Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

TTT-TP: Regularized Target Propagation for Multi-Layer Test-Time Learners

Dian Yu

dian.yu2@student.unsw.edu.au

UNSW Sydney

Research Article

Keywords: test-time training, fast weights, target propagation, credit assignment, recurrent state, online learning

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11159991/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

TTT-TP: Regularized Target Propagation for Multi-Layer Test-Time Learners

Dian Yu[1]

[1]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia
dian.yu2@student.unsw.edu.au

Abstract

Test-Time Training (TTT) layers use an online learner as recurrent state, so the learner's update rule is itself part of the sequence architecture. In TTT-MLP, the output fast layer has an explicit reconstruction target, whereas the lower fast layer has only a backpropagated gradient. We remove this asymmetry. TTT-TP maps the desired output displacement to the minimum-norm hidden displacement that approximately realizes it, through a Tikhonov-regularized inverse of the current upper fast weight, giving every fast layer an explicit local target. It retains the TTT-MLP forward architecture, upper-layer update, learned inner learning rate, parameter count, and recurrent-state size, and adds no feedback network. A transpose control recovers official TTT-MLP outputs and all fast states to numerical precision, isolating the proposed inverse as the change in lower-layer credit. On held-out episodic moving-hinge regression, TTT-TP reduces mean normalized query error from 0.4044 to 0.1899 across three seeds. On Penn Treebank and WikiText-2, compact two-layer language models match TTT-MLP test perplexity within 0.06%; the unfused reference reaches 82.6–88.7% of baseline throughput and uses 2.7% more peak allocated memory. The target interface also turns each layer's online update into an explicit erase–write Delta decomposition, restoring at the intermediate layer the same local form the output layer has always admitted. Unlike regularized target propagation used as a training algorithm, the inverse here is taken of a fast weight that is itself part of the recurrent state, and is recomputed as the sequence is read.

Keywords: test-time training · fast weights · target propagation · credit assignment · recurrent state · online learning

1 Introduction

Test-Time Training layers reinterpret recurrent state as a model that learns from the current sequence [Sun et al., 2025]. This view admits a spectrum of fast learners: a linear map yields TTT-Linear, while a two-layer GELU network yields the more expressive TTT-MLP. Once the recurrent state becomes multi-layer, however, an important asymmetry appears. The output fast layer is told what activation it should produce; the intermediate fast layer is told only how the final loss changes under an infinitesimal perturbation.

For ordinary offline optimization, the chain-rule gradient is sufficient. For a TTT layer, the distinction is architectural: the credit rule is executed online for every sequence and directly governs recurrent-state evolution. The output layer has an immediate residual and therefore an unambiguous local correction. The lower fast layer receives that residual only after multiplication by the transpose Jacobian of the current upper layer. It has no explicit statement of which hidden content should replace the content just retrieved. Consequently, retaining old adaptation, erasing retrieved content, or writing target content can be stated directly for the output layer, but not for the intermediate layer.

We ask whether a multi-layer TTT learner can preserve TTT-MLP's nonlinear fast state while giving every fast layer a concrete target. Our answer is TTT-TP. The current upper fast weight defines a regularized inverse that maps the desired output displacement to the minimum-norm hidden displacement that approximately produces it, with the ridge trading fidelity of the output correction against the size of the displacement.

Each layer then adapts toward its own target by a local squared loss. The upper-layer update is exactly the official TTT update; only lower-layer credit assignment changes. Two differences from prior work on regularized target propagation should be stated at the outset. First, Roulet and Harchaoui [2023] propose it as a *training algorithm*, a substitute for backpropagation through time motivated by vanishing and exploding gradients over long sequences. Here it is not a training algorithm: outer meta-training still uses ordinary backpropagation, and target propagation instead becomes part of the architecture, namely the rule by which the recurrent state updates itself while a sequence is read. Second, and consequently, the inverse is taken of a different object. As a training algorithm it inverts a fixed network layer; here it inverts a fast weight that is itself part of the recurrent state, so it is recomputed at every chunk from the learner that exists at that point in the sequence. To our knowledge this is the first use of target propagation inside a test-time learner.

Our contribution is a controlled change to the semantics of a multi-layer fast state. We establish three points:

1. A regularized-inverse hidden target can be inserted into the official TTT-MLP cell without changing its forward architecture, slow parameters, fast-state size, or upper-layer online update. The target is the solution to a well-defined local output-correction problem, not a learned auxiliary signal.

2. The target interface gives both fast layers local residuals and therefore an exact erase–write Delta decomposition. This makes structured control of intermediate fast state well-defined rather than an analogy imported from a single-layer memory.

3. The interface admits measurements that a chained gradient does not define. We report how much of the hidden target the update actually attains, how the transported direction relates to the transpose rule as a function of the ridge, how informative the lower-layer residual is, and how tightly the two fast layers are coupled. These delimit the design space the interface opens, by measurement rather than by assumption.

4. Correctness audits isolate the changed credit rule. It produces a strong, reproducible separation on a held-out episodic task that requires learning a new nonlinear function from each context, while preserving perplexity on two real text corpora at matched parameter and state budgets.

An LLM-based assistant was used for language editing, LaTeX preparation, and code refactoring; the author designed the method and experiments, verified the results, and takes responsibility for the manuscript.[1]

2 Background

2.1 TTT-MLP as recurrent fast weights

Consider one TTT head and row-vector notation. For a key-like training input $k \in \mathbb{R}^d$, the fast learner is

$$z_1 = kW_1 + b_1, \qquad h = \phi(z_1), \qquad z_2 = hW_2 + b_2, \tag{1}$$

where $W_1 \in \mathbb{R}^{d\times m}$, $W_2 \in \mathbb{R}^{m\times d}$, $m = 4d$, and ϕ is GELU [Hendrycks and Gimpel, 2016]. TTT constructs a self-supervised reconstruction target from the current token and differentiates a LayerNorm-based loss through the fast learner [Ba et al., 2016, Sun et al., 2025]. A learned token-dependent inner learning rate scales the update. The adapted learner is then queried by a query projection q.

Let $g_2 = \partial \mathcal{L}_{\mathrm{TTT}}/\partial z_2$. Official TTT-MLP transports this signal to the lower layer as

$$g_1^{\mathrm{BP}} = (g_2 W_2^\top) \odot \phi'(z_1). \tag{2}$$

This is an exact gradient for the current fast network. It also makes the lower update inseparable from the current upper-layer transpose Jacobian.

[1] The assistant does not meet authorship criteria and is not listed as an author.

Table 1: Matched components in the evaluated implementation.

Component	TTT-MLP	TTT-TP
Fast learner	Two-layer GELU MLP	Same
Upper update	Reconstruction gradient	Exactly the same
Lower transport	$g_2 W_2^\top$	$g_2 B_r$
Q/K/V, RoPE, convolution	Official cell	Same
Learned inner learning rate	Yes	Same
Added learned parameters	0	0
Added recurrent state	0	0

2.2 Target propagation and regularized inversion

Target propagation assigns desired activations rather than directly transmitting gradients [Lee et al., 2015, Meulemans et al., 2020]. Regularized target propagation uses analytic layer inverses to avoid learning a separate feedback network and to control poorly conditioned directions [Roulet and Harchaoui, 2023]. For a linear map with current weight W_2, the row-oriented Tikhonov inverse is

$$B_r = (W_2^\top W_2 + rI)^{-1} W_2^\top, \tag{3}$$

where $r > 0$. Since W_2 maps an m-dimensional hidden representation to a d-dimensional output, Equation 3 requires a $d \times d$ solve rather than an $m \times m$ solve.

3 TTT-TP

3.1 Explicit targets for both fast layers

TTT-TPretains Equation 1, the original reconstruction loss, and the learned inner learning rate. It constructs an output target and a hidden target:

$$z_2^* = z_2 - g_2, \tag{4}$$

$$h^* = h - g_2 B_r. \tag{5}$$

The corresponding lower-layer signal is

$$g_1^{\mathrm{TP}} = (g_2 B_r) \odot \phi'(z_1). \tag{6}$$

Both targets are evaluated from the same pre-update fast-state snapshot. The online optimizer applies the two local corrections using the existing TTT learning-rate schedule. The target step is one in all experiments.

Why the inverse is not merely another notation for backpropagation. Write the singular-value decomposition of the upper fast weight as $W_2 = U\Sigma V^\top$. Transpose transport weights a singular direction by σ_i, whereas the regularized inverse weights it by $\sigma_i/(\sigma_i^2 + r)$. The two rules therefore answer different questions. The transpose asks how a hidden perturbation changes the output loss; the inverse asks for a small hidden displacement whose forward image realizes the desired output displacement. The ridge suppresses directions that the current upper layer cannot reliably invert. Because W_2 is part of the recurrent fast state, this inverse is recomputed from the learner that exists at that point in the sequence. Outer training can consequently shape both the representation and the online credit geometry.

The method adds neither a trainable feedback pathway nor a recurrent tensor. The regularizer r is a scalar hyperparameter. Table 1 summarizes the controlled change.

3.2 Local generalized Delta form

The output-layer local loss is $\ell_2 = \frac{1}{2}\|z_2 - z_2^*\|^2$. Its negative weight gradient is

$$-\nabla_{W_2} \ell_2 = h^\top (z_2^* - z_2), \tag{7}$$

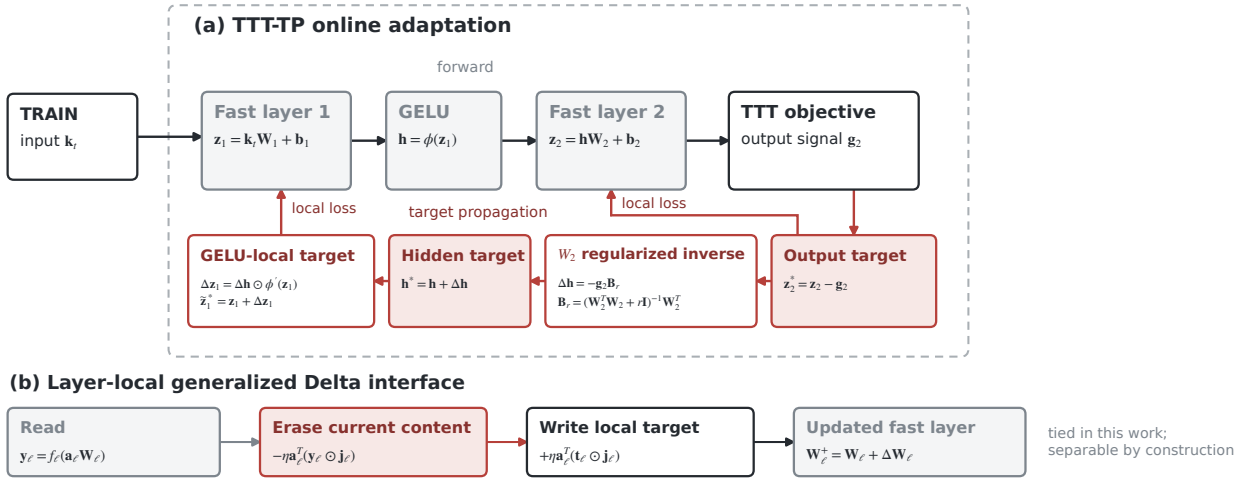

Figure 1: **TTT-TP replaces inter-layer backpropagation with an explicit regularized-inverse target while retaining the TTT-MLP fast learner.** (a) The forward learner remains a two-layer GELU MLP. The current upper fast state maps the output displacement to a post-activation hidden target; its residual is then transported locally through the GELU derivative to obtain an effective preactivation target. No inverse of GELU is used. (b) Each local target exposes a generalized Delta update as an erase term for currently retrieved content and a write term for target content. The evaluated model keeps their address and strength tied; independent routing is a future direction.

the classical residual outer-product Delta update. The lower local loss $\ell_1 = \frac{1}{2}\|h - h^*\|^2$ gives

$$-\nabla_{W_1}\ell_1 = k^\top\left[(h^* - h) \odot \phi'(z_1)\right]. \tag{8}$$

Thus regularized target propagation determines the layer-local targets, and generalized Delta corrections realize them. This does not make TTT-TP algebraically identical to DeltaNet. DeltaNet stores one linear associative matrix and updates it with a residual rule [Schlag et al., 2021, Yang et al., 2024]; TTT-TP retains a nonlinear two-layer TTT state.

The erase–write interpretation can be stated exactly for the updates that are executed. Augmenting an address with a constant coordinate absorbs the bias into W. For the upper layer, the address is h and the target is z_2^*. For the lower layer, define the effective preactivation target

$$\tilde{z}_1^* = z_1 + (h^* - h) \odot \phi'(z_1). \tag{9}$$

Both weight updates then share the layer-local form

$$W^+ = W + \beta a^\top(t - aW) = \underbrace{W - \beta a^\top(aW)}_{\text{old state minus retrieved content}} + \underbrace{\beta a^\top t}_{\text{write target content}}. \tag{10}$$

This is more than a verbal analogy: the two terms sum to the exact local gradient step used by the cell. Standard TTT-MLP exposes this target residual at the output layer only. TTT-TP exposes it at the intermediate layer as well, so retention strength, erase address, write address, or write target can in principle be controlled separately at every fast layer. The evaluated model keeps them tied exactly as in Equation 10; learning independent controls is a future direction analogous in spirit to the design space opened by gated Delta rules [Yang et al., 2025].

3.3 One enabled extension: anchor-relative retention

Fast TTT weights are initialized from offline-trained slow weights W_ℓ^0. A principled future retention rule should decay only sequence-specific displacement $F_{\ell,t} = W_{\ell,t} - W_\ell^0$:

$$F_{\ell,t} = \alpha_{\ell,t}F_{\ell,t-1} - \beta_{\ell,t}G_{\ell,t}^{\mathrm{TP}}, \qquad W_{\ell,t} = W_\ell^0 + F_{\ell,t}. \tag{11}$$

The present method fixes $\alpha = 1$ and uses the existing TTT inner-rate mechanism for β. Equation 11 illustrates the design freedom created by layer-local targets; it is not an evaluated component or an empirical claim.

4 Experimental Design

We test three questions. First, is the implementation matched to official TTT-MLP and is the inverse oriented correctly? Second, does changing only lower-layer credit produce a measurable difference when a context defines a new nonlinear function? Third, does the method preserve language-model quality on natural text at a matched budget? We fix $r = 0.3$ and the target step to one for every reported task and corpus rather than selecting them per dataset.

4.1 Correctness audit

The audit uses four checks. A transpose-bridge control replaces B_r with $W_2^\top$ and must recover official TTT-MLP. With target step one, the W_2, b_2 updates of TTT-TP must equal official TTT-MLP. The inexpensive dual solve in Equation 3 must match the explicit $m \times m$ primal solve and satisfy the Tikhonov normal equations. Finally, the analytic lower-layer signal must match automatic differentiation of the same hidden-target loss. We also verify equal initialization, parameter count, fast-state size, finite gradients, and causal prefixes.

4.2 Held-out moving-hinge regression

Each episode samples a new teacher. Inputs $x \in \mathbb{R}^8$ are mapped by

$$y = [\mathrm{ReLU}\,(1.6(xA - c))]\,C, \tag{12}$$

where $A \in \mathbb{R}^{8\times16}$, $C \in \mathbb{R}^{16\times4}$, and hinge centers c are independently resampled for every episode. The context contains 64 labeled support pairs followed by 16 unseen query inputs whose targets are hidden from the model. Outer training and evaluation use disjoint random streams. The locked test contains 32 fresh batches per seed. Shuffling support values across episodes while keeping inputs fixed tests whether predictions use the current episode rather than slow-weight priors.

All arms use width 64, two heads, chunk size 16, batch size 8, 600 AdamW steps [Loshchilov and Hutter, 2019], identical outer architecture, and identical data streams. Each has 57,750 trainable parameters and 16,704 fast-state scalars. We report normalized query MSE, defined as squared prediction error divided by target energy, over seeds 0, 1, and 2.

4.3 Language modeling

We train compact causal language models from scratch on the raw WikiText-2 corpus [Merity et al., 2017] and the standard Zaremba-preprocessed Penn Treebank splits [Zaremba et al., 2015]. The same frozen tokenizer is used for both corpora. Each model has two TTT-MLP layers, width 128, intermediate width 344, four heads, sequence length 256, and chunk size 16, totaling 4,564,904 parameters and 66,816 fast-state scalars. We train for 700 steps with batch size 16, AdamW learning rate 3×10^{-4}, cosine decay after 10% warmup, and weight decay 0.1. This corresponds to 2.856 million training tokens per run. Evaluation uses 64 deterministic windows and three seeds.

The language experiments are parity tests rather than scale claims. The official TTT reference is written in ordinary PyTorch, and our regularized inverse is not fused. Throughput excludes ten real warm-up steps and evaluation time; peak memory is maximum allocated CUDA memory.

5 Results

5.1 Implementation parity and episodic adaptation

The transpose control and official TTT-MLP differ by approximately 10^{-7} in locked-test NMSE across independently trained seeds. At the operator level, outputs and all four fast states match within floating-point

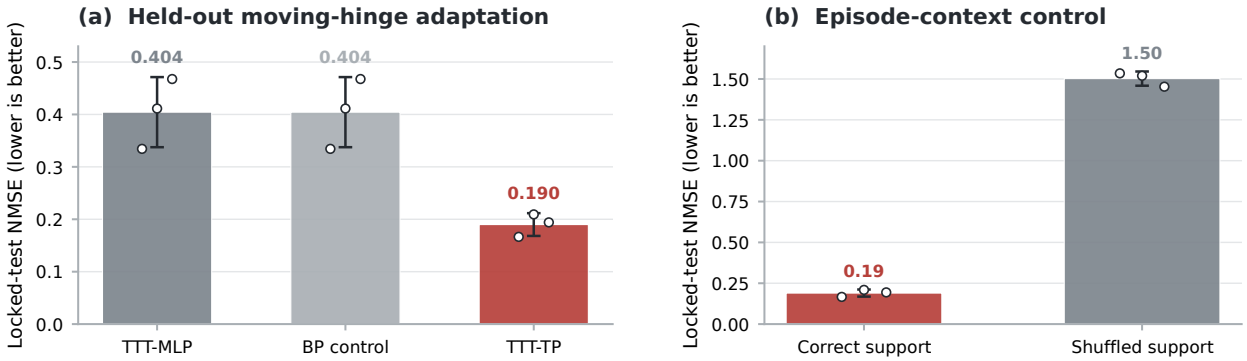

Figure 2: **TTT-TP improves held-out moving-hinge regression and relies on episode context.** (a) Locked-test NMSE across three independently trained seeds. Bars show means, error bars show sample standard deviations, and points show seeds. The BP control reproduces TTT-MLP. (b) Shuffling support values across episodes destroys TTT-TP performance, confirming reliance on current context.

Table 2: Locked-test moving-hinge NMSE. Mean and sample standard deviation are over three outer-training seeds. Lower is better.

Method	Seed 0	Seed 1	Seed 2	Mean	SD
TTT-MLP	0.3343	0.4112	0.4676	0.4044	0.0669
BP control	0.3343	0.4112	0.4676	0.4044	0.0669
TTT-TP	**0.1664**	**0.2093**	**0.1941**	**0.1899**	0.0217

tolerance. From the same pre-update snapshot, the one-step upper-state correction of TTT-TP also matches official TTT-MLP when the target step is one. Subsequent upper states may diverge because the lower state, and hence later hidden activations, has changed. These checks isolate the regularized inverse as the source of the changed recurrent dynamics.

Table 2 and Figure 2 show the main controlled result. TTT-TP reduces mean locked-test NMSE from 0.4044 to 0.1899, a relative reduction of 53.0%. All three TTT-TP seeds are below all three baseline seeds. Shuffling support raises TTT-TP NMSE to 1.5024 ± 0.0435, so the improvement cannot be explained by the slow model fitting an average teacher.

The result establishes more than numerical non-equivalence. After outer meta-training, the two credit rules induce measurably different online algorithms under an otherwise matched TTT architecture, and the target-based rule is substantially better when each context defines a new nonlinear function. It does not imply universal superiority across tasks.

5.2 Language-model parity and reference cost

Table 3 reports test perplexity. The seed-paired differences are small and do not consistently favor one rule. Mean TTT-TP perplexity is 0.11 lower on Penn Treebank and 0.14 lower on WikiText-2, corresponding to less than 0.06% of baseline perplexity. We therefore claim parity rather than an accuracy gain.

The cost result identifies the main engineering limitation. The current implementation calls a dense solve inside an unfused Python-level reference. It is not a competitive kernel. The recurrent state itself is unchanged; the measured 2.7% memory overhead comes from temporary computation. A fused solve or cached factorization within a chunk is outside the scope of this study.

6 Analysis of the target interface

Official TTT-MLP gives the lower fast layer a backpropagated gradient. One cannot ask of a gradient how far away the target is, or whether the update arrived. With an explicit target these questions have answers,

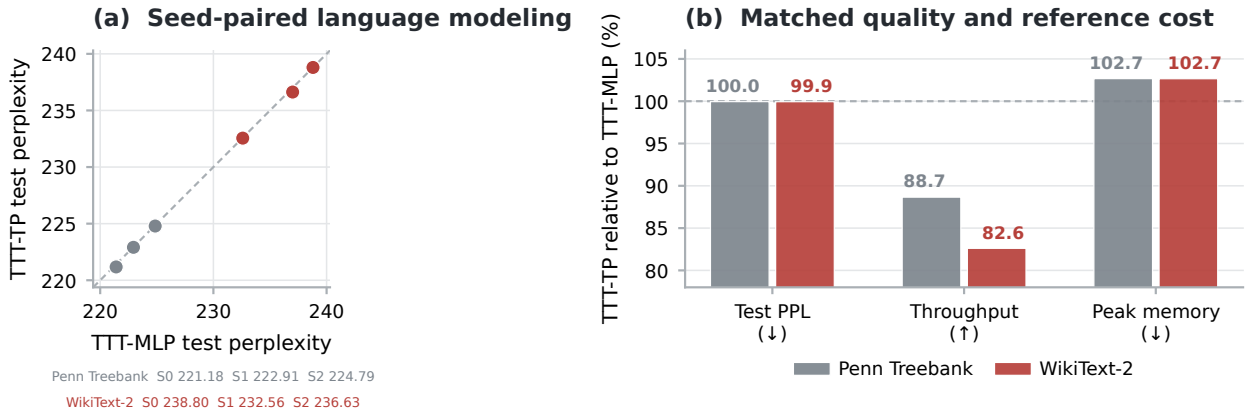

Figure 3: **TTT-TP preserves language-model quality at a matched parameter and fast-state budget.** (a) Seed-paired test perplexity on Penn Treebank and WikiText-2; the two colored rows list TTT-TP values for seeds S0–S2 and the dashed diagonal denotes equality. (b) TTT-TP metrics relative to TTT-MLP. Perplexity is matched within 0.06% on both corpora. The unfused reference reaches 82.6–88.7% of baseline throughput and uses approximately 2.7% more peak allocated memory.

Table 3: Test perplexity and implementation cost. Mean and sample standard deviation are over three seeds. Throughput and memory are ratios of three-seed means.

Corpus	Method	Test PPL	Throughput	Peak memory
Penn Treebank	TTT-MLP	223.07 ± 1.73	100.0%	100.0%
	TTT-TP	222.96 ± 1.81	88.7%	102.7%
WikiText-2	TTT-MLP	236.13 ± 3.19	100.0%	100.0%
	TTT-TP	235.99 ± 3.17	82.6%	102.7%

and this section reports them.

6.1 How much of the target the update reaches

The construction inverts the linear map W_2 exactly, but the conversion from a post-activation displacement to a pre-activation one is a single first-order step through ϕ', which is accurate only for small moves. We measure two things on the trained switching-hinge model over 128 probe episodes per seed, for seeds 0, 1 and 2.

Some targets cannot be reached at all. GELU has a minimum of $\phi_{\min} = -0.170$, so a coordinate of h^* below that value is not the image of any pre-activation. Figure 4(a) shows the distribution; $51.1 \pm 1.3\%$ of requested coordinates fall below $\phi_{\min}$.

Where the target is reachable, the step still falls short. Regressing the achieved displacement $\phi(z_1 - g_1) - h$ on the requested one gives a slope of 0.215 ± 0.004 (Figure 4(b)). The relative error $\|\phi(z_1 - g_1) - h^*\| / \|h^* - h\|$ pools to 0.824 ± 0.008; this is the larger number because it also counts movement in the wrong direction, whereas the slope counts only distance travelled.

The lower layer therefore pursues a target it does not reach in one step. Two constructions would address this — a difference-target correction, or a numerical inversion of ϕ — and both are well posed here only because the layer has a target. We have not evaluated whether either improves accuracy, and Appendix E gives a reason for caution: much of the episodic gain comes from the size of the lower-layer update rather than its direction, and any operator measured at a fixed target step inherits that ambiguity.

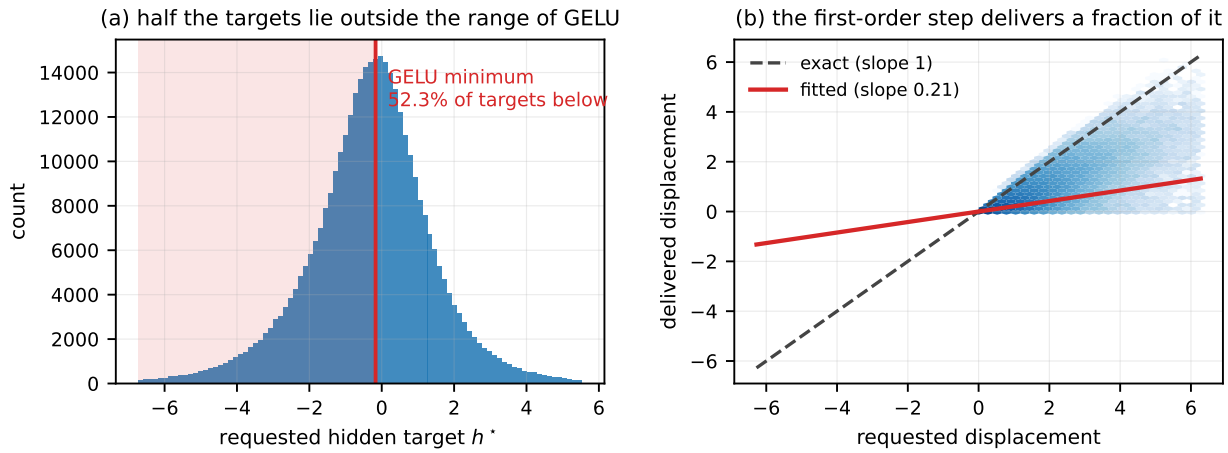

Figure 4: The first-order step does not reach the hidden target. (a) Requested targets h^* against the minimum of GELU; coordinates in the shaded region are not the image of any pre-activation. (b) Achieved against requested displacement per coordinate, with exact transport shown dashed and the fitted slope solid. Trained switching-hinge model, seed 0, 128 probe episodes; the pooled figures over three seeds are given in the text.

Table 4: The relation between the two transport rules changes with training. Mean ± standard deviation over three seeds on moving-hinge; $r = 0.3$ throughout. At initialisation the ridge dominates the spectrum and the bridge is close to a rescaled transpose. Training enlarges the spectrum, and the same ridge then produces a genuine reweighting.

	$\overline{\sigma^2}$	$r/\overline{\sigma^2}$	$\cos(B_r, W_2^\top)$	best scalar $c^\star$	gain spread
At initialisation	0.051 ± 0.000	5.83 ± 0.04	0.9975 ± 0.0001	2.76 ± 0.00	$1.32\times$
After training	0.190 ± 0.020	1.59 ± 0.16	0.9273 ± 0.0040	1.60 ± 0.07	$4.54\times$

6.2 Relation to the transpose rule

Section 3.1 distinguishes the two transport rules by the question each answers. They can also be compared term by term. With $W_2 = U\Sigma V^\top$, transpose transport weights a singular direction by σ_i and the regularized inverse weights it by $\sigma_i/(\sigma_i^2 + r)$. The singular vectors are shared, so the bridge reaches no direction the transpose misses; it only changes the weights.

How much those weights differ depends on r against the spectrum, and the spectrum is not fixed: W_2 is a slow parameter that outer training moves. We therefore measure the comparison twice, at initialisation and after training, over three seeds.

At initialisation, σ^2 has mean 0.051 and $r = 0.3$ exceeds it by a factor of 5.83. Then $\sigma/(\sigma^2 + r) \approx \sigma/r$ across the measured range: the gain $1/(\sigma^2 + r)$ spans a factor of 1.32 (Figure 5), the best scalar fit to B_r is $2.76\, W_2^\top$, and $\cos(B_r, W_2^\top) = 0.9975$. The transported direction is at that point close to a rescaling of the one TTT-MLP already uses.

Training changes this. The mean of σ^2 grows by a factor of 3.7, so $r/\overline{\sigma^2}$ falls to 1.59 and the ridge no longer dominates. The gain then spans a factor of 4.54 across the spectrum, $\cos(B_r, W_2^\top)$ falls to 0.9273, and a scalar multiple of the transpose leaves a residual of about 37% in Frobenius norm. The bridge performs a substantive reweighting in the trained model, not a rescaling.

Read together, the two rows say something about the role of the ridge that neither says alone. The construction begins close to backpropagation, which is why it can be inserted into a trained recipe without destabilising it, and which is the mechanism behind the language-model parity of Section 5.2. It does not stay there. As outer training enlarges the spectrum, the same fixed r moves the bridge away from the transpose of its own accord. Whether that drift is desirable is not something we test here; Section 8 notes that one fixed ridge may not suit a spectrum that changes with width, depth or training length, and this measurement

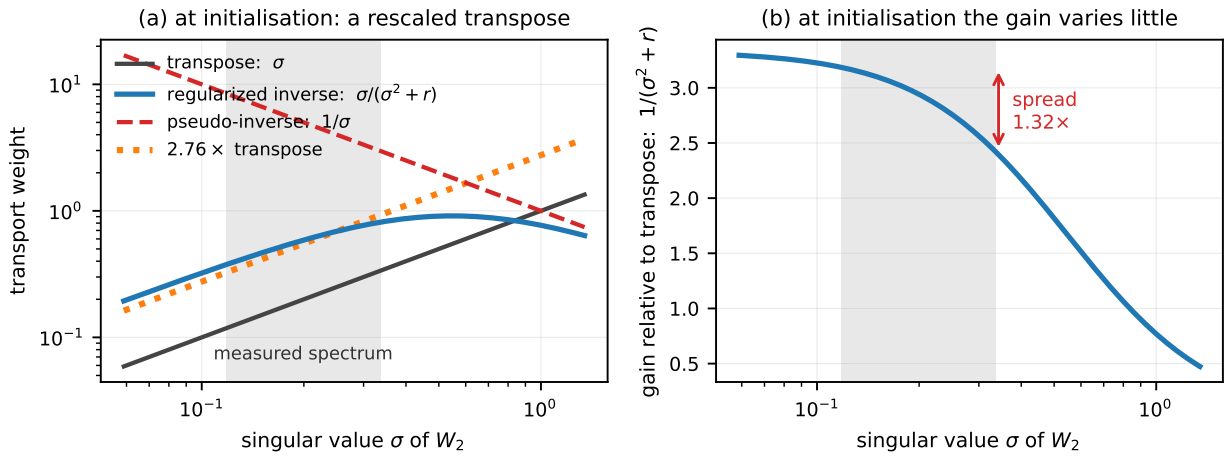

Figure 5: The two transport rules weight the same singular directions of W_2 differently. (a) Weight given to a direction with singular value σ; over the shaded spectrum at initialisation the regularized inverse follows a fixed multiple of the transpose. (b) Its gain relative to the transpose spans a factor of 1.32 over that range. After training the spectrum shifts right and the agreement no longer holds (Table 4).

makes that concern concrete rather than hypothetical.

6.3 What the lower-layer residual carries

The target supplies the lower layer with a local residual, the quantity the output layer has always had. On switching-hinge the teacher changes at a known position while the input distribution does not, so we can ask how well that residual separates the post-switch chunk from the stable ones. Under regularized target propagation the separation has Cohen's $d = 0.636$ and a signal-to-noise ratio of 18.2; the transpose control gives $d = 0.445$ and 3.8. The token representation carries no such signal, since inputs are drawn from one distribution throughout.

A retention gate conditioned on this residual did not improve held-out error in our experiments, so the measurement does not yield a better update rule. It does show that the residual is informative, not merely well defined.

6.4 Coupling between the two fast layers

Per-layer targets invite per-layer control. How much room is there? Across chunks and seeds the local residuals of the two fast layers correlate at $r = 0.989$. They move together almost exactly, leaving little signal for a rule that conditions retention, learning rate or erase strength on one layer and not the other. This bounds part of the design space the interface opens.

7 Related Work

Test-time learners. TTT layers make the recurrent state a model and update that model with a self-supervised objective at test time [Sun et al., 2025]. Our work keeps the published TTT-MLP forward learner and surrounding sequence block, but changes how online output error is assigned to the lower fast layer. This differs from changing the self-supervised objective, adding extra inner steps, or replacing the fast learner with a linear associative matrix.

Depth in fast learners. Recent systematic study of the TTT design space reports that deeper fast-weight networks do not improve over shallow ones, and attributes the difficulty to the way a one-step update interacts with a factorised fast weight: each factor's update is conditioned on the others along its forward–backward

path [Tang et al., 2026]. That work examines the fast-weight architecture, the loss, the learning rate, the decay and the normalisation as design dimensions, but retains the chained gradient as the credit rule for inner layers. The present paper changes exactly that rule, and the asymmetry it removes is one concrete mechanism by which an inner factor is disadvantaged: the output factor is told what activation to produce, the inner factor only how the loss responds to a perturbation. We therefore read Tang et al. [2026] as motivating the question this paper addresses rather than as contradicting it, while noting that our evidence is at small scale and does not establish that explicit targets make depth profitable. Work at the other end of the granularity axis enlarges the update chunk to improve hardware utilisation and reports that nonlinear fast weights outperform linear ones at matched state size [Zhang et al., 2026]; the connection between TTT objectives and linear attention has also been made explicit [Anonymous, 2026].

Target propagation. Target propagation replaces backward gradients with layerwise activation targets [Lee et al., 2015]. Subsequent work connected target propagation to approximate Gauss–Newton directions and studied limitations in non-invertible networks [Meulemans et al., 2020]. Regularized inversion provides analytic targets without a separately trained feedback network and has been studied for recurrent networks [Roulet and Harchaoui, 2023]. TTT-TP applies this principle inside the online learner that constitutes a TTT recurrent state. The distinction is not only where the method is applied but what is inverted: prior work inverts fixed layers as part of a training algorithm, whereas here the inverted operator is a fast weight belonging to the recurrent state and is rebuilt at every chunk. Outer meta-training still uses ordinary backpropagation. Unlike its use as an offline training rule, the inverse here is executed during sequence processing, is constructed from an upper weight that is itself fast state, and changes the recurrent algorithm experienced by later tokens. This placement—inside inference-time state evolution rather than outside it as the optimizer—is the central distinction.

Fast weights and Delta rules. Fast-weight programmers interpret sequence state as an associative map programmed by token-dependent updates [Schlag et al., 2021]. DeltaNet uses residual correction to edit this state, and later work parallelized its recurrence over sequence length [Yang et al., 2024]. Gated DeltaNet combines residual updates with adaptive retention [Yang et al., 2025]. Our generalized Delta interpretation is layer-local and nonlinear: it exposes analogous residual semantics for both layers of a TTT-MLP, but does not import a DeltaNet state or claim its parallel algorithm.

8 Limitations and Broader Impact

The experiments establish operator correctness, a controlled nonlinear-adaptation advantage, and small-scale language-model parity. They do not establish scaling laws, long-context extrapolation, downstream reasoning gains, or production decoding efficiency: the real-text models have 4.6M parameters and train for 2.856M tokens. The moving-hinge benchmark cleanly isolates online nonlinear function learning but is synthetic. Three outer-training seeds show a non-overlapping separation in that controlled setting, but are not intended for fine-grained statistical ranking on language modeling.

Regularized inversion introduces a solve and temporary activations. The current code is a correctness reference, not a fused implementation, and its throughput is lower than baseline. Conditioning may also vary across heads, layers, training stages, and precisions; a single fixed ridge may not remain appropriate at scale. Future work should study adaptive regularization, mixed precision, chunk-level factor reuse, and deeper fast learners.

The work is an architectural study with no direct human-subject or deployment component. Potential benefits include more interpretable control of online adaptation and alternatives for local learning. Potential risks follow those of adaptive language models generally: context-specific updates may memorize sensitive or adversarial content. Explicit retention and erasure interfaces could support mitigation, but this paper does not establish privacy, robustness, or reliable deletion guarantees.

9 Conclusion

We introduced TTT-TP, a parameter- and state-matched TTT-MLP variant that uses the current upper fast weight to construct a regularized-inverse target for the lower fast layer. The upper online update remains exactly official TTT-MLP, while the lower layer receives the minimum-energy hidden correction associated with the desired output change. Correctness controls isolate this change. The resulting online algorithm substantially improves held-out episodic nonlinear adaptation and preserves perplexity on two natural-text corpora, at a measurable unfused runtime cost. More broadly, TTT-TP resolves the target asymmetry inside a multi-layer fast learner: every layer now has a current retrieval, a desired target, and a local residual. That interface makes intermediate recurrent state addressable by erase, write, and retention operations stated in terms of that target. We also report in Appendix E that the greater part of the episodic improvement is attributable to the magnitude of the lower-layer update rather than to the transported direction, and in Section 6.2 that the construction begins close to backpropagation, which is why it can be inserted without destabilising a trained recipe, and that outer training then moves it away from the transpose of its own accord.

Statements and Declarations

Competing Interests. The author declares no competing financial or non-financial interests.
Funding. No external funding supported this work.
Author Contributions. D.Y. designed the method and experiments, implemented the code, analysed the results, and wrote the manuscript.
Data Availability. All datasets used are public: Penn Treebank and WikiText-2 for language modelling, and synthetic episodic regression generated by the released code. No new data were collected.
Code Availability. Code to reproduce every reported number is provided as supplementary material.

References

Anonymous. Test-time training with KV binding is secretly linear attention. *arXiv preprint arXiv:2602.21204*, 2026.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases*, pages 498–515. Springer, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Alexander Meulemans, Francesco Carzaniga, Johan A. K. Suykens, Joao Sacramento, and Benjamin F. Grewe. A theoretical framework for target propagation. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Vincent Roulet and Zaid Harchaoui. Target propagation via regularized inversion for recurrent neural networks. *Transactions on Machine Learning Research*, 2023.

Imanol Schlag, Kazuki Irie, and Juergen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366, 2021.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57503–57522, 2025.

Bohao Tang, Zhen Qin, Yuqi Pan, Zheng Li, Pengfei Liu, and Ya Zhang. Modular TTT: Rethinking test-time training as composable modules. *arXiv preprint arXiv:2608.07110*, 2026.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. In *International Conference on Learning Representations*, 2015.

Tianyuan Zhang et al. Test-time training done right. In *International Conference on Learning Representations*, 2026.

A Regularized inverse derivation

For an output displacement $\delta \in \mathbb{R}^d$, we seek the minimum-energy hidden displacement $u \in \mathbb{R}^m$ that approximately realizes δ through the current upper map:

$$\min_u \frac{1}{2}\|uW_2 - \delta\|^2 + \frac{r}{2}\|u\|^2. \tag{13}$$

The normal equation is

$$u(W_2W_2^\top + rI_m) = \delta W_2^\top. \tag{14}$$

Using the standard ridge identity gives

$$u = \delta(W_2^\top W_2 + rI_d)^{-1}W_2^\top = \delta B_r. \tag{15}$$

This dual expression is preferred because $d < m = 4d$. The numerical audit compares both expressions directly.

B Additional implementation details

The official PyTorch TTT implementation supplies query, key, and value projections, rotary position encoding, pre-convolution, output gating, LayerNorm reconstruction loss, and the learned inner-rate gate. For TTT-TP, each TTT-MLP instance is constructed normally, a replacement cell with the same parameter names and shapes is instantiated, and the state dictionary is loaded strictly. Construction is followed by resetting the random seed so that replacement does not alter later random streams.

The language models use shared query and key projections, pre-convolution, output gating, and tied token embeddings. No pretrained weights are loaded. Data order is identical across paired arms. Validation and printing are excluded from throughput timing, and both arms execute ten real training warm-up steps before measurement.

Table 5: Moving-hinge locked-test NMSE and shuffled-support control.

Method and metric	Seed 0	Seed 1	Seed 2
TTT-MLP NMSE	0.334337	0.411241	0.467567
BP control NMSE	0.334337	0.411241	0.467567
TTT-TP NMSE	0.166443	0.209286	0.194057
TTT-TP shuffled support	1.534457	1.519732	1.452866

Table 6: Seed-level language-model test perplexity.

Corpus	Method	Seed 0	Seed 1	Seed 2
Penn Treebank	TTT-MLP	221.409	222.938	224.858
	TTT-TP	221.176	222.910	224.790
WikiText-2	TTT-MLP	238.799	232.593	237.000
	TTT-TP	238.799	232.555	236.626

C Full locked-test records

D Sensitivity to the regularization strength

The ridge is fixed at $r = 0.3$ for every task and corpus reported in the main text. To check that the episodic result does not depend on that choice, we repeat the moving-hinge protocol at an effective ridge of 0.03. Held-out normalized query error is 0.1577 ± 0.0124 over seeds 0, 1 and 2, compared with 0.1899 at the reported setting. Over this range the method is not sensitive to r. These runs use 16 evaluation batches where the main results use 32, so they are reported separately and not pooled with Table 2.

E Update magnitude versus transported direction

Relative to the transpose control, the regularized inverse changes both the direction in which the lower layer moves and how far it moves, since B_r and $W_2^\top$ differ in scale by a factor between 1.6 and 2.8 depending on how far training has moved the spectrum (Table 4). Attribution requires separating the two.

We build an arm that keeps the magnitude the target-propagated update would have had, but takes its direction from the transpose rule, rescaling the transpose direction to the norm of the regularized-inverse update. On moving-hinge over three seeds this arm reaches 0.1611 ± 0.0120, against 0.1577 ± 0.0124 for the full method. Both use the 16-batch evaluation protocol of Appendix D, which is why the figure for the full method differs from the 0.1899 of Table 2; the comparison here is between the two arms, not against the main-text result. The gap is 0.0034, about 2% of the error.

The decomposition is available at all because the update is written as a displacement toward a target; a chained gradient does not separate into a magnitude and a direction that can be swapped independently. As for what it shows: most of the improvement over the transpose control follows from how far the lower layer moves, not from where it is sent. Increasing the lower-layer target step in the transpose control alone produces a comparable improvement. The contribution of this paper is the explicit target and the interface it defines. It is not a transport direction that outperforms the transpose, and the episodic numbers should be read with that in mind.

F Implementation of the regularized inverse

The bridge requires solving $(W_2^\top W_2 + rI)B = W_2^\top$ once per chunk per head. Two properties keep the cost down. The system is $d \times d$ in the head dimension rather than the hidden width, so its size does not grow with

the expansion factor of the fast MLP. And for $r > 0$ the matrix is symmetric positive definite by construction, so a Cholesky factorisation suffices without pivoting or iterative refinement.

The reference implementation uses a general dense solve. It is correct but wasteful: it materialises intermediate tensors at the hidden width where the head dimension would do. Restructuring the solve to stay in the head dimension leaves the computed bridge unchanged to floating-point tolerance and lowers peak allocated memory by 30 MiB in the wide-head configuration. We give this as an implementation note, not a throughput result. The timings in Section 5.2 come from the unfused reference and bound the cost of the construction from above; a tuned implementation would do better.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- 04supplementarycode.zip