also limits parallelization within training examples, which becomes a major bottleneck for long sequences.

Vaswani et al. (2017) [16] proposed the Transformer, an architecture in which self-attention, rather than recurrence or convolution, carries the full burden of representing inputs and outputs. As a consequence, computation parallelizes far more readily and the distance any two tokens must traverse to interact becomes independent of sequence length. The proposed architecture is shown in Figure 2.1.

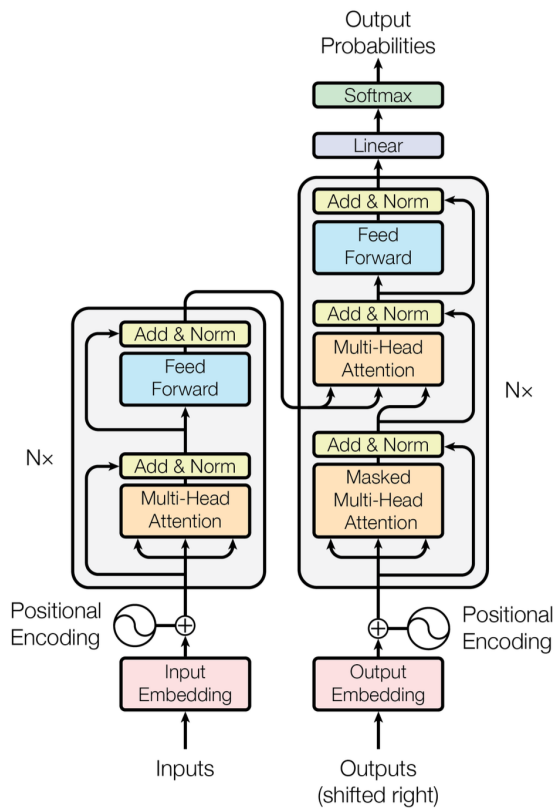


Figure 2.1: The original Transformer architecture. Left: encoder block; right: decoder block. Figure reproduced from Vaswani et al. [16].

Because the Transformer has neither recurrence nor convolution, it has no built-in notion of token

order before passing the data to encoder. To encode position information, positional encodings are added to input embeddings. The original model uses sine and cosine functions at different frequencies [16]. Applying the Transformer to images requires converting a 2D input into a sequence of tokens. Following Dosovitskiy et al. [17], an image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ is divided into N non-overlapping patches of size $P \times P$, giving $N = HW/P^2$ patches. Each patch is flattened and projected to a d_{model} -dimensional embedding via a learned linear layer, yielding a token sequence $X \in \mathbb{R}^{N \times d_{\text{model}}}$. A learnable classification token [CLS] is prepended to this sequence, and its final representation is used as the aggregate image representation passed to the prediction head. Because the Transformer has no built-in notion of token order, positional encodings are added to the patch embeddings before the first encoder block to inject spatial information.

The encoder itself is composed of two structurally distinct sub-layers: multi-head attention (MHA) and a position-wise feedforward network (FFN), each wrapped by a residual connection and layer normalization. Whereas the original Transformer normalizes the output of each sub-layer after the residual addition [16], the variant adopted here follows the pre-normalization convention used by the Vision Transformer, in which layer normalization precedes each sub-layer [17]. A single encoder block then takes the form

$$\tilde{X} = X + \text{MHA}(\text{LN}(X)), \quad Y = \tilde{X} + \text{FFN}(\text{LN}(\tilde{X})). \quad (2.5)$$

The FFN acts independently on each token and is typically parameterized as

$$\text{FFN}(x) = W_2 \phi(W_1 x + b_1) + b_2, \quad (2.6)$$

where ϕ is a non-linear activation function.

The core of the architecture is the attention mechanism, which allows the model to assign importance to different parts of the input sequence regardless of distance (see Figure 2.2). It is defined by three learned linear projections of the input X : queries (Q), keys (K), and values (V).

Mathematically, the attention output is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (2.7)$$

Here $Q, K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$, where d_k and d_v are the key and value dimensions per head. The term $\sqrt{d_k}$ in Equation (2.7) is a scaling factor that prevents dot products from becoming too large, which would push softmax into regions with very small gradients [16]. The term QK^T in Equation (2.7) is a similarity matrix that computes pairwise inner products between every query vector q_i and every key vector k_j in the sequence. If $Q, K \in \mathbb{R}^{n \times d_k}$, then:

$$(QK^T)_{ij} = \sum_{m=1}^{d_k} q_{im} k_{jm} \quad (2.8)$$

This operation computes dot-product similarity between tokens. In high-dimensional spaces, these scores determine how much attention token i should pay to token j . Since every token is compared against every other, forming the score matrix costs $O(n^2)$ in both time and memory; this quadratic growth in n is the principal efficiency constraint of standard self-attention [18].

The softmax function transforms a vector of raw scores z into a probability distribution in which all elements lie in $(0, 1)$ and sum to 1. For element z_i :

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad (2.9)$$

Equation (2.9) serves two purposes: the exponential term e^{z_i} amplifies larger values and suppresses smaller ones, and normalization ensures that attention weights sum to one, yielding a stable convex combination of value vectors V .

Rather than computing one attention map over the full embedding, the Transformer runs h attention operations in parallel (see Figure 2.2), each over its own low-rank projection of the queries, keys, and values. Splitting the computation this way gives every head access to a distinct learned subspace, so that complementary relational patterns can be captured simultaneously across positions. Because each head operates on only d_{model}/h dimensions, the h heads together cost no more than one full-width attention layer would.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.10)$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The projection matrices are $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

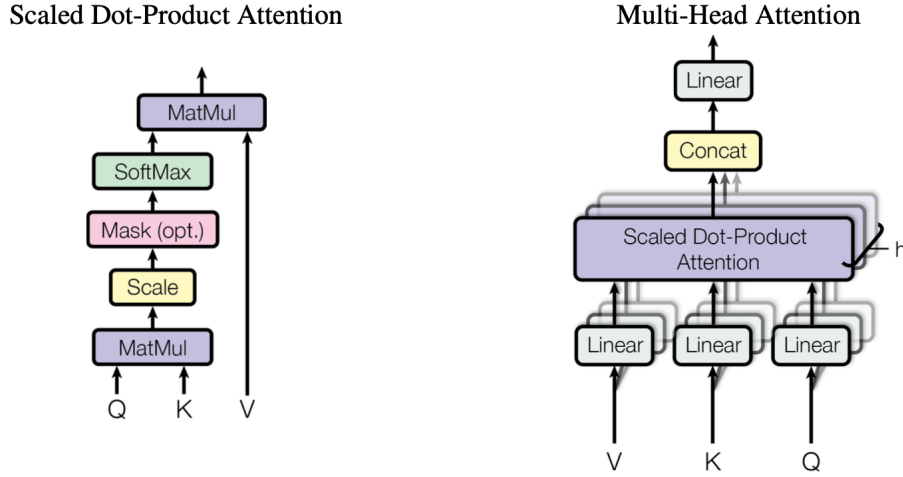


Figure 2.2: Left: scaled dot-product attention. Right: multi-head attention with parallel attention heads. Figure reproduced from Vaswani et al. [16].

Finally, unlike the original sequence-to-sequence Transformer, which uses both encoder and decoder stacks, the standard Vision Transformer (ViT) is encoder-only and applies a task-specific prediction head after encoder processing [17].

2.3. Physical and photonic neural networks

The hardware target of this work is a *physical neural network* (PNN): an analog physical system whose computation is carried out directly by its own dynamics, with tunable physical quantities — in photonics, the phase shifts and field amplitudes of light propagating through a programmable optical mesh — serving as the trainable weights [1]. A matrix–vector product, the dominant operation of a neural network, is then realized in the interference of light rather than executed as a sequence of digital multiply–accumulate operations [1]; this is what underlies the energy and latency advantages introduced in Chapter 1. The consequence that matters for training is that the weights are *embedded in the physics of the device* rather than stored in addressable memory. A digital network can read, transpose, and differentiate its weight matrices at will; a PNN cannot. This single fact is the root of the training obstacles examined next: the operations backpropagation depends on are not the operations an analog substrate naturally provides.

2.4. Limitations of Backpropagation and Backpropagation-Free Alternatives

Standard BP imposes three well-known computational constraints. First, every intermediate activation must be retained until the backward pass, so the memory footprint grows with network depth [19]. Second, repeated Jacobian products cause gradients to vanish or explode in deep architectures [20]. Third, weight updates are strictly sequential: a layer cannot be updated until the global error has fully propagated backwards through every downstream layer, which prevents parallel updates across depth [7].

On such physical hardware, these constraints become outright incompatibilities. BP requires an exact transpose channel ($W^{(l+1)\top}$) in the backward path. Because the weights are physically embedded rather than stored in memory, this transpose is not natively available in analog photonic circuits and can be obtained only through additional hardware blocks or physical reconfiguration [1], which forces existing photonic BP implementations to fall back on a digital twin [1]. Credit assignment is also non-local: a global error signal must reach every weight, which conflicts with in-situ hardware learning, in which each parameter can be adjusted only from signals physically present at that location [1]. A further obstacle is the simulation-reality gap. In-silico BP depends on a differentiable digital twin, yet fabrication variances, thermal drift, and nonlinear noise leave pre-trained gradients unreliable once transferred to the physical substrate [1].

The resulting picture, spanning algorithmic cost, and hardware incompatibility, explains the growing interest in backpropagation-free learning [21]. The remainder of this section follows that line of work from its origins in neuroscience, through forward-only gradient-estimation methods compatible with physical hardware, to the specific challenge of applying such methods to Transformers.

Towards Biological Plausibility

Backpropagation is widely regarded as biologically implausible because it requires each neuron to use the transpose of its downstream synaptic weights to compute updates - a constraint known as the weight transport problem. As Lillicrap et al. note, the cortex does not exhibit the point-to-point reciprocal connectivity that such weight symmetry would demand [22]. Motivated by such concerns, a range of more biologically plausible alternatives to backpropagation have been developed, including Feedback Alignment (FA), which replaces the transposed weights with fixed random feedback [23]; Predictive Coding (PC), rooted in hierarchical cortical models [24]; Equilibrium Propagation (EP) [25]; and the Forward-Forward algorithm (FF) [7].

One of the primary biologically plausible alternatives to backpropagation is Hebbian learning, which utilizes simple plasticity rules: synapses strengthen when pre- and post-synaptic neurons fire together. One of the extensions is synaptic modification based on the precise relative timing of the spikes (Spike-Timing-Dependent Plasticity). Beyond its classical role in weight adaptation, this timing-based rule has recently been re-purposed as a similarity operator inside Transformer attention, where query-key relevance is read directly from spike latencies rather than learned through gradient descent [26]. Recently, Chaudhary [27] showed that giving Transformers a self-updating short-term memory allows them to adapt to new information mid-sequence. The study found that different memory update rules, such as Hebbian and gradient-based, excel at different types of tasks.

Similarly, the spiking STDP Transformer (S²TDPT) introduced by [26] replaces dot-product attention with an STDP-driven mechanism that encodes token relevance through spike timing, thereby eliminating multiplications and achieving in-memory computation of attention weights. An earlier and foundational step in this direction, Spikformer [28], was among the first to bring self-attention into spiking neural networks: its Spiking Self-Attention operates on binary spike-form queries, keys, and values and drops softmax entirely, yet still reaches strong image-classification accuracy on both static and neuromorphic benchmarks. To address the scalability limits of purely local rules, Hua et al. [29] proposed the Global-guided Hebbian Learning (GHL) framework, which introduces a biologically inspired three-factor learning rule. By modulating local Hebbian updates with a global signal, GHL extends bio-inspired learning to very deep networks (up to ResNet-1202) and large-scale datasets such as ImageNet, reaching accuracies that come within a few percentage points of end-to-end backpropagation.

While these directions are compelling, their natural hardware substrate is neuromorphic and spiking rather than photonic. In addition to that, none of them matched backpropagation on the Transformer architectures targeted in this thesis. The rest of this section therefore turns to gradient-estimation methods that are compatible with photonic hardware.

Forward-Only Optimization

Other approaches retain backpropagation’s overall structure but relax its requirements - for example, Feedback Alignment [23] and Direct Feedback Alignment [30] replace the exact transpose in the backward pass with random projections. Both still require a differentiable computational graph, and Feedback Alignment in particular has been shown to converge only for Transformers of up to about two layers, failing entirely on deeper variants [31]. This motivates a shift toward methods that dispense with the backward pass altogether.

The gradient-approximation methods above still depend on access to the internal structure of the network such as derivatives of activation function, layer-wise error signals, or a differentiable computational graph. Obtaining exact gradients for a physical system is not possible generally [8], because such a system acts as a black box: its internal transformations are accessible only through input-output measurements. Furthermore, these schemes are degraded by fabrication tolerances, temperature drift and noise, which corrupt the estimated gradients on real hardware [1]. For these reasons, the zeroth-order optimization (ZO) offers a photonic hardware-compatible alternative.

Forward gradients [32] use forward-mode automatic differentiation to evaluate the Jacobian–vector product $(\nabla L) \cdot v$ - the exact directional derivative of the loss along a random direction v within a single forward pass, with no backward sweep. While this avoids a backward pass, it still assumes a differentiable computational graph - a constraint that does not hold for analog physical hardware. Zeroth-order optimization estimates directional derivatives using only function evaluations, without any access to the model’s internal computations, by using a central-difference scheme with perturbation v and step size ϵ :

$$\nabla_v \mathcal{L}(\omega) \approx \frac{\mathcal{L}(\omega + \epsilon \mathbf{v} / \|\mathbf{v}\|) - \mathcal{L}(\omega - \epsilon \mathbf{v} / \|\mathbf{v}\|)}{2\epsilon} \quad (2.11)$$

This doesn’t avoid gradients altogether, but produces an estimate of the overall gradient without analytical differentiation [33]. In the physical neural network setting, Guo et al. [8] demonstrated that combining layer-wise local learning with central-difference directional derivative estimation enables stable training of Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) [34], and extends naturally to photonic hardware simulations. Because it draws only on input-output evaluations, this estimator treats the activation as opaque and needs no analytic or surrogate model of it - a property forward-mode AD cannot offer, and the reason it transfers more readily to physical hardware.

Challenges of Backpropagation-Free Training for Transformers

Transformers present a particularly demanding challenge for BP-free training. Their attention mechanism requires computing a full $n \times n$ pairwise similarity matrix $QK^\top$, whose gradients propagate through multiple non-local operations - softmax, scaled dot-product, and value aggregation - creating intricate inter-layer dependencies. The multi-sub-layer encoder block structure (Multi-Head (MHA) followed by a Feed-Forward Network (FFN)) means that any layer-wise update scheme must account for two structurally distinct components per block. As Franceschi [31] noted, no existing work had applied backpropagation-free methods to Transformers before their study, and Kim & Yoo [35] frame the Transformer as an ideal stress-test for gradient-free training precisely because its stacked, attention-heavy layers make standard backpropagation so costly. This thesis therefore builds on Hinton’s Forward-Forward algorithm and its recent variants, which avoid the backward pass entirely and map cleanly onto the forward-only operations available on photonic hardware.

2.5. The Forward-Forward Algorithm and its Variants

Motivation for local learning

The Forward-Forward (FF) algorithm, proposed by Hinton [7], provides a local-learning alternative to BP that replaces the forward-backward cycle with two forward passes. A positive pass increases the layer-wise goodness, while a negative pass, operating on negative data (for example, corrupted inputs or examples paired with an incorrect label), decreases it. In other words, learning is represented as a discrimination task between positive and negative samples at each layer, bypassing backpropagation entirely. The goodness of the layer is defined as the sum of squared activations. The probability that a sample is positive is:

$$p(\text{positive}) = \sigma \left(\sum_j y_j^2 - \theta \right), \quad (2.12)$$

where θ is a threshold and $\sigma(\cdot)$ is the logistic sigmoid. This function tries to increase the probability for positive data and decrease it for the negative data, while updating the weights of the layer only locally.

Layer normalization is essential to this scheme. If the raw hidden vector were forwarded unchanged, a downstream layer could separate the two classes using nothing more than the overall magnitude of the incoming activity, short-circuiting any need to extract richer structure. Normalizing the vector removes this shortcut: magnitude is consumed locally to compute that layer's goodness, and only the *direction* of the activity is propagated, so each subsequent layer must encode genuinely new, length-invariant features. For classification, the class label is embedded directly in the input, and at inference the network runs one forward pass per candidate class, selecting the label whose goodness, accumulated over the hidden layers above the first, is largest [7].

However, the FF algorithm has notable limitations. The generation of high-quality negative data is a central difficulty, and Hinton identifies it as an open problem for the method. On the small-scale benchmarks examined, FF is slower than and does not generalize as well as backpropagation, leading Hinton to expect that very large models trained on very large datasets will continue to rely on BP [7]. A countervailing advantage, however, is that FF is well-suited to very low-power analog hardware, on which exact backpropagation is physically impractical.

A further motivation for FF in the hardware context is mortal computation [7]. Traditional software engineering assumes that trained weights are hardware-agnostic; because knowledge is treated as "immortal," a model can be seamlessly copied across devices. However, this assumption relies on highly precise and reproducible hardware, which is costly to manufacture and energy-intensive to operate. In mortal computation, if the learning procedure can discover effective parameters directly, then hardware can be simplified and made more energy-efficient. While the resulting trained parameters become device-specific and "die" with the hardware, the learned function can still be transferred to new devices via distillation rather than direct weight copying [7]. Because the FF algorithm does not require a perfectly differentiable model of the forward pass, it naturally supports this paradigm and can be trained directly on physical hardware with unknown internal operations. As Hinton puts it, "if you want your trillion parameter neural net to only consume a few watts, mortal computation may be the only option" [7]. This observation forms a core motivation for this thesis: to demonstrate that a Transformer architecture can be trained without backpropagation or automatic differentiation, using only operations that map onto the forward-only primitives available on photonic hardware. Whether this comes at a cost in accuracy, and on which task types, is the subject of Chapter 4.

Refining the Goodness Signal

The original FF goodness the sum of squared activations passed through a sigmoid (Equation 2.12) is a binary discriminator: positive samples should exceed a threshold, negative samples should fall below it [7]. This formulation has two known weaknesses. First, the objective contains no explicit margin term: it only pushes positive goodness above θ and negative goodness below it, without enforcing any minimum separation between the two. Second, the scalar output discards class structure: at inference time the model must run C separate forward passes, one per candidate class, and select the highest-scoring one.

A first family of variants reformulates the goodness signal itself to address these issues. Distance-Forward (DF) learning [36] reinterprets goodness as a squared Euclidean distance to class centroids and introduces an N-pair margin loss that explicitly enforces a separation m^+ between positive and the hardest negative goodness:

$$\mathcal{L}_{\text{DF}} = \max \left(m^+ + \max \{ G_k^{\text{neg}} \}_{k=1}^N - G^{\text{pos}}, 0 \right) + \lambda \max(\{ G_k^{\text{neg}} \}_{k=1}^N), \quad (2.13)$$

where the margin term replaces the implicit threshold θ with an enforceable geometric gap. Scalable Forward-Forward (SFF) [37] replaces the threshold with a log-sum-exp objective that drives the ground-truth goodness g_{pos} to exceed the soft maximum aggregated over all J classes, so that no separate negative samples are required:

$$\mathcal{L}_{\text{SFF}} = -\mathbb{E} \left[g_{\text{pos}} - \log \sum_{j=1}^J \exp(g f_{n,j}) \right]. \quad (2.14)$$

Channel-wise Competitive Learning (CwC) [38] applies a related insight to convolutional architectures: rather than a scalar goodness per layer, it computes one goodness score per class by partitioning channels into J class-specific groups and treating the resulting per-class scores as logits in a softmax cross-entropy objective. The shared idea across these methods is that a discriminative, multi-class goodness signal is a stronger learning target than a binary threshold.

Eliminating Negative Samples

Hinton explicitly identified the generation of high-quality negative samples as a primary open challenge for the FF algorithm [7], motivating subsequent variants that attempt to bypass this requirement entirely. Self-Contrastive FF (SCFF) [39] sidesteps the problem by generating negatives in a dataset-agnostic way: positive inputs concatenate a sample with itself and negative inputs concatenate two different samples from the batch (Figure 2.3), trained against a layer-wise sigmoid goodness with separate thresholds $\theta_{\text{pos}}^{(l)}, \theta_{\text{neg}}^{(l)}$ (Equations 2.15-2.18).

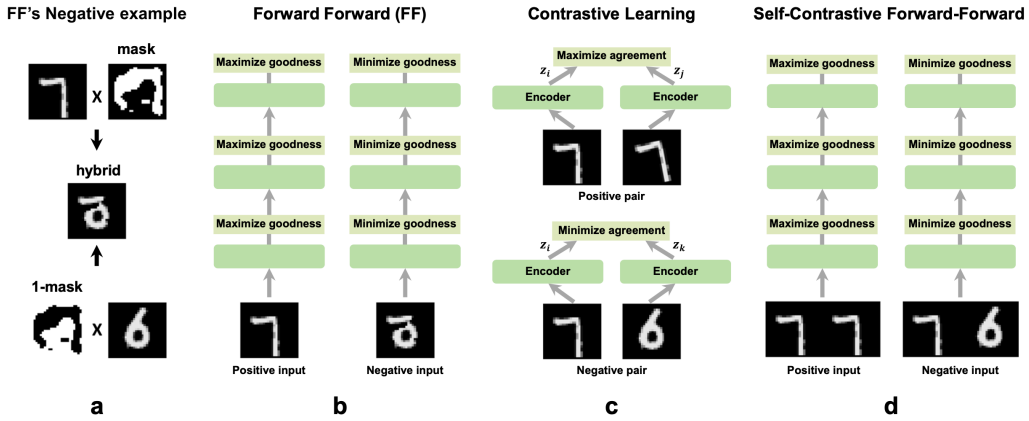


Figure 2.3: A, B: The classical Forward-Forward approach of negative sample generation and FF learning, given for comparison. C, D: In Contrastive learning, two encoders independently process input images to minimize agreement for negative pairs and maximize agreement for positive pairs. SCFF fuses the FF objective of panel B with the pairing logic of panel C: each layer is trained to raise its goodness on self-concatenated (matching) inputs and lower it on inputs built from two unlike samples. Figure reproduced from Chen et al. [39].

$$x_{i,\text{pos}}^{(0)} = [x_k, x_k]^T, \quad x_{j,\text{neg}}^{(0)} = [x_k, x_n]^T, \quad n \neq k, \quad (2.15)$$

$$G_i^{(l)} = \frac{1}{M^{(l)}} \sum_m y_{i,m}^{2(l)}, \quad (2.16)$$

$$p_{pos}(x_i) = \sigma(G_i^{(l)} - \theta_{pos}^{(l)}), \quad p_{neg}(x_j) = \sigma(\theta_{neg}^{(l)} - G_j^{(l)}), \quad (2.17)$$

$$\mathcal{L}_{SCFF} = -\mathbb{E}_{x_i \sim pos} \log p_{pos}(x_i) - \mathbb{E}_{x_j \sim neg} \log p_{neg}(x_j). \quad (2.18)$$

A more radical solution is to drop the positive/negative distinction altogether and learn directly from labels at each layer. Mono-Forward [40] achieves this by attaching a layer-specific goodness matrix M_i that maps activations to per-class scores,

$$g_i \triangleq a_i^\top M_i, \quad M_i \in \mathbb{R}^{n \times m}, \quad (2.19)$$

and optimizing a local Mono-Forward cross-entropy loss $\mathcal{L}_{MF}^{(i)}$ at each layer i :

$$\mathcal{L}_{MF}^{(i)} \triangleq - \sum_{c=1}^m y_c \log(\text{softmax}(g_i)_c), \quad (2.20)$$

with both layer weights and goodness matrices updated locally:

$$\begin{aligned} W_i &\leftarrow W_i - \eta \left(\frac{\partial \mathcal{L}_{MF}^{(i)}}{\partial G_i} \cdot \frac{\partial G_i}{\partial a_i} \cdot \frac{\partial a_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_i} \right) \\ M_i &\leftarrow M_i - \eta \left(\frac{\partial \mathcal{L}_{MF}^{(i)}}{\partial G_i} \cdot \frac{\partial G_i}{\partial M_i} \right). \end{aligned} \quad (2.21)$$

The Cascaded Forward algorithm (CaFo) [41] follows the same philosophy with attached predictors that emit full label distributions at each block, combined at inference by summation $\hat{Y} = \sum_{i=1}^r \hat{Y}^i$ (Figure 2.4). CwC [38], already discussed above, is also negative-free: its class-channel partitioning makes each layer act as its own multi-class classifier without ever forming a corrupted input.

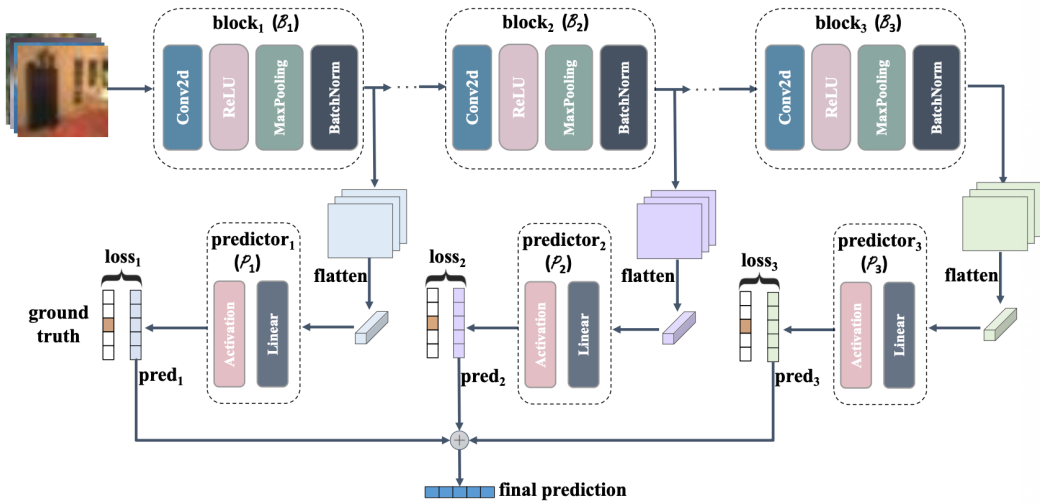


Figure 2.4: Architecture of the Cascaded Forward (CaFo) network. Each block is trained independently with an attached predictor that emits a class-distribution estimate; at inference, the per-block predictions are summed. Figure reproduced from Zhao et al. [41].

Scaling to Deep and Structured Architectures

A separate concern is whether FF can train architectures more complex than shallow MLPs. Pure layer-wise FF struggles to learn the intricate feature interactions present in residual networks, convolutional stacks, or attention-based models. Three strategies have emerged.

The first relaxes strict layer-wise locality. SFF [37] permits backpropagation *within* a localized block (such as a residual block) but strictly forbids it *across* blocks. This hybrid scheme retains FF’s hardware-friendly absence of a global backward pass while letting tightly coupled operations be jointly optimized, which is a pragmatic concession that achieves parity with BP on CIFAR-100 and Imagenette, and outperforms BP by approximately 9% on a 1,000-sample subset of Imagenette, suggesting that local objectives can act as regularizers in low-data regimes.

The second imposes architectural priors that are FF-compatible by construction. CwC’s CFSE block (Figure 2.5) reorganizes a convolutional layer into class-specific channel groups so that the goodness signal has a natural multi-class structure, and its Interleaved Layer Training (ILT) schedule activates layers in staggered windows so that downstream layers train against stabilized upstream features.

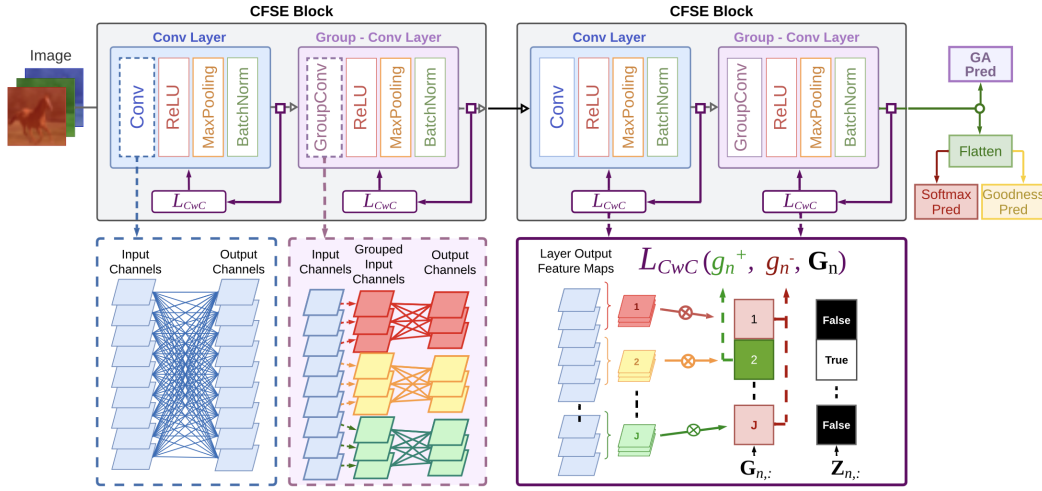


Figure 2.5: Architecture of the CFSE block. Convolutional channels are partitioned into class-specific groups, yielding one goodness score per class so that each layer acts as its own multi-class classifier. Figure reproduced from Papachristodoulou et al. [38].

The third extends FF to the Transformer encoder, the central concern of this thesis. Kim & Yoo [35] demonstrated that FF can replace BP across both MHA and FFN sub-layers using a margin-based local loss. On the IMDB dataset they report a roughly 20% reduction in training time and 30% reduction in memory, with model quality changing only marginally (perplexity 13.2 vs. 12.5; BLEU 27.8 vs. 28.3).

The Contrastive Forward-Forward (CFF) method [42] instead trains a Vision Transformer by replacing the squared-activation goodness with a supervised contrastive loss applied after each encoder block. Two augmented views of a batch are passed forward, and at every block the mean-pooled token representations are pulled together for same-class samples and pushed apart for different-class samples. For an anchor representation f_i with positive set $P(i)$ and full comparison set $A(i)$ within the batch, this loss takes the form:

$$\mathcal{L}_{\text{con}} = \sum_i \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(f_i \cdot f_p / \tau)}{\sum_{a \in A(i)} \exp(f_i \cdot f_a / \tau)}, \quad (2.22)$$

with temperature τ , each block detached from its predecessor so that no gradient crosses block boundaries. This permits single-pass inference, in contrast to the C forward passes Hinton’s formulation requires. The illustration of the algorithm is shown in Figure 2.6.

Existing FF-on-Transformer work is limited to these two studies, and both share two limitations relevant here. First, each attaches its local objective at sub-layer (MHA/FFN) or whole-block granularity,

so the four attention projections Q, K, V, O receive no dedicated local target. Second, both remove only the *global* backward pass while retaining ordinary automatic differentiation *within* each locally trained unit, and CFF additionally depends on the different-class negatives in $A(i)$; neither is therefore backpropagation- or negative-free in the sense required for photonic hardware.

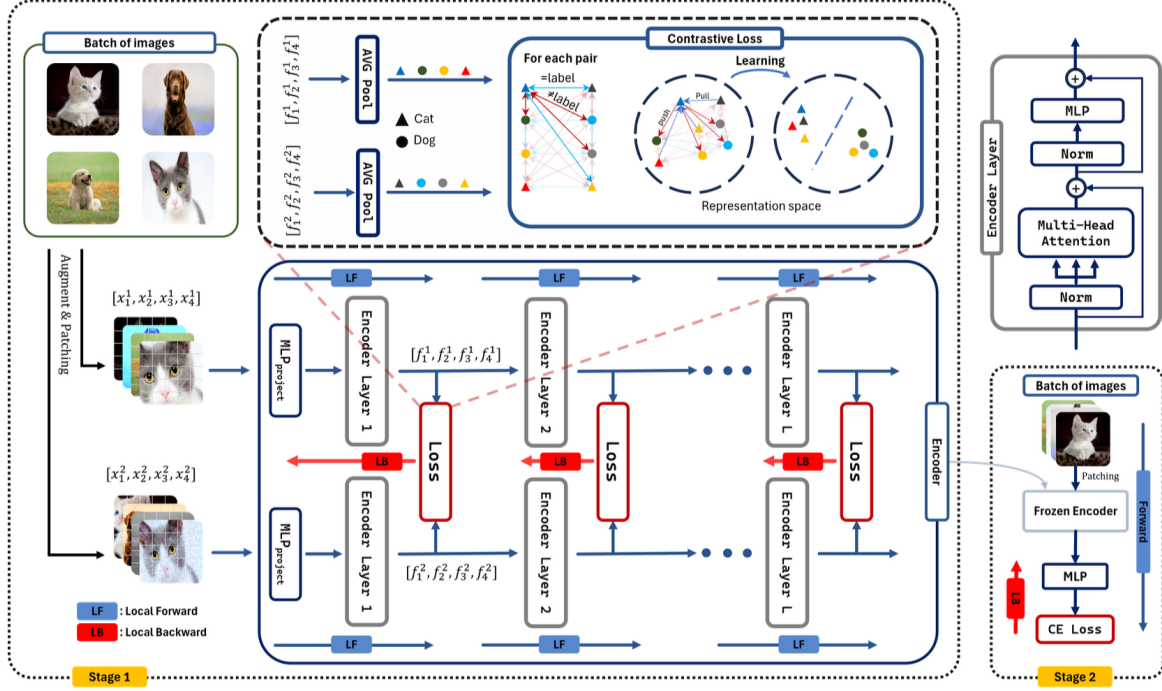


Figure 2.6: The Contrastive Forward-Forward scheme [42]. **Stage 1:** two augmented views of a batch are patched and passed through the shared encoder; after each encoder block the token representations are average-pooled and compared by a contrastive loss that pulls same-class samples together and pushes different-class samples apart, with a local backward pass updating only that block. **Stage 2:** the encoder is frozen and a small MLP head is trained with cross-entropy for single-pass inference. The local objective is attached at whole-block granularity and relies on different-class negatives, leaving the internal attention projections without a dedicated local target. Figure reproduced from Aghagolzadeh and Ezoji [42].

Hardware-Compatible Goodness via Fixed Prototypes

A final thread, and the one most directly relevant to this thesis, asks how the goodness target itself can be made hardware-compatible. The variants discussed so far still rely either on adaptive class centroids (DF learning), learned goodness matrices (Mono-Forward), or runtime class statistics (CwC, SFF). On photonic hardware these adaptive components introduce additional control complexity that scales poorly.

Hyperspherical Prototype Networks (HPN) [43] offer a way out. Standard prototype-based learning suffers from a "chicken-or-egg" dilemma: as network weights update, the mapping of training examples shifts, which moves the prototypes, which changes the optimization target. HPN resolves this by fixing prototypes *a priori* as maximally separated points on a hypersphere, reducing learning to minimizing the cosine distance between layer outputs and pre-assigned class targets. The optimization landscape becomes static, and large-margin separation is enforced from initialization.

FFzero [8] adapts this prior for hardware-compatible FF training by fixing, per layer, a set of C unit prototypes - one per class - that are sampled once and spread as far apart as possible on the unit sphere of the layer's output space (geometrically, the vertices of a randomly oriented regular simplex). Each layer m has its own prototype set ξ_m , and class-wise goodness is computed as cosine similarity between the layer output and the candidate class prototype:

$$\underbrace{G_1(\mathbf{W}_1|\hat{y}, \mathbf{x}) = \xi_1[\hat{y}] \cdot \mathbf{z}_1 / \|\mathbf{z}_1\|}_{\text{class-wise goodness of linear layer \#1}} \quad \text{and} \quad \underbrace{G_2(\mathbf{W}_2|\hat{y}, \mathbf{x}) = \xi_2[\hat{y}] \cdot \mathbf{z}_2 / \|\mathbf{z}_2\|}_{\text{class-wise goodness of linear layer \#2}}. \quad (2.23)$$

Inference aggregates per-layer votes by majority:

$$\mathbf{y} \leftarrow \text{mode} \left\{ \arg \max_{\hat{\mathbf{y}} \in \mathcal{Y}(N)} G_m(\mathbf{W}_m | \hat{\mathbf{y}}, \mathbf{x}) : m = 1, 2, 3, \dots, M \right\}. \quad (2.24)$$

FFzero further removes any reliance on automatic differentiation by estimating gradients from directional derivatives - two forward passes per random direction $\mathbf{v}$:

$$\nabla_{\mathbf{v}} G(\boldsymbol{\omega}) \approx \lim_{\varepsilon \rightarrow 0} \frac{G(\boldsymbol{\omega} + \varepsilon \mathbf{v} / \|\mathbf{v}\|) - G(\boldsymbol{\omega} - \varepsilon \mathbf{v} / \|\mathbf{v}\|)}{2\varepsilon}, \quad (2.25)$$

with the iterative weight update averaged over P directions:

$$\boldsymbol{\omega} \leftarrow \underbrace{\boldsymbol{\omega} + \lambda \frac{n}{P} \sum_{p=1}^P \nabla_{\mathbf{v}_p} G(\boldsymbol{\omega}) \frac{\mathbf{v}_p}{\|\mathbf{v}_p\|}}_{\text{weights update}} \quad \text{with } \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad n = \dim(\boldsymbol{\omega}) = \dim(\mathbf{v}). \quad (2.26)$$

The authors show analytically that, despite the noise in any single estimate, the update is unbiased in expectation, so the averaged trajectory matches that of exact partial derivatives. They then demonstrate empirically that this estimation noise stays confined within each locally trained layer, rather than accumulating with depth as it does under a global objective. The schematic of this update is shown in Figure 2.7, and the full FFzero classification architecture in Figure 2.8.

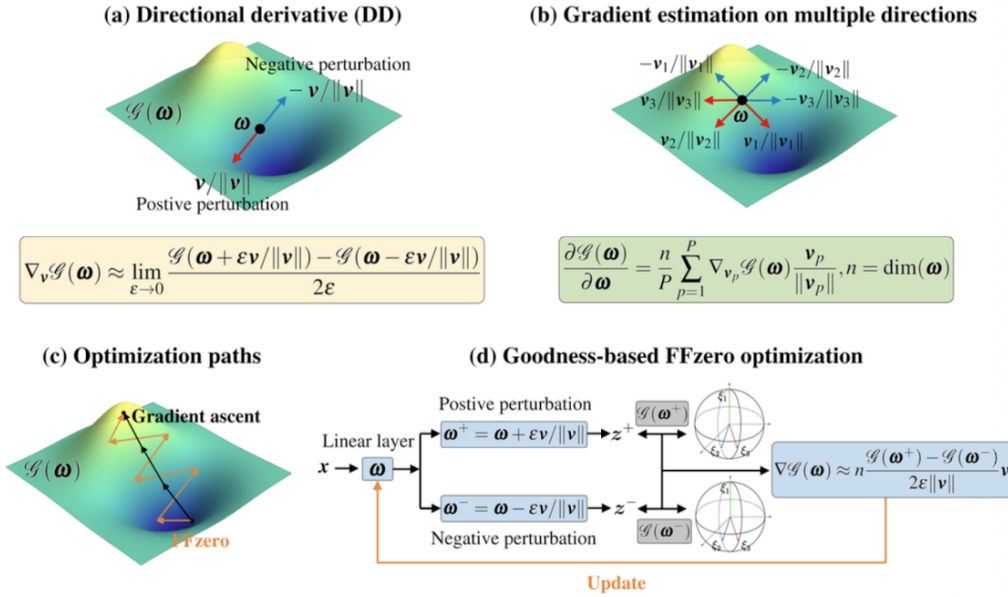


Figure 2.7: (a) Illustration of the directional derivative (DD). (b) Gradient estimation averaged over multiple random directions. (c) Optimization trajectory comparison between DD-based and exact-gradient (AD) updates. (d) Goodness-based DD update as used in FFzero. Figure reproduced from Guo et al. [8].

Summary

Taken together, these four threads - refining the goodness signal, eliminating negative samples, scaling to deep architectures, and fixing the goodness target via prototypes - trace the development of FF research from Hinton's original binary-threshold formulation toward a learning rule that is local, negative-free, backpropagation-free, and hardware-compatible. Two gaps remain. First, although Contrastive Forward-Forward [42] trains a Vision Transformer end-to-end, it (like Kim & Yoo [35]) attach

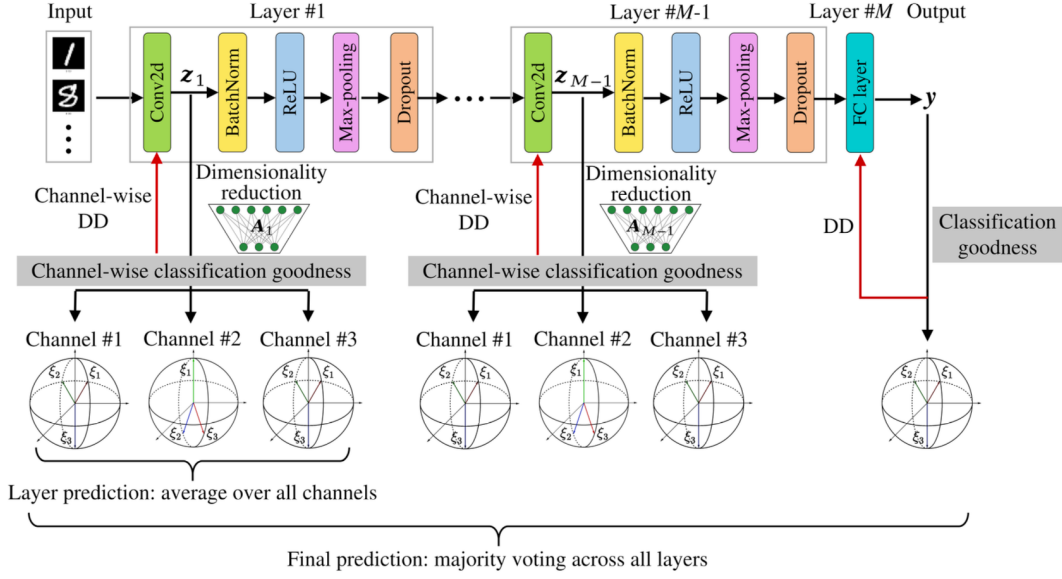


Figure 2.8: Schematic of the FFzero algorithm and architecture used for CNN classification. Figure reproduced from Guo et al. [8].

their local objective at whole-block or sub-layer granularity, so the four attention projections Q, K, V, O receive no dedicated local target and are updated only by automatic differentiation within each block; FFzero [8], the sole prototype-based and autodiff-free variant, is demonstrated only on MLPs and CNNs. Second, no existing work combines all four hardware-compatible properties (locality, negative-free training, prototype-based goodness, and directional-derivative updates) within a single Transformer training scheme. Closing both gaps is the contribution of Chapter 3.

3

Methodology

3.1. Overview and design rationale

The goal of this thesis is to adapt a transformer to be trained directly on a photonic hardware. This requires removing two operations from the standard training pipeline that an optical chip cannot perform. The first is the global backward pass and the second one is automatic differentiation. Therefore, the main approach is forward-only training, which eliminates the need in global backward pass, combined with directional-derivative gradient estimation - as they are both natively supported by a photonic chip.

There are six variants compared across two training paradigms. BP-AD (backpropagation with autograd) sets the accuracy baseline. BP-DD (backpropagation with directional-derivatives) stands as a hypothetical alternative for training on photonic chip using classical training approach. On the other side is Forward-Forward (FF) algorithm [7]: layer-wise training with local goodness signals which makes it plausible for in-situ training. There are two variants: FF-AD (Forward-Forward with autograd) where backpropagation is used but only within a single component and never across blocks, and FF-DD (Forward-Forward with directional derivatives) where backpropagation is absent and replaced by random directional perturbation, evaluated by forward passes. In both cases the attention is trained using Round-Robin technique (see [section 3.3](#)). FF-Joint-AD and FF-Joint-DD serve as ablation experiments that differ from FF-AD and FF-DD in only one respect: all weights in attention are updated at the same time. The main target is FF-DD because it is the only variant that combines all hardware-compatible properties: local learning, no backpropagation, Spherical Attention, and round-robin projection updates. All BP variants are based on a standard transformer architecture by Vaswani et al. [16]. All FF variants are based on a transformer architecture, adapted for photonic hardware by substituting softmax attention with Spherical Attention ([Figure 3.1](#)). Because all FF variants share the same forward model, accuracy differences within the FF family reflect only the training algorithm. Differences in performance between BP and FF methods reflect the architectural swap (Softmax MHA + linear head vs. Spherical Attention + SimplexHead). All training parameters are listed in [Appendix A](#).

3.2. Transformer Architecture

Transformer Block Formulation

Input processing differs by task. For tasks related to image classification (ViT), images are split into P non-overlapping patches, projected to the model dimension d and flattened. To aggregate global features, the [CLS] token is prepended. Under BP patch projection, token embedding, [CLS] token are updated during backpropagation, but FF freezes all embedding parameters and holds it at its initial values. Fixed sinusoidal positional encodings are added:

$$\text{PE}_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad \text{PE}_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right). \quad (3.1)$$

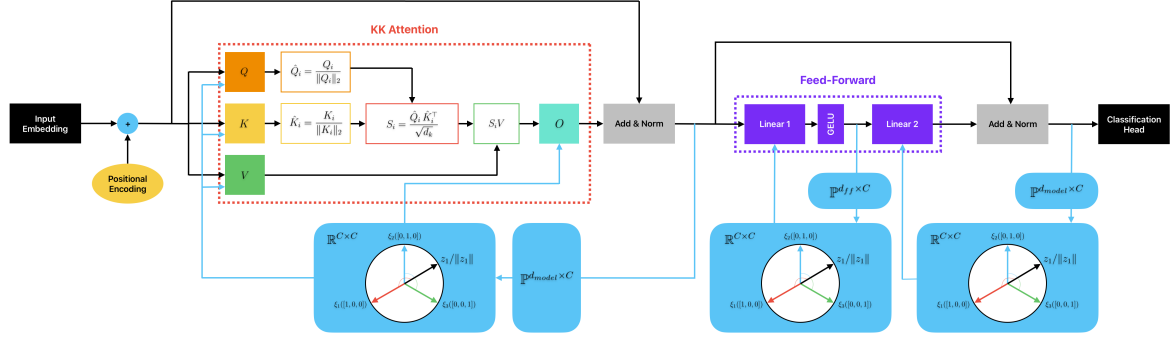


Figure 3.1: Architecture of the FF Transformer with Spherical Attention.

For Seq2Seq tasks, discrete tokens are mapped to continuous vectors by a learned embedding matrix and scaled by $\sqrt{d}$. The architecture relies on the Pre-Layer Normalization (Pre-LN) transformer block, which improves training stability compared to the original Post-LN design. The forward pass through a single block under standard BP training is shown in Equation 2.5. Multi-Head Attention for BP version of a Transformer is defined in Equation 2.7 - Equation 2.10:

In the methodological implementation of a Transformer, the feed-forward sub-layer is a two-layer MLP with GELU activation:

$$\text{FFN}(\mathbf{x}) = W_2 \text{GELU}(W_1 \mathbf{x} + b_1) + b_2, \quad d_{ff} = 4d, \quad (3.2)$$

where the feed-forward hidden dimension d_{ff} is fixed at four times the model dimension d across all tasks. A final Layer Normalization is applied to the output of the full block stack before the classification or language modeling head.

Under FF training the skip path carries the normalized input $\hat{\mathbf{x}} = \text{LN}(\mathbf{x})$ instead of the raw input:

$$\hat{\mathbf{x}} = \text{LN}(\mathbf{x}), \quad (3.3)$$

$$\mathbf{x}' = \hat{\mathbf{x}} + \text{MHA}(\hat{\mathbf{x}}, \hat{\mathbf{x}}, \hat{\mathbf{x}}), \quad (3.4)$$

$$\hat{\mathbf{x}}' = \text{LN}(\mathbf{x}'), \quad (3.5)$$

$$\mathbf{x}'' = \hat{\mathbf{x}}' + \text{FFN}(\hat{\mathbf{x}}'). \quad (3.6)$$

Each block is read out by a fixed projection M_l using cosine similarity, so absolute scale is irrelevant. With a raw-input skip the residual stream accumulates across the frozen blocks and grows with depth, so each block's bounded update becomes a vanishing fraction of the direction; the readout is then dominated by lower-layer content and M_l can no longer separate the classes. Normalizing the skip path resets the carried signal to unit scale at every block, keeping each block's contribution comparable across depth.

Throughout FF training, the LayerNorm parameters (γ, β) preceding each sub-layer are applied with stop-gradient:

$$\text{LN}_{\text{frozen}}(x) = \text{LayerNorm}(x; s_g(\gamma), s_g(\beta)) \quad (3.7)$$

This prevents the LN scale and shift from accumulating gradients from the local goodness signal of any single component: the parameters stay at their initial values ($\gamma = 1, \beta = 0$) through FF training, so every component is optimized against a fixed normalization.

Spherical Multi-Head Attention

Standard multi-head attention depends on the softmax operation, which maps poorly onto coherent optical hardware: it requires an element-wise exponential followed by a global normalization, neither of which has an efficient passive-optical realization [3]. First of all, softmax requires calculating exponential functions for each input e^{x_i} , and then dividing that by the sum of all exponential for normalization (see Equation 2.9).

Second, Tian et al. note that conventional photonic accelerators are optimized for weight-stationary data-flows, leaving them poorly matched to attention, whose operands are both dynamic at runtime [3]. A further difficulty is that scaling such optical interconnects is bounded by accumulated propagation loss, which constrains how large a coherent attention array can practically be made [3]. The hardware footprint is also unfavorable: a softmax-based weighting unit requires four phase shifters, whereas a single KKA interference operator needs only two [3]. Tian et al. address this with Kramers-Kronig Attention (KKA), in which the multiply-accumulate operations of standard attention, including the value aggregation, are replaced throughout by the real part of an optical interference:

$$a \square b \sim \text{Re}(ab^*), \quad (3.8)$$

$$\text{softmax}\left(Q^\top K / \sqrt{d}\right) V \sim (Q_{\text{kka}}^\top \square K_{\text{kka}}) \square V_{\text{kka}}. \quad (3.9)$$

The amplitude-phase coupling dictated by the Kramers-Kronig relations supplies an input-dependent nonlinearity in place of the softmax, which in turn removes the $1/\sqrt{d}$ factor that standard attention needs only to bound the score variance [3].

Whether this kernel can be used to train in-situ depends on which gradient signal drives the update (see Appendix B). Based on the results, is that finite-difference directional-derivative estimator cannot follow the periodic structure of the cosine, which motivated the use of a non-periodic, monotonically angle-dependent score function. Therefore, Spherical Attention was used, inspired by Tian et al. [3], which preserves two main properties of KKA attention: attention scores are bounded and softmax is removed. Unlike KKA, where the amplitude-phase nonlinearity absorbs the score variance and the $\sqrt{d}$ scaling can be dropped, Spherical Attention has no such nonlinearity, so the $1/\sqrt{d_k}$ factor is retained to keep the output magnitude independent of head dimension (Eq. 3.10). Empirically, removing the factor degraded the accuracy across all variants. The two mechanisms (Kramers-Kronig and Spherical Attention) are not mathematically equivalent, but a softmax-free score is shared, and the score function is smooth enough for DD's finite-difference estimator to navigate.

The per-head function drops the softmax and normalizes the queries and keys:

$$\text{SphAttn}(Q, K, V) = \frac{\hat{Q} \hat{K}^\top}{\sqrt{d_k}} V, \quad \hat{Q} = \frac{Q}{\|Q\|_2}, \quad \hat{K} = \frac{K}{\|K\|_2}, \quad (3.10)$$

to be read against the standard Attention(Q, K, V) = softmax($QK^\top / \sqrt{d_k}$) V of Equation 2.7: the softmax is removed and Q, K are replaced by their L_2 norm counterparts $\hat{Q}, \hat{K}$. The multi-head wrapper is then identical to Equation 2.10, with SphAttn substituted for Attention:

$$\text{SphAttn-MHA}(Q, K, V) = \text{Concat}(\text{head}_1^{\text{sph}}, \dots, \text{head}_h^{\text{sph}}) W^O, \quad \text{head}_i^{\text{sph}} = \text{SphAttn}(QW_i^Q, KW_i^K, VW_i^V). \quad (3.11)$$

For BP training, future positions are masked additively, $(M_c)_{ij} = -\infty$ for $j > i$ and 0 otherwise, so that softmax maps masked entries to zero attention weight. For FF training, the softmax-free Spherical Attention cannot use $-\infty$ masking, which would propagate as NaN; instead, forbidden entries are zero-filled directly on the score matrix.

The choice of the L_2 norm is deliberate: it is the unique normalization for which the inner product of the resulting unit vectors equals the cosine of the angle between them, $\hat{Q} \hat{K}^\top = \cos \theta$. It bounds the score by

construction - each entry lies in $[-1, +1]$, and after the $1/\sqrt{d_k}$ scaling in $[-1/\sqrt{d_k}, +1/\sqrt{d_k}]$ - taking over the variance-bounding role that softmax previously played. Second, the score becomes invariant to the magnitudes $\|Q\|$ and $\|K\|$, depending only on direction. Finally, L_2 normalization is consistent with the rest of the pipeline: the simplex prototypes and the goodness projection (Equation 3.16) also operate on the L_2 unit sphere, so the entire network - attention scores and goodness scores alike - shares one spherical geometry, from which the method takes its name. The norms L_1 and L_∞ would also bound the score but would not yield an angular similarity on the sphere, breaking this consistency.

The contribution here is its use as a photonic-compatible adaptation without softmax of a training pipeline. Together with the Transformer Block Formulation above, this completes the architectural specification of the network. The next section describes the six training methods used to fit this network.

3.3. Training Methods

The six variants introduced in section 3.1 differ in three orthogonal axes: the loss they minimize (global cross-entropy for BP, local simplex goodness for FF), the gradient signal they use (automatic differentiation for AD variants, central-difference directional derivatives for DD variants), and the order in which the four attention projections are updated (round-robin and joint for FF; joint for BP). This section describes the BP baseline first, then the local goodness loss shared by all FF variants, the per-component round-robin and two-phase schedules, the layer-wise pipeline that ties them together, and finally a side-by-side comparison.

BP: Global Backpropagation

The baseline method trains the entire network end-to-end with standard backpropagation using the AdamW optimizer [44] and a cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{n=1}^N \log \frac{\exp(z_{y_n})}{\sum_c \exp(z_c)}, \quad (3.12)$$

where z are the final logits. For vision tasks, $z = \text{LinearHead}(\mathbf{h}_{\text{CLS}})$ is computed once per image from the [CLS] token. For sequence tasks, the loss is averaged over all next-token predictions: $z_t = \text{LinearHead}(\mathbf{h}_t)$ at every position t , with y_t the next token in the sequence.

The global gradient $\nabla_{\theta} \mathcal{L}_{\text{CE}}$ is computed by automatic differentiation through every layer, including attention projections, feed-forward weights, and layer norms. Parameters are updated as:

$$\theta \leftarrow \theta - \eta \text{AdamW}(\nabla_{\theta} \mathcal{L}_{\text{CE}}). \quad (3.13)$$

Training uses $\eta = 10^{-3}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay $\lambda = 10^{-4}$, gradient clipping at norm 1.0, and runs for 100 epochs with early stopping (stop after 10 epochs with no improvement with $\text{min_delta} = 10^{-3}$ on validation accuracy). Positional encodings are fixed sinusoidal buffers; all other parameters are jointly optimized.

Simplex Prototype Goodness Loss

All Forward-Forward components, on the other hand, are trained with a local goodness objective. The objective is a prototype-based loss [8], [43], introduced in section 2.5 of the background chapter: a fixed set of maximally separated class prototypes is sampled *a priori*, and the loss pushes the normalized projection of each activation toward the prototype of the correct class and away from all others. Following FFzero, each trainable component is associated with its own simplex prototype matrix $R \in \mathbb{R}^{C \times r}$, where C is the number of classes and r the prototype-space dimension; its rows R_c are the maximally separated class prototypes on the unit sphere in $\mathbb{R}^r$. A fixed orthogonal projection M maps the component's output into this r -dimensional prototype space (its input dimension is d for the attention projections and the second FFN phase, and d_{ff} for the first FFN phase).

Given an activation $z \in \mathbb{R}^d$ and its true class label y , the goodness pipeline is:

$$\hat{z} = \frac{Mz}{\|Mz\|_2} \quad (\text{project into prototype space; L2-normalize}), \quad (3.14)$$

$$s_c = \hat{z} \cdot R_c^\top \quad (\text{cosine similarity to each class prototype}), \quad (3.15)$$

$$\mathcal{L}_{\text{good}}(z, y) = \frac{1}{C-1} \sum_{c \neq y} \max(0, \alpha + s_c - s_y) \quad (\text{margin loss}), \quad (3.16)$$

where $\alpha > 0$ is the margin hyperparameter. The loss pushes the projection of z toward the prototype R_y of the true class and away from all other class prototypes by at least α . Prediction at inference is $\hat{y} = \arg \max_c s_c$, i.e. the class whose prototype is closest to the projected, normalized activation.

Where FFzero attaches one prototype set per linear layer of an MLP, the present work attaches one prototype set for attention block, shared by all four projections W^Q, W^K, W^V, W^O , and one to each FFN layer, so that every Transformer component has its own local goodness signal. The prototype sets $\{R, M\}$ are sampled once per component using the deterministic seeding scheme of Algorithm 1 and remain fixed throughout training.

Forward-Forward Attention Training

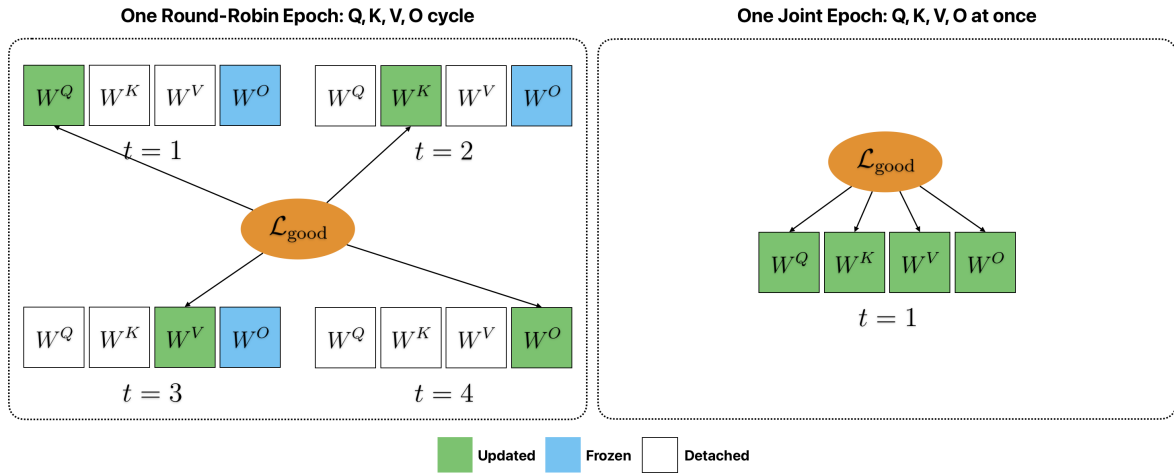


Figure 3.2: Per-step update of the four attention projections under the two FF schedules. Each step computes the local goodness loss $\mathcal{L}_{\text{good}}$ on the attention output and updates the projections shown in green. **Left (round-robin):** the projections are trained one at a time, $W^Q \rightarrow W^K \rightarrow W^V \rightarrow W^O$. On each sub-step only the active projection (green) is updated. The non-active query/key/value projections are *detached* (white), so no gradient flows through them; W^O is instead *frozen* (blue) on the $Q/K/V$ steps - it is kept in the computation graph so the gradient can pass through it to reach the active projection, but its own weights are not updated. **Right (joint):** all four projections are updated simultaneously from a single goodness signal. One round-robin epoch comprises four sequential sub-steps; one joint epoch, a single step. The round-robin schedule reflects the physical constraint that on photonic hardware the four projections are distinct programmable elements that cannot be updated coherently in parallel.

FF (Round-Robin). In each training epoch the four projections are visited in order $p \in \{Q, K, V, O\}$. On the step for part p :

1. Projections other than p have their outputs detached from the computation graph. The exception is W^O when $p \in \{Q, K, V\}$: W^O is left in the graph (so gradients can flow through it back to W^p) but is held fixed via `requires_grad=False`, so it receives no update on this step.
2. The attention output and residual are computed:

$$z = \text{LN}(x) + \text{SphAttn}(\text{LN}(x)) \quad (3.17)$$

where input x is detached so no gradient flows to previous blocks.

3. The goodness loss $\mathcal{L}_{\text{good}}(z_0, y)$ is evaluated on the [CLS] token z_0 (vision) or all sequence positions (Seq2Seq), and only W^p is updated via Adam.

One full epoch therefore consists of four sequential sub-steps, each updating a single projection.

FF-Joint. All four projections are updated simultaneously in a single Adam [45] step per batch, using the same goodness loss:

$$\{W^Q, W^K, W^V, W^O\} \leftarrow \{W^Q, W^K, W^V, W^O\} - \eta \nabla_{\{W^Q, W^K, W^V, W^O\}} \mathcal{L}_{\text{good}}(z_0, y). \quad (3.18)$$

Gradients flow through all four projections in a single backward pass but remain strictly block-local: the input x is detached, so no signal reaches earlier blocks.

Forward-Forward FFN Training

A transformer block consists of a Multi-Head Attention sub-layer followed by a position-wise Feed-Forward Network (FFN), separated by residual connections and LayerNorm (section 3.2). Once the attention sub-layer of the current block has been trained and frozen by the round-robin schedule of the previous subsection, the FFN sub-layer is trained next, before the pipeline moves on to the next block in the stack. The two linear layers of the FFN, $W_1 : \mathbb{R}^d \rightarrow \mathbb{R}^{d_{ff}}$ and $W_2 : \mathbb{R}^{d_{ff}} \rightarrow \mathbb{R}^d$, are trained sequentially in two phases so that each phase’s objective is strictly local to one linear layer:

Phase 1. Train W_1 on the goodness of the up-projected activation:

$$\mathcal{L}_{F_1} = \mathcal{L}_{\text{good}}(W_1 \text{LN}(x), y). \quad (3.19)$$

Phase 2. Freeze W_1 ; train W_2 on the goodness of the full FFN output including its residual connection:

$$\mathcal{L}_{F_2} = \mathcal{L}_{\text{good}}(\text{LN}(x) + W_2 \text{GELU}(W_1 \text{LN}(x)), y). \quad (3.20)$$

The residual is included in $\mathcal{L}_{F_2}$ because this is the quantity that the *next encoder block in the stack* (block $\ell+1$) receives as its input during inference. Training W_2 on the residual-summed output therefore aligns the local FF objective with what the downstream blocks will actually see.

Layer-Wise Training Pipeline

FF-AD trains each attention projection and each feed-forward phase *independently*, using the local goodness loss (Eq. 3.16) with standard autograd and the Adam optimizer.

FF-DD replaces the automatic-differentiation gradient of FF-AD with a directional-derivative estimate that requires no backpropagation through the component, following the FFzero algorithm of Guo et al. [8]. No backpropagation is performed at any point; only forward passes through the local loss are required.

Directional-Derivative Update Rule

Let $\omega \in \mathbb{R}^n$ denote the parameters of a single trainable component (one projection W^p or one FFN phase). A Gaussian perturbation direction $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is drawn and globally normalized:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}. \quad (3.21)$$

The directional derivative is estimated via **central differences**:

$$\hat{g} = \frac{\mathcal{L}_{\text{good}}(\omega + \varepsilon \hat{\mathbf{v}}) - \mathcal{L}_{\text{good}}(\omega - \varepsilon \hat{\mathbf{v}})}{2\varepsilon}, \quad (3.22)$$

where $\varepsilon = 0.5$ is the perturbation magnitude chosen empirically out of $\varepsilon \in [0.5, 0.1, 0.01, 0.001]$. Such a large value is required because $\hat{\mathbf{v}}$ is normalized over all n parameters, making the loss difference $\Delta \mathcal{L} \approx 2\varepsilon \hat{g}$ vanishingly small. The gradient *estimate* and parameter update are then:

$$\widehat{\nabla}_{\omega} \mathcal{L} = n \hat{g} \hat{\mathbf{v}}, \quad (3.23)$$

$$\omega \leftarrow \omega - \eta \widehat{\nabla}_{\omega} \mathcal{L}, \quad (3.24)$$

where n is the total parameter count of the component. The scaling by n compensates for the normalization of $\hat{\mathbf{v}}$, preserving the expected step magnitude regardless of component size. Equations (3.22)-(3.23) are written for a single perturbation direction. In general the estimate may be averaged over P random directions, each requiring two forward passes; this work uses $P = 1$, so no averaging term appears. This matches the single-direction setting ($P = 1$) used in every FFzero benchmark of Guo et al. [8], including their photonic-emulation experiment.

Stochastic operations (e.g. any active dropout) must see *identical* randomness in both forward passes so that the finite difference isolates the parameter perturbation. The CPU and CUDA RNG states are saved immediately before the $+\varepsilon$ pass and restored before the $-\varepsilon$ pass, guaranteeing symmetric evaluation.

The round-robin structure (attention) and two-phase structure (FFN) are identical to FF-AD; only the inner parameter update rule changes. FF-DD uses a larger learning rate $\eta = 5 \times 10^{-2}$ to compensate for the noise in the single-direction gradient estimate.

All FF variants train blocks greedily from input to output according to Algorithm 1. Here A_{ℓ} and F_{ℓ} are the attention and feed-forward sub-blocks of layer ℓ , $\hat{A}_{\ell}$ and $\hat{F}_{\ell}$ denote the same modules after their parameters have been frozen, and ϕ_{emb} is the (frozen) embedding stage. $\text{Stream}(D, f)$ is a streaming data loader that applies the composed transform f on-the-fly to each batch from D (see "Streaming and memory efficiency" below), and $\text{compose}(\cdot)$ is left-to-right function composition.

Algorithm 1 Layer-Wise Forward-Forward Training

Require: Blocks $\{(\mathcal{A}_{\ell}, \mathcal{F}_{\ell})\}_{\ell=0}^{L-1}$, train loader $\mathcal{D}$, val loader $\mathcal{D}_v$

- 1: Freeze all embedding parameters
 - 2: pipeline $\leftarrow [\phi_{\text{emb}}]$
 - 3: **for** $\ell = 0$ **to** $L - 1$ **do**
 - 4: Sample $(R_{\mathcal{A}}, M_{\mathcal{A}})$ with seed $s_{\mathcal{A}} + \ell \times 1000$
 - 5: Train $\mathcal{A}_{\ell}$ on $\text{Stream}(\mathcal{D}, \text{compose}(\text{pipeline}))$
 - 6: Freeze $\mathcal{A}_{\ell}$; append $\hat{\mathcal{A}}_{\ell}$ to pipeline
 - 7: Sample $(R_{F_1}, M_{F_1}, R_{F_2}, M_{F_2})$ with seeds $s_{F_1} + \ell \times 1000, s_{F_2} + \ell \times 1000$
 - 8: Train $\mathcal{F}_{\ell}$ on $\text{Stream}(\mathcal{D}, \text{compose}(\text{pipeline}))$
 - 9: Freeze $\mathcal{F}_{\ell}$; append $\hat{\mathcal{F}}_{\ell}$ to pipeline
 - 10: Evaluate end-to-end depth- $(\ell+1)$ accuracy on $\mathcal{D}_v$
 - 11: **end for**
 - 12: Replace head with $\text{SimplexHead}(R_{F_2}, M_{F_2})$ from the last block
-

The same projection-and-similarity computation defines the inference-time classifier, the *SimplexHead*. It reuses the prototype set (R_{F_2}, M_{F_2}) of the final block's second FFN phase: given the network's output representation (the [CLS] token for vision, or each sequence position for Seq2Seq), it applies the projection M_{F_2} , L_2 -normalizes, and returns the cosine similarities s_c to each class prototype, predicting $\hat{y} = \arg \max_c s_c$. The SimplexHead carries no trainable parameters of its own: the prototypes R_{F_2} and the projection M_{F_2} are fixed quantities sampled before training, and it replaces the linear classification head used under BP. This is the head referred to in the final line of Algorithm 1.

Streaming and memory efficiency. At depth ℓ , each training batch must pass through the frozen prefix $f^{(\ell)} = \hat{\mathcal{F}}_{\ell-1} \circ \dots \circ \phi_{\text{emb}}$ before reaching the trainable block. Rather than pre-computing and caching all intermediate activations — which would require memory proportional to the dataset size — $f^{(\ell)}$ is composed into a single callable and applied on-the-fly at fetch time inside a streaming data loader. No activations are retained between batches, keeping memory cost $O(\text{batch_size})$ regardless of depth L or dataset size.

Deterministic simplex geometry. Each trainable component at each block requires a simplex reference R and orthogonal projection M , sampled once before training begins. To guarantee reproducibility across runs and model depths, the RNG is seeded deterministically:

$$\text{seed}(\text{component}, \ell) = s_{\text{component}} + \ell \times 1000, \quad (3.25)$$

with fixed offsets $s_{\mathcal{A}} = 0$, $s_{F_1} = 1$, $s_{F_2} = 2$. The stride of 1000 ensures no two (component, block) pairs share a seed for any realistic depth. Both the NumPy and PyTorch RNG states are saved and restored around each sampling call so that global training randomness is unaffected.

Relevance to Photonic Hardware FFzero is particularly attractive for optical neural networks because it requires only *two forward passes per component per batch* to estimate the gradient, with no need for adjoint or error back-propagation circuits. The physically measurable scalar loss values $\mathcal{L}^+$ and $\mathcal{L}^-$ are sufficient to drive learning, making the algorithm compatible with in-situ training on photonic chips where backpropagation is physically infeasible.

Training Paradigm Comparison

Method	Attn. mech.	Attn. update	Local loss	FFN update	Head	η	Optimiser
BP-AD	Softmax	Joint	$\times$	Joint	Linear	1×10^{-3}	AdamW
BP-DD	Softmax	Joint	$\times$	Joint	Linear	5×10^{-2}	SGD + DD
FF-AD	Spherical	$W^Q \rightarrow W^K \rightarrow W^V \rightarrow W^O$	$\checkmark$	W_1 then W_2	SimplexHead	1×10^{-3}	Adam (per part)
FF-DD	Spherical	$W^Q \rightarrow W^K \rightarrow W^V \rightarrow W^O$	$\checkmark$	W_1 then W_2	SimplexHead	5×10^{-2}	SGD + DD
FF-Joint-AD	Spherical	$W^{Q,K,V,O}$ jointly	$\checkmark$	W_1 then W_2	SimplexHead	1×10^{-3}	Adam (joint)
FF-Joint-DD	Spherical	$W^{Q,K,V,O}$ jointly	$\checkmark$	W_1 then W_2	SimplexHead	5×10^{-2}	SGD + DD

Table 3.1: The comparison BP-AD $\leftrightarrow$ FF-DD entangles the training algorithm with the photonic-compatible architecture (Spherical Attention, SimplexHead) and reports the combined cost of moving to a forward-only optical system; the within-FF rows isolate the optimiser (AD vs. DD) and schedule (round-robin vs. joint) choices.

BP uses AdamW, consistent with established practice for end-to-end transformer training. The FF variants use plain Adam: weight decay acts on global loss surfaces, and its effect on block-local goodness objectives is unclear; excluding it keeps the comparison controlled.

The key distinguishing axes are:

- **Locality.** BP requires a global backward pass through every layer; all FF variants compute a block-local loss, making them compatible with layer-wise or asynchronous hardware deployment.
- **Backpropagation.** FF-AD and FF-Joint still use autograd *within* a component; only FF-DD and FF-Joint-DD are fully backpropagation-free and therefore amenable to in-situ training on photonic hardware.
- **Round-robin vs joint.** Round-robin training (FF-AD, FF-DD) explicitly decouples the four attention projections, potentially improving conditioning and reducing interference during learning. FF-Joint and FF-Joint-DD serve as ablations to quantify this effect.

Scaling evaluation. All six variants are evaluated across network depth ($L \in \{1, \dots, 5\}$ blocks) to test how each training algorithm behaves as the model grows. The motivation is a specific hypothesis carried over from FFzero [8]. Under a global objective the per-layer estimation noise is passed downstream and builds up with depth, so BP+DD should deteriorate as layers are added; under a layer-local objective each component is fitted on its own, leaving that noise contained, so the FF variants - FF+DD especially - should hold up as depth grows. Guo et al. [8] established this depth-resilience of local learning for multilayer perceptrons and convolutional networks; whether it holds for a Transformer is open, because self-attention couples positions across the entire sequence and may carry credit-assignment dependencies that a strictly local, layer-wise objective cannot capture. The depth sweep therefore tests two questions: (i) whether local FF training resists the depth-induced collapse expected of global directional-derivative training, and (ii) whether this resilience survives the non-local structure of attention. Chapter 4 reports accuracy as a function of L for all six variants.

3.4. Experimental Protocol

All six configurations of [section 3.3](#) are evaluated: BP-AD, BP-DD, FF-AD, FF-DD, FF-Joint-AD, and FF-Joint-DD. Recall that they vary along three independent axes: training rule (BP, FF, or FF-Joint), gradient method (autograd or directional derivative), and attention mechanism (softmax for BP, Spherical for FF) ([Table 3.1](#)). Full hyperparameters are listed in [Appendix A](#).

Every (rule, gradient) configuration is evaluated across encoder depths $L \in \{1, \dots, 5\}$ with 5 random seeds (42-46). BP is trained separately at each depth. FF and FF+Joint are trained once at $L = 5$ and the per-block validation accuracy is recorded for all shallower depths from the same greedy run. This gives 50 BP runs and 10 + 10 FF and FF+Joint runs, 70 in total per task. The depth sweep is needed because the FF objective is expected to be more sensitive to depth than BP: the credit-assignment path seen by each layer is one block long, and whether further blocks add useful representation or wash it out cannot be predicted in advance.

Tasks

Seven tasks are used, grouped into two families that probe different capabilities. The two vision tasks are image classification benchmarks trained with a Vision Transformer; the five sequence tasks are trained with a single decoder-only (GPT-style) causal language model and ordered by increasing difficulty. All sequence tasks share one format - the source, a separator token, and the target are concatenated into one stream and trained by next-token prediction, and one evaluation protocol: accuracy is measured by greedy auto-regressive generation from the prompt (not teacher forcing), so the reported numbers reflect free-running inference. For every sequence task, train and validation use a hashed 90/10 split with an explicit check that no validation input appears in training, so accuracy measures generalization to unseen inputs. The tasks are summarized in [Table A.4](#).

Vision (classification). **MNIST** (28×28 grayscale, 10 digit classes) tests whether a transformer with Spherical Attention can separate ten near-linearly-separable classes; it is presented without data augmentation. **CIFAR-10** (32×32 RGB, 10 object classes) raises the difficulty with natural images and is trained with standard augmentation (random crop, horizontal flip, TrivialAugmentWide, and random erasing). Both report top-1 classification accuracy on the [CLS] token, an attention map for each head, and a full confusion matrix is retained for per-class analysis.

Sequence (next-token prediction). **Copy** emits the source verbatim, testing faithful position-to-position routing and token reproduction. **Reverse** emits the source in reversed order, testing position-mirrored routing in which each output position must attend to its mirror in the input. **Argsort** emits the permutation that sorts the input - the rank indices, not the sorted values - so the model must perform a global comparison across all positions; inputs are sampled without replacement to keep the mapping strictly bijective (no ties). **Recall** presents a set of key-value pairs (distinct letter keys, digit values) followed by a query key, and must return the single value bound to that key, testing content-addressed associative lookup rather than positional routing. **Arithmetic** is the most demanding: each source is an expression $n_1 \square n_2$ with $\square \in \{+, -, \times\}$ in equal proportion and operands of up to $D = 5$ digits (subtraction constrained to $n_1 \geq n_2$; multiplication operands bounded to keep the product tractable). The answer is emitted digit-reversed (least-significant first) so that carry and borrow propagate in generation order. Unlike the routing and lookup tasks, it requires genuine multi-step computation—per-column carry/borrow and partial-product composition—that no amount of positional routing can shortcut.

Reported metric. The vision tasks report top-1 accuracy. The sequence tasks report position-wise token accuracy (the fraction of target positions matched under auto-regressive decoding), which awards partial credit. The tasks are summarized in [Table A.4](#).

4.1. Vision Tasks

MNIST Dataset

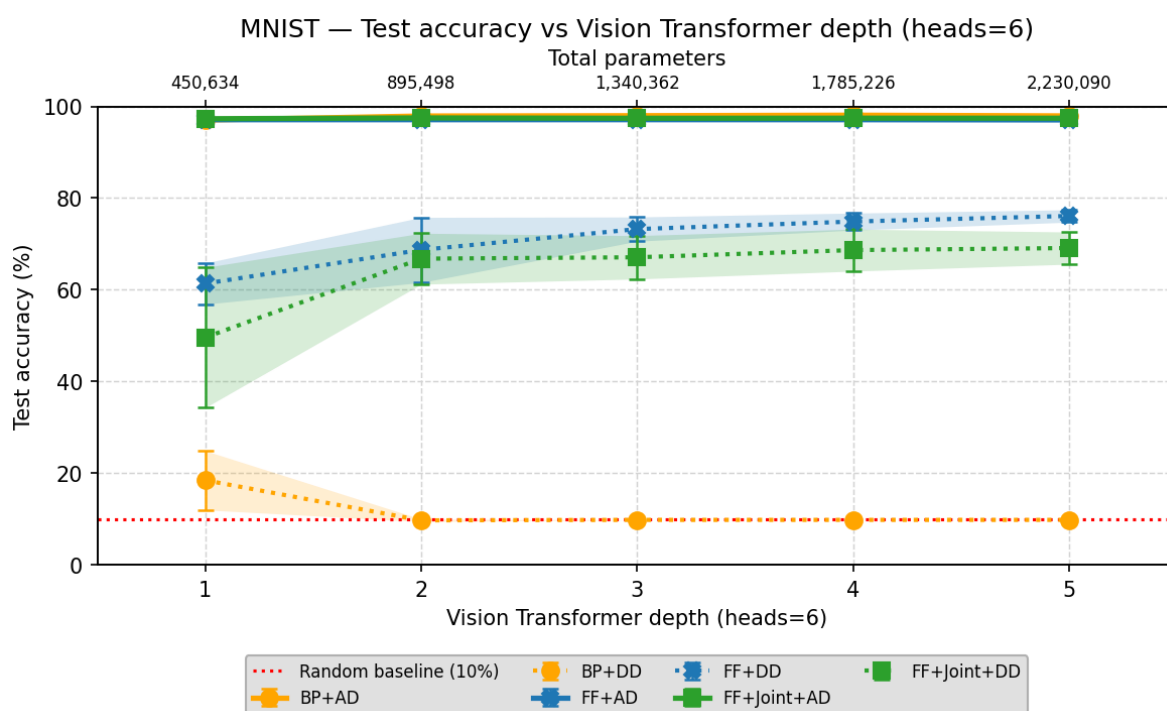


Figure 4.1: Results for MNIST on a Vision Transformer with 6 heads, 5 layers and 5 seeds.

Figure 4.1 splits the methods into three groups. Every autograd method (BP+AD, FF+AD, and FF+Joint+AD) reaches 97-98% and stays flat across depth. MNIST is therefore solved regardless of the training rule or the attention mechanism, as long as the gradient is exact. Adding layers neither helps nor hurts these methods, even though each layer adds parameters (top axis).

The directional-derivative methods separate from this cluster. FF+DD rises from $\approx 61\%$ at $L = 1$ to $\approx 77\%$ at $L = 5$, and FF+Joint+DD from $\approx 49\%$ to $\approx 68\%$. Both improve with depth and then flatten, and FF+DD stays about 7 pp ahead of FF+Joint+DD. BP+DD is the exception: it starts near 19% and collapses to the random baseline (10%) from $L = 2$ onward.

The attention maps in Figure 4.2 match this picture. BP+AD attends to a single small region in roughly the same place for every digit. FF+AD and FF+Joint+AD instead spread their attention and change

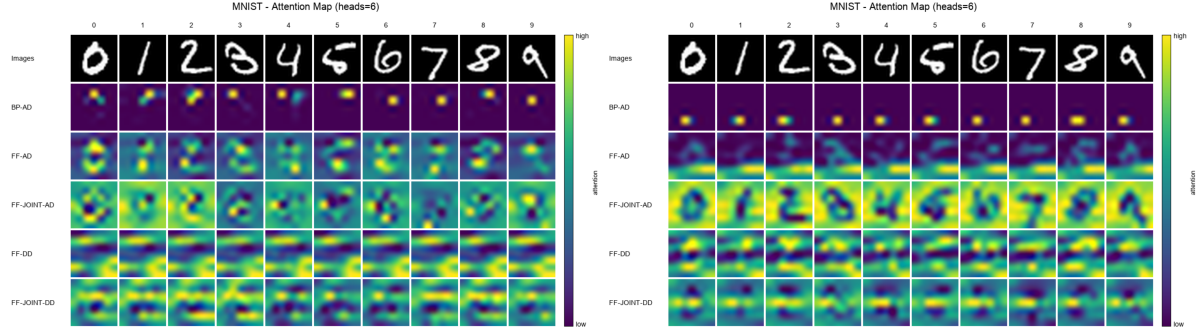


Figure 4.2: MNIST attention maps for $L = 5$, $H = 6$ (Left: head 5, Right: head 2). Rows: training method; columns: object class.

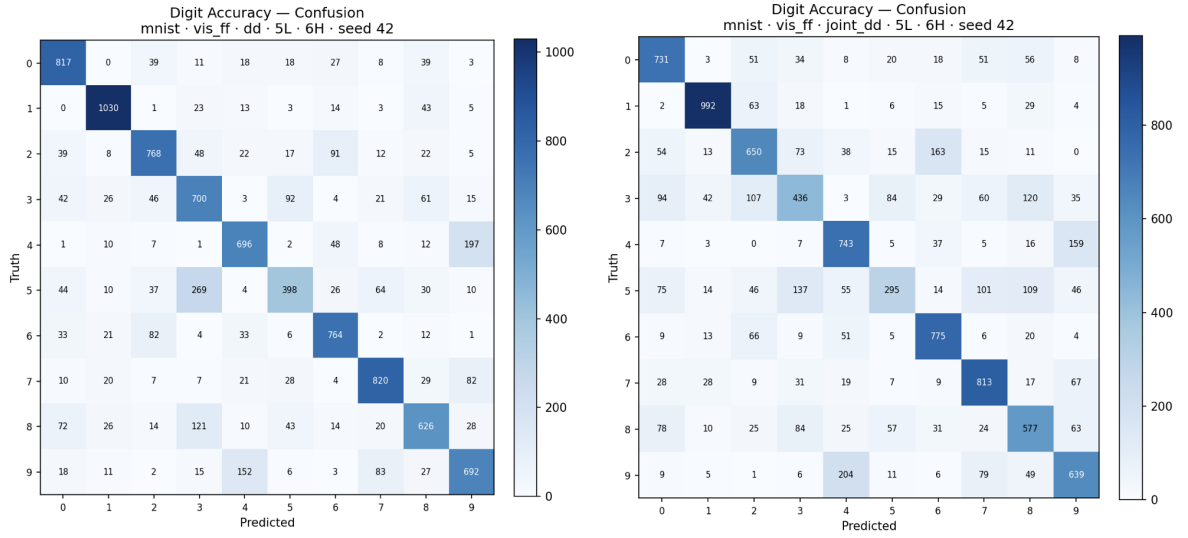


Figure 4.3: MNIST confusion maps for $L = 5$, $H = 6$ (Left: FF-DD, Right: FF-Joint-DD).

it with the digit, faintly tracing the digit shape. The two directional-derivative variants, FF+DD and FF+Joint+DD, collapse to a wavy stripe pattern that is nearly identical across all ten digits. Under the directional-derivative estimate the head stops reacting to the input and settles into a fixed positional filter, which is why these variants learn but never reach the autograd accuracy.

The confusion maps in [Figure 4.3](#), both taken at seed 42, allow a controlled comparison of the two variants. Both keep a clear diagonal and make structured rather than random errors, confusing visually similar digits: 4 and 9 in both directions, 5 most often as 3, 2 as 6, and 3 with 8. What separates the variants is the strength of that diagonal. FF-DD (left) classifies most digits well, its main weakness being the digit 5, which it often reads as 3. FF-Joint-DD (right) is weaker on most digits and collapses hardest on 3, classifying it correctly under half the time (436 of 1,010) and scattering it mostly to 8, 2, and 0. This heavier collapse on 3 and 5 is consistent with its lower overall accuracy ([Figure 4.1](#)).

Finally, the ≈ 7 pp advantage of FF+DD over FF+Joint+DD follows from the update schedule. FF+DD perturbs one projection at a time, so each directional-derivative estimate covers few parameters and stays relatively accurate; FF+Joint+DD perturbs all four projections at once, and the variance of a single-direction estimate grows with the number of parameters perturbed. The joint estimate is therefore noisier in attention.

CIFAR-10 Dataset

CIFAR-10 separates the methods more than MNIST ([Figure 4.4](#)). BP+AD is highest, rising from $\approx 58\%$ at $L = 1$ to $\approx 68\%$ for $L \geq 2$. The two autograd FF variants, FF+AD and FF+Joint+AD, sit together around 46-48% and stay flat with depth. Their directional-derivative counterparts are lower again: FF+DD

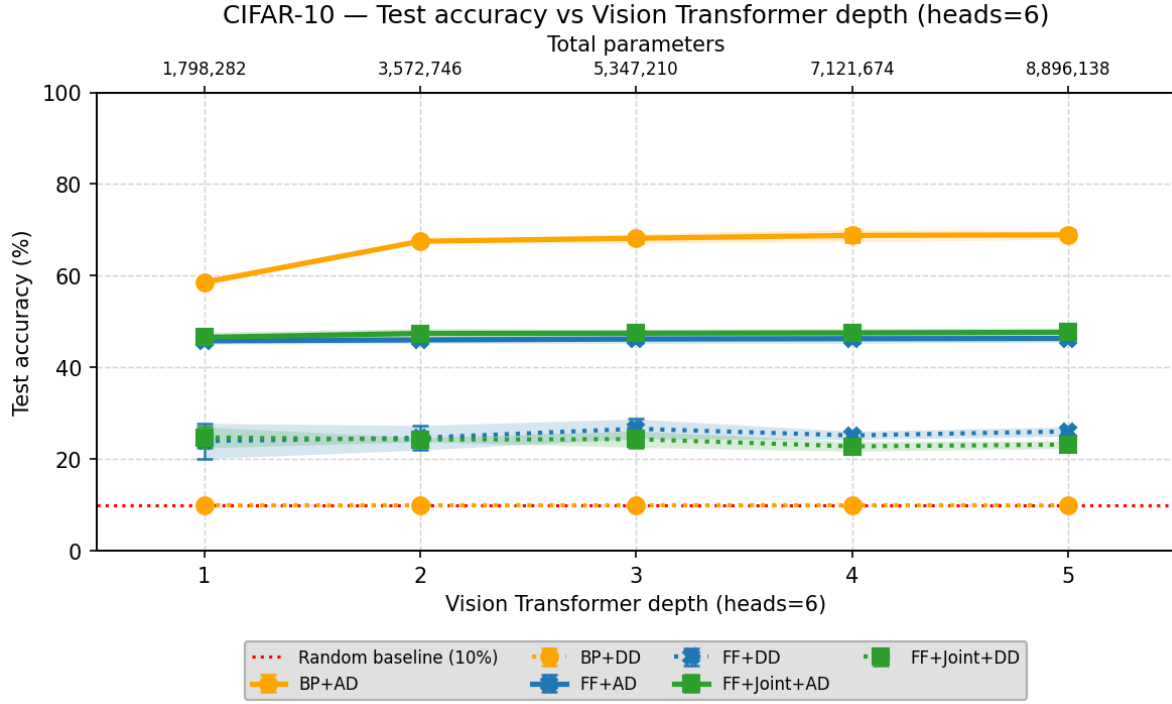


Figure 4.4: Results for CIFAR-10 on a Vision Transformer with 6 heads, 5 layers and 5 seeds.

around 26% and FF+Joint+DD around 23%, also flat. BP+DD stays at the random baseline (10%) at every depth.

Two differences from MNIST stand out. First, the FF variants trail BP+AD by about 20 pp even with exact gradients (FF+AD and FF+Joint+AD reach $\approx 47\%$), whereas on MNIST every autograd method reached 97%. Two factors contribute, and these experiments cannot separate them. (i) FF freezes the entire embedding stage - the patch-projection (a Conv2d feature extractor) and the CLS token are held at their random initialization, and only the transformer blocks and the simplex head are trained; BP trains the embedding end-to-end. CIFAR-10 needs learned low-level features such as edges and textures, which a random fixed extractor cannot provide, whereas MNIST is near-linearly-separable and random patch features already suffice - which is why the same frozen embedding does not hurt FF on MNIST. (ii) The layer-local goodness objective is a weaker training signal than the global cross-entropy loss. Both factors worsen with task difficulty, so the larger gap on CIFAR-10 is consistent with either or both. Second, the autograd-to-directional-derivative cost is comparable on the two vision tasks (≈ 21 pp on MNIST, ≈ 20 pp on CIFAR-10).

The round-robin and joint schedules are close on this task: the difference is at most a few points and does not point the same way under AD and DD (FF+Joint+AD is marginally ahead of FF+AD, FF+DD marginally ahead of FF+Joint+DD). On CIFAR-10 the limiting factors are the FF objective and the DD noise, not the update schedule. Depth does not help any FF variant: each extra layer adds parameters (top axis) but no accuracy.

The attention maps (Figure 4.5) tell the same story as MNIST. BP+AD attends to a few small, bright regions that move with the image and roughly follow the object, rather than sitting at one fixed location as on MNIST: CIFAR-10 objects vary in position, so the attention has to track them. FF+AD and FF+Joint+AD use broader, brighter attention that changes with the class. FF+DD and FF+Joint+DD again collapse to a striped, position-only texture that looks almost the same for every class. The collapse of the directional-derivative variants to a fixed positional pattern is therefore not specific to MNIST; it appears on CIFAR-10 as well and depends on a head.

The confusion maps (Figure 4.6), both at seed 42, show what the low directional-derivative accuracy looks like in practice. Neither variant spreads its errors evenly: a few classes act as sinks that the others

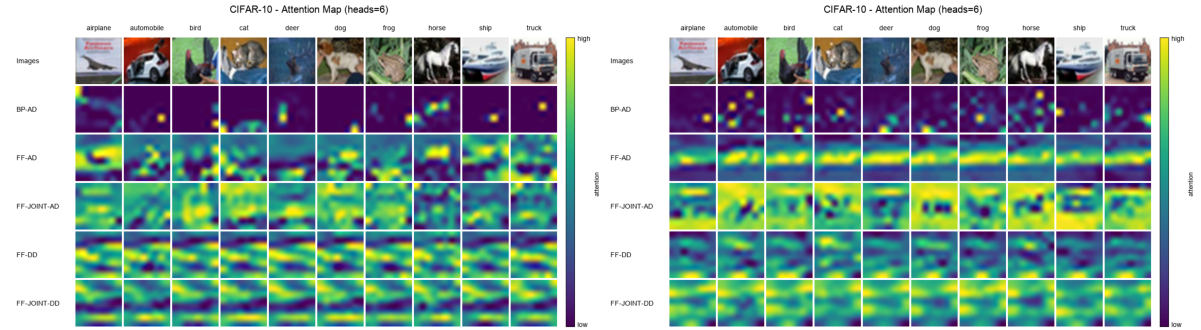


Figure 4.5: CIFAR-10 attention maps for $L = 5, H = 6$ (Left: head 5, Right: head 2). Rows: training method; columns: object class.

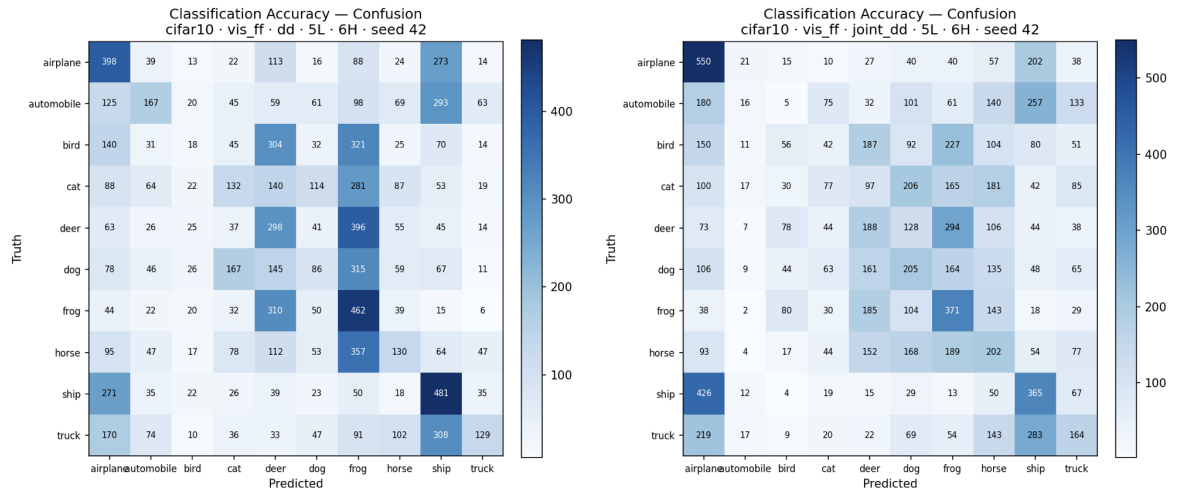


Figure 4.6: CIFAR-10 confusion maps for $L = 5, H = 6$ (Left: FF-DD, Right: FF-Joint-DD).

collapse into, with *frog* drawing the animals and *ship* drawing the vehicles in both. For FF-DD (left), every animal class is most often predicted as frog, and *automobile* and *truck* pile onto ship, while *bird* is almost never recovered (18 of roughly 1,000). FF-Joint-DD (right) fails along the same lines but more severely on the vehicle side: *automobile* and *truck* still fall onto ship, ship itself is then mostly read as *airplane* (426 of its samples, against 365 correct), and *automobile* all but vanishes (16 correct). On the animal side the joint schedule is in fact slightly better behaved, with dog and horse keeping a usable diagonal. The heavier vehicle collapse is consistent with FF-Joint-DD’s lower accuracy in [Figure 4.4](#).

4.2. Sequence Tasks

Copy Dataset

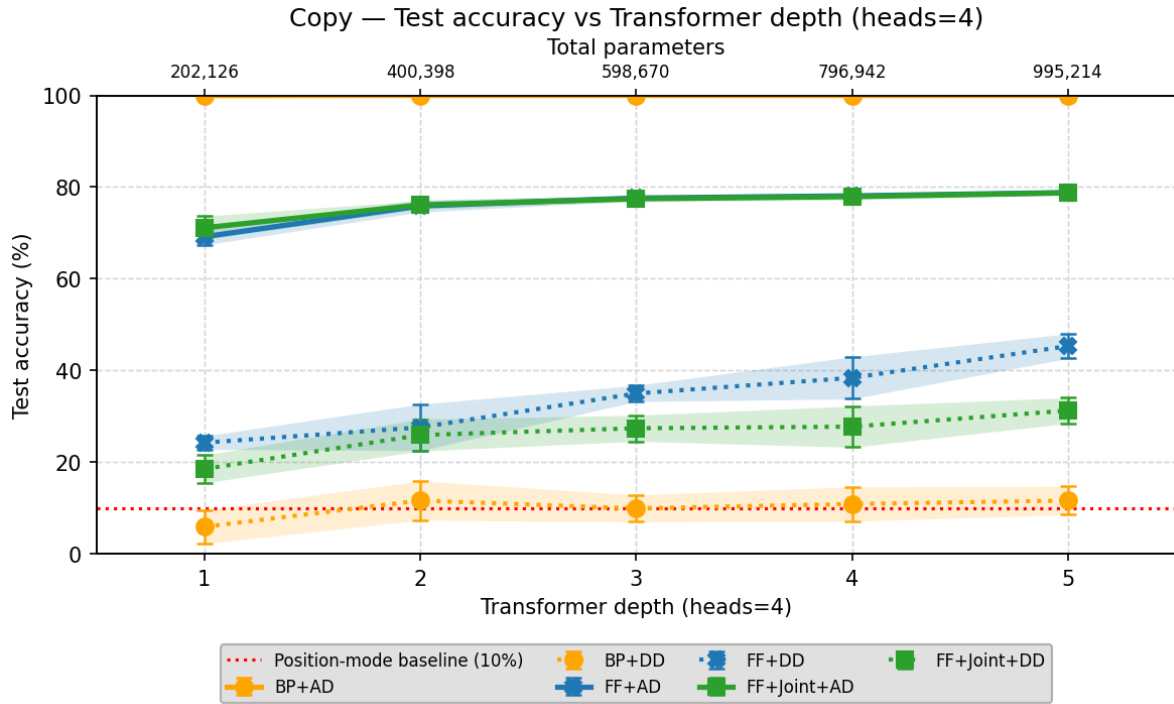


Figure 4.7: Results for Copy on a Seq2Seq Transformer with 4 heads, 5 layers and 5 seeds.

The Copy task asks the model to reproduce the input sequence of digits verbatim, scored by the fraction of output positions with the correct token. [Figure 4.7](#) shows a new behavior absent from the vision tasks: the FF accuracy *grows with depth* instead of staying flat. BP+AD solves the task perfectly (100%) at every depth - one softmax layer is enough to learn the copy routing with exact gradients. The autograd FF variants, FF+AD and FF+Joint+AD, start around 69% at $L = 1$ and rise together to $\approx 79\%$ at $L = 5$, tracking each other closely. The directional-derivative variants also rise with depth: FF+DD from $\approx 24\%$ to $\approx 45\%$, and FF+Joint+DD from $\approx 18\%$ to $\approx 33\%$. BP+DD stays at the position-mode baseline (10%) at every depth.

The depth dependence is the key difference from the vision tasks. Copying a variable-length sequence means routing information from each input position to its matching output position, and this routing genuinely needs several attention layers to build up. On MNIST and CIFAR-10 a single patch-aggregation layer already did most of the work, so extra layers added nothing; on Copy each extra layer adds accuracy. The FF objective is able to exploit that depth.

The gap between FF+DD and FF+Joint+DD also widens with depth, from a few points at $L = 1$ to about 15 pp at $L = 5$. The round-robin schedule gives each of the four projections its own directional-derivative estimate, while the joint schedule drives all four from one shared estimate. On a task whose difficulty grows with depth, round-robin’s lower-variance per-projection estimates inject less noise at each layer, and that advantage compounds across layers, so FF+DD pulls steadily ahead. BP+DD, by contrast, never leaves the baseline: it uses softmax attention and a global objective, and the directional-derivative

noise prevents it from using the extra layers at all.

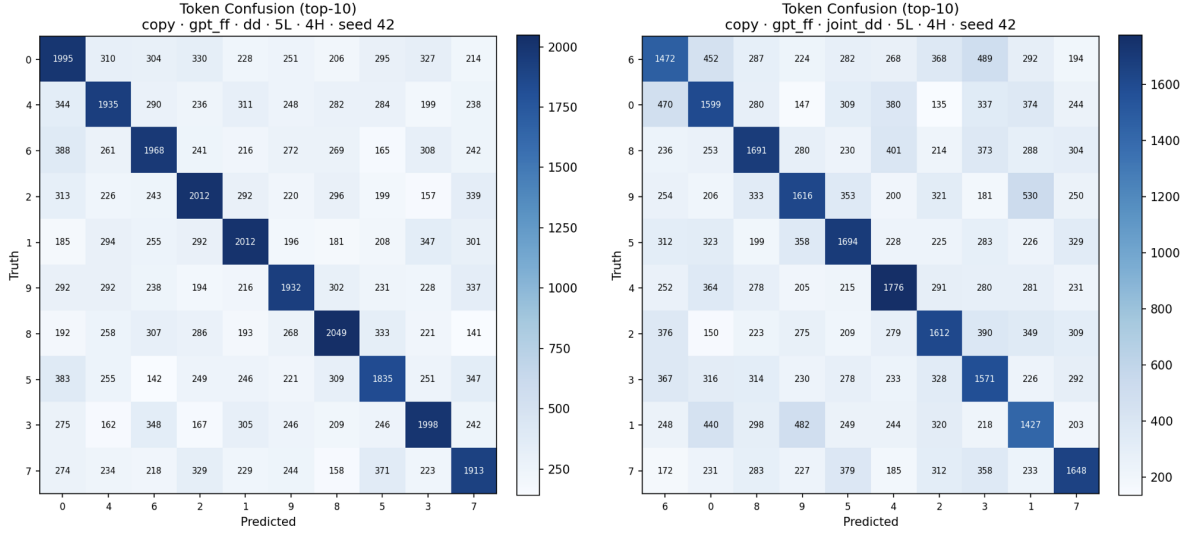


Figure 4.8: Copy confusion maps for $L = 5$, $H = 4$ (Left: FF-DD, Right: FF-Joint-DD).

The confusion maps in Figure 4.8, both at seed 42, compare the two directional-derivative variants for Copy task. Each map counts, per digit, how often it is copied correctly (diagonal) versus replaced by another digit (off-diagonal). FF-DD (left) has a strong, clean diagonal: every digit is copied correctly most of the time, and the errors are spread roughly evenly over the other digits. FF-Joint-DD (right) has a markedly weaker diagonal and much more off-diagonal mass - it makes more copy errors on every digit. This matches the ≈ 15 pp accuracy gap between the two in Figure 4.7.

Reverse Dataset

The Reverse task asks the model to emit the input sequence in reverse order, scored by per-position token accuracy. It is a harder routing problem than Copy: each output position must attend to a different input position, and the mapping depends on the sequence length, which varies between 5 and 10. Figure 4.9 shows the same qualitative pattern as Copy but at a lower ceiling. BP+AD solves the task essentially perfectly - $\approx 99\%$ at $L = 1$ and 100% from $L = 2$ on. The autograd FF variants improve with depth and then plateau: FF+Joint+AD from $\approx 40\%$ to $\approx 63\%$, and FF+AD from $\approx 38\%$ to $\approx 60\%$, with the joint schedule a few points ahead. The directional-derivative variants also rise with depth: FF+DD from $\approx 18\%$ to $\approx 37\%$, FF+Joint+DD from $\approx 19\%$ to $\approx 27\%$. BP+DD stays near the baseline ($\approx 13\text{--}17\%$) and gains nothing from depth.

Two points stand out against Copy. First, the FF ceiling drops from $\approx 79\%$ on Copy to $\approx 60\%$ on Reverse, with the same parameter count. Reversing a variable-length sequence requires a more complex, length-dependent routing than identity-copy, and the layer-local goodness objective approximates it less well. Even with depth and exact local gradients, FF+Joint+AD still trails BP+AD by about 40 pp - the cost of replacing the global error signal with a local objective. Second, the FF+DD-over-FF+Joint+DD gap again widens with depth (to ≈ 10 pp at $L = 5$), for the same reason as on Copy: round-robin perturbs one projection at a time, keeping each estimate low-variance, and that advantage compounds with depth, whereas the joint variant perturbs all four at once.

The confusion maps (Figure 4.10), both at seed 42, mirror the Copy result. FF-DD (left) has a clear diagonal: each token is reproduced correctly most of the time, with errors spread roughly evenly over the other tokens. FF-Joint-DD (right) has a weaker diagonal and more off-diagonal mass, making more token errors throughout. As on Copy, this matches the ≈ 10 pp accuracy gap between the two variants in Figure 4.9.

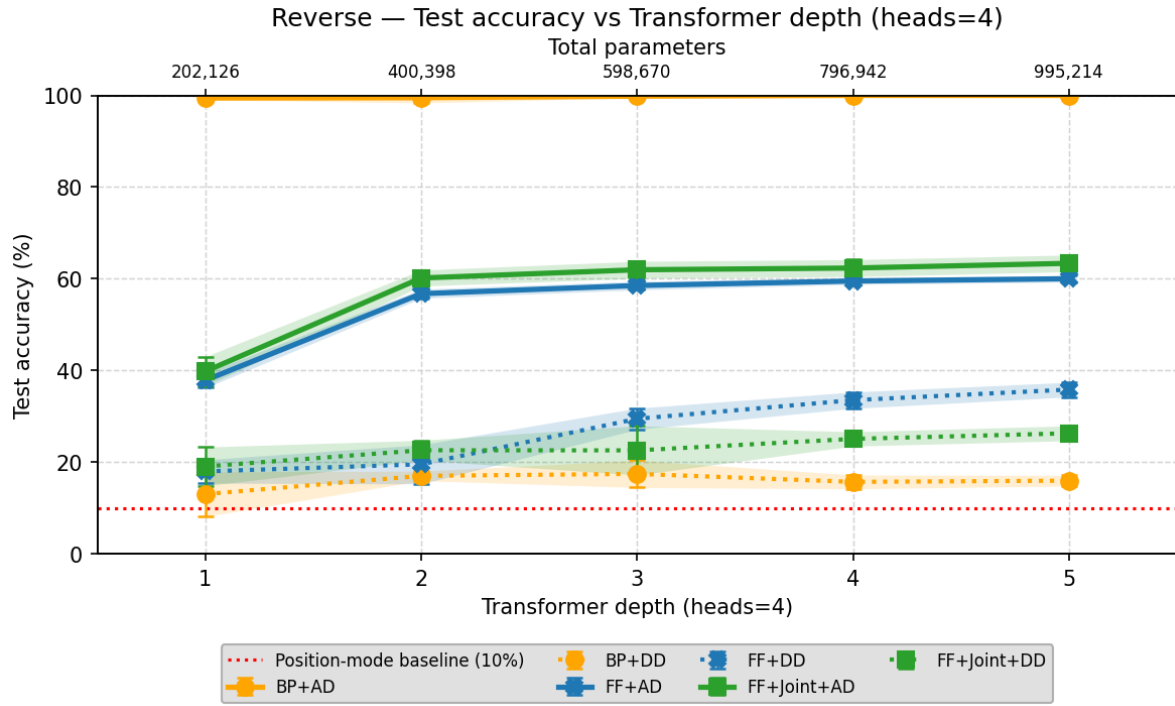


Figure 4.9: Results for Reverse on a Seq2Seq Transformer with 4 heads, 5 layers and 5 seeds

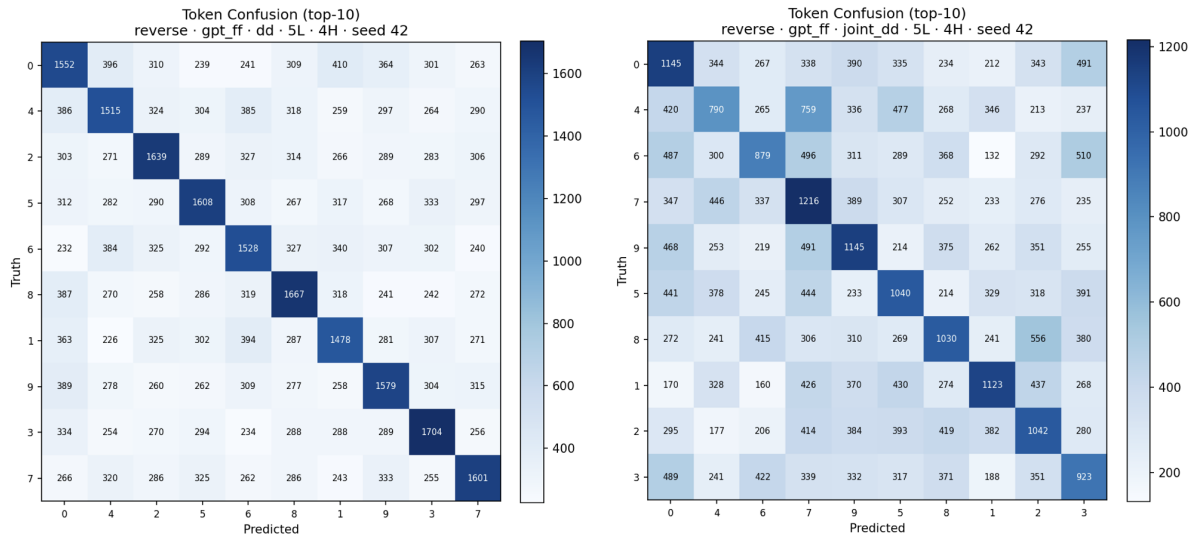


Figure 4.10: Reverse confusion maps for $L = 5$, $H = 4$ (Left: FF-DD, Right: FF-Joint-DD).

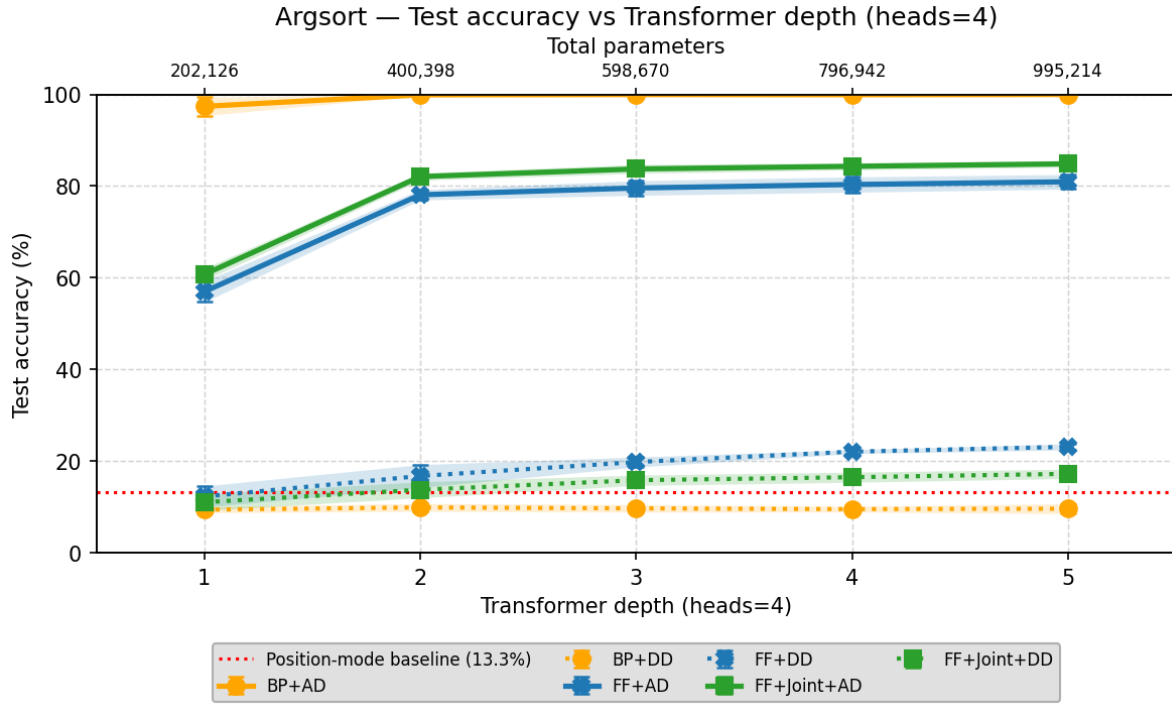


Figure 4.11: Results for Argsort on a Seq2Seq Transformer with 4 heads, 5 layers and 5 seeds

Argsort Dataset

The Argsort task requires the model to output the input tokens in sorted order by index, scored by per-position token accuracy. Figure 4.11 shows the largest gap between the autograd and directional-derivative methods of any task in the suite. BP+AD solves it almost perfectly ($\approx 97\%$ at $L = 1$, $\approx 100\%$ from $L = 2$). The autograd FF variants also do well: FF+Joint+AD rises from $\approx 60\%$ at $L = 1$ to $\approx 84\%$ at $L = 5$ and FF+AD from $\approx 57\%$ to $\approx 81\%$, with most of the gain coming from the first-to-second layer step. This is the highest FF+AD ceiling of any task. The directional-derivative variants, however, stay low: FF+DD reaches only $\approx 23\%$ and FF+Joint+DD $\approx 17\%$ at $L = 5$, both just above the 13.3% baseline. BP+DD sits *below* the baseline ($\approx 9\text{-}10\%$) at every depth.

Argsort therefore isolates the cost of the directional-derivative estimator particularly cleanly. The layer-local goodness objective handles sorting well when the gradient is exact - FF+AD reaches $\approx 84\%$ - so the drop to $\approx 23\%$ under FF+DD is almost entirely due to the noise in the single-direction estimate, not to the objective itself. Sorting needs precise ranking decisions about which token belongs in each output position, and the noisy scalar estimate cannot resolve them. As on the other sequence tasks, FF+DD stays a few points ahead of FF+Joint+DD (round-robin perturbs one projection at a time, keeping each estimate low-variance), and BP+DD fails outright, here dropping below even the trivial position-mode predictor.

The confusion maps (Figure 4.12), both at seed 42, show what the $\approx 20\%$ accuracy looks like. Neither directional-derivative variant produces a clean diagonal. Both collapse toward predicting the most frequent, low-rank tokens and confuse neighboring ranks heavily, while the highest tokens (8 and especially 9) are almost never produced correctly. FF-Joint-DD (right) is more collapsed than FF-DD (left): its diagonal is weaker and its predictions are concentrated on even fewer tokens. This matches the lower accuracy of FF-Joint-DD in Figure 4.11.

Recall Dataset

The Recall task is associative key-value lookup. The input is a sequence of (key, value) pairs followed by a query key, and the model must emit the value bound to that key. The output is a single token, so token accuracy equals sequence accuracy. The task isolates attention’s core function: there is no positional or local shortcut, because the query and the keys are drawn from the same alphabet and appear at

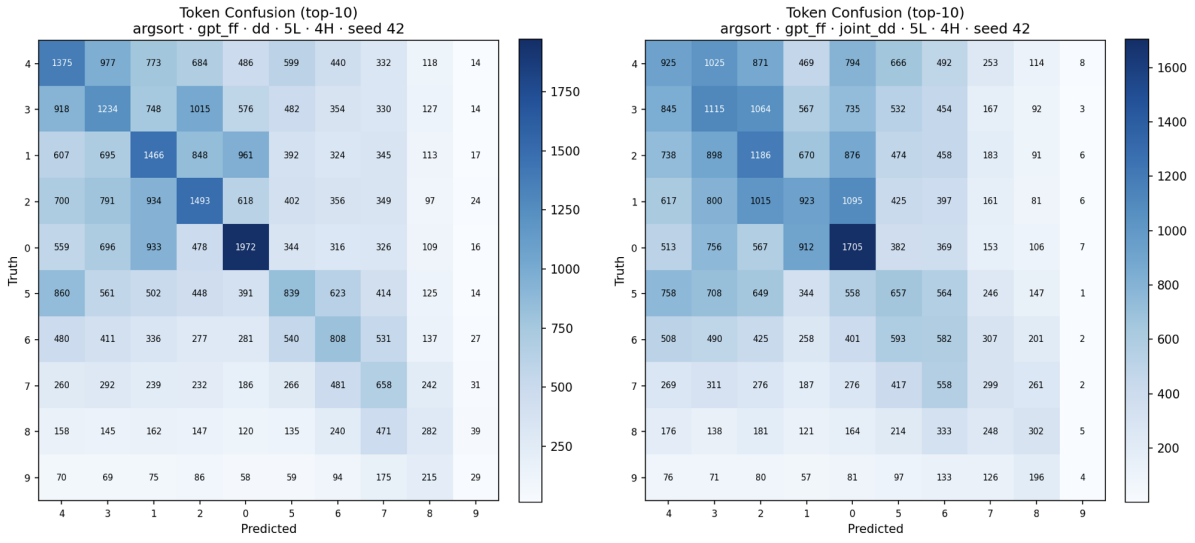


Figure 4.12: ArgSort confusion maps for $L = 5$, $H = 4$ (Left: FF-DD, Right: FF-Joint-DD).

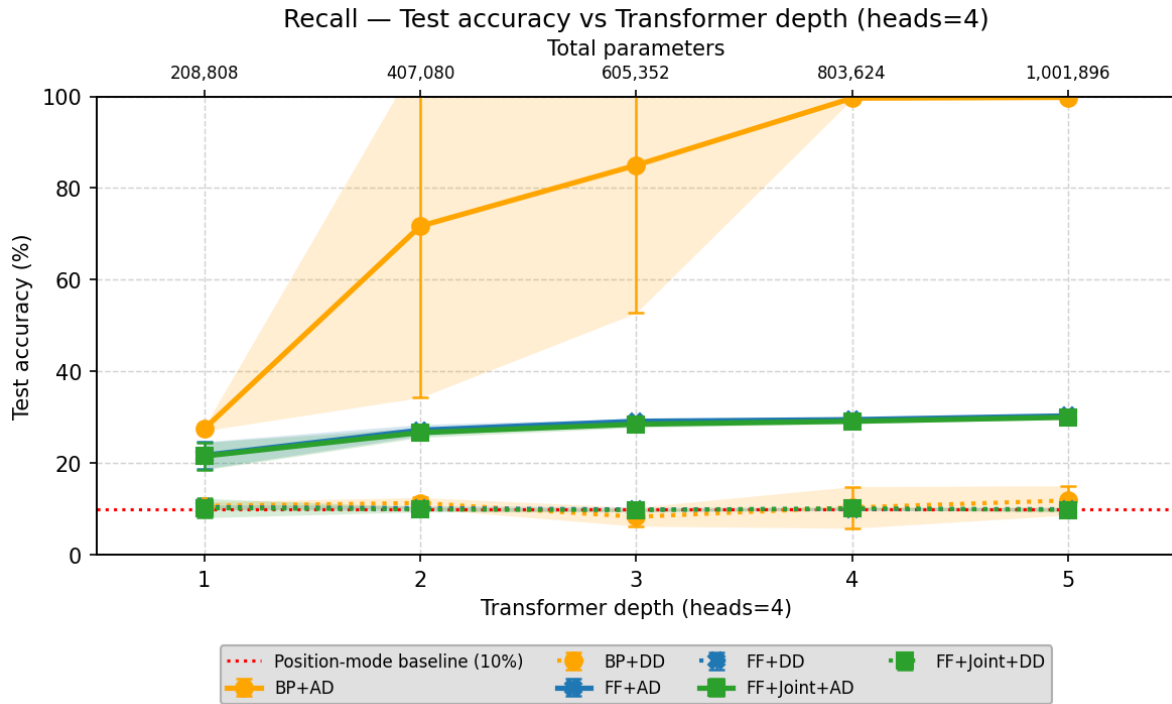


Figure 4.13: Results for Recall on a Seq2Seq Transformer with 4 heads, 5 layers and 5 seeds.

arbitrary positions, so the model must attend to the matching key and copy its value.

Figure 4.13 shows the widest spread between methods of any task. BP+AD undergoes a phase transition with depth: $\approx 27\%$ at $L = 1$, $\approx 72\%$ at $L = 2$ (with a very wide confidence band), and $\approx 100\%$ from $L = 4$ on. The wide bands at $L = 2$ and $L = 3$ are bimodal across seeds: some initializations find the key-matching attention pattern and solve the task, others do not. The retrieval solution is therefore a narrow basin that exact gradients reach only once enough layers are stacked. The autograd FF variants show no such transition; FF+AD and FF+Joint+AD track each other and plateau around 30%, far below BP+AD at every depth. All three directional-derivative variants (FF+DD, FF+Joint+DD, BP+DD) sit together at the 10% baseline.

Recall is the one task where even FF+AD, with exact gradients, falls far short. The likely reason is architectural. Retrieving a single value requires sharp, near one-hot attention weights that concentrate on the matching key. Softmax produces such peaked weights through its exponential normalization; Spherical attention has no softmax, and its bounded cosine scores cannot be sharpened into a one-hot distribution. The FF goodness objective gives no direct incentive to peak the attention either. The bottleneck here is therefore the attention mechanism, not the optimizer, which is why the AD/DD distinction that orders every other task disappears on Recall. FF also differs from BP in its objective and its frozen embedding, so the failure cannot be pinned on attention alone; even so, the inability to form one-hot weights is the most direct explanation.

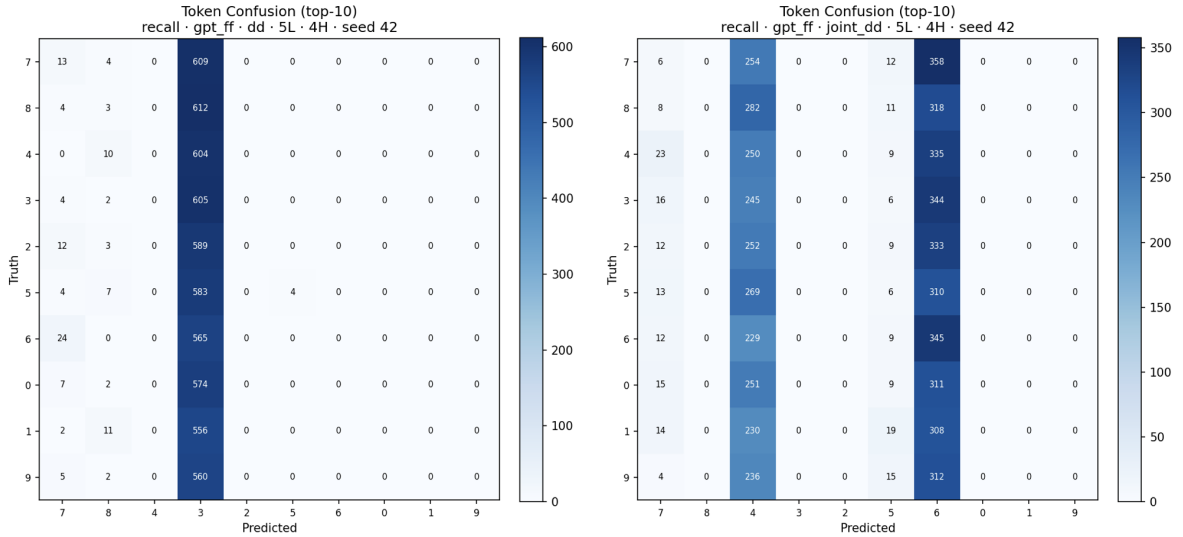


Figure 4.14: Recall confusion maps for $L = 5$, $H = 4$ (Left: FF-DD, Right: FF-Joint-DD).

The confusion maps (Figure 4.14), both at seed 42, make the failure of the directional-derivative variants explicit. FF-DD (left) predicts the *same* token for almost every query: the “predicted 3” column is filled while the rest are essentially empty, so the model ignores the key and outputs a constant. FF-Joint-DD (right) is almost as degenerate, splitting its predictions between just two tokens (mostly 6, with some 4) regardless of the query. Neither performs any content-based retrieval, and both sit at the 10% baseline because always emitting a fixed value is correct about one time in ten.

Math Dataset

The Math task asks the model to produce the answer to a short arithmetic expression, scored by per-position token accuracy. Figure 4.15 shows it is the hardest task in the suite for every method. BP+AD reaches $\approx 83\%$ at $L = 1$ and rises to $\approx 95\%$ by $L = 4$ - high, but not the perfect score it achieved on Copy, Reverse, and Argsort. Arithmetic carries genuine output uncertainty that exact gradients do not fully remove at this model size. The autograd FF variants plateau around 50% (FF+Joint+AD $\approx 51\%$, FF+AD $\approx 49\%$), about 45 pp below BP+AD - the largest FF-BP gap of any task. The directional-derivative variants are lower still: FF+DD plateaus near 31%, while FF+Joint+DD peaks at $\approx 30\%$ ($L = 4$) and then drops to $\approx 22\%$ at $L = 5$. BP+DD declines with depth from $\approx 15\%$ to $\approx 8\%$, staying below the 16.5%

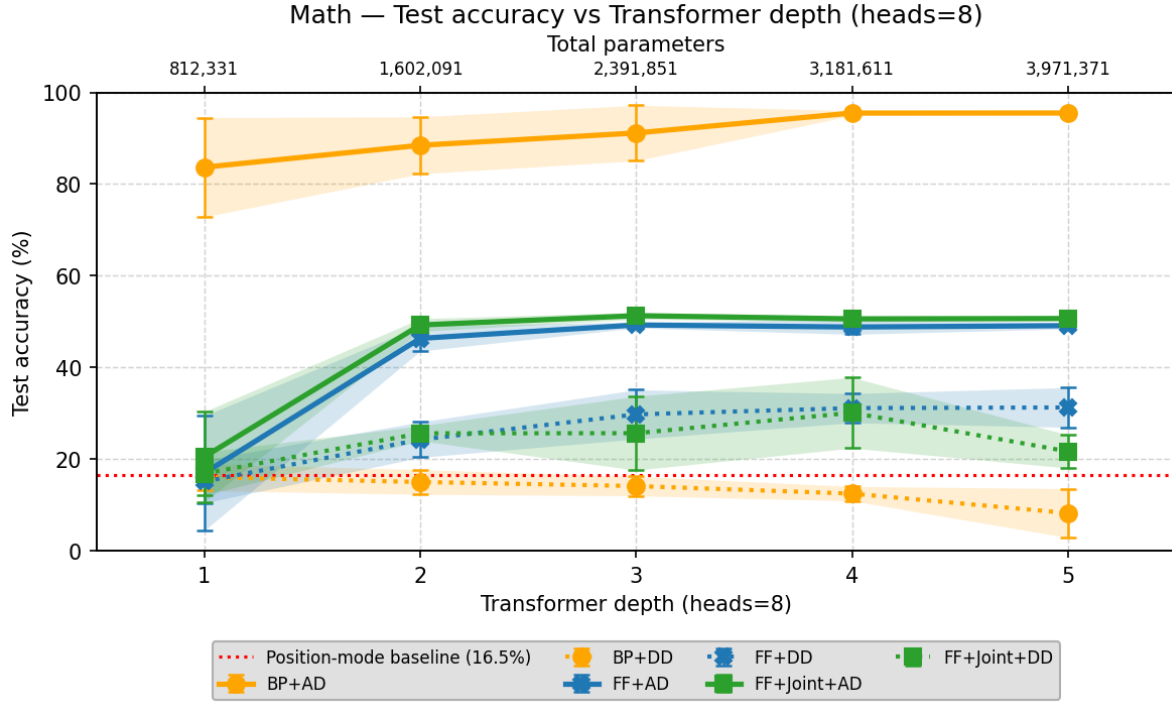


Figure 4.15: Results for Math on a Seq2Seq Transformer with 8 heads, 5 layers and 5 seeds.

baseline at every depth.

The large FF-BP gap reflects the structure of the task. Arithmetic requires carry propagation across positions which a layer-local goodness objective rewards only indirectly, so even FF+AD with exact gradients caps near 50%. This is the opposite of Argsort, where the local objective handled the task well ($\approx 84\%$) and only the gradient estimator was the limit.

Math is also the first task where added depth begins to *hurt* the directional-derivative variants rather than just plateau. FF+Joint+DD and BP+DD both decline at the deepest settings, and BP+DD falls further below the baseline as layers are added. With 8 heads per layer, each added layer increases the variance of the single-direction estimate; for BP+DD, the global softmax objective amplifies this further, so the estimate becomes not just uninformative but actively misleading. (Because BP+DD differs from the FF variants in both the objective and the attention mechanism, these two effects cannot be separated here.) This is the clearest case in the suite that directional-derivative training does not scale with depth under softmax attention.

The confusion maps (Figure 4.16), both at seed 42, show what the low directional-derivative accuracy looks like on the answer digits. Neither variant performs the arithmetic; each fills the answer positions with a few high-frequency digits regardless of the true value, so a small number of predicted columns dominate the map while the rest stay near-empty. FF-DD (left) splits its output mainly between the tokens 1 and 0, with some 2. FF-Joint-DD (right) concentrates instead on a single dominant token, 1, with smaller spillover to 2, 0, and 8. The heavier collapse onto one digit leaves the joint variant correct on fewer true digits, consistent with its lower Math accuracy in Figure 4.15.

4.3. Summary

Across all seven tasks the method ordering is consistent: BP+AD is best everywhere, the autograd FF variants come next, and the directional-derivative FF variants follow. BP+DD is last on every task except Recall, where all three directional-derivative variants collapse to the random baseline and are statistically indistinguishable. Table 4.1 collects the performance summary across all tasks.

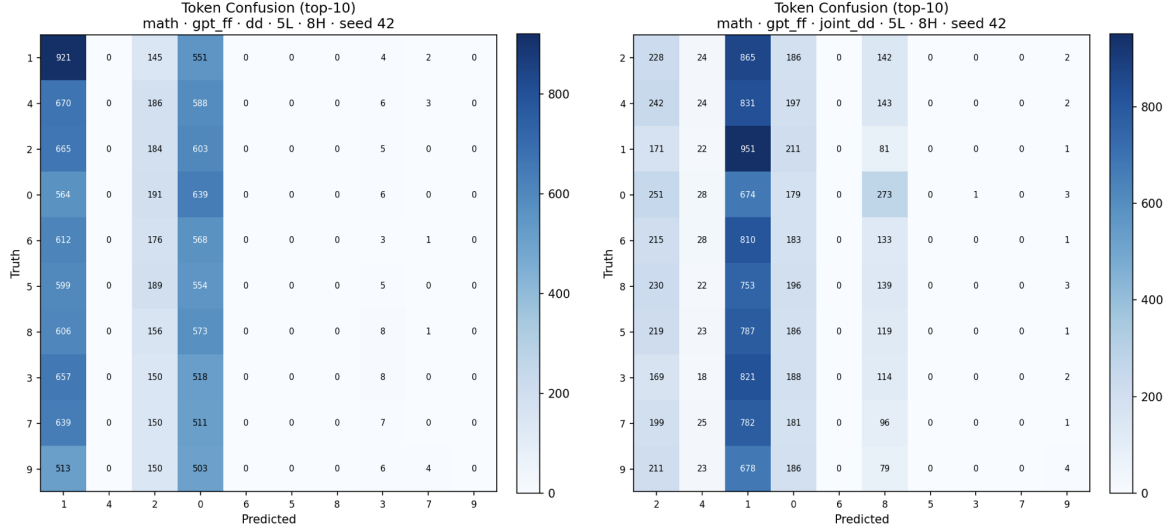


Figure 4.16: Math confusion maps for $L = 5$, $H = 8$ (Left: FF-DD, Right: FF-Joint-DD).

Task	BP-AD	BP-DD	FF-AD	FF-DD	FF-Joint-AD	FF-Joint-DD
MNIST	97.93 ± 0.25	9.80 ± 0.00	97.04 ± 0.33	76.10 ± 1.35	97.42 ± 0.18	69.10 ± 3.51
CIFAR-10	68.92 ± 0.96	10.00 ± 0.00	46.32 ± 0.73	26.04 ± 1.01	47.67 ± 1.00	23.17 ± 0.84
Copy	99.99 ± 0.02	11.62 ± 3.09	78.80 ± 0.59	45.31 ± 2.68	78.75 ± 0.50	31.25 ± 2.79
Reverse	99.99 ± 0.02	15.96 ± 1.15	60.00 ± 0.72	35.82 ± 1.59	63.37 ± 1.82	26.30 ± 1.62
Argsort	99.98 ± 0.02	9.62 ± 1.07	80.94 ± 1.59	23.12 ± 0.66	84.85 ± 0.73	17.23 ± 1.01
Recall	99.78 ± 0.18	11.86 ± 3.20	30.32 ± 0.26	9.95 ± 0.41	29.99 ± 0.39	9.82 ± 0.45
Math	95.50 ± 0.41	8.23 ± 5.33	49.11 ± 0.70	31.27 ± 4.33	50.68 ± 0.52	21.69 ± 3.61

Table 4.1: Test accuracy (% , mean $\pm$ standard deviation over 5 seeds) at the deepest depth ($L = 5$) for all six training variants. BP uses softmax attention; all FF variants use Spherical attention. Results compared across FF-AD vs FF-Joint-AD, and FF-DD vs FF-Joint-DD - the best are highlighted.

The forward-only gap is real but task-dependent, and its cause differs by task. Replacing global backpropagation with the layer-local FF objective costs essentially nothing on MNIST, where every autograd method solves the task. The gap grows on harder tasks, and for three distinct reasons. On CIFAR-10 and Math the *local objective* is the bottleneck: it cannot learn the features CIFAR-10 needs or the cross-position carry propagation Math needs, so even FF+AD with exact local gradients trails BP+AD by 20-46 pp. On Argsort the objective is not the problem, since the FF autograd variants reach 81-85%; there the limit is the *DD estimator*. On Recall the limit is *architectural*: even FF+AD reaches only $\approx 30\%$, because Spherical attention cannot form the sharp, near one-hot weights that content-addressed retrieval requires. Identifying which of the three binds on a given task is the main analytical result of this chapter.

The directional-derivative estimator adds a further, separable cost. Within the FF family, switching from autograd to the single-direction directional-derivative estimate lowers accuracy on every task, and because the objective and the architecture are held fixed, this cost is isolated cleanly. It is moderate on most tasks (≈ 18 -34 pp) but largest on Argsort (≈ 58 pp), where the precise per-position ranking decisions are exactly what a noisy scalar estimate cannot resolve. The directional-derivative variants also vary far more across seeds than their autograd counterparts (Math FF+DD ± 4.33 , MNIST FF-Joint-DD ± 3.51), as a noisy estimate should cause. Even so, where the round-robin advantage is claimed it exceeds that variance: on Copy, 45.31 ± 2.68 against 31.25 ± 2.79 , the two do not overlap. FF+DD remains trainable everywhere except Recall, holding above the random baseline on six of seven tasks; only on Argsort is the margin narrow ($\approx 23\%$ against a 13.3% baseline). That it trains at all is the feasibility

result on which the photonic case rests.

Round-robin is the better directional-derivative schedule. Between the two photonic-deployable variants, FF+DD (round-robin) matches or beats FF+Joint+DD (joint) on every task, and the advantage *widens with depth* on the sequence-routing tasks (up to ≈ 14 pp on Copy, ≈ 10 pp on Reverse). By perturbing one projection at a time, the round-robin schedule keeps the number of parameters perturbed per estimate small, so each directional-derivative estimate stays accurate; the joint schedule perturbs all four projections at once from a single shared direction, and the variance of the estimate grows with the perturbed dimension. Round-robin is thus a finer-grained form of the locality that makes forward-only training viable, and its lower per-layer noise accumulates into a widening advantage as depth grows.

Global directional-derivative training (BP+DD) does not work. BP+DD never learns any task. It stays within a few points of the baseline everywhere, drops below it on Argsort and Math, and on Math *declines* with depth, in every case far below every other method. This matches the central FFzero hypothesis, that directional-derivative errors compound across layers under a global objective, and it is most visible on Math and MNIST, where BP+DD degrades as depth increases; on the other tasks it already fails at $L = 1$, leaving no signal to compound. The contrast with the trainable FF+DD variants is the clearest evidence that locality, not forward-only gradients alone, is what makes directional-derivative training viable.

Depth. Depth behaves differently across task families. On the vision tasks the autograd methods are flat with depth, because a single attention layer already does most of the work; the directional-derivative variants are the exception on MNIST, where they climb from $\approx 61\%$ at $L = 1$ to $\approx 77\%$ at $L = 5$ before flattening, while on CIFAR-10 even they stay flat. On the sequence-routing tasks (Copy, Reverse, Argsort) the FF variants improve with depth, because the routing genuinely needs several composed layers. On Math, depth begins to *hurt* the directional-derivative variants once the parameter and head counts are large, as estimator variance grows faster than the useful signal.

Limitations. Three limitations of the forward-only approach emerge. (1) Relative to backpropagation, the local goodness objective is a weaker training signal on tasks that require global computation, and FF additionally trains on a frozen, randomly-initialized input embedding, so it cannot learn input features; these two effects are confounded in the present experiments. (2) Relative to softmax, Spherical attention cannot produce one-hot weights, so tasks that depend on sharp content-based retrieval (Recall) are out of reach regardless of the optimizer. (3) The directional-derivative estimator adds a noise floor that caps accuracy and is most damaging on tasks requiring precise decisions or at large depth. None of these is fatal in general - FF+DD trains to a useful regime on six of seven tasks - but together they bound how close a forward-only, photonic-compatible Transformer can come to a backpropagation-trained one.

5

Discussion & Conclusion

5.1. Principal findings

The central question of this thesis, whether a Transformer can be trained without backpropagation using only the forward operations available to a photonic substrate, is answered in the affirmative. The fully forward-only variant, FF-DD, trains to a useful operating point on six of the seven tasks, staying well above the random baseline everywhere except Recall (Chapter 4). It does not match the backpropagation baseline, but it learns, and it does so with no backward pass, no automatic differentiation, and no softmax.

The comparison across the six variants locates why this is possible. A forward-only gradient estimate on its own is not enough: BP-DD, which applies the same directional-derivative estimate to a global objective, fails on every task and degrades further with depth. What separates it from FF-DD is locality. Estimating the gradient one block at a time keeps the perturbed dimension small and stops the estimation error from compounding through the network, whereas the global estimate accumulates that error across layers until the signal is lost. Locality, not the forward-only gradient alone, is what makes directional-derivative training viable; this is the clearest single result of the comparison, and it carries the FFzero hypothesis [8] from multilayer perceptrons and convolutional networks to a Transformer, and specifically its attention mechanism.

This also settles the question the methodology left open: whether the stability of local, layer-by-layer training under increasing depth still holds when the stacked layers are attention layers that mix information across the whole sequence. It does, conditionally: on the sequence-routing tasks the FF variants improve monotonically with depth, so stacking attention layers composes useful routing rather than washing it out, and the resilience breaks only where the computation must couple positions across the whole sequence at scale.

Under exact local gradients, FF-AD produces attention maps unlike those of the backpropagation baseline: where BP-AD concentrates on a few sharp regions, FF-AD spreads its weight into diffuse silhouettes that follow the object rather than pinpointing it (chapter 4). This is a qualitatively different regime, and the interpretability and pruning tools built around sharp, peaked attention may not carry over to it. An open direction for making Forward-Forward more effective in Vision Transformers is to characterize these distributed maps and design forward-only objectives that exploit them rather than work against them.

5.2. Limitations

The experiments expose where forward-only training of a Transformer reaches its ceiling, and the general lesson is that the ceiling has two fundamentally different origins: the learning rule and the architecture.

The cost of removing backpropagation. Two of the limits are inherent to forward-only learning and would arise in any such method, not only this one. Backpropagation is the only mechanism that assigns credit exactly and globally; remove it and a method is forced into a trade-off. It can replace the global objective with a local one, in which case each layer is optimized only for its own goodness and cannot learn a representation whose value lies in what a later layer does with it. Or it can keep a global target but estimate its gradient from forward evaluations alone, in which case the estimate carries an irreducible variance that grows with the number of parameters perturbed. FF-DD pays both costs: it is local and it estimates its gradient. Neither can be removed, only reduced. A forward-only Transformer will therefore always trail a backpropagation-trained one, and the gap grows the more a task depends on global computation or precise decisions. This is an inherent price of the approach, not a flaw of this particular implementation.

The cost of a hardware-compatible architecture. The third limit is unrelated to forward-only learning and would persist even under exact gradients. Making attention photonic-compatible meant removing the softmax, and with it the network's only means of forming sharp, near one-hot attention weights. Workloads that hinge on such selection are therefore out of reach regardless of how the network is trained. This is an architectural constraint, not a learning one, and unlike the two costs above it would be lifted by restoring a sharpening non-linearity rather than by changing the training rule. The distinction matters for anyone building on this work: the first two limits are the standing cost of forward-only learning, while the third is a design choice that can be revisited.

5.3. Addressable Bottlenecks and Future Work

Several of these limits have concrete levers, none of which reintroduces a global backward pass. This work used a single perturbation direction per update; averaging over more directions lowers the variance of the estimate at a proportional cost in forward passes, and because the gain saturates after only a handful of directions [8], a modest increase may recover part of the gap cheaply. The inability to form sharp selection could be removed without touching the training rule, for example by devising a new attention computation in the spirit of the Kramers-Kronig kernel [3]. The input embedding is frozen at initialization here; letting it adapt, or starting from a pre-trained feature extractor, would test how much of the natural-image gap is feature learning rather than the objective. The benchmarks also cover only vision and synthetic algorithmic tasks, so whether the layer-local objective handles the hierarchical statistics of natural language is untested. Most important of all, every result here comes from a PyTorch simulation rather than a fabricated photonic chip; confirming that the method survives on real hardware is the decisive next step.

5.4. Implications and real-world translation

The stakeholders affected by the training-energy problem do not all gain equally from this result. Operators of large-scale infrastructure would benefit from the favorable energy scaling of optical matrix multiplication; manufacturers of photonic accelerators, from a forward-only rule that removes the gradient-mirroring hardware required by on-chip backpropagation. The method trains to a useful level short of backpropagation, so its near-term value is not to replace backpropagation at scale but to enable in-situ learning where backpropagation is physically unavailable. The main risk is unchanged: every result here comes from simulation, so the reported accuracies are an optimistic bound on what hardware would deliver.

5.5. Conclusion

This thesis set out to determine whether a Transformer can be trained without backpropagation, using only the forward-pass operations available to a photonic substrate, and to characterize what such training costs. The answer is that it can. Trained layer by layer with a local Forward-Forward prototype-based objective, directional-derivative updates, and a softmax-free Spherical attention adapted from the Kramers-Kronig kernel, the network reaches a useful operating point on vision and algorithmic transduction tasks, with no backward pass, no automatic differentiation, and no softmax. The four attention projections are trained one at a time, in a round-robin schedule rather than jointly.

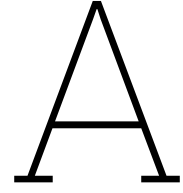
The contribution is therefore a characterization as much as a method. Every operation the trained model requires reduces to a forward pass and a measured scalar loss, with no transpose channel and no differentiable model of the device. What this thesis cannot supply is the measurement on real hardware: compiling a trained model onto a fabricated photonic chip and retraining it in-situ from measured losses is the experiment this work points to, and the one that would show whether the accuracy traded for a backpropagation-free, photonic-compatible Transformer is a price worth paying.

References

- [1] A. Momeni et al., “Training of physical neural networks,” *Nature*, vol. 645, no. 8079, pp. 53–61, 2025. DOI: [10.1038/s41586-025-09384-2](https://doi.org/10.1038/s41586-025-09384-2).
- [2] F. Ashtiani, A. J. Geers, and F. Aflatouni, “An on-chip photonic deep neural network for image classification,” *Nature*, vol. 606, no. 7914, pp. 501–506, 2022. DOI: [10.1038/s41586-022-04714-0](https://doi.org/10.1038/s41586-022-04714-0). [Online]. Available: <https://www.nature.com/articles/s41586-022-04714-0>.
- [3] Y. Tian et al., “Photonic transformer chip: Interference is all you need,” *Photonix*, vol. 6, no. 45, 2025. DOI: [10.1186/s43074-025-00182-7](https://doi.org/10.1186/s43074-025-00182-7).
- [4] M. Morsali et al., *Opto-vit: Architecting a near-sensor region of interest-aware vision transformer accelerator with silicon photonics*, 2025. arXiv: [2507.07044](https://arxiv.org/abs/2507.07044) [cs.AR]. [Online]. Available: <https://arxiv.org/abs/2507.07044>.
- [5] F. Ashtiani, M. H. Idjadi, and K. Kim, *Integrated photonic deep neural network with end-to-end on-chip backpropagation training*, 2025. arXiv: [2506.14575](https://arxiv.org/abs/2506.14575) [physics.optics].
- [6] S. Bandyopadhyay et al., “Single-chip photonic deep neural network with forward-only training,” *Nature Photonics*, vol. 18, no. 12, pp. 1335–1343, Dec. 2024, ISSN: 1749-4893. DOI: [10.1038/s41566-024-01567-z](https://doi.org/10.1038/s41566-024-01567-z). [Online]. Available: <https://doi.org/10.1038/s41566-024-01567-z>.
- [7] G. Hinton, *The forward-forward algorithm: Some preliminary investigations*, 2022. arXiv: [2212.13345](https://arxiv.org/abs/2212.13345) [cs.LG].
- [8] Y. Guo, F. Braun, B. Ketelaar, S. Tan, R. Norte, and S. Kumar, *Local learning for stable backpropagation-free neural network training towards physical learning*, 2026. arXiv: [2603.24790](https://arxiv.org/abs/2603.24790) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2603.24790>.
- [9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- [10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [11] A. Paszke et al., *PyTorch: An imperative style, high-performance deep learning library*, 2019. arXiv: [1912.01703](https://arxiv.org/abs/1912.01703) [cs.LG].
- [12] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: [1409.0473](https://arxiv.org/abs/1409.0473) [cs.CL].
- [13] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1412–1421. DOI: [10.18653/v1/D15-1166](https://doi.org/10.18653/v1/D15-1166).
- [14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [15] J. Chung, C. Gulcehre, and K. Cho, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [16] A. Vaswani et al., *Attention is all you need*, 2017. arXiv: [1706.03762](https://arxiv.org/abs/1706.03762) [cs.CL].
- [17] A. Dosovitskiy et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: [2010.11929](https://arxiv.org/abs/2010.11929) [cs.CV].
- [18] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, *Linformer: Self-attention with linear complexity*, 2020. arXiv: [2006.04768](https://arxiv.org/abs/2006.04768) [cs.LG].
- [19] T. Chen, B. Xu, C. Zhang, and C. Guestrin, *Training deep nets with sublinear memory cost*, 2016. arXiv: [1604.06174](https://arxiv.org/abs/1604.06174) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1604.06174>.

- [20] R. Pascanu, T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks*, 2013. arXiv: [1211.5063](https://arxiv.org/abs/1211.5063) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1211.5063>.
- [21] R. Ye, C. Ye, C. Huang, M. Tang, and Y. Liu, “Beyond-backpropagation training: Methods, applications, and perspectives,” *Preprint*, Jan. 2026.
- [22] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, “Backpropagation and the brain,” *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020. doi: [10.1038/s41583-020-0277-3](https://doi.org/10.1038/s41583-020-0277-3).
- [23] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, *Random feedback weights support learning in deep neural networks*, 2014. arXiv: [1411.0247](https://arxiv.org/abs/1411.0247) [q-bio.NC].
- [24] R. Rao and D. Ballard, “Predictive coding in the visual cortex,” *Nature Neuroscience*, vol. 2, pp. 79–87, 1999. doi: [10.1038/4580](https://doi.org/10.1038/4580).
- [25] B. Scellier and Y. Bengio, *Equilibrium propagation: Bridging the gap between energy-based models and backpropagation*, 2017. arXiv: [1602.05179](https://arxiv.org/abs/1602.05179) [cs.LG].
- [26] K. Mondal and A. Kumar, *Attention via synaptic plasticity is all you need: A biologically inspired spiking neuromorphic transformer*, 2025. arXiv: [2511.14691](https://arxiv.org/abs/2511.14691) [cs.NE].
- [27] S. Chaudhary, *Enabling robust in-context memory and rapid task adaptation in transformers with Hebbian and gradient-based plasticity*, 2025. arXiv: [2510.21908](https://arxiv.org/abs/2510.21908) [cs.NE].
- [28] Z. Zhou et al., *Spikformer: When spiking neural network meets transformer*, 2022. arXiv: [2209.15425](https://arxiv.org/abs/2209.15425) [cs.NE].
- [29] W. Hua, K. Zhao, L. Leng, R. Cheng, Y. Ma, and Q. Guo, *Hebbian learning with global direction*, 2026. arXiv: [2601.21367](https://arxiv.org/abs/2601.21367) [cs.AI]. [Online]. Available: <https://arxiv.org/abs/2601.21367>.
- [30] A. Nøkland, *Direct feedback alignment provides learning in deep neural networks*, 2016. arXiv: [1609.01596](https://arxiv.org/abs/1609.01596) [stat.ML]. [Online]. Available: <https://arxiv.org/abs/1609.01596>.
- [31] D. Franceschi, *Backpropagation free transformers*, Preprint. Under review, Work performed while at Columbia University, Saarbrücken, Germany. [Online]. Available: <https://dinkofranceschi.com/docs/bft.pdf>.
- [32] A. G. Baydin, B. A. Pearlmutter, D. Syme, F. Wood, and P. Torr, *Gradients without backpropagation*, 2022. arXiv: [2202.08587](https://arxiv.org/abs/2202.08587) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2202.08587>.
- [33] S. Malladi et al., *Fine-tuning language models with just forward passes*, 2024. arXiv: [2305.17333](https://arxiv.org/abs/2305.17333) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2305.17333>.
- [34] Y. LeCun, “Convolutional networks for images, speech, and time-series,” *The Handbook of Brain Theory and Neural Networks*, 1998.
- [35] H. Kim and S. Yoo, “Replacing backpropagation with the forward-forward (ff) algorithm in transformer models: A theoretical and empirical study on scalable and efficient gradient-free training,” *Tehnički glasnik*, vol. 19, pp. 452–460, Jun. 2025. doi: [10.31803/tg-20250225100638](https://doi.org/10.31803/tg-20250225100638).
- [36] Y. Wu et al., *Distance-forward learning: Enhancing the forward-forward algorithm towards high-performance on-chip learning*, 2024. arXiv: [2408.14925](https://arxiv.org/abs/2408.14925) [cs.NE]. [Online]. Available: <https://arxiv.org/abs/2408.14925>.
- [37] A. Krutsylo, *Scalable forward-forward algorithm*, 2025. arXiv: [2501.03176](https://arxiv.org/abs/2501.03176) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2501.03176>.
- [38] A. Papachristodoulou, C. Kyrkou, S. Timotheou, and T. Theodoridis, “Convolutional channel-wise competitive learning for the forward-forward algorithm,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, pp. 14 536–14 544, Mar. 2024, ISSN: 2159-5399. doi: [10.1609/aaai.v38i13.29369](https://doi.org/10.1609/aaai.v38i13.29369). [Online]. Available: <http://dx.doi.org/10.1609/aaai.v38i13.29369>.
- [39] X. Chen, D. Liu, J. Laydevant, and J. Grollier, *Self-contrastive forward-forward algorithm*, 2025. arXiv: [2409.11593](https://arxiv.org/abs/2409.11593) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2409.11593>.
- [40] J. Gong, B. Li, and W. Abdulla, *Mono-forward: Backpropagation-free algorithm for efficient neural network training harnessing local errors*, 2025. arXiv: [2501.09238](https://arxiv.org/abs/2501.09238) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2501.09238>.

- [41] G. Zhao, T. Wang, Y. Li, Y. Jin, C. Lang, and H. Ling, *The cascaded forward algorithm for neural network training*, 2023. arXiv: [2303.09728](https://arxiv.org/abs/2303.09728) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2303.09728>.
- [42] H. Aghagolzadeh and M. Ezoji, "Contrastive forward-forward: A training algorithm of vision transformer," *Neural Networks*, vol. 192, p. 107 867, Dec. 2025, issn: 0893-6080. DOI: [10.1016/j.neunet.2025.107867](https://doi.org/10.1016/j.neunet.2025.107867). [Online]. Available: <http://dx.doi.org/10.1016/j.neunet.2025.107867>.
- [43] P. Mettes, E. van der Pol, and C. G. M. Snoek, *Hyperspherical prototype networks*, 2019. arXiv: [1901.10514](https://arxiv.org/abs/1901.10514) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1901.10514>.
- [44] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: [1711.05101](https://arxiv.org/abs/1711.05101) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1711.05101>.
- [45] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2015. arXiv: [1412.6980](https://arxiv.org/abs/1412.6980) [cs.LG].



Model Configurations

All experiments were conducted using the hyperparameters defined in the project configuration. Table A.1 details the architectural dimensions for each task, while Table A.2 summarizes the training settings. Because AD operates with much cleaner gradients, it doesn't need higher learning rate to converge. Table A.3 shows the difference between AD and DD learning rates. Task summary is shown in Table A.4. All these configurations can be found on GitHub.

Task	Max Layers (L)	d_{model}	Heads (H)	d_{ff}
Copy	8	64	2	128
Reverse	8	64	2	256
Sort	8	128	4	512
Recall	8	128	4	512
Math	8	256	8	1024
MNIST	8	192	6	768
CIFAR-10	8	384	6	1536
CIFAR-100	8	384	6	1536

Table A.1: Model architecture parameters per task. All experiments sweep depths $L \in \{1, \dots, 8\}$ using 8 random seeds (42-49).

Task Group	Batch Size	Epochs	Betas	Weight Decay
Sequence (Copy-Math)	32-64	100	[0.9, 0.98]	0.01
Vision (MNIST)	256	100	[0.9, 0.999]	0.05
Vision (CIFAR-10/100)	128	100	[0.9, 0.999]	0.05

Table A.2: Training hyperparameters and optimizer settings. All runs use 8 seeds (42-49); reported metrics are means with standard deviations over seeds.

Method	Learning Rate
BP + AD / BP + DD	1×10^{-3}
FF + AD / FF_joint + AD	1×10^{-3}
FF + DD / FF_joint + DD	5×10^{-2}

Table A.3: Learning rates for different methods.

Dataset	Train	Val	Classes / Vocab	Input size	Task
MNIST	60,000	10,000	10 classes	$28 \times 28 \times 1$	Classification
CIFAR-10	50,000	10,000	10 classes	$32 \times 32 \times 3$	Classification
Copy / Reverse / Argsort	60,000	6,000	14 tokens	5–10	Seq2Seq
Recall	60,000	6,000	40 tokens	$2n+2, n \leq 10$	Seq2Seq
Arithmetic	30,000	3,000	43 tokens	$\leq 2D+1, D=5$	Seq2Seq

Table A.4: Tasks summary. Sizes for the algorithmic and arithmetic datasets are configurable; the values used in the experiments are listed. The arithmetic vocabulary consists of 10 digits, 3 operators, and 4 special tokens.

B

Kramers-Kronig Attention

To confirm that Spherical attention is not adopted arbitrarily over the original Kramers-Kronig attention (KKA) of Tian et al. [3], the KKA kernel of their Eq. 3 was implemented digitally and trained with the four Forward-Forward variants of [section 3.1](#) on MNIST.

Digital KKA construction. Real-valued Q and K projections are interpreted as optical phases, scaled by a learnable per-head factor α (Xavier-initialized projections, $\alpha = 1$ at initialization). The amplitude-phase-coupled interference of Eq. 3 is reconstructed through its closed-form trigonometric expansion,

$$\text{score}(i, j) = \sum_d \cos^2(\alpha Q_{id}) \cos(\alpha K_{jd}) + \frac{1}{2} \sin(2\alpha Q_{id}) \sin(\alpha K_{jd}), \quad (\text{B.1})$$

which preserves the input-dependent nonlinearity that Tian et al. identify as the softmax substitute. The value step is a plain linear multiply-accumulate, $\text{out}(i, d) = \sum_j \text{score}(i, j) V_{jd}$ - a simplification of Tian’s design. The only systematic difference from the chip is the absence of the per-modulator initial phases $\phi_{\text{ini}}, \varphi_{\text{ini}}$, removed on-chip by calibration and here absorbed into the learnable α .

Result. The six variants were run on MNIST across Vision Transformer depths $L \in \{1, \dots, 5\}$ (Figure B.1); BP-AD and BP-DD use the standard softmax baseline for reference. Three points stand out. (i) FF-AD and FF-Joint-AD exceed 95% at every depth, matching the softmax baseline: the digital KKA kernel is itself trainable under a local goodness objective when the gradient is exact, which also validates the implementation as faithful to Eq. 3. (ii) FF-DD and FF-Joint-DD plateau at 32-40% on the *same* kernel, with no gain from depth - so the failure is due neither to the kernel nor to FF locality (both handled by the AD variants) but to the directional-derivative estimator.

Mechanism. The main hypothesis why it fails for FF-DD builds from the interaction between central-difference estimation and the periodicity of cos. The estimator evaluates the loss at two points spaced $2\varepsilon\|\hat{\mathbf{v}}\|$ apart along a random direction ([section 3.3](#)). With $\varepsilon = 10^{-3}$ and $\hat{\mathbf{v}}$ normalized over the component’s $n \sim 10^5$ parameters, the effective phase step on the cosine surface exceeds its oscillation period for moderate α , so the finite difference aliases the high-frequency loss surface: the estimate has expectation near zero and no consistent direction. AD is immune, recovering the exact local gradient regardless of high-frequency content.

Consequence. The directional derivative is the only one of the two estimators that maps to photonic hardware - it requires no backpropagation or analytical differentiation - so making FF-DD work is the goal of this thesis, and the KKA kernel cannot support it. This motivates replacing the periodic cosine with a non-periodic, monotonically angle-dependent score. L_2 -normalized Spherical attention (??) is the minimal such construction, preserving the two properties it borrows from KKA: bounded scores and the removal of softmax.

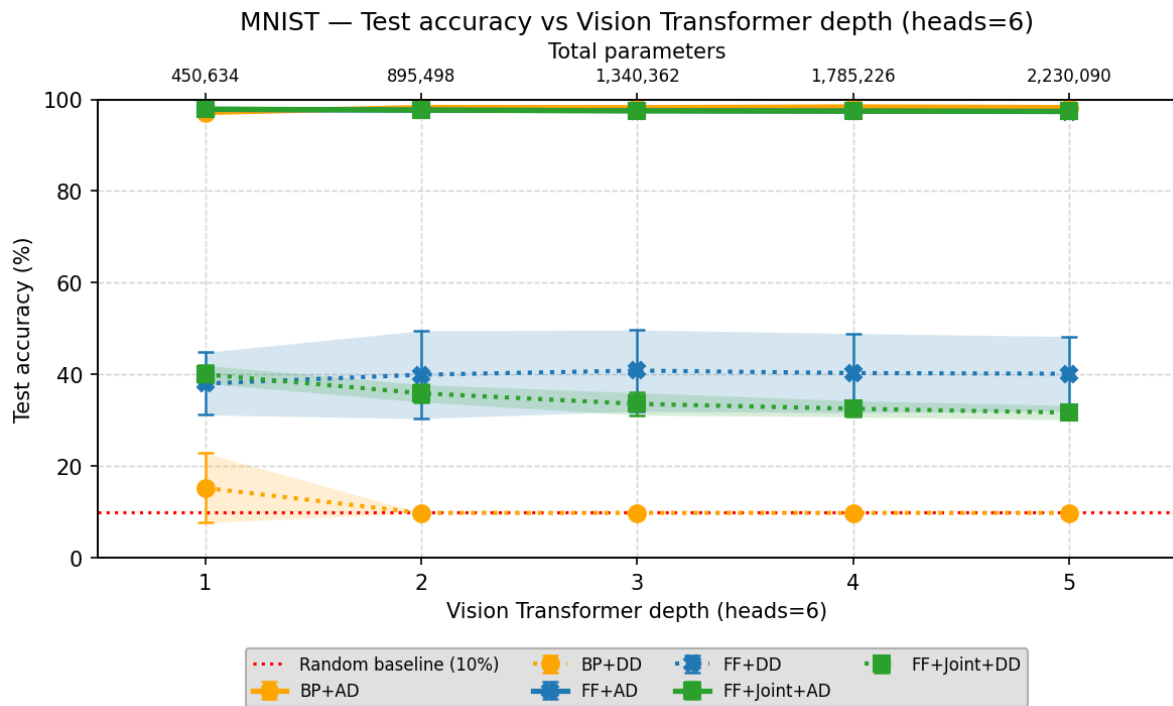
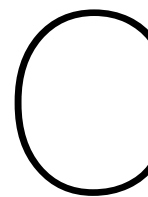


Figure B.1: MNIST test accuracy versus Vision Transformer depth. BP-AD and BP-DD use standard softmax MHA (reference baselines); the four FF variants use the digital KKA kernel of Eq. B.1. FF-AD and FF-Joint-AD reach > 95%, matching the softmax baseline; FF-DD and FF-Joint-DD plateau at 32-40% regardless of depth.



Declaration of Generative AI Usage

In accordance with the institutional guidelines on the use of Generative AI in BSc and MSc End Projects, this appendix explicitly discloses the use of generative AI tools in the research, development, and writing process of this thesis.

Tools Used

The following generative AI platforms were utilized during this project:

- **Claude** (Anthropic)
- **Gemini** (Google)

Scope and Purpose of Application

These tools were strictly employed to support independent academic work in the following capacities:

- **Conducting Research:** Utilized for brainstorming problem statements, idea testing to explore alternative perspectives, and supporting literature searches by identifying relevant papers and connections between research areas.
- **Code Assistance:** Employed to support programming tasks by generating boilerplate code and providing debugging hints to resolve errors.
- **Writing and Structuring:** Used to enhance the quality and clarity of academic writing, which included generating structural outlines, checking grammar, and refining sentences to better match academic style and fluency.

Responsibility and Verification

I hereby declare that I retain full creative and intellectual responsibility for all stages of the research, development, and writing process, as well as for the final outcomes of this thesis.

Recognizing the potential for AI tools to produce inaccurate, biased, or unverifiable content, I have independently verified the accuracy, validity, originality, and reliability of all content included in this work. All ideas, arguments, interpretations, and text remain my own, and any material derived from external sources has been properly referenced.

Furthermore, I confirm that all usage complies with relevant data protection regulations and that no sensitive, confidential, or proprietary information was entered into these generative AI tools.