

ViT³: Unlocking Test-Time Training in Vision

Dongchen Han¹ Yining Li¹ Tianyu Li¹ Zixuan Cao¹
Ziming Wang² Jun Song² Yu Cheng² Bo Zheng² Gao Huang^{1*}

¹ Tsinghua University ² Alibaba Group

Abstract

Test-Time Training (TTT) has recently emerged as a promising direction for efficient sequence modeling. TTT reformulates attention operation as an online learning problem, constructing a compact inner model from key-value pairs at test time. This reformulation opens a rich and flexible design space while achieving linear computational complexity. However, crafting a powerful visual TTT design remains challenging: fundamental choices for the inner module and inner training lack comprehensive understanding and practical guidelines. To bridge this critical gap, in this paper, we present a systematic empirical study of TTT designs for visual sequence modeling. From a series of experiments and analyses, we distill six practical insights that establish design principles for effective visual TTT and illuminate paths for future improvement. These findings culminate in the Vision Test-Time Training (ViT³) model, a pure TTT architecture that achieves linear complexity and parallelizable computation. We evaluate ViT³ across diverse visual tasks, including image classification, image generation, object detection, and semantic segmentation. Results show that ViT³ consistently matches or outperforms advanced linear-complexity models (e.g., Mamba and linear attention variants) and effectively narrows the gap to highly optimized vision Transformers. We hope this study and the ViT³ baseline can facilitate future work on visual TTT models. Code: github.com/LeapLabTHU/ViTTT.

1. Introduction

Vision Transformers (ViT) [12, 55] have become a cornerstone of modern computer vision, driving state-of-the-art results in image classification [12, 34, 55], generation [41], object detection [4, 81], and segmentation [63, 66]. However, a fundamental limitation lies in the quadratic complexity $\mathcal{O}(N^2)$ of Softmax attention [57] with respect to sequence length. This scaling bottleneck makes processing long visual sequences prohibitively expensive.

*Corresponding author.

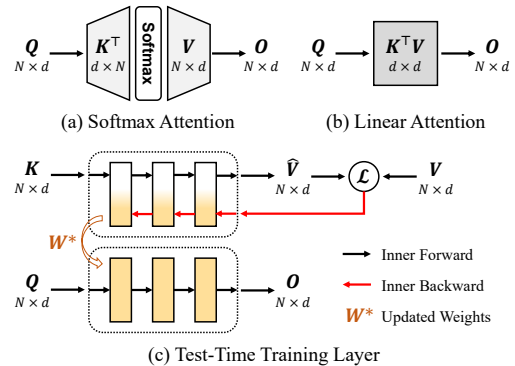


Figure 1. Illustration of Softmax attention [57], linear attention [30], and Test-Time Training (TTT) module [54]. (a) Softmax attention can be viewed as building a two-layer MLP that directly uses the uncompressed keys K and values V , where the hidden width equals the sequence length N and the nonlinearity is Softmax. While effective, this N -width MLP leads to $\mathcal{O}(N^2)$ costs when applied to the queries $Q \in \mathbb{R}^{N \times d}$. (b) Linear attention compresses $K, V \in \mathbb{R}^{N \times d}$ into a $d \times d$ linear layer weights using matrix multiplication $K^T V$, yielding $\mathcal{O}(N)$ complexity. However, the $d \times d$ linear state is limited and the naive $K^T V$ compression can discard important information, often resulting in inferior performance. (c) TTT generalizes the inner model to an arbitrary module $\mathcal{F}_W(\cdot): \mathbb{R}^d \rightarrow \mathbb{R}^d$ and compresses $K, V \in \mathbb{R}^{N \times d}$ into the module weights W through a few self-supervised online training steps. Output is calculated with the updated weights W^* . When $\mathcal{F}_W(\cdot)$ is implemented as a linear-complexity module, e.g., a two-layer MLP with hidden dimension $4d$, TTT can deliver powerful $\mathcal{O}(N)$ sequence modeling. Please refer to Sec. 3 for details.

To break this quadratic barrier, the community has explored linear-complexity $\mathcal{O}(N)$ alternatives. A representative approach is linear attention [30], which replaces Softmax with a linear kernel. This enables a reordering of computation from $(QK^T)V$ to $Q(K^TV)$, achieving the desired $\mathcal{O}(N)$ efficiency. Nonetheless, previous works [6, 18, 44] show that linear attention suffers from limited expressive power, making it impractical for many applications.

Test-Time Training (TTT) model [54], a recent design, offers a new promising path forward. Specifically, TTT reformulates the entire attention operation as an online learning process: in each context a compact inner model is trained or constructed from the key-value pairs and then

applied to every query. As shown in Fig. 1, from this viewpoint, Softmax attention can be interpreted as constructing a two-layer MLP with hidden dimension N and Softmax activation, whereas linear attention corresponds to building a $d \times d$ linear layer via matrix multiplication $K^\top V$. In contrast to these *fixed* designs, TTT allows the inner model to be an *arbitrary* module $\mathcal{F}_W(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$. It treats the key-value pairs as a “mini-dataset” and performs a few steps of self-supervised online training to learn (update) the parameters W of this inner model at test time. This approach opens a far richer, more powerful linear-complexity design space, yielding impressive $\mathcal{O}(N)$ sequence modeling results in various scenarios [5, 8, 76].

However, the very flexibility of TTT introduces a critical bottleneck: a large and under-explored design space. Key choices regarding the inner training process and the inner model architecture lack systematic understanding in existing work [2, 54, 76], which prevents the design of powerful visual TTT models. This gap motivates our central research question: *What design principles can be established for building efficient yet expressive Test-Time Training module, and how to enable future improvement?*

To answer this question, we systematically explore the new design space of Test-Time Training models in vision. We focus on two fundamental aspects: inner training settings (loss function, learning rate, batch size, number of epochs), and inner model design (architecture and model size). Through a series of experiments, we summarize our empirical observations into six practical insights: 1) loss functions \mathcal{L} for which $\frac{\partial^2 \mathcal{L}}{\partial V \partial V}$ vanishes are not suitable for TTT; 2) a single epoch of full-batch (batch gradient) inner training works well for vision; 3) a relatively large inner learning rate (we use 1.0) is effective; 4) increasing inner model capacity consistently improves performance; 5) in current TTT settings, deep inner models suffer from optimization difficulties; 6) convolutional architectures are particularly appropriate as inner models for visual tasks.

Guided by these findings, we propose **Vision Test-Time Training (ViTTT, ViT³)** model, a simple, parallelizable and pure TTT architecture tailored for visual sequence modeling. We evaluate ViT³ on a broad suite of tasks, including image classification, generation, object detection, and semantic segmentation. ViT³ consistently matches or outperforms advanced $\mathcal{O}(N)$ models, such as Mamba and linear attention variants, while effectively narrowing the performance gap to highly optimized $\mathcal{O}(N^2)$ vision Transformers. These results confirm that the proposed ViT³ offers impressive efficiency and capacity, demonstrating the potential of test-time training approaches on visual tasks.

Our main contributions and takeaways are as follows:

- We present a systematic empirical study of Test-Time Training designs for vision, covering inner training regimes (loss function, learning rate, batch size, epochs)

and inner model design (architecture and model size).

- We offer six practical insights for building effective yet efficient TTT module, providing detailed analyses of the TTT design space. Our analyses also reveal several valuable future research directions for TTT models.
- We build the Vision Test-Time Training (ViT³) model, a simple TTT architecture that implements these insights. With $\mathcal{O}(N)$ complexity, ViT³ achieves competitive results across image classification, image generation, object detection, and semantic segmentation, serving as a strong baseline for future research on visual TTT models.

2. Related Work

Attention [57] and vision Transformers [12, 22, 55, 56] have driven substantial progress in recent years. However, the quadratic computation complexity $\mathcal{O}(N^2)$ of self-attention leads to unmanageable cost when processing long visual sequences. To mitigate this, many works reduce the number of tokens involved in attention by introducing locality [11, 23, 34] or sparsity [59, 61, 79]. An alternative family of approaches, linear attention methods [17, 30, 69, 70], redesign the attention paradigm to achieve global modeling with linear complexity $\mathcal{O}(N)$. With growing demand to process longer visual sequences, these linear-complexity models [18, 33, 42, 67] have attracted increasing interest. In this paper, we complement these lines of work by exploring TTT-based linear-complexity design.

Test-Time Training (TTT) model [54] is a recently emerged approach for efficient sequence modeling, which reformulates attention operation as an online learning problem. This paradigm increases the flexibility of sequence models and has enabled scalable, linear-complexity designs across multiple domains [1, 8, 76]. For instance, LaCT [76] attains effective linear-time language modeling by applying large chunk test-time training. TTT layers allow pretrained diffusion Transformers that could only generate 3-second clips to produce coherent one-minute videos [8]. TTT3R [5] views 3D reconstruction as a test-time training process, achieving a $2\times$ improvement in global pose estimation. TTT is also closely related to fast weight programmers [48], meta-learning [16, 38], Test-Time Scaling [40, 52] and recurrent depth [3, 47]. Despite these promising results, the superiority of TTT methods remain locked by the under-explored design space (i.e., inner training and inner module settings). In this paper, we systematically evaluate TTT in vision, summarizing several insights for effective TTT models and potential future works.

3. Preliminaries

3.1. Attention Mechanism

Softmax attention [57], also referred to as dot-product attention, is the predominant attention mechanism used in

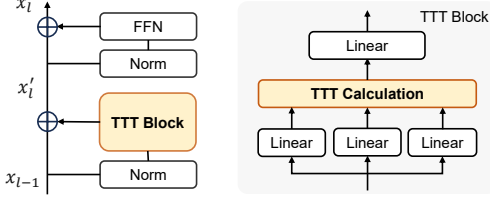


Figure 2. Illustration of the TTT model building block. TTT shares the same macro architecture as Transformer.

modern Transformer architectures. Let $x \in \mathbb{R}^{N \times C}$ denote an input sequence of N tokens with dimension C . A single head of Softmax attention can be written as:

$$O_i = \sum_{j=1}^N \frac{\exp(Q_i K_j^\top)}{\sum_{j=1}^N \exp(Q_i K_j^\top)} V_j, \quad (1)$$

where $Q = xW_Q$, $K = xW_K$, $V = xW_V$, $Q, K, V \in \mathbb{R}^{N \times d}$ represent queries, keys and values, and $W_{Q/K/V} \in \mathbb{R}^{C \times d}$ are projection matrices. We omit the factor $\frac{1}{\sqrt{d}}$, as it can be achieved equivalently by scaling Q, K . To facilitate analysis, we reformulate Eq. (1) as:

$$O = \sigma(QK^\top)V \triangleq \sigma(QW_1)W_2 = \text{MLP}(Q), \quad (2)$$

where $\sigma(\cdot)$ is the row-wise Softmax operation. This expression highlights that Softmax attention can be viewed as a two-layer MLP of width N acting on Q , with parameters $W_1 = K^\top$, $W_2 = V$, and Softmax serving as the activation. This N -width MLP leads to $\mathcal{O}(N^2)$ costs when evaluating $O = \text{MLP}(Q)$, resulting in excessive computational costs in long-sequence modeling scenarios.

Linear attention [30] is an alternative attention paradigm that addresses the quadratic cost by reducing complexity to $\mathcal{O}(N)$. Specifically, linear attention replaces the non-linear Softmax with a linear kernel-based normalization:

$$O_i = \sum_{j=1}^N \frac{Q_i K_j^\top}{\sum_{j=1}^N Q_i K_j^\top} V_j = \frac{Q_i \left(\sum_{j=1}^N K_j^\top V_j \right)}{Q_i \left(\sum_{j=1}^N K_j^\top \right)}, \quad (3)$$

where $Q = \phi(xW_Q)$, $K = \phi(xW_K)$, $V = xW_V$, and $\phi(\cdot)$ is the kernel function. This separate non-linearity on Q and K enables the rearrangement of the computation order from $(QK^\top)V$ to $Q(K^\top V)$ based on the associative property of matrix multiplication, thus reducing complexity to $\mathcal{O}(N)$. If we momentarily ignore the scalar normalization factor $Q_i \left(\sum_{j=1}^N K_j^\top \right) \in \mathbb{R}$, Eq. (3) simplifies to:

$$O = Q(K^\top V) \triangleq QW = \text{FC}(Q). \quad (4)$$

Consequently, linear attention corresponds to compressing K and V into the $d \times d$ linear (FC) layer weights $W = K^\top V$. While this approach enables $\mathcal{O}(N)$ efficiency, the limited linear state and naive $K^\top V$ compression tend to be inferior in practice [18, 19, 44].

3.2. Test-Time Training Model

Test-Time Training (TTT) [54] reframes sequence modeling as an online learning problem and thereby generalizes linear attention into a unified, more powerful compression-and-adaptation framework. The key idea is to compress contextual key-value information (K, V) into the weights of a compact neural network $\mathcal{F}_W: \mathbb{R}^d \rightarrow \mathbb{R}^d$, and then extract output features using queries. The compression is achieved by a fast self-supervised online learning on the key-value ‘‘dataset’’ $\mathcal{D} = \{(K_i, V_i) | i = 1, \dots, N\}$:

$$\hat{V}_B = \mathcal{F}_W(K_B), \quad W \leftarrow W - \eta \cdot \frac{\partial \mathcal{L}(\hat{V}_B, V_B)}{\partial W}, \quad (5)$$

where $K_B, V_B \in \mathbb{R}^{B \times d}$ denote a mini-batch of size B sampled from \mathcal{D} . \mathcal{L} is a self-supervised reconstruction objective that encourages $\mathcal{F}_W(K_i)$ to predict the associated value V_i . The initial W_0 are implemented as learnable parameters of the outer network. After a few inner updates, the adapted parameters W^* are used to produce the output $O = \mathcal{F}_{W^*}(Q)$. The approach is referred to as a Test-Time Training (TTT) layer because the inner module \mathcal{F}_{W_0} is briefly trained (adapted) per input sequence at test time. Importantly, the inner adaptation procedure of Eq. (5) is differentiable and is optimized jointly with the outer network during training (i.e., the inner updates are unrolled and learned end-to-end), so models with TTT layers remain fully end-to-end trainable. We refer to \mathcal{F}_W and Eq. (5) as *inner module* and *inner training (inner loop)*, and refer to the entire network and its training on real data as *outer model* and *outer training (outer loop)*.

The per-sequence computation and memory cost of a TTT layer scale with the cost of the inner module. Therefore, when the inner module is implemented using a linear-complexity architecture, e.g., a two-layer MLP with ratio 4.0, the TTT layer inherits $\mathcal{O}(N)$ time and memory complexity while benefiting from the richer, learning-based compression and non-linear expressive states.

4. Exploring Test-Time Training Designs

In this section, we conduct a systematic empirical study of Test-Time Training designs in vision. We employ the classical vision Transformer architecture DeiT-S [55] and substitute the attention blocks with TTT layers to create our baseline. Each inner training (loss function, learning rate, batch size and epochs) and model (architecture and size) designs are introduced *separately* to the baseline model to assess their impact. Experiments are conducted on ImageNet-1K with the standard 300 epoch training settings.

4.1. Inner Training Configuration

We first implement the inner model as a two-layer MLP (SiLU activation, ratio 1.0) to study the inner training.

Loss Function	#Params	FLOPs	FPS	Top-1
Dot Product Loss	23.5M	4.58G	1315	78.9
MSE (L2) Loss	23.5M	4.63G	1296	79.2
RMSE Loss	23.5M	4.63G	1269	78.8
MAE (L1) Loss	23.5M	4.63G	1292	76.5
Smooth L1 loss	23.5M	4.63G	1292	78.1

Table 1. Results of different inner training loss functions. Please refer to the Appendix for detailed formulas of each loss function.

Insight 1: inner loss functions \mathcal{L} for which the mixed second derivative $\frac{\partial^2 \mathcal{L}}{\partial \hat{V} \partial V}$ vanishes are not suitable for TTT. As discussed in Sec. 3, the inner training process of TTT layer is optimized jointly with the outer network. Concretely, the value projection matrix $W_V \in \mathbb{R}^{C \times d}$ obtains outer-loop gradients through backpropagation applied to the inner update term $G = \frac{\partial \mathcal{L}(\hat{V}_B, V_B)}{\partial W}$ in Eq. (5). In other words, we differentiate through the inner training steps and taking gradients of gradients, which is a commonly studied operation in meta-learning [16]. Note that

$$\frac{\partial G}{\partial W_V} = \frac{\partial^2 \mathcal{L}(\hat{V}_B, V_B)}{\partial W \partial W_V} = \frac{\partial \hat{V}_B}{\partial W} \cdot \frac{\partial^2 \mathcal{L}(\hat{V}_B, V_B)}{\partial \hat{V}_B \partial V_B} \cdot \frac{\partial V_B}{\partial W_V}. \quad (6)$$

If the mixed derivative $\frac{\partial^2 \mathcal{L}}{\partial \hat{V} \partial V}$ is (near) zero, the outer-loop gradient signal to W_V vanishes after backpropagation through the inner step, which undermines learning and degrades performance. Our experiments in Tab. 1 support this analysis: MAE (L1) loss — whose derivative is a sign function and whose mixed second derivative is zero almost everywhere — yields the worst accuracy. Smooth L1 loss also performs poorly because its mixed derivative vanishes in certain regions, whereas the three other losses which do not exhibit this issue achieve approximately 79.0% accuracy.

Remark 1. Follow prior TTT work [54, 76], we focus on loss functions that encourages $\mathcal{F}_W(K_i)$ to predict its corresponding value V_i . This training target is implicitly satisfied by the widely used Softmax attention [57]. As analyzed in Sec. 3, Softmax attention can be viewed as constructing an inner two-layer MLP $\mathcal{F}(\cdot)$ using K and V . For an input K_i , the output of this inner MLP is $\mathcal{F}(K_i) = \sigma(K_i K^\top) V$, where $\sigma(\cdot)$ denotes the row-wise Softmax. Under common conditions (e.g., sufficiently distinct key vectors and appropriate temperature scaling), the self-similarity $K_i K_i^\top$ is typically the largest value in $K_i K^\top$, so $\sigma(K_i K^\top)$ concentrates most score on the i -th position and becomes approximately one-hot. Therefore, $\mathcal{F}(K_i) = \sigma(K_i K^\top) V \approx V_i$, which is exactly the training target of TTT.

Insight 2: a single epoch of full-batch (batch gradient) inner training works well for vision. We first study the effect of the inner-training batch size, presenting the results in Tab. 2. Here $B = N$ denotes full-batch gradient descent using all N key-value pairs as a single inner batch, whereas $B = \frac{N}{2}, \frac{N}{3}, \frac{N}{4}$ refer to mini-batch gradient descent, partitioning the dataset into 2, 3, 4 sequential mini-batches

Epoch	Batch Size	#Params	FLOPs	FPS	Top-1
1	N	23.5M	4.58G	1315	78.9
1	$N/2$	23.5M	4.58G	1201	78.6
1	$N/3$	23.5M	4.58G	1131	78.3
1	$N/4$	23.5M	4.58G	1101	78.1
2	N	23.5M	4.81G	971	79.1
3	N	23.5M	5.04G	787	79.2
4	N	23.5M	5.27G	659	57.0*

Table 2. Results of various batch sizes and epochs. * refers to the best accuracy before divergence during training.

Learning Rate	0.1	0.2	0.5	1.0	2.0	5.0	10.0	Dynamic
Top-1	77.5	78.1	78.7	78.9	78.9	76.7*	76.9*	78.7

Table 3. Results of different inner learning rates. * refers to the best accuracy before divergence during training.

and conducting 2, 3, 4 inner updates per epoch. In contrast to prior TTT results on language modeling that reported gains from smaller mini-batches [54], we find that $B = N$ leads to the best outcome in our vision experiments. We attribute this discrepancy to the difference between causal and non-causal data modalities. Specifically, sequential mini-batch gradient descent imposes a causal bias: (1) updates from earlier mini-batches change the inner-module weights, thus influencing the gradients of later batches; (2) updates from later mini-batches can overwrite those produced earlier. This causal dependency is ideally suited for causal data like language, but could be suboptimal for vision [20, 71].

Multiple epochs of full-batch inner-training improve accuracy. However, they notably reduce model throughput and can introduce training instability.

Remark 2. Existing linear sequence-modeling approaches for vision (e.g., linear attention [18, 20] and Mamba [33, 65]) can be categorized into two broad classes: parallel and sequential. Parallel designs [18, 20] model the entire input sequence in a non-causal (global) manner, which is similar to the single-epoch, full-batch inner update in our TTT setup. In contrast, sequential models [33, 65] process inputs along carefully designed scan paths, capturing richer and more expressive spatial dependencies. Therefore, while naive sequential mini-batch exhibits subpar performance in our experiments, designing mini-batch inner-training algorithms tailored for vision remains a promising future work.

Insight 3: a relatively large inner learning rate of 1.0 is effective. Tab. 3 reports the results of different inner learning rates, ranging from 0.1 to 10. Small inner learning rates produce insufficient updates to the inner model weights W , while excessively large ones can result in training instability. We observe that $\eta = 1.0$ yields meaningful inner updates while preserving stability of the outer optimization. We additionally evaluate an input-dependent, token-wise dynamic rate $\eta_i = \eta \cdot \text{Sigmoid}(x_i W_\eta)$, $\eta = 1.0$ as proposed in prior work [54, 76]. In our vision experiments this scheme is generally less effective.

Remark 3. In some special cases, the inner learning rate

Inner Model	#Params	FLOPs	FPS	Top-1
MLP(x), $r1, l2$	23.5M	4.58G	1315	78.9
MLP(x), $r2, l2$	24.1M	4.92G	1119	79.2
MLP(x), $r3, l2$	24.7M	5.27G	938	79.5
MLP(x), $r4, l2$	25.2M	5.62G	836	79.6
FC(x)	23.2M	4.34G	1708	79.1
MLP(x), $r1, l2$	23.5M	4.58G	1315	78.9
MLP(x), $r1, l3$	23.8M	4.81G	1086	77.5
SiLU(FC(x))	23.2M	4.40G	1456	79.4
SwiGLU(x)	23.8M	4.75G	1103	79.0
FC(x) \odot SiLU(FC(x))	23.5M	4.58G	1194	79.7
Conv(x)	25.5M	5.27G	979	79.9
DWConv(x)	22.9M	4.25G	1366	80.1

Table 4. Results of different inner model designs. The r and l denote width ratio and layer-wise depth of a MLP. For example, $r3, l2$ refers to a 2-layer MLP with hidden dimension $3d$, where d is the input and output dimension of an inner model.

can be absorbed into the scaling of K and V . For example, when the inner model is a linear layer $\mathcal{F}_W(x) = xW$, $W \in \mathbb{R}^{d \times d}$ and the inner loss is mean squared error (MSE), the gradient-based update term can be written as:

$$\eta \cdot \frac{\partial \mathcal{L}(\hat{V}, V)}{\partial W} = \eta \cdot K^\top (KW - V) = \tilde{K}^\top (\tilde{K}W - \tilde{V}), \quad (7)$$

where $\tilde{K} = \sqrt{\eta}K$, $\tilde{V} = \sqrt{\eta}V$. This shows that scaling K and V is mathematically equivalent to changing η . Nevertheless, η remains a crucial practical hyperparameter because such rescaling could be hard to learn or incompatible with other model components (e.g., initialized parameter scales, or normalization layers). The impact of inner learn rate is fully demonstrated by our results in Tab. 3. Similarly, the famous $\frac{1}{\sqrt{d}}$ scaling in Softmax attention can be absorbed into Q and K , but is well known to be critical in practice [57].

4.2. Inner Model Design

To study the inner model designs, we hold the inner training configuration as follows: dot product loss, one epoch of full-batch gradient descent, and learning rate 1.0.

Insight 4: increasing inner model capacity consistently improves performance. To validate this, we instantiate the inner model $\mathcal{F}_W(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ as a two-layer MLP with SiLU activation and progressively enlarge its hidden dimension from d to $4d$. Tab. 4 reports that the accuracy increases consistently as inner model capacity grows. This demonstrates a key advantage of the TTT paradigm over previous linear attention methods [19, 30, 44]: rather than restricting the inner model to a linear $d \times d$ mapping, TTT allows for a more complex nonlinear module as the inner model, and consistently benefit from its capacity.

Remark 4. Increasing inner model size incurs more computation than enlarging an outer model. At test time, an outer module $\mathcal{F}_W(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ requires only a single forward pass $y = \mathcal{F}_W(x)$. In contrast, an inner module needs to perform: (i) a forward pass on keys $\hat{V} = \mathcal{F}_W(K)$, (ii) the

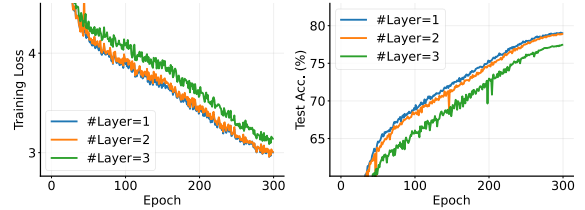


Figure 3. Results of TTT models with inner modules of 1, 2, 3 layers (FC, two-layer and three-layer MLP). Deeper inner models lead to higher training loss, and thus lower test accuracy.

backward pass of loss \mathcal{L} , and (iii) a forward pass on queries $O = \mathcal{F}_W(Q)$. A backward pass typically costs twice the FLOPs of a forward pass, because backpropagation computes gradients of both parameter and input; for example, for the forward $Y = XW$ of a linear layer, we compute $\frac{\partial \mathcal{L}}{\partial W} = X^\top \frac{\partial \mathcal{L}}{\partial Y}$ and $\frac{\partial \mathcal{L}}{\partial X} = \frac{\partial \mathcal{L}}{\partial Y} W^\top$ (to propagate to previous layers). Hence, one inner training epoch consumes approximately $1 + 2 + 1 = 4$ forward-equivalent FLOPs, i.e., about $4 \times$ the compute of an outer module with the same architecture. Therefore, while simply scaling inner model demonstrates promising results, we believe designing lightweight and expressive inner model is an important topic.

Insight 5: in current TTT settings, deep inner models suffer from optimization difficulties. Scaling inner model width leads to consistent gains, as suggested by Insight 4. A common alternative is to scale network depth [25, 56, 58]. We evaluate three inner models of different depths: a single $d \times d$ linear (FC) layer, a two-layer MLP, and a three-layer MLP. For both MLP variants, the hidden dimensions are set to d , matching the input and output dimensionality. Tab. 4 shows that the final test accuracy does not scale with inner model depth, but instead demonstrates a clear degradation. This finding is counterintuitive, because deeper inner modules contain more parameters and therefore greater capacity.

We hypothesize that increasing the depth of the inner module exacerbates optimization difficulties, leading to diminishing returns. To validate this hypothesis, Fig. 3 plots training loss and test accuracy for the three architectures. Models with deeper inner modules exhibit higher training loss, and thus lower test accuracy. In other words, while deeper inner modules are theoretically more expressive and easier to fit the training data, they practically deliver lower performance due to underfitting. This disparity between theory and practice suggests an optimization bottleneck. We further corroborate this claim using a constrained inner-model $\mathcal{F}_W(x) = \text{SiLU}(\text{FC}(x))$, which can be viewed as a two-layer MLP whose output linear layer is fixed to the identity. Replacing the full two-layer MLP with this constrained design increases accuracy from 78.9% to 79.4%. A similar effect is also observed with SwiGLU [49]: removing the output layer (i.e., a SwiGLU with identity output layer) raises the accuracy from 79.0% to 79.7%. These constrained special cases achieve much better results than

Method	Type	#Params	FLOPs	Top-1
ConvNeXt-T [35]	ConvNet	29M	4.5G	82.1
InternImage-T [60]	ConvNet	30M	5.0G	83.5
BiFormer-S [79]	Transformer	26M	4.5G	83.8
TransNeXt-T [50]	Transformer	28M	5.7G	84.0
RMT-S [15]	Transformer	27M	4.5G	84.1
FasterViT-1 [‡] [24]	Transformer	53M	5.3G	83.2
VMamba-T [33]	Mamba	31M	4.9G	82.5
LocalVMamba-T [29]	Mamba	26M	5.7G	82.7
SOFT-S++ [37]	Linear	27M	4.5G	82.6
VVT-S [53]	Linear	26M	5.6G	82.7
MILA-T [‡] [20]	Linear	25M	4.2G	83.5
H-ViT ³ -T	TTT	29M	4.9G	83.5
H-ViT ³ -T [‡]	TTT	29M	4.9G	84.0
InceptionNeXt-B [72]	ConvNet	87M	14.9G	84.0
InternImage-B [60]	ConvNet	97M	16.0G	84.9
CSwin-B [11]	Transformer	78M	15.0G	84.2
iFormer-L [51]	Transformer	87M	14.0G	84.8
TransNeXt-B [50]	Transformer	90M	18.4G	84.8
FasterViT-3 [‡] [24]	Transformer	160M	18.2G	84.9

Table 5. Comparison with hierarchical architectures on ImageNet-1K. We focus on representative ConvNet, Transformer, Mamba, and Linear attention methods. ‡ indicates the model is trained with MESA [13], a strategy that can alleviate overfitting at little cost.

Method	Type	#Params	FLOPs	Top-1
ConvNeXt-S [35]	ConvNet	50M	8.7G	83.1
InternImage-S [60]	ConvNet	50M	8.0G	84.2
BiFormer-B [79]	Transformer	57M	9.8G	84.3
TransNeXt-S [50]	Transformer	50M	10.3G	84.7
RMT-B [15]	Transformer	54M	9.7G	85.0
FasterViT-2 [‡] [24]	Transformer	76M	8.7G	84.2
VMamba-S [33]	Mamba	50M	8.7G	83.6
LocalVMamba-S [29]	Mamba	50M	11.4G	83.7
SOFT-M++ [37]	Linear	48M	8.7G	83.7
VVT-M [53]	Linear	48M	9.4G	83.8
MILA-S [‡] [20]	Linear	43M	7.3G	84.4
H-ViT ³ -S	TTT	54M	8.8G	84.4
H-ViT ³ -S [‡]	TTT	54M	8.8G	84.9
Mamba2D-B [31]	Mamba	94M	—	83.0
VMamba-B [33]	Mamba	89M	15.4G	83.9
SOFT-L++ [37]	Linear	85M	15.4G	84.1
MILA-B [‡] [20]	Linear	96M	16.2G	85.3
H-ViT ³ -B	TTT	94M	16.7G	84.9
H-ViT ³ -B [‡]	TTT	94M	16.7G	85.5

Table 5. Comparison with hierarchical architectures on ImageNet-1K. We focus on representative ConvNet, Transformer, Mamba, and Linear attention methods. ‡ indicates the model is trained with MESA [13], a strategy that can alleviate overfitting at little cost.

Inner Model	#Params	FLOPs	FPS	Top-1
SiLU(xW_1) W_2+x	23.5M	4.58G	1294	78.8
SiLU(xW_1)(W_2+I)	23.5M	4.58G	1294	79.1
SiLU(xW_1) W_2 , W_2 init as I	23.5M	4.58G	1315	79.0

Table 6. Results of applying residual connections and weight initialization strategies to the MLP inner model. Despite achieving some improvements, these methods still underperform the constrained design $\mathcal{F}_W(x) = \text{SiLU}(\text{FC}(x))$, i.e., $\text{SiLU}(xW_1)$.

original designs, validating the optimization problem. Together, these results fully support our hypothesis that deeper inner modules are harder to optimize and can therefore underperform their shallower counterparts in current settings.

Remark 5. The optimization difficulties of deep inner models can be attributed to two complementary aspects:

- Outer-loop problem. The inner model initialization W_0 of $\mathcal{F}_W(\cdot)$ (i.e., initial weights for Eq. (5)) is learned end-to-end as part of the full network parameters. For deep inner modules, this initialization becomes difficult to optimize during outer-loop (end-to-end) training.
- Inner-loop problem. Increasing inner depth causes exploding or vanishing inner-loop gradients $\frac{\partial \mathcal{L}}{\partial W}$, which hinder effective compression of the K, V context.

We empirically find that standard residual connections [25] and naive initialization schemes provide limited mitigation for these issues (see Tab. 6). Notably, theoretical works [9, 14, 39, 43, 45] suggest that deeper neural networks offer exponentially greater capability, which is a key factor behind the success of modern deep learning models. Therefore, addressing the optimization challenges and enabling deep inner models within the TTT framework represents a fundamental and promising research direction.

Insight 6: convolutional architectures are particularly appropriate as inner models for visual tasks. Convolutional operations have long been a cornerstone of vi-

Method	Type	#Params	FLOPs	Top-1
DeiT-T [55]	Transformer	6M	1.2G	72.2
Vim-T [80]	Mamba	7M	1.5G	76.1
Agent-DeiT-T [21]	Linear	6M	1.2G	74.9
ViT ³ -T	TTT	6M	1.2G	76.5
ConvNeXt-S (iso.) [35]	ConvNet	22M	4.3G	79.7
DeiT-S [55]	Transformer	22M	4.6G	79.8
Vim-S [80]	Mamba	26M	5.1G	80.3
Agent-DeiT-S [21]	Linear	23M	4.4G	80.5
ViT ³ -S	TTT	24M	4.8G	81.6
ConvNeXt-B (iso.) [35]	ConvNet	87M	16.9G	82.0
DeiT-B [55]	Transformer	87M	17.6G	81.8
ViT ³ -B	TTT	90M	18.0G	82.6

Table 7. Comparison with non-hierarchical designs on ImageNet.

sual models [25, 28, 46] before the widespread adoption of Transformers [12]. Owing to the flexibility of TTT framework, the inner model $\mathcal{F}_W(\cdot)$ can be implemented as a compact convolutional network instead of being restricted to modules based on linear layers (e.g., MLP or GLU) in prior work [8, 54, 76]. As a showcase, we evaluate two simple designs: a standard 3×3 convolution, and a lightweight 3×3 depthwise convolution [27]. Results in Table 4 show that both variants yield strong accuracy improvements. We argue that using a convolutional inner module provides a natural and elegant integration of global and local information. Recall that TTT compresses the global context K, V into the parameters of $\mathcal{F}_W(\cdot)$. When $\mathcal{F}_W(\cdot)$ is implemented as a convolution, global information is effectively encoded into the local convolution kernel weights. Consequently, computing the output $O = \mathcal{F}_W(Q)$ realizes both global (via the updated kernel weights) and local (via the convolution receptive field) interactions, thus delivering a natural integration of global and local features and performance gains.

Remark 6. As discussed in Sec. 3, TTT fits the inner model using a key-value dataset $\mathcal{D} = \{(K_i, V_i) | i = 1, \dots, N\}$. When the inner model is convolutional, we actually gen-

Method	Type	FLOPs	AP ^b	AP ₅₀ ^b	AP ₇₅ ^b	AP ^m	AP ₅₀ ^m	AP ₇₅ ^m
Mask R-CNN 1×Schedule								
InternImage-T [60]	C	270G	47.2	69.0	52.1	42.5	66.1	45.8
CSWin-T [11]	T	279G	46.7	68.6	51.3	42.2	65.6	45.4
FocalNet-T [68]	T	268G	46.1	68.2	50.6	41.5	65.1	44.5
Vmamba-T [33]	M	271G	47.3	69.3	52.0	42.7	66.4	45.9
SOFT-T++ [37]	L	-	43.8	66.0	47.5	40.1	63.0	43.0
MILA-T [20]	L	255G	46.8	69.5	51.5	42.1	66.4	45.0
H-ViT ³ -T		271G	47.3	69.8	52.3	42.8	66.8	46.2
InternImage-S [60]	C	340G	47.8	69.8	52.8	43.3	67.1	46.7
CSWin-S [11]	T	342G	47.9	70.1	52.6	43.2	67.1	46.2
TransNeXt-T [50]	T	356G	49.9	71.5	54.9	44.6	68.6	48.1
Vmamba-S [33]	M	357G	48.7	70.0	53.4	43.7	67.3	47.0
SOFT-S++ [37]	L	-	46.6	67.8	51.2	42.0	64.8	45.2
H-ViT ³ -S		349G	49.1	71.5	53.7	44.1	68.4	47.4
InternImage-B [60]	C	501G	48.8	70.9	54.0	44.0	67.8	47.4
CSWin-B [11]	T	526G	48.7	70.4	53.9	43.9	67.8	47.3
TransNeXt-S [50]	T	516G	51.1	72.6	56.2	45.5	69.8	49.1
VMamba-B [33]	M	485G	49.2	70.9	53.9	43.9	67.7	47.6
SOFT-B++ [37]	L	-	47.0	68.3	51.7	42.2	65.2	45.4
H-ViT ³ -B		510G	50.0	71.8	55.0	44.6	69.0	47.7

Method	Type	FLOPs	AP ^b	AP ₅₀ ^b	AP ₇₅ ^b	AP ^m	AP ₅₀ ^m	AP ₇₅ ^m
Mask R-CNN 3×Schedule								
InternImage-T [60]	C	270G	49.1	70.4	54.1	43.7	67.3	47.3
CSWin-T [11]	T	279G	49.0	70.7	53.7	43.6	67.9	46.6
DAT-T++ [62]	T	301G	50.5	71.9	55.7	45.1	69.2	48.7
Vmamba-T [33]	M	270G	48.9	70.6	53.6	43.7	67.7	46.8
LocalVMamba-T [29]	M	291G	48.7	70.1	53.0	43.4	67.0	46.4
PolaFormer-T [18]	L	268G	47.0	68.9	51.5	42.3	66.0	45.8
MILA-T [20]	L	255G	48.8	71.0	53.6	43.8	68.0	46.8
H-ViT ³ -T		271G	48.9	71.0	53.4	44.0	68.0	47.5
InternImage-S [60]	C	340G	49.7	71.1	54.5	44.5	68.5	47.8
QFormer _h -S [75]	T	-	49.5	71.2	54.2	44.2	68.3	47.6
CSWin-S [11]	T	342G	50.0	71.3	54.7	44.5	68.4	47.7
DAT-S++ [62]	T	430G	51.2	72.6	56.3	45.7	69.9	49.7
Vmamba-S [33]	M	384G	49.9	70.9	54.7	44.2	68.2	47.7
MILA-S [20]	L	319G	50.5	71.8	55.2	44.9	69.1	48.2
H-ViT ³ -S		349G	50.5	72.0	55.5	45.0	69.1	48.8
InternImage-B [60]	C	501G	50.3	71.4	55.3	44.8	68.7	48.0
SWin-B [34]	T	526G	48.6	70.0	53.4	43.3	67.1	46.7
CSWin-B [11]	T	526G	50.8	72.1	55.8	44.9	69.1	48.3
H-ViT ³ -B		510G	51.0	72.1	55.9	45.3	69.3	49.1

Table 8. Results on COCO dataset. C, T, M, L represent ConvNet, Transformer, Mamba, and Linear attention, respectively. The FLOPs are computed with an input resolution of 1280×800.

eralize this definition to $\mathcal{D} = \{(K_i^{\text{loc}}, V_i) | i = 1, \dots, N\}$, where K_i^{loc} represents the local neighborhood of K_i . For instance, with a 3×3 convolution inner model, TTT learns pairs $(K_i^{3 \times 3}, V_i)$, where $K_i^{3 \times 3}$ represents the $3 \times 3 = 9$ key tokens in the local window centered at K_i .

5. ViT³: A Test-Time Training Architecture

In Sec. 4, we present a controlled study of visual TTT, distilling six practical insights that inform effective designs and potential future works. Guided by these findings, in this section we introduce Vision Test-Time Training (ViT³) model, a pure TTT *baseline* with linear complexity for benchmarking TTT methods on diverse visual tasks.

Specifically, for inner training, we use a single epoch of full-batch gradient descent with learning rate 1.0, optimizing a dot-product loss. As shown in Sec. 4.1, this simple configuration is efficient and effective for visual TTT. The inner model comprises two useful modules we identified in Sec. 4.2: a simplified gated linear unit $\mathcal{F}_1 = \text{FC}(x) \odot \text{SiLU}(\text{FC}(x))$ and a depthwise convolution $\mathcal{F}_2 = \text{DWConv}(x)$. The gated linear unit doubles the capacity of a naive $d \times d$ linear state while remaining easy to optimize, whereas the depthwise convolution offers a natural integration of local and global information. Within each TTT block, we use \mathcal{F}_2 in a single attention head and instantiate the remaining heads with \mathcal{F}_1 . The resulting design is a drop-in replacement for standard attention blocks and can be integrated into various vision Transformer backbones. We practically build two model families, ViT³ (non-hierarchical) and H-ViT³ (hierarchical 4-stage), following the design philosophies of classical vision Transformers [12, 55] and modern 4-stage architectures [34, 59], respectively. We further adapt our approach to diffusion image Transformers

(DiT) [41] for generative tasks, building DiT³ models. Detailed model architectures are shown in the Appendix.

5.1. Image Classification

The ImageNet-1K [10] dataset contains 1.28M training images and 50K validation images across 1,000 classes. To ensure a fair comparison with prior work, we follow the training protocol used in Swin Transformer [34]. Concretely, all models are trained from scratch for 300 epochs with the AdamW optimizer [36], a cosine learning-rate decay and a linear warm-up for the first 20 epochs. We use a weight decay of 0.05. The total batch size is 4096 and initial learning rate is set to 4×10^{-3} . Standard augmentation and regularization methods are applied, including RandAugment [7], Mixup [74], CutMix [73], and random erasing [77]. We also report results with MESA [13] training strategy.

The results are presented in Tab. 5 and Tab. 7. We observe that ViT³ and H-ViT³ models consistently outperform various linear attention and Mamba variants, while remaining competitive with state-of-the-art vision Transformer models. For example, H-ViT³-S attains higher performance than the larger SOFT-L++ [37] and VMamba-B [33], despite using roughly half of parameters and FLOPs. These results validate the advantages of the TTT paradigm over prior linear-complexity designs and highlight its potential for efficient, scalable $\mathcal{O}(N)$ visual sequence modeling.

5.2. Object Detection

The COCO [32] dataset is a widely adopted benchmark for object detection and instance segmentation. We employ the standard 1x and 3x Mask R-CNN [26] training schedules, presenting the results in Tab. 8. With high-resolution inputs, token sequences in object detection are typically much

Backbone	Type	#Params	FLOPs	mIoU
ConvNeXt-T [35]	ConvNet	60M	939G	46.0
Focal-T [68]	Transformer	62M	998G	45.8
FasterViT-2 [23]	Transformer	-	974G	47.2
VMamba-T [33]	Mamba	62M	949G	47.9
SOFT-T++ [37]	Linear	60M	948G	46.5
VVT-S [53]	Linear	56M	960G	46.8
H-ViT ³ -T	TTT	58M	946G	48.0
<hr/>				
ConvNeXt-S [35]	ConvNet	82M	1027G	48.7
CSwin-S [11]	Transformer	65M	1027G	50.4
TransNeXt-S [50]	Transformer	80M	1089G	52.2
LocalVMamba-S [29]	Mamba	81M	1095G	50.0
VVT-M [53]	Linear	78M	1040G	48.1
SOFT-S++ [37]	Linear	81M	1040G	48.9
H-ViT ³ -S	TTT	84M	1026G	50.2
<hr/>				
ConvNeXt-B [35]	ConvNet	122M	1170G	49.1
CSwin-B [11]	Transformer	109M	1222G	51.1
TransNeXt-B [50]	Transformer	121M	1268G	53.0
VMamba-B [33]	Mamba	122M	1170G	51.0
VVT-L [53]	Linear	92M	1068G	48.8
SOFT-B++ [37]	Linear	121M	1204G	49.2
H-ViT ³ -B	TTT	124M	1195G	51.7

Table 9. Results of semantic segmentation. FLOPs are calculated with an input resolution of 512×2048 .

longer than in image classification [71], resulting in $N \gg d$. Under these conditions, existing linear-complexity designs such as VMamba [33] and SOFT++ [37] tend to be constrained by limited state capacity and therefore frequently underperform. In contrast, with nonlinear inner modules and the online learning procedure, H-ViT³ attains stronger global modeling: it matches or surpasses state-of-the-art linear-complexity methods and greatly narrows the performance gap to highly optimized vision Transformers.

5.3. Semantic Segmentation

The ADE20K [78] is a well-established benchmark for semantic segmentation. We use UPerNet [64] as the framework. As shown in Tab. 9, similar to the trend in detection task, H-ViT³ establishes a strong linear-complexity baseline for semantic segmentation and consistently outperforms VMamba [33], VVT [53] and SOFT++ [37]. Nevertheless, H-ViT³ remains inferior to highly optimized vision Transformers such as TransNeXt [50]. We argue that designing deeper, more expressive inner modules is a promising direction to further boost capacity and to build TTT models comparable with state-of-the-art Transformers.

5.4. Image Generation

We further benchmark TTT method on the class-conditional image generation task using the ImageNet-1K dataset. Specifically, we replace the Softmax attention module in DiT [41] with our ViT³ block, yielding the DiT³ model family. Following standard protocol, we report FID on 50 000 validation samples (FID-50K) at 256^2 resolution. Tab. 10 shows that DiT³ consistently improves DiT in various settings without extra tuning. These results validate the effec-

Model	#Params	FLOPs	FID↓	IS↑	Prec.↑	Rec.↑
DiT-S/8 [41]	33M	0.36G	153.60	-	-	-
DiT ³ -S/8	35M	0.40G	143.49	8.34	0.15	0.13
DiT-S/4 [41]	33M	1.41G	100.41	-	-	-
DiT ³ -S/4	35M	1.57G	93.77	14.42	0.27	0.40
DiT-S/2 [41]	33M	6.06G	68.40	-	-	-
DiT ³ -S/2	35M	6.23G	62.65	21.59	0.39	0.57
<hr/>						
DiT-B/8 [41]	131M	1.42G	122.74	-	-	-
DiT ³ -B/8	135M	1.51G	120.41	10.94	0.20	0.26
DiT-B/4 [41]	130M	5.56G	68.38	-	-	-
DiT ³ -B/4	134M	5.88G	65.25	22.29	0.37	0.55
DiT-B/2 [41]	130M	23.01G	43.47	-	-	-
DiT ³ -B/2	134M	23.35G	39.31	36.99	0.51	0.62

Table 10. Results of class-conditional image generation.

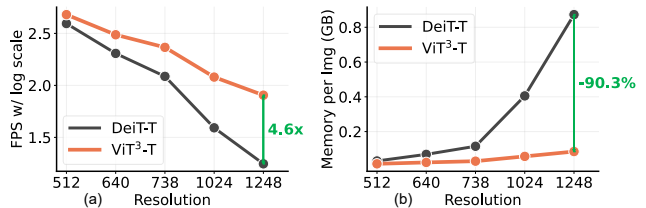


Figure 4. Comparisons between DeiT and ViT³ in (a) FPS on RTX3090, and (b) per image GPU memory usage.

tiveness of ViT³ for image generation, and establish strong TTT baselines for linear-complexity approaches.

5.5. Efficiency Analysis

The above evaluations primarily examine the expressiveness of TTT method. Here we would like to highlight its efficiency advantages over the widely adopted Softmax attention. Fig. 4 compares the FPS and GPU memory usage of ViT³-T and DeiT-T. Owing to its linear time and memory complexity, ViT³ scales more efficiently in both throughput and memory consumption as image resolution increases. At 1248^2 resolution (i.e., 6,084 tokens per image), ViT³-T attains a 4.6× speedup over DeiT-T and reduces GPU memory consumption by 90.3%, demonstrating its efficiency.

6. Conclusion

Designing efficient yet expressive sequence-modeling paradigms remains a key pursuit in computer vision. In this paper, we systematically study the design space of Test-Time Training (TTT), a new promising approach for building scalable models with linear complexity. Through a series of experiments, we distill our observations into six practical insights, shedding some light on the design principles for effective visual TTT and possible future directions. Our findings and analyses culminated in the Vision Test-Time Training (ViT³) model, a pure TTT architecture for visual sequence modeling. ViT³ achieves competitive results across multiple tasks, serving as a strong baseline for linear-complexity methods. We hope our findings and analyses can stimulate further research on visual TTT models.

Acknowledgements

This work is supported in part by the National Key R&D Program of China under Grant 2024YFB4708200, the National Natural Science Foundation of China under Grants U24B20173 and U2541227, and the Scientific Research Innovation Capability Support Project for Young Faculty under Grant ZYGXQNJSKYCXNLZCXM-I20.

References

- [1] Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025. 2
- [2] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025. 2
- [3] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *NeurIPS*, 2022. 2
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1
- [5] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025. 2
- [6] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *ICLR*, 2021. 1
- [7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020. 7
- [8] Karan Dalal, Daniel Kocreja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *CVPR*, 2025. 2, 6
- [9] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *NeurIPS*, 2011. 6
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 7
- [11] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, 2022. 2, 6, 7, 8
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 6, 7
- [13] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In *NeurIPS*, 2022. 6, 7
- [14] Ronen Eldan and Ohad Shamir. The power of depth for feed-forward neural networks. In *Conference on learning theory*, 2016. 6
- [15] Qihang Fan, Huaibo Huang, Mingrui Chen, Hongmin Liu, and Ran He. Rmt: Retentive networks meet vision transformers. In *CVPR*, 2024. 6
- [16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 2, 4
- [17] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2
- [18] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *ICCV*, 2023. 1, 2, 3, 4, 7
- [19] Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *NeurIPS*, 2024. 3, 5
- [20] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. In *NeurIPS*, 2024. 4, 6, 7
- [21] Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *ECCV*, 2024. 6
- [22] Dongchen Han, Tianyu Li, Ziyi Wang, and Gao Huang. Vision transformers are circulant attention learners. In *AAAI*, 2026. 2
- [23] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *CVPR*, 2023. 2, 8
- [24] Ali Hatamizadeh, Greg Heinrich, Hongxu Yin, Andrew Tao, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. Fastervit: Fast vision transformers with hierarchical attention. In *ICLR*, 2024. 6
- [25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6
- [26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 7
- [27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*, 2017. 6
- [28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 6
- [29] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. In *ECCVW*, 2024. 6, 7, 8
- [30] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020. 1, 2, 3, 5
- [31] Shufan Li, Harkanwar Singh, and Aditya Grover. Mamband: Selective state space modeling for multi-dimensional data. In *ECCV*, 2024. 6
- [32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence

- Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 7
- [33] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. In *NeurIPS*, 2024. 2, 4, 6, 7, 8
- [34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 2, 7
- [35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 6, 8
- [36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 7
- [37] Jiachen Lu, Junge Zhang, Xi Tian Zhu, Jianfeng Feng, Tao Xiang, and Li Zhang. Softmax-free linear transformers. *IJCV*, 2024. 6, 7, 8
- [38] Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. In *ICLR*, 2019. 2
- [39] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NeurIPS*, 2014. 6
- [40] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. sl: Simple test-time scaling. In *EMNLP*, 2025. 2
- [41] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1, 7, 8
- [42] Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. In *AAAI*, 2025. 2
- [43] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NeurIPS*, 2016. 6
- [44] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *ICLR*, 2022. 1, 3, 5
- [45] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, 2017. 6
- [46] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 6
- [47] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *ICLR*, 2025. 2
- [48] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021. 2
- [49] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 5
- [50] Dai Shi. Transnext: Robust foveal visual perception for vision transformers. In *CVPR*, 2024. 6, 7, 8
- [51] Chenyang Si, Weihao Yu, Pan Zhou, Yichen Zhou, Xinchao Wang, and Shuicheng Yan. Inception transformer. In *NeurIPS*, 2022. 6
- [52] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. 2
- [53] Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. Vicinity vision transformer. *TPAMI*, 2023. 6, 8
- [54] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. In *ICML*, 2025. 1, 2, 3, 4, 6
- [55] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 2, 3, 6, 7
- [56] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021. 2, 5
- [57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 4, 5
- [58] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *TPAMI*, 2024. 5
- [59] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 2, 7
- [60] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *CVPR*, 2023. 6, 7
- [61] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, 2022. 2
- [62] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Dat++: Spatially dynamic vision transformer with deformable attention. *arXiv preprint arXiv:2309.01430*, 2023. 7
- [63] Zhuofan Xia, Dongchen Han, Yizeng Han, Xuran Pan, Shiji Song, and Gao Huang. Gsva: Generalized segmentation via multimodal large language models. In *CVPR*, 2024. 1
- [64] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 8
- [65] Yicheng Xiao, Lin Song, Jiangshan Wang, Siyu Song, Yixiao Ge, Xiu Li, Ying Shan, et al. Mambatree: Tree topology is all you need in state space model. In *NeurIPS*, 2024. 4
- [66] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. 1

- [67] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *AAAI*, 2021. [2](#)
- [68] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021. [7](#), [8](#)
- [69] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *ICML*, 2024. [2](#)
- [70] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *NeurIPS*, 2024. [2](#)
- [71] Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? In *CVPR*, 2025. [4](#), [8](#)
- [72] Weihao Yu, Pan Zhou, Shuicheng Yan, and Xinchao Wang. Inceptionnext: When inception meets convnext. In *CVPR*, 2024. [6](#)
- [73] Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. [7](#)
- [74] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. [7](#)
- [75] Qiming Zhang, Jing Zhang, Yufei Xu, and Dacheng Tao. Vision transformer with quadrangle attention. *TPAMI*, 2024. [7](#)
- [76] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. In *ICLR*, 2026. [2](#), [4](#), [6](#)
- [77] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. [7](#)
- [78] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019. [8](#)
- [79] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *CVPR*, 2023. [2](#), [6](#)
- [80] Lianghai Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: efficient visual representation learning with bidirectional state space model. In *ICML*, 2024. [6](#)
- [81] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. [1](#)