How Far Can Vision Test Time Training Models Adapt at Test Time?

Anonymous authors
Paper under double-blind review

Abstract

Test-time training (TTT) lets a model adapt its computation to every input through an inner parameter update, and how far that update can move before adaptation stops working has not been examined. We find that vision TTT adapts over a narrow range. Modest changes in update magnitude can erase the benefit of adaptation, and the preferred magnitude shifts as the input geometry changes. We show that this range is learned during training, and we test this with a minimal change to the training recipe, sampling the magnitude of the inner update on each forward pass and sharing it across all TTT blocks, so that the model learns from a continuum of adapted states while each update keeps its sample-specific direction. Range-trained models remain effective across a much broader span of update magnitudes at a small cost at the standard setting, their learned computation still depends strongly on sample-specific update directions, and the learned range persists under distribution shift, changes in input resolution and downstream fine-tuning. Vision TTT can thus learn how far to adapt at inference, which makes adaptation range a trainable property of the model.

1 Introduction

Test-time training (TTT) layers turn part of a network's forward pass into learning. A TTT layer fits a small inner model to the current input and then uses the fitted model to produce its output (Sun et al., 2025; Behrouz et al., 2025; Zhang et al., 2026). Attention and state-space layers apply the same weights to every input at inference (Vaswani et al., 2017; Gu & Dao, 2024), while the weights of a TTT layer are fitted anew for each input. This makes the layer a direct form of input-conditioned computation (Han et al., 2022; Ha et al., 2017; Yang et al., 2019) in the tradition of fast weights (Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021). First developed for language modeling, TTT layers now serve as the token mixer of vision backbones (Han et al., 2026), video generators (Dalal et al., 2025) and view synthesis models (Zhang et al., 2026), where they pair a cost linear in the number of tokens with per-input adaptation.

Each adaptation is a gradient step on the inner model, and the step has a magnitude. In current vision TTT models the magnitude is set by the inner learning rate, a hyperparameter chosen once by comparing separately trained models and then fixed for every block and every input (Han et al., 2026). Several ordinary uses of a trained model move the step away from that value. The inner gradient averages over the tokens of an image, so changing the input resolution changes the step the network actually takes. Skipping the update, the natural way to measure what adaptation contributes, sets the step to zero. Taking more inner steps, or choosing the step for each input, asks the network to compute at magnitudes it never met in training. A TTT model supports these uses only if its adapted computation keeps working over a range of magnitudes.

Existing analyses of TTT layers identify the operator a trained layer implements, such as a learned form of linear attention for TTT with key-value binding (Liu et al., 2026), and relate the inner objective to associative memory (Wang et al., 2025; Behrouz et al., 2026b). They take the step as given. How far the step can move before the adapted computation stops working, and what decides that range, has not been examined.

We find that current vision TTT models adapt over a narrow range. Their accuracy peaks at the trained magnitude and falls quickly when the step is made shorter or longer. Skipping the update altogether hurts more than removing the TTT branch from every block, and the magnitude the network

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

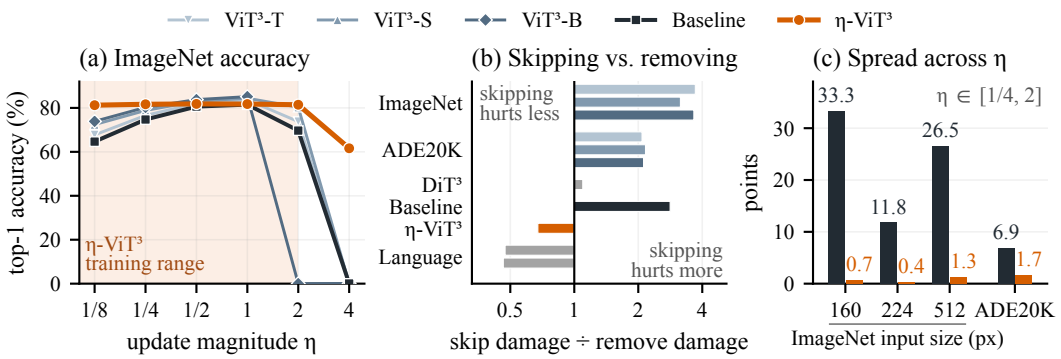

Figure 1: (a) ImageNet top-1 accuracy as the update magnitude η of every TTT block is changed at test time. ViT^3 and our baseline, trained at $\eta = 1$, work only near that value. η-ViT^3 is trained with η drawn from the shaded interval and stays accurate from $\eta = 1/8$ to $\eta = 2$. (b) Damage of skipping the update divided by the damage of removing the TTT branch. Above one, the network does better without its TTT layers than with unadapted ones. Rows give ViT^3 on ImageNet and ADE20K, DiT^3 for image generation, the baseline and η-ViT^3 on ImageNet, and the language models TTT-Linear and TTT-MLP. (c) Difference between the best and the worst result over $\eta \in [1/4, 2]$, in ImageNet top-1 accuracy at three input sizes and in ADE20K mIoU after fine-tuning.

prefers moves with the input resolution. Language TTT models tolerate a wide span of magnitudes under the same operations, so the narrow range is not a property of test-time training itself. Following a vision model through training shows the gap between the damage of skipping the update and that of removing the branch widening steadily, as the network commits to the one magnitude it is trained with. The usable range of test-time adaptation is learned during training.

Training is also where the range can be widened. Every training pass of ViT^3 uses the same magnitude, so the layers after each TTT block only learn to read outputs from fast weights one standard step away from their initial values. Language TTT layers update their fast weights many times within a training sequence and see outputs at many such distances (Sun et al., 2025). We introduce η-ViT^3, which gives a vision TTT network the same variety through a minimal change to the ViT^3 training recipe. On every forward pass it draws the update magnitude from an interval that starts at zero and is centered on the standard value, and shares it across all TTT blocks. The network thus learns from a continuum of adapted states, including the unadapted one, while the direction of every update is still computed from the current input. At inference η-ViT^3 runs exactly like ViT^3, with no added parameters and no added computation.

η-ViT^3 widens the usable range while keeping the adaptation specific to each image. On ImageNet it loses at most 0.5 points of top-1 accuracy between one eighth and twice the standard magnitude, where a baseline trained with the same recipe loses up to 16.8 points, and the two models are on par at the standard setting (81.9% against 81.8%). The wider range carries over to larger classifiers and to image generation. It persists under distribution shift, after fine-tuning for ADE20K segmentation, and across input resolutions from 160 to 512 pixels, where the accuracy of η-ViT^3 varies by at most 1.3 points between a quarter and twice the standard magnitude. Replacing every update of η-ViT^3 with a random direction of the same length drops it to chance, more than six times the damage of removing its TTT branch, so the learned computation still depends on the direction computed for each image.

The narrow range of ViT^3 comes from what its update does to the output of each TTT block (Section 6). Its update cancels an output computed at the initial weights that is harmful on its own, so any step other than the trained one leaves part of that output in place or overshoots it. Training over a range of magnitudes turns the initial-weight output into a useful one, and the step can then change without breaking the network.

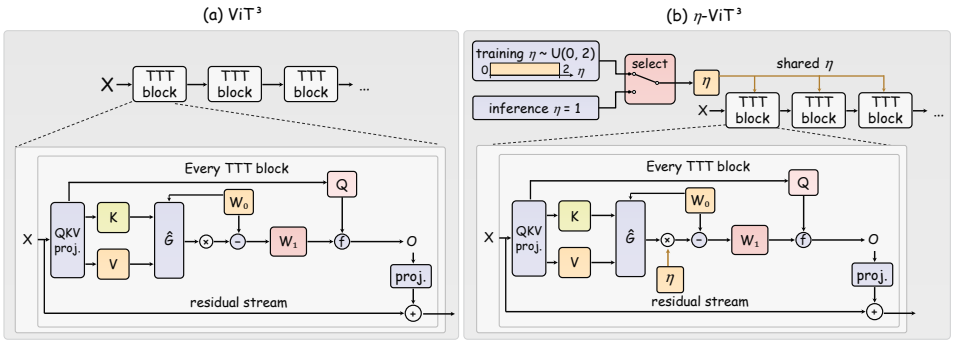

Figure 2: One TTT block of ViT^3 (left) and of η-ViT^3 (right). Linear projections produce Q, K and V from the N token features of one image. The inner model evaluates the dot-product loss between its predictions from the keys and the values at the initial weights W_0, the gradient G is normalized to $\hat{G}$, and one step of magnitude η gives the adapted weights W_1, which the inner model applies to the queries. The output O is projected and added to the residual stream. The two models differ only in η. ViT^3 uses $\eta = 1$ in every training forward pass and at inference. η-ViT^3 draws one $\eta \sim \mathcal{U}(0, 2)$ per training forward pass, shares it across all TTT blocks and uses $\eta = 1$ at inference.

2 Test-Time Training in Vision

The TTT block of ViT^3. ViT^3 builds a vision backbone from test-time training (TTT) layers (Han et al., 2026), shown in Figure 2. Each block keeps the residual structure of a vision transformer and replaces self-attention with a TTT layer. Given the N token features of one image, linear projections produce queries, keys and values $Q, K, V \in \mathbb{R}^{N \times d}$ for every head. Inside each head sits a small inner model $f(\cdot\,; W)$ whose weights are fitted to the current image. Most heads use a gated linear unit (Shazeer, 2020) with the SiLU activation (Elfwing et al., 2018) and without an output layer,

$$f(K; W) = (KW_a) \odot \mathrm{SiLU}(KW_b), \qquad W = \{W_a, W_b\}, \quad W_a, W_b \in \mathbb{R}^{d \times d}, \tag{1}$$

and one additional head uses a 3×3 depthwise convolution (Howard et al., 2017) over the keys laid out on the two-dimensional token grid, which adds local spatial context (Han et al., 2026). The inner model is fitted to predict each value from its key under a dot-product loss,

$$\mathcal{L}(W; K, V) = -\frac{s}{N} \sum_{i=1}^{N} \big\langle f(K; W)_i,\, \boldsymbol{v}_i \big\rangle, \tag{2}$$

where $f(K; W)_i$ is the prediction for token i, $\boldsymbol{v}_i$ is its value and s is a fixed scale.

Every image starts from the same initial weights W_0 of the inner model. These are ordinary parameters of the network, learned by the outer training together with the projections. For each image, the block computes the gradient of Eq. (2) at W_0 and takes a single normalized step,

$$W_1 = W_0 - \eta\, \hat{G}, \qquad \hat{G} = \frac{G}{\|G\| + 1}, \qquad G = \nabla_W \mathcal{L}(W_0; K, V), \tag{3}$$

where the normalization is applied separately to every column of W_a and W_b and to every convolution kernel. The adapted inner model is then applied to the queries, and its output $O = f(Q; W_1)$ is projected back to the model width and added to the residual stream. The inner step is differentiable, and training back-propagates through it, so the network learns W_0 jointly with the projections that produce Q, K and V (Finn et al., 2017; Sun et al., 2025).

The step in Eq. (3) has a direction and a magnitude. The direction $\hat{G}$ is computed from the current image, so two images produce two different sets of fast weights and hence two different computations on their queries. This per-input computation separates a TTT layer from attention and state-space layers, whose weights stay fixed at inference (Vaswani et al., 2017; Gu & Dao, 2024). The magnitude is set by the inner learning rate η. Every column of $\hat{G}$ has norm below one and η scales the whole step, so we refer to η as the update magnitude. ViT^3 chose $\eta = 1$ from a sweep over separately trained models (Han et al., 2026), and every ViT^3 model uses this value in every block, for every image, in training and at inference.

Changing the update at test time. To measure how far adaptation can move, we change the update of a trained network at test time and leave every learned weight untouched. Each change is applied to all TTT blocks at once.

Scaling the update evaluates the network with a magnitude $\eta \neq 1$ in Eq. (3). Magnitudes below one adapt less than in training and magnitudes above one adapt more. At $\eta = 0$ the update is skipped, and every inner model computes its output with its initial weights W_0.

Removing the TTT branch sets the output of every TTT layer to zero, so the residual stream passes the layer unchanged while the rest of each block stays in place. The two operations answer different questions. Removing the branch asks how much the network relies on the TTT layer at all. Skipping the update asks how much it relies on adaptation.

Changing the direction keeps the length of every update and replaces its direction. For each image, head and convolution kernel, the replacement is rescaled to the norm of the original update. We use three replacements. The first is a random Gaussian direction, the second is the update computed on another image of the same batch, and the third is the average update over the batch. If the network relies on the direction computed for each image, all three should cost accuracy.

We report each change as the drop it causes relative to the unmodified model, in top-1 accuracy for classification, mean intersection over union (mIoU) for segmentation, Fréchet inception distance (FID) for generation and negative log-likelihood per token for language modeling. Classification is evaluated on a fixed subset of 5,000 ImageNet validation images. The classification and segmentation models contain batch normalization (Ioffe & Szegedy, 2015), and changing the update also changes the statistics these layers receive. After each change we re-estimate them on training images under the same change, as is common when the inputs of these layers shift (Li et al., 2018; Schneider et al., 2020), taking the median over batches (Appendix A).

3 Vision TTT Adapts over a Narrow Range

We apply the operations of Section 2 to the ViT^3 ImageNet classifiers in three sizes, ViT^3-T, ViT^3-S and ViT^3-B with 29M, 54M and 94M parameters and 17, 34 and 34 TTT blocks, all trained at 224×224 resolution (Han et al., 2026; Russakovsky et al., 2015). We also use the ViT^3 ADE20K segmentation models, which put an UperNet head on each backbone (Zhou et al., 2017; Xiao et al., 2018), and the class-conditional image generator DiT^3-B/2, a diffusion transformer built from the same TTT layers (Peebles & Xie, 2023; Han et al., 2026). For comparison we apply the same operations to the language TTT models TTT-Linear and TTT-MLP with 125M parameters, evaluated on books from PG19 (Sun et al., 2025; Rae et al., 2020).

Accuracy peaks at the trained magnitude. Figure 1a plots ImageNet accuracy against the update magnitude. All three classifiers reach their best accuracy at $\eta = 1$, the value used in training, and lose accuracy quickly on either side. At one eighth of the trained magnitude they lose 11.3 to 16.0 points, and at one quarter 4.9 to 7.1 points. Doubling the magnitude costs ViT^3-T 9.9 points and ViT^3-S 4.1 points, and at four times the magnitude all three models are at chance. ViT^3 adapts well at one magnitude and poorly a factor of two away from it.

Skipping the update is worse than removing the TTT branch. Skipping the update leaves every TTT layer in place and only withholds adaptation, so it would be natural to expect it to cost at most as much as removing the layers altogether. Tables 1 and 3 and Figure 1b show the opposite for nearly every vision model. On ImageNet, skipping the update costs 37.4 to 46.3 points, 3.2 to 3.7 times the cost of removing the TTT branch. The same ordering holds for ADE20K segmentation, where skipping costs 2.1 to 2.2 times as much as removing. In image generation with DiT^3, skipping the update raises the FID about as much as removing the branch. The inner model at its initial weights produces a damaging output, which only the update at its trained magnitude turns into a useful one (Section 6).

The language models behave differently (Table 1, bottom). Halving, doubling or quadrupling the update magnitude of TTT-Linear and TTT-MLP raises the negative log-likelihood by at most 0.06 nats per token, and skipping the update costs less than half as much as removing the TTT layers. The narrow range comes from how the vision models are trained (Section 4).

Table 1: Changing the update of TTT models at test time. Each entry is the damage a change causes relative to the unmodified model, as a drop in ImageNet top-1 accuracy or ADE20K mIoU, a rise in FID for generation, or a rise in negative log-likelihood (nats per token) on PG19. The ratio column divides the damage of skipping the update by the damage of removing the TTT branch, so values above one mean the network does better without its TTT layers than with unadapted ones. ADE20K uses 2,000 validation images for ViT^3-T and 500 for ViT^3-S and ViT^3-B, which are evaluated in the three settings the ratio needs.

Task	Model	Unmodified	Remove branch	Skip update	Ratio	$\eta = 1/2$	$\eta = 2$
ImageNet top-1	ViT^3-T	83.7	12.5	46.3	3.71	1.6	9.9
	ViT^3-S	84.4	11.8	37.4	3.16	1.4	4.1
	ViT^3-B	85.1	10.6	38.6	3.65	1.4	85.1
ADE20K mIoU	ViT^3-T	46.1	17.3	35.9	2.08	2.1	10.3
	ViT^3-S	42.3	14.9	32.3	2.16	–	–
	ViT^3-B	43.9	18.0	38.1	2.12	–	–
Generation FID	DiT^3-B/2	46.5	241.3	264.5	1.10	68.0	174.6
PG19 NLL	TTT-Linear	2.70	2.66	1.26	0.47	0.05	0.02
	TTT-MLP	2.67	2.73	1.26	0.46	0.03	0.02

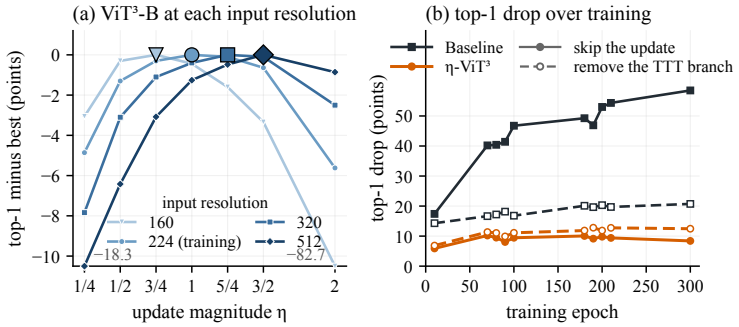

Figure 3: (a) Accuracy of ViT^3-B relative to its best magnitude at each input resolution. The best magnitude, marked on each curve, moves from 0.75 at 160 pixels to 1.5 at 512 pixels. (b) Damage of skipping the update and of removing the TTT branch over the course of training. For the baseline, skipping the update becomes steadily more damaging than removing the branch. For η-ViT^3 it stays below the cost of removal throughout.

The preferred magnitude moves with input resolution. The inner gradient in Eq. (3) averages over all tokens of the image, and the number of tokens grows with the input resolution. Running a ViT^3 classifier at a resolution other than 224 changes the update even at $\eta = 1$. From 160 to 512 pixels, the change that the update makes to the gated-unit weights shrinks steadily as the resolution grows, by 15 to 19 percent over this span in every model we test. The magnitude with the best accuracy moves with it (Figure 3a). For ViT^3-B it is 0.75 at 160 pixels, 1 at the training resolution, 1.25 at 320 pixels and 1.5 from 384 pixels on, and ViT^3-T and ViT^3-S also prefer 1.5 at 512 pixels. Each input size thus calls for its own magnitude, and the span of magnitudes the model tolerates narrows toward both ends of the resolution range (Section 5).

The narrow range forms during training. Our ViT^3-T baseline (Section 5) lets us follow the range through training (Figure 3b). After 10 epochs, skipping the update already costs 17.4 points, slightly more than removing the branch at 14.3. The gap widens as training continues. At epoch 100 skipping the update costs 46.8 points against 16.8 for removal, and at the end of training 58.5 points against 20.7. The network commits more and more to the one magnitude it is trained with, so training is where the usable range is decided and where it can be widened.

4 η-ViT^3

The narrow range forms during training (Section 3). Every training forward pass of ViT^3 uses $\eta = 1$, so the layers that read a TTT block only ever see outputs from fast weights exactly one unit step away from W_0. Outputs at any other magnitude, including the output at W_0 itself, are inputs the rest of the network was never trained to read.

Language TTT layers are trained differently. TTT-Linear and TTT-MLP split a sequence into mini-batches of tokens and update their fast weights after every mini-batch, with a learned inner learning rate that depends on the token (Sun et al., 2025). Within one training sequence, the rest of the network reads outputs from fast weights at many distances from their initialization, and these models tolerate a wide span of update magnitudes at test time (Section 3). We give a vision TTT network the same variety through the update magnitude, the one quantity that controls how far its fast weights move.

For every training forward pass, we draw one update magnitude from a uniform distribution and use it in every TTT block,

$$\eta \sim \mathcal{U}(0, 2), \qquad W_1^{(\ell)} = W_0^{(\ell)} - \eta\, \hat{G}^{(\ell)} \quad \text{for every TTT block } \ell, \tag{4}$$

and all images in the pass share the same draw. Everything else follows ViT^3, including the inner model, the loss in Eq. (2), the gradient normalization, the architecture, the optimizer and the schedule. Training back-propagates through the inner step at the sampled magnitude, so the initial weights and every later layer learn from a continuum of adapted states. We call the resulting model η-ViT^3. At inference we set $\eta = 1$, and it computes exactly as a ViT^3 model with the same weights.

The interval in Eq. (4) is centered on the standard magnitude, so the standard setting lies in the middle of the range seen in training. Its lower end reaches zero, which makes the initial weights W_0 one of the trained states, and the output they compute must then be useful to the rest of the network on its own. One magnitude is shared by all blocks, which keeps the whole network in a single consistent state of adaptation at every pass and leaves one number to set at inference.

The change amounts to one random scalar per forward pass. It adds no parameters and no computation, and an η-ViT^3 checkpoint runs and fine-tunes exactly like a ViT^3 checkpoint (Appendix F).

Choosing the training interval. We set the interval with ViT^3-T models trained on ImageNet-100 for 100 epochs, one model per interval (Table 6 in Appendix C). Its lower end matters most, because it decides whether the initial weights become usable. With intervals that start at 0.25, skipping the update at test time still costs about twice as much as removing the TTT branch (10.2 and 11.6 points against 4.8 and 5.6). With intervals that start at 0, skipping the update costs less than removing the branch (2.1 and 2.5 points against 3.2 and 3.3).

The most direct alternative skips the update in a small fraction of training passes, in the spirit of dropout and stochastic depth (Srivastava et al., 2014; Huang et al., 2016). Skipping it in 5% of the passes also makes the initial weights usable, but the network then learns to ignore the direction of the update. For that model, replacing the update with a random direction costs 2.7 points, less than the 3.5 points of removing the branch. The η-ViT^3 model trained on the same data loses 9.0 points under the same replacement, nearly three times its removal cost. Sampling the magnitude makes the initial weights usable and keeps the direction informative at the same time.

5 Experiments

Setup. We train two ViT^3-T models on ImageNet-1k (Deng et al., 2009; Russakovsky et al., 2015) for 300 epochs with the ViT^3 code and training recipe at 224×224 resolution, with a total batch size of 512 (Appendix B). The baseline keeps $\eta = 1$ throughout training, exactly as ViT^3 does. η-ViT^3 uses the same code, recipe and hyperparameters and differs only in drawing η from $\mathcal{U}(0, 2)$ at every forward pass, as in Eq. (4). We make the same change to two larger classifiers, ViT^3-S and ViT^3-B, trained for 100 epochs with the same recipe, and to the class-conditional image generator DiT^3-B/2, trained for 400k steps with the DiT^3 recipe (Peebles & Xie, 2023; Han et al., 2026). We call this generator η-DiT^3 and compare generators by FID (Appendix A).

Table 2: Changing the update of the baseline and η-ViT^3 at test time on ImageNet. Top-1 is the accuracy of the unmodified model on the full validation set. All other entries are drops in top-1 accuracy on 5,000 validation images caused by setting the update magnitude to the value shown, by removing the TTT branch, or by replacing every update with a direction of the same length taken from a random draw, from another image in the batch, or from the batch average. Lower damage is bold in the magnitude columns.

		Update magnitude η						Remove branch	Replaced direction		
Model	Top-1	0	1/8	1/4	1/2	2	4		Random	Other image	Batch mean
Baseline	81.8	58.5	16.8	6.8	0.9	11.8	81.4	20.7	78.3	59.1	26.8
η-ViT^3	81.9	**8.4**	**0.5**	**0.1**	**-0.1**	**0.3**	**20.1**	12.5	81.7	79.0	20.2

We measure every change of Section 2 as a drop in top-1 accuracy on 5,000 ImageNet validation images, with batch-normalization statistics re-estimated under the change. For distribution shift we use ImageNet-C with its 15 corruptions at severities 3 and 5 (2,000 images per corruption and severity), ImageNet-R and ImageNet-A (5,000 images each) (Hendrycks & Dietterich, 2019; Hendrycks et al., 2021a;b). Under shift we keep the batch-normalization statistics from training, so that the update of the TTT layers is the only form of test-time adaptation. For resolution we evaluate every model at nine input sizes from 160 to 512 pixels on 5,000 validation images each. For downstream fine-tuning we train an UperNet segmentation model on ADE20K from each backbone for 160k iterations with the ViT^3 segmentation recipe, which keeps $\eta = 1$ throughout, and evaluate on 2,000 validation images (Zhou et al., 2017; Xiao et al., 2018; MMSegmentation Contributors, 2020).

A wider usable range at a small cost at the standard setting. Figure 1a and Table 2 compare η-ViT^3 with the baseline. From one eighth to twice the standard magnitude, η-ViT^3 loses at most 0.5 points, while the baseline loses up to 16.8 points over the same span. At four times the standard magnitude the baseline is at chance and η-ViT^3 still classifies 61.6% of the images correctly. The wider range comes at a small cost at the standard setting. η-ViT^3 reaches 81.9% top-1 on the full validation set against 81.8% for the baseline, and the larger classifiers stay within 0.5 points of their baselines (Table 3).

At the lower end of the range, skipping the update now costs 8.4 points, less than the 12.5 points of removing the TTT branch. η-ViT^3 can switch adaptation off and still run as a working model, from the first epochs of training to the last (Figure 3b), and the contribution of adaptation can be read directly as the gap between the adapted and the unadapted network.

The range holds across model sizes and in image generation. Table 3 repeats these changes on ViT^3-S, ViT^3-B and the class-conditional image generator DiT^3-B/2, each trained once at $\eta = 1$ and once with sampled magnitudes. From one eighth to twice the standard magnitude, η-ViT^3-S and η-ViT^3-B lose at most 0.9 and 0.4 points, against up to 11.6 and 11.0 points for their baselines. Skipping the update costs both less than removing the TTT branch, and a random direction of the same length leaves both with at most 1.1% top-1 accuracy. Halving or doubling the magnitude changes the FID of η-DiT^3 by at most 2.9, while the FID of its baseline rises by 75.6 and 163.6, and skipping the update raises the FID of η-DiT^3 by less than half as much as removing the branch.

The range holds across input resolutions. At every resolution from 160 to 512 pixels, the accuracy of η-ViT^3 varies by at most 1.3 points over $\eta \in [1/4, 2]$. The same spread is 8.5 to 33.3 points for the baseline and 5.3 to 82.7 points for ViT^3, and it is smallest near the training resolution and grows toward both ends of the range (Figures 1c and 4). One magnitude setting serves η-ViT^3 at every input size, while ViT^3 would need a separate setting for each size, as Section 3 showed.

6 The Initial-Weight Output Sets the Range

A TTT layer that computes with its initial weights hurts the network more than no TTT layer at all (Section 3). We split the output of each TTT layer into two parts to find the source of this damage. At the standard magnitude the layer outputs $O = f(Q; W_1)$. We write it as

$$O = O_0 + U, \qquad O_0 = f(Q; W_0), \qquad U = O - O_0, \tag{5}$$

Table 3: Changing the update at test time in larger classifiers and in an image generator, each trained once at $\eta = 1$ (baseline) and once with $\eta \sim \mathcal{U}(0, 2)$. Unmodified is the top-1 accuracy on the full ImageNet validation set or the FID of 5,000 samples. Other entries are drops in top-1 accuracy on 5,000 validation images or rises in FID. Lower damage is bold in the magnitude columns.

			Update magnitude η						Remove branch	Random direction
Backbone	Model	Unmodified	0	1/8	1/4	1/2	2	4		
ViT^3-S	Baseline	82.6	38.1	11.6	5.3	1.4	6.2	82.9	13.1	51.2
	η-ViT^3	82.1	**5.4**	**0.9**	**0.5**	**0.2**	**0.0**	**4.5**	8.2	81.3
ViT^3-B	Baseline	82.8	32.6	11.0	5.0	1.1	4.7	82.6	12.6	47.6
	η-ViT^3	82.5	**5.1**	**0.4**	**0.3**	**0.0**	**0.2**	**1.3**	7.2	81.8
DiT^3-B/2	Baseline	47.1	222.8	–	–	75.6	163.6	–	246.1	–
	η-DiT^3	50.1	**80.1**	–	–	**-0.1**	**2.9**	–	203.9	–

where O_0 is the output the inner model gives at its initial weights and U is the change brought by the update. Both parts are computed inside each block from the same input.

The update cancels the initial-weight output. Table 4 reports the geometry of the two parts, measured per token on 512 validation images and averaged over tokens and blocks. In ViT^3 and in the baseline, O_0 and U have almost the same length and point in nearly opposite directions, with a cosine of -0.44 to -0.53. Adding them cancels much of both, and the output that reaches the residual stream is only about half as long as the two parts laid end to end. A large share of what the update does is to remove the initial-weight output.

O_0 is as long as O on average. Rescaling O to the length of O_0 costs nothing (at most 0.3 points), and rescaling O_0 by a fixed factor per block so that its average length matches O leaves the damage of skipping almost unchanged (45.5 against 46.8 points for ViT^3-T and 57.2 against 58.7 for the baseline, on 2,000 images). The damage lies in the direction of O_0.

Training decides whether the initial-weight output helps. To see how the rest of the network reads O_0, we skip the update and keep only a fraction α of O_0 in every block. At $\alpha = 1$ this is skipping the update, and at $\alpha = 0$ it is removing the branch. Figure 5 in Appendix G traces the damage between the two ends. For the baseline, it falls steadily from 58.7 points at $\alpha = 1$ to 22.6 points at $\alpha = 0$, so every part of the initial-weight output that reaches the residual stream hurts. For η-ViT^3, the curve runs the other way, from 7.3 points at $\alpha = 1$ to 12.9 points at $\alpha = 0$, so the initial-weight output is useful on its own and removing it hurts. The geometry of the two parts changes little between the two models (Table 4). Training with sampled magnitudes changes how the network reads O_0.

A last measurement locates the damage. Following analyses that separate the direct effect of a component from its effect through later layers (Vig et al., 2020; McGrath et al., 2023), we skip the update in every block and pin the outputs of all later branches to their values in the unmodified forward pass. The change then reaches the classifier only along the residual stream, and the remaining damage is the direct effect of O_0. For the baseline, 63.3 of the 76.4 points lost by skipping the update remain, and for ViT^3-S 69.5 of 75.9. The initial-weight output damages the prediction directly, and later layers do not undo it. For η-ViT^3 the direct damage is 4.1 of 12.3 points.

The narrow range of Section 3 comes from the cancellation between the update and the initial-weight output. When training uses a single magnitude, the update has to cancel a harmful initial-weight output at exactly that magnitude, so a smaller step leaves part of it in place and a larger one overshoots. η-ViT^3 makes the initial-weight output useful, so the magnitude of the update can change without breaking the network.

7 Related Work

Test-time training and fast weights. Test-time training was first used to adapt a whole model to a shifted test distribution through a self-supervised task (Sun et al., 2020; Gandelsman et al., 2022),

Table 4: Geometry of the two parts of the TTT output and the direct damage of skipping the update. O_0 is the output at the initial weights, U the change brought by the update and $O = O_0 + U$ the actual output, each measured per token and averaged over 512 validation images and all TTT blocks. The last two columns give the top-1 drop from skipping the update on 2,000 validation images, in total and with all later branches pinned, with the batch-normalization statistics from training.

Model	$\|O\|/\|O_0\|$	$\cos(O_0, U)$	$\|U\|/\|O_0\|$	Skip, total	Skip, direct
ViT^3-T	1.03	-0.53	1.08	–	–
ViT^3-S	1.04	-0.44	0.97	75.9	69.5
Baseline	1.02	-0.53	1.08	76.4	63.3
η-ViT^3	1.13	-0.41	1.06	12.3	4.1

in the same spirit as test-time adaptation methods that update normalization or other parameters on unlabeled test data (Wang et al., 2021; Li et al., 2018; Schneider et al., 2020). TTT layers move this learning inside the forward pass and fit fast weights for every input, continuing the line of fast weight programmers (Schmidhuber, 1992; Ba et al., 2016; Schlag et al., 2021) and linear attention (Katharopoulos et al., 2020), and recent sequence models differ mainly in their inner objective and update rule, from delta-rule updates (Yang et al., 2024; 2025) to gradient steps on neural memories (Sun et al., 2025; Behrouz et al., 2025; 2026a; Zhang et al., 2026). Analyses of these layers treat the update as a mechanism for storing and retrieving associations (Wang et al., 2025; Behrouz et al., 2026b; von Oswald et al., 2023) and study the range of the update coefficient as a question of expressivity (Grazzi et al., 2025). Closest to our work, Liu et al. (2026) show that TTT layers with key-value binding, including those of ViT^3, compute a learned form of linear attention, and they vary the number of inner steps of trained models. We study the magnitude of the single step that vision TTT models take, find its usable range narrow and learned in training, and widen it.

Linear-cost vision backbones. Vision backbones with a cost linear in the number of tokens replace softmax attention (Dosovitskiy et al., 2021; Touvron et al., 2021; Liu et al., 2021) with linear attention (Han et al., 2023; 2024), state-space layers (Zhu et al., 2024; Liu et al., 2024) or TTT layers (Han et al., 2026), and TTT layers also drive long video generation (Dalal et al., 2025) and novel view synthesis (Zhang et al., 2026). Such backbones suit high-resolution inputs, where train and test resolutions often differ (Touvron et al., 2019; Dehghani et al., 2023; Tian et al., 2023). In a TTT backbone the resolution also changes the effective update, which η-ViT^3 tolerates.

Training over a family of settings. Randomizing a structural choice during training is a common way to make one network serve many settings. Dropout, stochastic depth and LayerDrop remove units or layers at random (Srivastava et al., 2014; Huang et al., 2016; Fan et al., 2020), slimmable and once-for-all networks train many widths and depths in one model (Yu et al., 2019; Yu & Huang, 2019; Cai et al., 2020; Kusupati et al., 2022), and FlexiViT samples the patch size (Beyer et al., 2023). Meta-learning methods learn the inner learning rate as a parameter (Finn et al., 2017; Li et al., 2017; Antoniou et al., 2019). η-ViT^3 randomizes the magnitude of a per-input update, which keeps its direction informative and makes the network robust to its length.

8 Conclusion

Vision TTT models such as ViT^3 adapt well only at their trained update magnitude and prefer a different magnitude at every input resolution. This narrow range forms during training, and η-ViT^3 widens it by drawing one update magnitude per forward pass and sharing it across all TTT blocks. η-ViT^3 keeps its accuracy from one eighth to twice the standard magnitude at a small cost at the standard setting, still depends on the direction computed for each image, and keeps this range under shift, resolution change and fine-tuning. The same change widens the range of larger classifiers and of an image generator. How far a vision TTT model can adapt at test time is a property that training sets.

REFERENCES

Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In *ICLR*, 2019. URL https://openreview.net/forum?id=HJGven05Y7.

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *NeurIPS*, pp. 4331–4339, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/9f44e956e3a2b7b5598c625fcc802c36-Abstract.html.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

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. In *ICML*, 2026a. URL https://openreview.net/forum?id=HV1R5bMSpa.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. In *ICLR*, 2026b. URL https://openreview.net/forum?id=gZyEJ2kMow.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *CVPR*, pp. 14496–14506, 2023. doi: 10.1109/CVPR52729.2023.01393.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-All: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. URL https://openreview.net/forum?id=HylxE1HKwS.

Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc Le. RandAugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d85b63ef0ccb114d0a3bb7b7d808028f-Abstract.html.

Karan Dalal, Daniel Koceja, 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*, pp. 17702–17711, 2025. doi: 10.1109/CVPR52734.2025.01649. URL https://openaccess.thecvf.com/content/CVPR2025/html/Dalal_One-Minute_Video_Generation_with_Test-Time_Training_CVPR_2025_paper.html.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M. Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey A. Gritsenko, Mario Lucic, and Neil Houlsby. Patch n' Pack: NaViT, a Vision Transformer for any aspect ratio and resolution. In *NeurIPS*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/06ea400b9b7cfce6428ec27a371632eb-Abstract-Conference.html.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing Transformer depth on demand with structured dropout. In *ICLR*, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-time training with masked autoencoders. In *NeurIPS*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/bcdec1c2d60f94a93b6e36f937aa0530-Abstract-Conference.html.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *ICLR*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *COLM*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *ICLR*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. FLatten Transformer: Vision Transformer using focused linear attention. In *ICCV*, pp. 5938–5948, 2023. doi: 10.1109/ICCV51070.2023.00548.

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. URL https://papers.nips.cc/paper_files/paper/2024/hash/e618724ac897c6cf3fbfb273f8695d67-Abstract-Conference.html.

Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. ViT^3: Unlocking test-time training in vision. In *CVPR*, pp. 51–61, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Han_ViT3_Unlocking_Test-Time_Training_in_Vision_CVPR_2026_paper.html.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE TPAMI*, 44(11):7436–7456, 2022. doi: 10.1109/TPAMI.2021.3117837.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, pp. 8320–8329, 2021a. doi: 10.1109/ICCV48922.2021.00823.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, pp. 15262–15271, 2021b. doi: 10.1109/CVPR46437.2021.01501. URL https://openaccess.thecvf.com/content/CVPR2021/html/Hendrycks_Natural_Adversarial_Examples_CVPR_2021_paper.html.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, pp. 6626–6637, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

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. *arXiv preprint arXiv:1704.04861*, 2017. URL https://arxiv.org/abs/1704.04861.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pp. 646–661, 2016. doi: 10.1007/978-3-319-46493-0_39.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015. URL https://proceedings.mlr.press/v37/ioffe15.html.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive Transformers with linear attention. In *ICML*, pp. 5156–5165, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham M. Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *NeurIPS*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html.

Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018. doi: 10.1016/j.patcog.2018.03.005.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017. URL https://arxiv.org/abs/1707.09835.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with KV binding is secretly linear attention. In *ICML*, 2026. URL https://openreview.net/forum?id=MpBeXMurIb.

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. URL https://papers.nips.cc/paper_files/paper/2024/hash/baa2da9ae4bfed26520bb61d259a3653-Abstract-Conference.html.

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*, pp. 9992–10002, 2021. doi: 10.1109/ICCV48922.2021.00986.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Thomas McGrath, Matthew Rahtz, János Kramár, Vladimir Mikulik, and Shane Legg. The Hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023. URL https://arxiv.org/abs/2307.15771.

MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

William Peebles and Saining Xie. Scalable diffusion models with Transformers. In *ICCV*, pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive Transformers for long-range sequence modelling. In *ICLR*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers are secretly fast weight programmers. In *ICML*, pp. 9355–9366, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/85690f81aadc1749175c187784afc9ee-Abstract.html.

Noam Shazeer. GLU variants improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. URL https://dl.acm.org/doi/10.5555/2627435.2670313.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, pp. 9229–9248, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *CVPR*, pp. 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

Rui Tian, Zuxuan Wu, Qi Dai, Han Hu, Yu Qiao, and Yu-Gang Jiang. ResFormer: Scaling ViTs with multi-resolution training. In *CVPR*, pp. 22721–22731, 2023. doi: 10.1109/CVPR52729.2023.02176.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, pp. 776–794, 2020. doi: 10.1007/978-3-030-58621-8_45.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *NeurIPS*, pp. 8250–8260, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d03a857a23b5285736c4d55e0bb067c8-Abstract.html.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pp. 10347–10357, 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

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*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Investigating gender bias in language models using causal mediation analysis. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html.

Johannes von Oswald, Eyvind Niklasson, Maximilian Schlegel, Seijin Kobayashi, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Blaise Agüera y Arcas, Max Vladymyrov, Razvan Pascanu, and João Sacramento. Uncovering mesa-optimization algorithms in Transformers. *arXiv preprint arXiv:2309.05858*, 2023. URL https://arxiv.org/abs/2309.05858.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025. URL https://arxiv.org/abs/2501.12352.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pp. 432–448, 2018. doi: 10.1007/978-3-030-01228-1_26.

Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. CondConv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, pp. 1305–1316, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f2201f5191c4e92cc5af043eebfd0946-Abstract.html.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear Transformers with the delta rule over sequence length. In *NeurIPS*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/d13a3eae72366e61dfdc7eea82eeb685-Abstract-Conference.html.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta rule. In *ICLR*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, pp. 1803–1811, 2019. doi: 10.1109/ICCV.2019.00189.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *ICLR*, 2019. URL https://openreview.net/forum?id=H1gMCsAqY7.

Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pp. 6022–6031, 2019. doi: 10.1109/ICCV.2019.00612.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

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. URL https://openreview.net/forum?id=Tb9qAxT3xv.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, pp. 13001–13008, 2020. doi: 10.1609/aaai.v34i07.7000.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, pp. 5122–5130, 2017. doi: 10.1109/CVPR.2017.544.

Lianghui 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*, pp. 62429–62442, 2024. URL https://proceedings.mlr.press/v235/zhu24f.html.

A Evaluation details

Evaluation subsets. ImageNet results use a fixed random subset of 5,000 of the 50,000 validation images, drawn once with a fixed seed and shared by all models and all changes. The only exception is the top-1 column of Table 2, which is measured on the full validation set. ImageNet-C uses 2,000 images per corruption and severity, and ImageNet-R and ImageNet-A use 5,000 images each. ADE20K results use 2,000 validation images, except for the ViT^3-S and ViT^3-B segmentation models in Table 1, which use 500.

Re-estimating batch-normalization statistics. The ViT^3 classifiers contain batch normalization in the convolutional stem, in the downsampling layers between stages and before the classifier. A change to the update of the TTT layers changes the activations these layers receive. After every change, we pass training images through the changed network in batches of 128 with the validation preprocessing, record the mean and variance of the input of every batch-normalization layer for each batch, and set its statistics to the median over batches. We use 10,240 training images for the results in Tables 1, 2 and 3 and in Figure 1a, and 2,048 images for the resolution, decomposition and

multi-step results. The median keeps a few batches with extreme statistics under large changes from dominating the estimate. The original statistics are restored before the next change. Distribution-shift results and the direct-effect measurement in Table 4 keep the statistics from training.

Table 5 repeats the main ImageNet measurements with the statistics from training. Every ordering discussed in the main text holds without re-estimation. ViT^3-B is the one model whose accuracy at $\eta = 2$ depends on the statistics used, 5.4 points lower with the training statistics and at chance after re-estimation.

Table 5: Drops in ImageNet top-1 accuracy on the evaluation subset when the batch-normalization statistics from training are kept, for comparison with Tables 1, 2 and 3.

Model	Top-1	$\eta = 0$	1/8	1/4	1/2	2	4	Remove	Random	Other image	Batch mean
ViT^3-T	83.8	73.2	30.8	10.6	1.9	7.3	83.7	20.0	78.5	51.9	23.7
ViT^3-S	84.8	74.8	29.2	8.9	1.9	4.1	84.7	15.7	79.0	41.1	20.8
ViT^3-B	85.2	67.9	23.3	7.9	1.6	5.4	85.1	17.6	73.1	32.7	18.1
Baseline-T	82.3	76.8	23.6	8.9	1.6	4.8	82.2	66.2	80.6	63.6	33.0
η-ViT^3-T	82.5	12.7	1.1	0.6	0.1	1.2	23.3	33.8	82.3	76.2	23.8
Baseline-S	83.1	74.3	27.3	9.4	1.8	6.3	82.7	26.7	77.3	40.4	19.4
η-ViT^3-S	83.0	6.9	0.9	0.5	0.2	0.9	45.8	11.6	81.4	64.7	12.9
Baseline-B	83.9	69.9	22.1	7.3	1.5	4.8	83.7	23.3	74.8	34.7	18.3
η-ViT^3-B	83.3	6.9	0.0	0.0	0.0	0.4	21.2	9.2	82.4	59.0	11.0

Generation and language models. For DiT^3-B/2 we draw 5,000 class-conditional samples at 256×256 with 250 sampling steps and no classifier-free guidance, using the same class labels and noise for every change, and compute FID against 5,000 ImageNet validation images (Heusel et al., 2017). For TTT-Linear and TTT-MLP we evaluate the negative log-likelihood per token on 32 books from the PG19 validation split with 2,048-token contexts. Neither model contains batch normalization. The two generators of Table 3 are evaluated with the exponential moving average of their weights under the same protocol, with the same labels, noise and reference images.

B Training details

The two ViT^3-T models on ImageNet-1k use the ViT^3 code for the tiny hierarchical model with the following recipe. Training runs for 300 epochs at 224×224 with AdamW (Loshchilov & Hutter, 2019) and a cosine schedule with 20 warm-up epochs, weight decay 0.05 and gradient clipping at 5.0, a total batch size of 512 with the learning rate set to 5×10^{-4} by the linear scaling rule of the ViT^3 code, mixed precision, RandAugment, Mixup 0.8, CutMix 1.0, random erasing 0.25, label smoothing 0.1 and stochastic depth 0.2 (Touvron et al., 2021; Cubuk et al., 2020; Zhang et al., 2018; Yun et al., 2019; Zhong et al., 2020; Szegedy et al., 2016). The only difference is the update magnitude used in training, fixed at $\eta = 1$ for the baseline and drawn from $\mathcal{U}(0, 2)$ once per forward pass for η-ViT^3. The ImageNet-100 models of Section 4 use the same recipe for 100 epochs on the 100-class subset of Tian et al. (2020). The ADE20K models follow the ViT^3 segmentation recipe with an UperNet head for 160k iterations at 512×512 crops.

The ViT^3-S and ViT^3-B pairs of Table 3 use the same recipe for 100 epochs with 7 warm-up epochs and the stochastic depth rates of the ViT^3 configurations for these sizes, 0.4 and 0.6. The DiT^3-B/2 pair follows the DiT^3 training recipe, 400k steps at 256×256 resolution with a global batch size of 256 and AdamW at a learning rate of 10^{-4} (Peebles & Xie, 2023; Han et al., 2026). The generator with sampled magnitudes draws one $\eta \sim \mathcal{U}(0, 2)$ per training step and shares it across all TTT blocks.

C Choosing the training interval

Table 6 lists the ImageNet-100 models used in Section 4. All are ViT^3-T trained for 100 epochs with the same recipe and evaluated on the 5,000 ImageNet-100 validation images with re-estimated statistics.

Table 6: Training intervals for the update magnitude on ImageNet-100. Entries are drops in top-1 accuracy from removing the TTT branch, skipping the update, replacing it with a random direction of the same length, and changing its magnitude at test time. The last row keeps $\eta = 1$ but skips the update in a random 5% of the training passes.

Magnitude in training	Remove	Skip	Random dir.	$\eta = 1/2$	$\eta = 2$
$\eta = 1$ (baseline)	7.72	9.36	10.26	1.18	8.70
$\eta \sim \mathcal{U}(0.25, 1.75)$	5.56	11.60	25.52	0.32	1.08
$\eta \sim \mathcal{U}(0.25, 2)$	4.80	10.18	20.92	0.06	0.40
$\eta \sim \mathcal{U}(0, 1.75)$	3.22	2.06	10.10	0.30	-0.06
$\eta \sim \mathcal{U}(0, 2)$ (η-ViT^3)	3.34	2.52	8.98	-0.08	0.22
$\eta = 1$, 5% of passes skipped	3.54	2.26	2.66	0.32	7.02

D DIRECTION REPLACEMENTS ACROSS MODEL SIZES

A flat accuracy curve could also arise if the network had learned to ignore its TTT layers. The direction replacements in Tables 2 and 7 rule this out. Replacing every update of η-ViT^3 with a random direction of the same length costs 81.7 points, which is chance level and more than six times the cost of removing the branch. Using the update computed on another image costs 79.0 points, and using the batch-averaged update costs 20.2 points. In all three model sizes, every replacement costs η-ViT^3 more than removing its branch. Removing the branch measures how much a network uses its TTT layers at all, and dividing the damage of a replacement by it measures how much of that use rests on the direction computed for each image. By this measure η-ViT^3 relies more on the per-image direction than its baseline for every replacement and every size (Table 7), with ratios of 6.5, 6.3 and 1.6 against 3.8, 2.9 and 1.3 for the tiny models. η-ViT^3 keeps the direction computed for each image and has learned to tolerate changes in the length of that step.

Table 7 lists these drops for all three model sizes, together with each replacement divided by the damage of removing the TTT branch.

Table 7: Damage of each direction replacement and its ratio to the damage of removing the TTT branch, as drops in ImageNet top-1 accuracy on the evaluation subset with re-estimated statistics, under the protocol of Tables 2 and 3. Remove is the damage of removing the branch. Higher ratios mean that more of what the network draws from its TTT branch depends on the direction computed for each image.

			Drop from the replacement			Ratio to removing the branch		
Size	Model	Remove	Random	Other image	Batch mean	Random	Other image	Batch mean
T	Baseline	20.7	78.3	59.1	26.8	3.8	2.9	1.3
	η-ViT^3	12.5	81.7	79.0	20.2	**6.5**	**6.3**	**1.6**
S	Baseline	13.1	51.2	31.7	17.9	3.9	2.4	1.4
	η-ViT^3	8.2	81.3	68.3	12.1	**9.9**	**8.3**	**1.5**
B	Baseline	12.6	47.6	27.9	15.2	3.8	2.2	1.2
	η-ViT^3	7.2	81.8	67.5	11.9	**11.4**	**9.4**	**1.7**

E A SECOND INNER STEP

We also run more than one inner step at inference. To take K inner steps at inference, every TTT block repeats Eq. (3) K times from W_0, each time recomputing the gradient of the inner loss at the current weights, normalizing it and stepping with the trained magnitude $\eta = 1$. Table 8 reports ImageNet top-1 accuracy on the evaluation subset with re-estimated statistics. With a second step the baseline falls to chance, while η-ViT^3 still classifies 38.8% of the images correctly.

Table 8: ImageNet top-1 accuracy with K inner steps per TTT block at inference. $K = 1$ is the standard model and $K = 0$ skips the update.

Model	$K = 0$	$K = 1$	$K = 2$
Baseline	22.7	81.4	0.1
η-ViT^3	73.2	81.7	38.8

F THE RANGE UNDER DISTRIBUTION SHIFT AND FINE-TUNING

Distribution shift leaves the range of η-ViT^3 in place (Table 9, left). Averaged over the ImageNet-C corruptions, its accuracy varies by less than 2 points between $\eta = 1/2$ and $\eta = 2$, and it still reaches 46.3% at $\eta = 3$, where the baseline is at 4.6%. ImageNet-R and ImageNet-A show the same pattern. The usable range learned on clean training images carries over to inputs the model has never seen.

Finally, the range survives full downstream fine-tuning (Table 9, right). After 160k iterations on ADE20K with η fixed at 1, both segmentation models reach the same mIoU, 44.15 for η-ViT^3 and 44.13 for the baseline. Between a quarter and twice the standard magnitude, the η-ViT^3 model loses 0.09 to 1.58 mIoU and the baseline model 2.74 to 6.92. At four times the magnitude the baseline model falls to 0.83 mIoU while the η-ViT^3 model keeps 27.73. Replacing the update with a random direction still costs the η-ViT^3 model 42.9 mIoU, three times the cost of removing its branch. At input sizes of 384, 512 and 640 pixels, the η-ViT^3 model varies by at most 1.91 mIoU over the same span of magnitudes, against 6.92 to 9.51 for the baseline model.

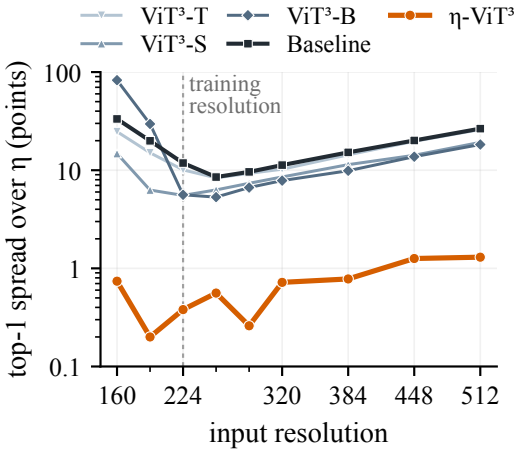

Figure 4: Spread of ImageNet accuracy over $\eta \in [1/4, 2]$ at each input resolution. For ViT^3 and the baseline it is smallest near the training resolution and grows toward both ends. η-ViT^3 stays within 1.3 points at every resolution.

Table 9: The range of η-ViT^3 under distribution shift and after downstream fine-tuning. Left, ImageNet-C (mean over 15 corruptions at severities 3 and 5), ImageNet-R and ImageNet-A top-1 accuracy at each update magnitude, with the batch-normalization statistics from training. Right, drop in ADE20K mIoU after fine-tuning each backbone for 160k iterations at $\eta = 1$, and the spread of mIoU over $\eta \in [1/4, 2]$ at three input sizes.

		Top-1 at update magnitude η				
Data	Model	1/2	1	3/2	2	3
IN-C	Baseline	50.2	54.9	48.6	39.3	4.6
	η-ViT^3	55.3	55.8	55.5	54.1	46.3
IN-R	Baseline	43.1	47.7	46.7	42.0	7.5
	η-ViT^3	47.1	47.9	47.4	46.1	41.4
IN-A	Baseline	20.5	33.8	35.1	28.8	3.5
	η-ViT^3	32.5	35.4	36.5	36.8	33.1

ADE20K	Baseline	η-ViT^3
mIoU, unmodified	44.13	44.15
Drop at $\eta = 1/4$	6.92	**1.58**
Drop at $\eta = 1/2$	2.74	**0.09**
Drop at $\eta = 2$	5.21	**1.22**
Drop at $\eta = 4$	43.30	**16.42**
Drop, remove branch	18.55	14.22
Drop, random direction	37.05	42.88
Spread at 384 px	9.51	**1.74**
Spread at 512 px	6.92	**1.68**
Spread at 640 px	8.03	**1.91**

G KEEPING A FRACTION OF THE INITIAL-WEIGHT OUTPUT

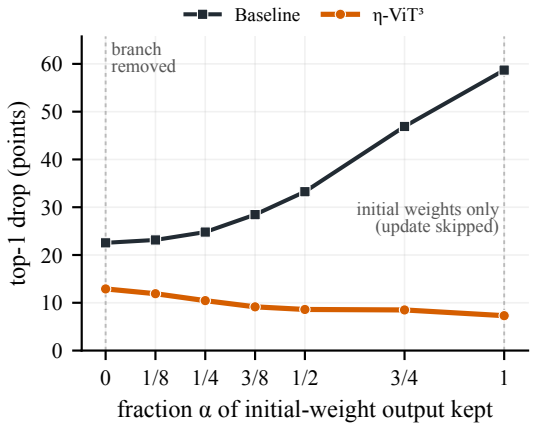

Figure 5: Damage of keeping only a fraction α of the initial-weight output O_0 in every block. For the baseline every bit of O_0 hurts. For η-ViT^3, O_0 is useful on its own.

H ACCURACY SPREAD ACROSS INPUT RESOLUTIONS

Table 10 lists the values plotted in Figure 4, the difference between the best and the worst ImageNet top-1 accuracy over $\eta \in \{1/4, 1/2, 3/4, 1, 5/4, 3/2, 2\}$ at each input resolution, on 5,000 validation images with re-estimated statistics.

Table 10: Spread of ImageNet top-1 accuracy over update magnitudes from 1/4 to 2 at each input resolution.

Model	160	192	224	256	288	320	384	448	512
ViT^3-T	24.72	15.06	10.08	8.36	9.28	10.24	14.34	19.70	25.96
ViT^3-S	14.78	6.28	5.54	6.30	7.36	8.54	11.40	14.20	19.24
ViT^3-B	82.68	29.62	5.62	5.32	6.66	7.84	9.88	13.74	18.28
Baseline	33.32	19.94	11.84	8.52	9.62	11.28	15.20	20.10	26.54
η-ViT^3	0.74	0.20	0.38	0.56	0.26	0.72	0.78	1.26	1.30

I REPRODUCIBILITY STATEMENT

Section 4 and Eq. (4) specify the only change η-ViT^3 makes to the ViT^3 training code. Appendix B lists the full training recipe, and Appendix A describes the evaluation subsets, the re-estimation of batch-normalization statistics and the protocols for generation and language models. All test-time changes of Section 2 act on the weights of trained models without retraining. We will release the training code, the evaluation code and the trained checkpoints.

J USAGE OF AI ASSISTANTS

AI assistants served as a writing aid for polishing the text and as a search aid for identifying related work. Each edit was reviewed and each reference verified by us before it entered the paper, and we remain accountable for everything the paper contains.