LATENT FLOW TRANSFORMER: FLOW MATCHING THE HIDDEN DYNAMICS OF LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers, the standard implementation for large language models (LLMs), typically consist of tens to hundreds of discrete layers. While depth improves performance, it is an inefficient use of capacity next to the continuous formulations that diffusion and flow-based models use for image generation. We propose the Latent Flow Transformer (LFT), which replaces a contiguous block of layers with a single transport operator trained by flow matching, a post-training procedure that leaves the surrounding architecture unchanged. To identify replaceable spans, we introduce the Recoupling Ratio, an optimal-transport measure of how far a span's endpoint latents deviate from an optimal pairing. We further introduce Flow Walking (FW), which addresses the failure of existing flow-based methods to preserve latent pair coupling. On Pythia-1.4B, an LFT trained with FW replaces 12 of 24 layers with a single operator and retains 92% of the teacher's downstream accuracy (57.5% vs. 62.8%) at 63% of its parameters and 72% of its per-token FLOPs, whereas skipping the same span attains 53.8%. FW matches a regression transport layer at six replaced layers, surpasses it at twelve, and is less sensitive than standard flow matching to the latent coupling of the replaced span, bridging the gap between autoregressive and flow-based generation paradigms.

1 INTRODUCTION

The scale of Transformer-based large language models (LLMs) enables remarkable capabilities, but also introduces substantial memory and compute demands. Work in compression techniques such as pruning and knowledge distillation (Zafrir et al., 2021; Lu et al., 2024; Xu et al., 2024) demonstrates that many layers are overparemeterized and can be removed with minimal performance loss (Men et al., 2024; Gromov et al., 2025; Chen et al., 2025). Meanwhile, in image generation, diffusion and flow matching (FM) models (Lipman et al., 2022) have demonstrated remarkable efficiency, with recent approaches generating high-quality images in as few as one sampling pass (Frans et al., 2024). Inspired by this success, works by (Sun et al., 2019; Xu et al., 2024) have explored adaptin FM and diffusion models for language modeling.

In this work, we aim to explore reducing the redundancy of transformer capacity through matching the latent dynamics of a model. Recent studies frame transformer forward passes as discretized continuous dynamical processes (Chen et al., 2018; Li et al., 2022; Fein-Ashley, 2025), and show that latent trajectories across layers can be nonlinear and complex. Concurrently, in image modeling, it has been shown that efficient surrogate models can be trained to approximate the latent mapping of deep networks using straighter transport dynamics (Das et al., 2023; Song et al., 2023). These developments motivate exploring whether flow-based approaches can learn more efficient inter-layer mappings in transformers, offering a new path to structural compression.

Building upon these insights, we propose the **Latent Flow Transformer (LFT)**, a post-trained transformer variant designed for parameter-and-compute efficient language modeling by replacing a contiguous block of transformer layers with a single learned transport operator trained via flow matching (Lipman et al., 2022). To guide which layers are suitable for replacement, we introduce the *Recoupling Ratio*, an interpretable Optimal Transport-based metric that accurately predicts flow matching quality. Our analysis finds that middle layers with near zero ratios are particularly compressible, supporting the intuition that early LLM layers function qualitatively differently from later ones (Men et al., 2024; Gromov et al., 2025; Sun et al., 2026).

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.

We show that standard flow matching enables effective transport operator training, with perturbations smaller than those caused by removing a couple of transformer layers (Sajjad et al., 2023; Chen et al., 2025). However, performance plateaus as flow path proximity introduces velocity ambiguity (Kim et al., 2024; Park et al., 2024)—a known limitation in FM models. While reflow techniques are a common strategy to mitigate this issue, they are inherently incompatible with the LFT's objective of preserving the original input-output latent relationship. To address this, we introduce **Flow Walking (FW)**, a scalable algorithm for learning latent transport using numerical integration. By training a transformer-like policy to explore paths with a clearer view to the target, FW enables the model to navigate and resolve local ambiguities arising from multiple nearby flow paths, thereby significantly improving transport *alignment* and overcoming the performance plateau observed with standard flow matching.

We validate the framework on *Pythia*, using 410M for the layer-selection study and 1.4B for the compression results. First, layer selection is critical: sweeping spans of *equal length* across the recoupling range shows that replacement quality tracks the *Recoupling Ratio*. As the ratio rises, the latent matching error of standard flow matching increases by two orders of magnitude, while that of FW increases by less than a factor of two. Second, replacing a zero-recoupling span at 1.4B is feasible: six layers reduce to a single transport operator at 10.05 perplexity, against 183.57 for skipping the same span. Finally, compression at 12 of 24 layers shows that LFT-FW leads both standard flow matching and a regression transport layer on latent matching, perplexity and downstream accuracy, and after repair retains 92% of the teacher's downstream accuracy at 63% of its parameters. These results demonstrate that LFT-FW solves, at least part of, the primary challenge in aligning latent transport across distant transformer layers, bridging autoregressive and flow-based modeling paradigms[1].

2 PRELIMINARIES

Continuous-Time perspective of transformer layers A neural net with L discrete layers, especially with a residual connection, can be seen as refining its input hidden state h_{l-1} at layer l, $l = 1, 2, ..., L$, to a better state at its output, h_l. A continuous-time counterpart can be expressed by the following dynamics (Chen et al., 2018; 2023; Papamakarios et al., 2021): $v_t = \frac{dh_t}{dt} = u_\theta(h_t, t)$, where θ parameterizes the ordinary differential equation (ODE), and h_t matches with h_l whenever t matches l/L. Learning θ involves conducting expensive simulation (Kidger, 2022): $h_{t_2} = h_{t_1} + \int_{t_1}^{t_2} u_\theta(h_t, t)\, dt$.

Flow Matching for Simulation-Free Training Flow-based approaches (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022; Liu et al., 2022) provide a simulation-free solution to the ODE defined in continuous-time dynamics. Flow matching moves a particle at initial position x_0 to its target position x_1 via a predefined path in unit time, and learns the corresponding velocity field (Lipman et al., 2022) conditioned on a set of

$$\mathcal{L}_{FlowMatching} = \mathbb{E}_t \left[\|u_\theta(x_t, t) - v_t\|^2 \right], \tag{1}$$

where x_t and v_t are the position and the velocity of the particle at time t, respectively. A straight line, constant speed trajectory is a popular and practical choice for flow matching. With such a target trajectory, x_t and v_t are simply

$$x_t = (1 - t) \cdot x_0 + t \cdot x_1, \quad v_t = x_1 - x_0. \tag{2}$$

During discrete-time inference, datapoints are transported along the flow in discrete steps. When moving from time points t to $t + d$, a step is:

$$x_{t+d} = x_t + d \cdot u_\theta(x_t, t). \tag{3}$$

To enhance the stability and the accuracy of the flow trajectory, an alternative is to set the velocity at the midpoint of the currently estimated forward step:

$$x_{t+d} = x_t + d \cdot u_\theta(x_t + \frac{d}{2} \cdot u_\theta(x_t, t), t + \frac{d}{2}). \tag{4}$$

[1]We release our code for the community to reproduce our results at `https://to_be_released`

Challenges in Flow Matching for Paired Data In flow matching with paired samples, it is crucial to preserve the deterministic correspondence between the support of the source and target distributions. Such setting, where interpolation trajectories intersect, challenges standard flow matching methods (Lipman et al., 2022; Liu et al., 2022) as these methods tend to average conflicting velocity signals near these intersections, yielding biased estimates and failing to transport source points accurately to their paired targets (Park et al., 2024; Kim et al., 2024; Frans et al., 2024). To overcome this, Work by (Park et al., 2024) introduces an auxiliary acceleration field that regulates the rate of change along each trajectory, thereby sharpening the alignment to the true paired mapping. Alternatively, (Kim et al., 2024) propose learning latent-space projections that untangle intersecting pairs prior to flow estimation, effectively preventing trajectory crossings before velocity estimation.

3 Latent Flow Transformer

We propose the Latent Flow Transformer (LFT), a novel architecture for language modeling that seeks to leverage the benefits of flow-related concepts, as demonstrated in image generation, for transformer-based LLMs. The LFT is designed to potentially achieve advantages such as improved parameter efficiency, simulation-free training, and latent trajectory straightening.

The LFT reduces model size by replacing a contiguous block of transformer layers from the teacher with a single learned transport operator, called the **latent flow layer**. This operator is trained using flow concepts (Lipman et al., 2022) to accurately map the input latent at the input of the block to its corresponding output over a desired trajectory. With a suitable architectural choice, the learned operator can be unrolled over predefined intermediate time steps to effectively recover a transformer-like structure.

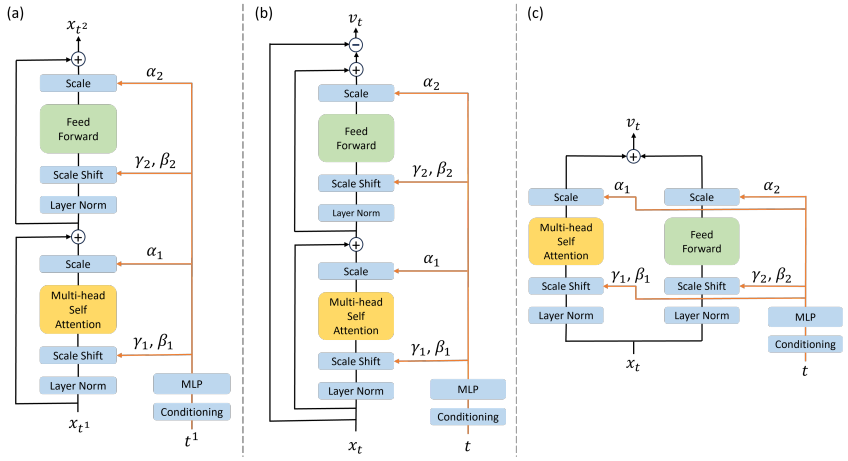


Figure 1: **Velocity field estimator.** (a) The DiT block, mapping an input hidden state to an output hidden state conditioned on time t. (b) Velocity field estimator derived from the DiT block, mapping an input state x_t and time t to a velocity v_t. (c) Velocity field estimator for *Pythia*.

3.1 Velocity field estimating network

Specifying an LFT requires defining the network architecture for the velocity field estimator $u_\theta(x_t, t)$; following the approach in (Peebles & Xie, 2023), our method augments a teacher transformer layer with additional scale and shift operators and an MLP predicting their factors, as shown in Figure 1(a). We derive the velocity estimate by subtracting the input latent from this augmented network's output (Figure 1(b)). Figure 1(c) details the specific case for *Pythia* models. We remark that these networks are used solely for learning the velocity field. During LFT inference, the networks are augmented with skip connections to produce latents, where $u_\theta(x_t, t)$ is evaluated at different values of t from token to token.

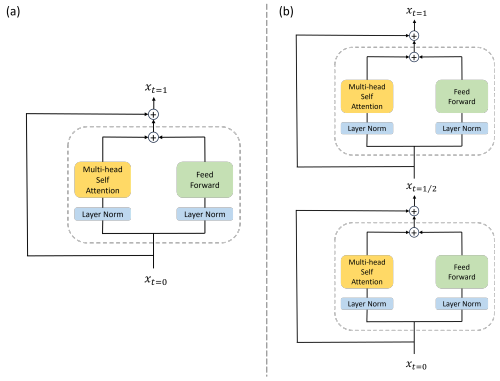


Figure 2: **Static structure of an unrolled LFT.** (a) Single step and (b) two step reconstruction, assuming Equation 3.

Algorithm 1 Learning the velocity field via standard flow matching (Lipman et al., 2022)

$\mathcal{D}$ is a set of training data, $\phi(d, \ell)$ is the teacher LLM's latent at layer ℓ over data d, $AdamW$ is the AdamW optimizer.
while termination condition not met **do**
  $d \sim \mathcal{D}$
  $(x_0, x_1) \leftarrow (\phi(d, m), \phi(d, n))$
  $t \sim U$
  $x_t \leftarrow (1-t)x_0 + tx_1$
  $\theta \leftarrow AdamW(\theta, \nabla_\theta \|u_\theta(x_t, t) - (x_1 - x_0)\|^2)$
end while

3.2 Predict flow matching quality with the Recoupling Ratio

Selecting the optimal block of layers for replacement is critical for LFT performance. Learnability via flow matching is bottlenecked by flow path crossing (Peyré et al., 2019; Lipman et al., 2022), a challenge exacerbated in LFT as preserving original input-output pairings precludes the use of reflow methods (Liu et al., 2022; Das et al., 2023; Kim et al., 2024). To guide layer selection, we introduce an estimator which we refer to as the *Recoupling ratio*. Given sample latent pairs from layers m to n, this estimator quantifies the deviation between their original pairing and the pairing dictated by Optimal Transport (OT). Optimal Transport identifies the minimal cost mapping between two distributions. That is

$$M = \arg\min_{\gamma} \sum_{i,j} \gamma_{i,j}\, d\big(h_m^{(i)},\, h_n^{(j)}\big), \tag{5}$$

where the OT matrix M represents the ideal pairing between latents at layer m and layer n that minimizes a transport cost, such as Euclidean distance.

We define the Recoupling Ratio R as the percentage of pairing relationship that disagrees with M. If M is a square matrix of order O_M:

$$R := 1 - \mathbf{E}\big[\tfrac{\mathrm{Tr}(M)}{O_M}\big]. \tag{6}$$

Because the Recoupling Ratio quantifies the misalignment with original pairings, therefore, controlling for O_M, lower R indicates better alignment and thus predicting fewer flow-crossing issues and higher feasibility for learning the LFT. In our experiments, a practically small O_M is sufficient for the prediction to aid decision making on the selection of layers.

3.3 Learning the velocity field

The velocity estimators shall learn the velocity field from a collection of x_0, x_1 pairs. For an LFT to replacing all layers from layer m to layer n (inclusive), omitting the details of preserving the shared context, we take the input latent of layer m for a given token as x_0, and the output latent of layer n as the corresponding x_1. To learn the velocity field between x_0 and x_1, standard flow matching (Lipman et al., 2022) can be applied as shown in Algorithm 1. We remark that LFT can be trained by many other compatible flow-related algorithms, including our proposed Flow Walking method in Section 4.

3.4 Unrolled LFT has a Transformer structure

At inference, unrolling the latent flow layer over a fixed set of time points $t_0 = 0 < t_1 < t_2 ... < 1$ hardens the latent evolution process into a static processing graph, beneficial for visualizing data flow and optimizing hardware implementation. For the specific case of a single step flow matching combined with the simple reconstruction approximation rule of Equation 3, the latent flow layer becomes equivalent to a single standard transformer layer (see Figure 2). If more than one step is

used, the latent flow layer is equivalent to a stack of transformer layers with cross-layer attention (Li et al., 2020).

The structural similarity between the unrolled LFT and a standard transformer is of significant practical importance, enabling researchers and practitioners to leverage the extensive ecosystem and highly optimized infrastructure developed for transformer-based LLMs.

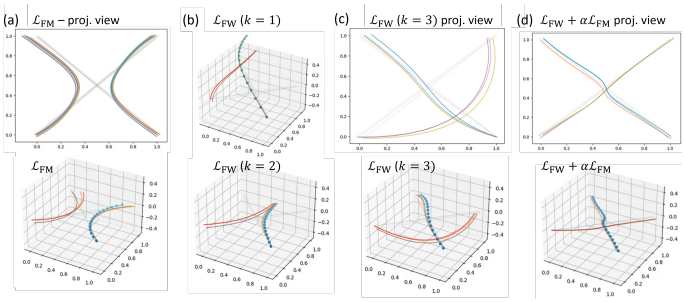


Figure 3: **Toy trajectories for paired data.** Faded lines show the ground-truth trajectories of paired points, while solid curves depict predictions from various flow-matching algorithms. (a) Standard flow matching fails to maintain pairwise correspondence. (b) FW results for $k = 1$ (top) and $k = 2$ (bottom): using fewer integration steps leads to poor trajectory estimates. (c) FW with $k = 3$: projected view (top) and full trajectories (bottom) demonstrate how the learned velocity field generates smooth, curved paths that avoid intersections. (d) A hybrid of FW and standard flow matching with $\alpha = 0.001$ preserves the constant velocity of paired data while allowing curvature at intersections to prevent flow crossings.

4 Flow Matching with Paired Data

To address the challenge of crossing trajectory in Section 2 while to enable the unrolling of latent flow layer back to a Transformer structure in Section 3.4, we introduce the Flow Walking (FW) algorithm for training and inference the latent flow layer of LFT (see Algorithm 2 and Algorithm 3).

FW approximates x_0 to x_1 using numerical integration with discrete time points. We define a step as $s_\theta(x_t, t, t') = x_{t+d}$, where $d = t' - t$ and x_{t+d} can be determined by Equation 3 or Equation 4. Our key intuition is that we can learn non-crossing trajectories by slightly separate trajectories around intersecting points. We define the learning of velocity fields with:

$$\mathcal{L}_{\text{FlowWalking}(k)} = \mathbb{E}_{t_1,\ldots,t_{k-1}} \left\| x_0 + \sum_{i=1}^{k} \Delta_{\theta,t_i} - x_1 \right\|^2, \tag{7}$$

where $\Delta_{\theta,t_i} = s_\theta\big(x_{t_{i-1}},\, t_{i-1}, t_i\big) - x_{t_{i-1}}$, $t_0 = 0$, $t_k = 1$, and $t_i \sim [0, 1]$ for $i \in [1, k-1]$. Unlike linear inference paths (Section 2) that randomly samples different numbers of points k (Geiping et al., 2025), we would like to choose the minimum k with randomly sampled t to avoid expensive simulation. We empirically choose an efficient and generalized training loss using $k = 3$ (Figure 3) as choosing $k = 2$ and $t_1 \sim [0, 1]$ cannot generalize to inference time points $k > 2$.

Algorithm 2 LFT training (FW)

$\mathcal{D}$ is a set of training data, $\phi(d, \ell)$ is the teacher LLM's latent at layer ℓ over data d, $AdamW$ is the AdamW optimizer.
while termination condition not met **do**
  $d \sim \mathcal{D}$
  $(x_0, x_1) \leftarrow (\phi(d, m), \phi(d, n))$
  $t_1, t_2 \sim U[0, 1]$ ▷ Sorted $t_2 > t_1$
  $x_{t_1} \leftarrow s_\theta(x_0, 0, t_1)$ ▷ 1st step
  $x_{t_2} \leftarrow s_\theta(x_{t_1}, t_1, t_2)$ ▷ 2nd step
  $\hat{x}_1 \leftarrow s_\theta(x_{t_2}, t_2, 1)$ ▷ 3rd step
  $\theta \leftarrow AdamW(\theta, \nabla_\theta \|\hat{x}_1 - x_1\|^2)$
end while

Algorithm 3 LFT inference

$x \leftarrow \text{EmbedIn}(token)$
$d \leftarrow 1/k$
$t \leftarrow 0$
$t' \leftarrow d$
for $i \in [0, \ldots, k-1]$ **do**
  $x \leftarrow s_\theta(x, t, t')$
  $t \leftarrow t + d$
  $t' \leftarrow t' + d$
end for
logits = $\text{EmbedOut}(x)$
return logits

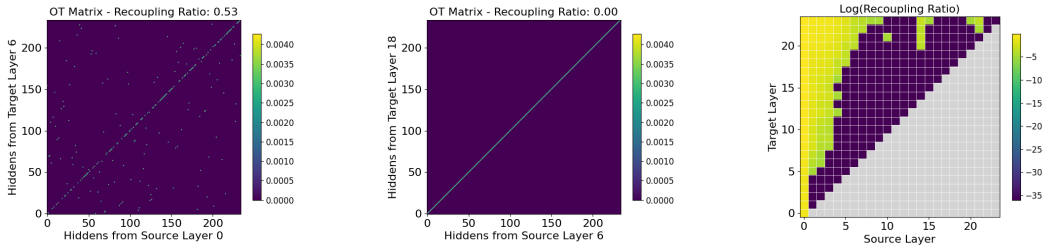
We compare FW with alternative training schemes on toy paired-data experiments (Figure 3. In panel (a), standard flow matching - unstable for paired data - exhibits trajectory "recoupling," though in the absence of crossings it can recover a straight, constant speed path. Panels (b) show FW with $k = 1$ and $k = 2$ (top and bottom, respectively), which fail to learn a generalizable velocity field, and FW with $k = 3$ preserves the original coupling at the expense of straightness. Finally, in panel (d), we add a straightness regularizer via standard flow matching, i.e.,$\mathcal{L} = \mathcal{L}_{\text{FlowWalking}} + \alpha\mathcal{L}_{\text{FlowMatching}}$, which yields a non-crossing, straight interpolation of the paired data. While this stability-regularized variant shows promise for paired-data application, we leave its comprehensive study to future work and here focus on demonstrating FW's feasibility to LFT.

5 Experiments

We implemented the LFT based on `Pythia` and present a series of experiments evaluating the choice of distilled layers, the convergence of flow matching distillations, and inference as an integrated transformer. We trained LFT variants via Standard Flow Matching (LFT-SFM) in Algorithm 1 and Flow Walking (LFT-FW) in Algorithm 2. To ensure a fair comparison between SFM and FW, we use the same recursive update rule Equation 4 to compute x_{t+d}.

Baselines We compare the proposed LFT - SFM and LFT - FW against two baselines: (i) layer-skipping, which directly treats the source hidden state as the target, and (ii) a regression model (Gromov et al., 2025; Chen et al., 2025), which uses a single transformer layer (initialized from the teacher's source layer) trained with an MSE loss to map source to target hidden states. For details of the selection of the compressed layers and the training setup, see Appendix A.

Evaluation To assess the local fidelity of a latent flow layer, we measure (i) the **normalized mean squared error** (NMSE) and (ii) the **KL divergence** between its output $\hat{x}$ and the target latent x, denoted as $\mathbf{KL}_{x||\hat{x}}$. The NMSE is defined as $\frac{\mathbb{E}\,\|\hat{x}-x\|^2}{\mathbb{E}\,\|x\|^2}$. For an end-to-end language-modeling evaluation, we compute the empirical KL divergence between the teacher distribution P and the LFT output distribution Q, denoted as $\mathbf{KL}_{P||Q}$. We also report **perplexity** (PPL) based on the LFT's output logits over a held-out subset of The Pile and downstream tasks reported in accuracy.



(a) (b) (c)

Figure 4: **Optimal Transport Matrices and Recoupling Ratio.** (a) OT Matrix of paired hiddens between layer 0 and layer 6. (b) OT Matrix of paired hiddens between layer 6 and layer 18. (c) Recoupling Ratios of different source target layers.

5.1 Can Recoupling Ratio guide latent flow layer selection?

We first evaluate the Recoupling Ratio metric on `Pythia-410M` for layer selection. By sampling 256 tokens from the first portion of the Pile, we computed the recoupling matrix over different choices of starting layer m and last layer n and report results in Figure 4. Figure 4(a) is the Optimal Transport matrix M for $(m, n) = (0, 6)$. A high recoupling ratio signals that one might encounter a lot of issues related to flow crossing. In comparison, in Figure 4(b), which gives the OT matrix M for $(m, n) = (6, 18)$, with a zero recoupling ratio, the existing pairings for this batch of data are already optimal. In Figure 4(c), the recoupling ratio over all possible choices of m and n is given. We note

that early layers exhibit high recoupling ratio, indicating high misalignment between latents of the layers, while that for middle or later layers of the model are lower, suggesting a lower misalignment issue between latents. This complements prior studies, where in a typical transformer LLM, the first and the last few layers often apply transformations to the hidden state in a way qualitatively different from those by the middle layers (Gromov et al., 2025; Sun et al., 2026).

We train the latent flow layer over spans of differing recoupling ratio, holding the number of replaced layers fixed at six so that R is the only quantity that varies. A higher ratio translates to worse replacement quality (Figure 5), and the degradation is not shared equally by the two objectives: as $\ln R$ rises from -35 to -0.55, the latent matching $\mathrm{KL}_{x||\hat{x}}$ of LFT-SFM increases by two orders of magnitude, whereas that of LFT-FW ($k = 3$) increases by less than a factor of two, remaining below the regression baseline on latent matching and perplexity at every span scanned.

Therefore, layer latents with low recoupling ratios are more suitable for learning the latent flow layer where velocity signals near the trajectory intersections of the latents are less conflicting. We carry this to further LFT training experiments in Section 5.2.

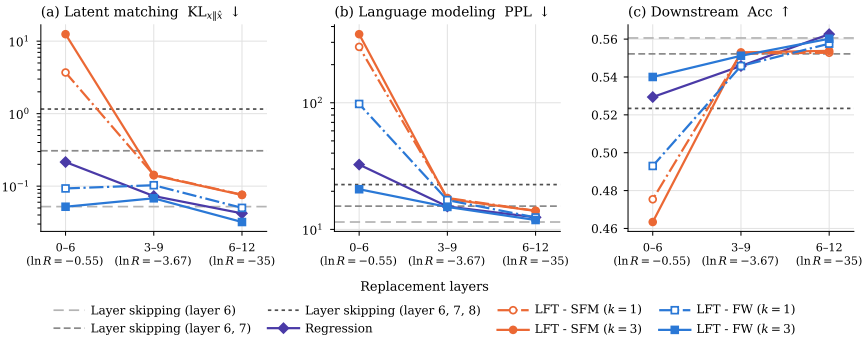
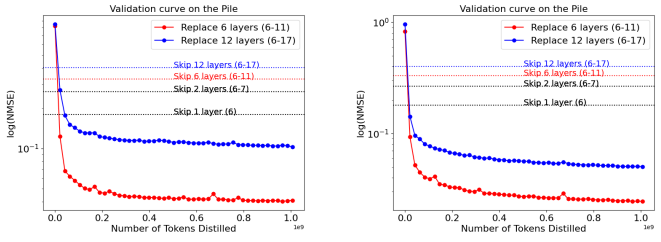


Figure 5: **Replacement quality against the recoupling ratio of the replaced span.** We report results for `Pythia-410M` with a span of six layers replaced by a single learned layer: (a) latent matching $\mathrm{KL}_{x||\hat{x}}$, (b) Pile perplexity, (c) mean downstream accuracy. Ticks give the replaced span and its $\ln R$ (Equation 6). Numbers in Table 4.

5.2 Training LFT: Latent Flow Layer Training and Model Repairing

Latent flow layer convergence We distill on 2.6 billion tokens from The Pile by replacing either layers 6–12 (25 % of layers) or layers 6–18 (50 % of layers) of `Pythia-410M` with a single flow-matching layer. As shown in Figure 6, both LFT-SFM and LFT-FW converge rapidly and substantially outperform naive layer-skipping.



(a) LFT-SFM (b) LFT-FW

Figure 6: **Latent distillation with flow matching vs layer-skipping baselines.** Validation NMSE between predicted and original hidden states vs. training tokens. The dotted lines show Pythia layer-skipping baselines. Flow-matching layers approximate original representations with lower reconstruction error than layer-skipping baselines across both setting.

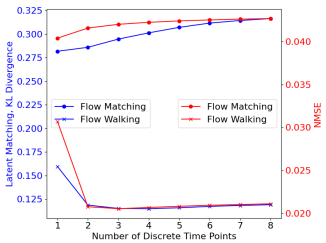


Figure 7: **Inference performance ($\mathrm{KL}_{x||\hat{x}}$ and NMSE) vs number of discrete points k.** We report results of LFT-SFM and LFT-FW with latent flow layer replacing layer 6-18 of Pythia-410M.

Method	Layer replacement only			After full-model repair		Model efficiency	
	$\mathrm{KL}_{x\|\|\hat{x}}\downarrow$	PPL ↓	Acc ↑	$\mathrm{Acc}_{\mathrm{KL}}\uparrow$	$\mathrm{Acc}_{\mathrm{LM}}\uparrow$	Params (M) ↓	GFLOPs/token ↓
Pythia-1.4B (teacher)	–	7.26	0.628	–	–	1414.6	2.623
Replacing Layer 6-12 (25% of layers, $\ln R = -35$)							
Layer skipping (Gromov et al., 2025)	14.370	183.57	0.467	0.592	0.593	1112.5	2.019
Regression (Chen et al., 2025)	$\mathbf{1.4\times10^{-4}}$	10.07	0.594	0.599	0.601	1192.8	2.179
LFT - SFM ($k=1$)	3.6×10^{-4}	11.39	0.589	0.592	0.601	1192.8	2.179
LFT - SFM ($k=3$)	2.5×10^{-4}	11.32	0.588	0.592	0.600	1192.8	2.500
LFT - FW ($k=1$)	5.0×10^{-4}	10.57	0.584	0.599	**0.605**	1192.8	2.179
LFT - FW ($k=3$)	1.6×10^{-4}	**10.05**	**0.595**	**0.604**	0.603	1192.8	2.500
Replacing Layer 6-18 (50% of layers, $\ln R = -35$)							
Layer skipping (Gromov et al., 2025)	14.375	11264.30	0.386	0.538	0.535	810.3	1.415
Regression (Chen et al., 2025)	0.005	19.87	0.539	0.571	0.573	890.6	1.575
LFT - SFM ($k=1$)	0.010	31.85	0.525	0.560	0.560	890.6	1.575
LFT - SFM ($k=3$)	0.010	33.23	0.522	0.558	0.560	890.6	1.896
LFT - FW ($k=1$)	0.008	26.95	0.539	0.569	0.573	890.6	1.575
LFT - FW ($k=3$)	**0.003**	**18.30**	**0.546**	**0.575**	**0.574**	890.6	1.896

Table 1: **Layer replacement, model repairing, and model efficiency.** Replacement methods compared at 1.4B, each replacing a span on the $\ln R = -35$ recoupling plateau (Equation 6). The *layer replacement only* columns describe the spliced model; the *after full-model repair* columns report downstream accuracy after repairing the full model under each of the two repair objectives of (Gromov et al., 2025). Parameter and FLOP counts follow (McLeish et al., 2025). See Table 3 for per-task breakdown. **Boldface** marks the best method within each replaced span.

Learned latent dynamics and model repairing We demonstrate the learned latent flow layer and the repairing of the full model on `Pythia-1.4B` (See Appendix B for `Pythia-410M` results). We perform inference at multiple discrete time points and compare to the regression baseline in Table 1 (see 5.3 for analysis on time-point selection). Although LFT-SFM exceeds layer-skipping, it fails to match the regression model, owing to Standard Flow Matching struggling to learn fine-grained velocity fields when paired trajectories are closely clustered in latent space (Section 4). In contrast, Flow Walking outperforms both baselines in latent-state matching and downstream language modeling; notably, LFT-FW ($k = 1$) achieves parity with the regression model and outperforms LFT-SFM ($k = 3$), suggesting that Flow Walking's implicit velocity estimates more accurately guide the model toward its target hidden state.

Following (Gromov et al., 2025; Chen et al., 2025), we experiment with KL loss and standard language modelling loss for repairing with the distilled latent flow layer. The same ordering in performance holds after repair: on `Pythia-1.4B`, skipping layers 6–18 outright collapses downstream accuracy to 0.386, while the same span replaced by an LFT-FW layer already scores 0.546 before any repair and 0.575 after it, retaining 92% of the teacher's downstream accuracy (0.575 vs. 0.628) while replacing half of the teacher's layers with a single latent flow layer.

5.3 Effect of discrete time points on inference performance

The number of discrete time points k is a key hyperparameter of the LFT at inference time (see Algorithm 3). While minimizing the total number of learnable parameters is crucial, it is equally important to choose a small k, since each additional time point incurs extra FLOPs during inference. In Figure 7, we report both the $\mathrm{KL}_{x||\hat{x}}$ and NMSE of LFT as functions of k. For LFT-SFM, both KL and NMSE decrease as k is reduced, implying that early velocity estimates steer more correctly the hidden state toward its target hidden state. As to LFT-FW, it attains its best performance at $k = 3$, matching the three-step integration in Equation 7. As k approaches 1, KL divergence rises sharply, indicating that the implicit velocity estimation at $t = 0$ is inaccurate and must be refined through multi-step corrections. Notably, despite never trained with more than three integration steps, LFT-FW extrapolates robustly to $k = 8$, showing only minor degradation.

6 Related work

Theory of Transformer. Transformer architectures (Vaswani et al., 2017) are the foundation of LLMs, enabling strong performance across various NLP tasks. As model scale increases, understanding

their internal structure becomes essential. Recent works explore the evolution of representations across layers, attention distributions, and layer-level modularity. For example, "Frankenstein" models recombine layers from different checkpoints to study knowledge localization and compatibility (Matena & Raffel, 2022; Ilharco et al., 2023; Kim et al., 2023). Other research investigates latent trajectory and attention alignment across layers (Wang et al., 2024; Gromov et al., 2025), offering insights into the compositional behavior of LLMs.

Efficient Transformer. Improving transformer efficiency has been approached through pruning, parameter sharing, and distillation. Magnitude-based pruning (Gordon et al., 2020) and structured approaches (Zafrir et al., 2021) have shown that substantial parameter reduction is possible with limited performance drop (Lu et al., 2024). Parameter-sharing architectures like ALBERT (Lan et al., 2020) compress models by reusing weights across layers. Distillation methods such as DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2019) transfer knowledge from larger models to smaller ones through intermediate supervision and multi-stage training.

Flow matching for image generation. Flow-based models have emerged as efficient alternatives to diffusion in generative modeling. Flow Matching (FM) (Lipman et al., 2022) learns continuous velocity fields for optimal transport and has been extended with Rectified Flow (Liu et al., 2022) and Conditional Flow Matching (Li et al., 2023) to improve training stability and sample quality. Recent works (Frans et al., 2024) demonstrate high-quality one-step generation by learning direct endpoint mappings, highlighting FM's potential in reducing generation steps.

Diffusion and flow matching models for language modeling. Recent efforts have adapted diffusion and FM frameworks for language generation. Diffusion-based models such as GENIE (Lin et al., 2023) apply denoising processes to continuous token embeddings, while AR-Diffusion (Wu et al., 2023) introduces autoregressive properties for better sequence modeling. FM-based approaches include FlowSeq (Hu et al., 2024), which improves sampling efficiency, and CaLMFlow (He et al., 2024), which applies Volterra integral formulations to better align continuous and discrete modeling in language.

7 Discussion

We demonstrated that a contiguous stack of transformer layers can be compressed into a single flow-matching transport operator, giving a post-trained variant - Latent Flow Transformer - with reduced parameter count and compute cost. On Pythia 410M and 1.4B, the resulting models outperform both layer-skipping and regression baselines, suggesting that the transformation carried out by a block of layers is, at least in part, well approximated by a continuous transport in latent space.

Limitations Despite the promising results of LFT, several limitations remain. We validated LFT only on `Pythia` 410M and 1.4B checkpoints; more recent dense and MoE architectures remain underexplored. We conjecture that for dense models such as OLMo or Llama, it may be possible to compress more layers under a matched parameter budget, given the issues reported for deeper layers in these architectures (Sun et al., 2026), but this remains untested. Furthermore, LFT-FW requires three unrolling steps at train time to compute the FW loss, requiring more train-FLOPs per token than the regression baseline. LFT-FW is therefore preferable when train-FLOPs isn't the binding constraint.

Future work First, the most immediate direction is to improve the learning of the latent flow layer. Our current formulation keeps the input and output layers of the source model fixed, which leaves the flow trajectories unnecessarily entangled; jointly optimizing the trajectories (Kim et al., 2024), or adopting recent one-step flow matching techniques such as drift models and mean flows (Geng et al., 2025; Deng et al., 2026), could learn the layer closer to a single step generation and reduce the number of layers by an order of magnitude or more. Second, rather than treating flow replacement as a post-training edit, the compressed architecture could be folded into pretraining itself, in the spirit of pruning-aware pretraining (Xing et al., 2025): continuing to pretrain a flow-replaced model, or scheduling the replacement during training, would test whether the shallower architecture can be trained directly instead of distilled from a full-depth teacher. Finally, because a flow-compressed model is substantially smaller than its source while remaining close to it in distribution, it is a natural draft model for speculative decoding (Li et al., 2025; Chen et al., 2026), where the KL results reported here suggest a favorable acceptance rate.

AI USE STATEMENT

We used AI tools to help accelerate experiment execution, draft plotting and table-generation code, review the manuscript's wording, check the accuracy of references, and adopt paper template. The authors take responsibility for the final content of this work, including all text, references, experimental results, figures, and tables.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding, 2026. URL https://arxiv.org/abs/2602.06036.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models, 2025. URL https://arxiv.org/abs/2403.19135.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36:47143–47175, 2023.

Ayan Das, Stathi Fotiadis, Anil Batra, Farhang Nabiei, FengTing Liao, Sattar Vakili, Da shan Shiu, and Alberto Bernacchia. Image generation with shortest path diffusion, 2023. URL https://arxiv.org/abs/2306.00501.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting, 2026. URL https://arxiv.org/abs/2602.04770.

Jacob Fein-Ashley. Flowing through layers: A continuous dynamical systems perspective on transformers, 2025. URL https://arxiv.org/abs/2502.05656.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One Step Diffusion via Shortcut Models, October 2024. URL http://arxiv.org/abs/2410.12557. arXiv:2410.12557 [cs].

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Mitch Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, 2020.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025. URL https://arxiv.org/abs/2403.17887.

Sizhuang He, Daniel Levine, Ivan Vrkic, Marco Francesco Bressana, David Zhang, Syed Asad Rizvi, Yangtian Zhang, Emanuele Zappala, and David van Dijk. Calmflow: Volterra flow matching using causal language models, 2024. URL https://arxiv.org/abs/2410.05292.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. Flow matching for conditional text generation in a few sampling steps. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 380–392, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-short.33/.

Gabriel Ilharco, Mitchell Wortsman, et al. Editing models with task arithmetic. In *ICML*, 2023.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of EMNLP*, 2019.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

Semin Kim, Jaehoon Yoo, Jinwoo Kim, Yeonwoo Cha, Saehoon Kim, and Seunghoon Hong. Simulation-free training of neural odes on paired data. *Advances in Neural Information Processing Systems*, 37:60212–60236, 2024.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, February 2020. URL http://arxiv.org/abs/1909.11942. arXiv:1909.11942.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8335–8351, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.571. URL https://aclanthology.org/2022.acl-long.571/.

Xiangning Li, Shuang Li, Zhengyang Zhou, et al. Diffusion via conditional transport. *arXiv preprint arXiv:2305.08891*, 2023.

Yangyang Li, Qin Huang, Xuan Pei, Yanqiao Chen, Licheng Jiao, and Ronghua Shang. Cross-layer attention network for small object detection in remote sensing imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:2148–2161, 2020.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL https://arxiv.org/abs/2503.01840.

Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise. In *International Conference on Machine Learning*, pp. 21051–21064. PMLR, 2023.

Yotam Lipman, Talia Ringer Cohen, Yujia Zhang, Ehsan Hajiramezanali, et al. Flow matching for generative modeling. In *NeurIPS*, 2022.

Minghao Liu, Belinda Tzen, Maxim Raginsky, et al. Rectified flow: Learning deep generative models using optimal transport. *arXiv preprint arXiv:2209.14687*, 2022.

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. Reassessing Layer Pruning in LLMs: New Insights and Methods, November 2024. URL http://arxiv.org/abs/2411.15558. arXiv:2411.15558.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *arXiv preprint arXiv:2206.14841*, 2022.

Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws, 2025. URL https://arxiv.org/abs/2502.06857.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL https://arxiv.org/abs/2403.03853.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Dogyun Park, Sojin Lee, Sihyeon Kim, Taehoon Lee, Youngjoon Hong, and Hyunwoo J. Kim. Constant acceleration flow, 2024. URL https://arxiv.org/abs/2411.00322.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, January 2023. ISSN 0885-2308. doi: 10.1016/j.csl.2022.101429. URL http://dx.doi.org/10.1016/j.csl.2022.101429.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *arXiv preprint arXiv:1910.01108*, 2019.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression, 2019. URL https://arxiv.org/abs/1908.09355.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models, 2026. URL https://arxiv.org/abs/2502.05795.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. Latent space chain-of-embedding enables output-free llm self-evaluation. *arXiv preprint arXiv:2410.13640*, 2024.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Yelong Shen, Jian Jiao, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. Ar-diffusion: auto-regressive diffusion model for text generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Xingrun Xing, Zheng Liu, Shitao Xiao, Boyan Gao, Yiming Liang, Wanpeng Zhang, Haokun Lin, Guoqi Li, and Jiajun Zhang. Efficientllm: Scalable pruning-aware pretraining for architecture-agnostic edge language models, 2025. URL https://arxiv.org/abs/2502.06663.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Prune once for all: Sparse pre-trained language models. In *arXiv preprint arXiv:2102.07436*, 2021.

Setting	Pythia-410M	Pythia-1.4B
Teacher and layer surgery		
Teacher checkpoint	Pythia-410M	Pythia-1.4B
Transformer layers / hidden size	24 / 1024	24 / 2048
Replacement layer	one transformer block + time embedding + adaLN	
Replacement-layer initialisation	Source layer of the teacher (adaLN zero-init, time MLP $\mathcal{N}(0, 0.02^2)$)	
Attention implementation	SDPA, self-attention on x_t	
Stage 1: latent flow layer training		
Corpus	The Pile (2020)	The Pile (2020)
Max sequence length	2048 (truncation, no packing)	
Batch size (tokens)	0.26M	0.5M
Training tokens	≈2.6B	≈5.2B
Learning rate	1×10^{-3}	2×10^{-4}
LR schedule	cosine, warmup ratio 0.01	
Optimiser	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01	
Gradient clipping	1.0	
Trainable parameters	replacement layer only	
Stage 2: full-model repair		
Repair objectives	KL loss and LM loss	KL loss only
Corpus	The Pile (2020)	
Effective batch (sequences)	64	256
Optimisation steps	400	1907
Repair tokens	≈18M	≤1.0B
Learning rate	1×10^{-5}	1×10^{-5}
LR schedule	cosine, warmup ratio 0.02	
Optimiser	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01	
KL temperature	1.0	
Trainable parameters	all	
Evaluation		
Language Modelling eval	The Pile (2020, held out set from training)	
Downstream harness	`lm-evaluation-harness`, zero-shot, `acc`	

Table 2: **Hyperparameters for the two training stages and the evaluation protocol behind Table 1.** Stage 1 trains the single replacement layer against latents taken from the frozen teacher; Stage 2 trains the whole spliced model end-to-end under one of the two repair objectives. Within a scale, the KL and LM objectives share the same data, steps, batch size, learning rate and seed, and differ only in the loss. Token counts are approximate: documents are truncated at 2048 tokens rather than packed. All trainings were done on NVIDIA A6000x8 nodes.

A Experiment setup

Table 2 summarizes hyperparameters setup for producing Table 1.

Assets and licenses All assets used are public and used within their terms: the `Pythia` 410M and 1.4B checkpoints (Apache 2.0), The Pile (MIT), and `lm-evaluation-harness` (MIT).

Models and layer surgery Both teachers are the stock Pythia checkpoints, 410M and 1.4B, used without any continued pre-training; each has 24 transformer layers. We replace one contiguous span of layers, either $6 \to 12$ (six layers, a quarter of the stack) or $6 \to 18$ (twelve layers, half the stack), by a single learned transport layer, keeping the same layer indices at both scales. That layer is one GPT-NeoX transformer block augmented with a sinusoidal timestep embedding and an adaLN modulation head; its block weights are initialised from teacher layer 6, its time-embedding MLP from $\mathcal{N}(0, 0.02^2)$ and its adaLN projection from zero, so it starts as (approximately) the identity of the layer it replaces. LFT-SFM and LFT-FW use the residual-free parallel form of the block (Figure 1 (c)), while the Regression baseline keeps the standard residual block and is applied once at $t = 0$. The layer-skipping baseline sets the target latent to the source latent.

B Full layer-replacement results

Table 4 is the unabridged version of Table 1 and the raw results of Figure 5.

Method	Repair	ARC-e acc ↑	HellaSwag acc ↑	PIQA acc ↑	SciQ acc ↑	WinoGrande acc ↑
Pythia-410M (24 layers)						
Pythia-410M (teacher)	–	0.515	0.361	0.685	0.815	0.526
Skip 1 layer (6)	none	0.478	0.349	0.666	0.796	0.514
Skip 1 layer (6)	KL loss	0.510	0.361	0.673	0.802	0.541
Skip 1 layer (6)	LM loss	0.506	0.361	0.677	0.811	0.541
Skip 2 layers (6–7)	none	0.436	0.345	0.657	0.794	0.529
Skip 2 layers (6–7)	KL loss	0.494	0.356	0.667	0.809	0.530
Skip 2 layers (6–7)	LM loss	0.496	0.366	0.666	0.817	0.538
Skip 3 layers (6–8)	none	0.419	0.332	0.631	0.732	0.503
Skip 3 layers (6–8)	KL loss	0.498	0.350	0.665	0.805	0.527
Skip 3 layers (6–8)	LM loss	0.491	0.364	0.653	0.809	0.503
Replacing layers 6–12 (25% of layers)						
Regression	none	0.481	0.346	0.656	0.798	0.532
Regression	KL loss	0.484	0.351	0.656	0.793	0.530
Regression	LM loss	0.473	0.351	0.653	0.800	0.529
LFT - SFM ($k = 3$)	none	0.470	0.341	0.653	0.766	0.539
LFT - SFM ($k = 3$)	KL loss	0.486	0.339	0.649	0.784	0.532
LFT - SFM ($k = 3$)	LM loss	0.480	0.347	0.646	0.789	0.548
LFT - FW ($k = 3$)	none	0.475	0.343	0.662	0.801	0.520
LFT - FW ($k = 3$)	KL loss	0.482	0.344	0.672	0.798	0.521
LFT - FW ($k = 3$)	LM loss	0.476	0.347	0.667	0.807	0.521
Replacing layers 6–18 (50% of layers)						
Regression	none	0.433	0.337	0.613	0.711	0.511
Regression	KL loss	0.446	0.334	0.627	0.743	0.511
Regression	LM loss	0.444	0.335	0.631	0.753	0.513
LFT - SFM ($k = 3$)	none	0.384	0.338	0.619	0.671	0.513
LFT - SFM ($k = 3$)	KL loss	0.418	0.337	0.627	0.746	0.514
LFT - SFM ($k = 3$)	LM loss	0.414	0.333	0.631	0.744	0.513
LFT - FW ($k = 3$)	none	0.441	0.330	0.615	0.758	0.510
LFT - FW ($k = 3$)	KL loss	0.446	0.333	0.640	0.796	0.516
LFT - FW ($k = 3$)	LM loss	0.445	0.338	0.640	0.789	0.509
Pythia-1.4B (24 layers)						
Pythia-1.4B (teacher)	–	0.597	0.395	0.712	0.866	0.569
Replacing layers 6–12 (25% of layers)						
Layer skipping	none	0.353	0.309	0.598	0.561	0.515
Layer skipping	KL loss	0.536	0.375	0.696	0.825	0.528
Layer skipping	LM loss	0.532	0.376	0.692	0.830	0.536
Regression	none	0.550	0.373	0.688	0.826	0.534
Regression	KL loss	0.558	0.369	0.691	0.849	0.529
Regression	LM loss	0.552	0.383	0.693	0.849	0.526
LFT - SFM ($k = 1$)	none	0.542	0.377	0.684	0.823	0.521
LFT - SFM ($k = 1$)	KL loss	0.548	0.372	0.692	0.839	0.511
LFT - SFM ($k = 1$)	LM loss	0.554	0.380	0.699	0.841	0.532
LFT - SFM ($k = 3$)	none	0.539	0.377	0.682	0.823	0.521
LFT - SFM ($k = 3$)	KL loss	0.544	0.372	0.692	0.841	0.510
LFT - SFM ($k = 3$)	LM loss	0.552	0.381	0.699	0.843	0.525
LFT - FW ($k = 1$)	none	0.534	0.368	0.684	0.822	0.514
LFT - FW ($k = 1$)	KL loss	0.564	0.371	0.697	0.851	0.513
LFT - FW ($k = 1$)	LM loss	0.565	0.375	0.703	0.856	0.527
LFT - FW ($k = 3$)	none	0.556	0.377	0.691	0.829	0.524
LFT - FW ($k = 3$)	KL loss	0.572	0.372	0.702	0.843	0.529
LFT - FW ($k = 3$)	LM loss	0.566	0.377	0.701	0.851	0.522
Replacing layers 6–18 (50% of layers)						
Layer skipping	none	0.282	0.295	0.539	0.306	0.509
Layer skipping	KL loss	0.451	0.344	0.625	0.747	0.525
Layer skipping	LM loss	0.438	0.340	0.627	0.747	0.525
Regression	none	0.479	0.337	0.633	0.734	0.513
Regression	KL loss	0.514	0.350	0.657	0.809	0.525
Regression	LM loss	0.519	0.352	0.654	0.813	0.529
LFT - SFM ($k = 1$)	none	0.442	0.339	0.626	0.694	0.523
LFT - SFM ($k = 1$)	KL loss	0.500	0.342	0.654	0.790	0.512
LFT - SFM ($k = 1$)	LM loss	0.494	0.347	0.655	0.786	0.518
LFT - SFM ($k = 3$)	none	0.433	0.336	0.622	0.696	0.525
LFT - SFM ($k = 3$)	KL loss	0.493	0.342	0.657	0.786	0.511
LFT - SFM ($k = 3$)	LM loss	0.496	0.344	0.655	0.789	0.517
LFT - FW ($k = 1$)	none	0.474	0.340	0.622	0.735	0.523
LFT - FW ($k = 1$)	KL loss	0.513	0.350	0.654	0.805	0.522
LFT - FW ($k = 1$)	LM loss	0.516	0.352	0.655	0.814	0.527
LFT - FW ($k = 3$)	none	0.508	0.335	0.636	0.732	0.517
LFT - FW ($k = 3$)	KL loss	0.525	0.357	0.663	0.815	0.513
LFT - FW ($k = 3$)	LM loss	0.533	0.355	0.658	0.818	0.508

Table 3: **Per-task downstream results before and after repair, for both model scales.** Accuracy on the five tasks averaged into the Downstream Acc mean: ARC-easy, HellaSwag, PIQA, SciQ and WinoGrande.

	Layer replacement only					After full-model repair		Model efficiency	
	Latent matching		Language modeling		Downstream	KL loss	LM loss	Memory	Inference compute
Method	$\mathrm{KL}_{x\|\|\tilde{x}}\downarrow$	NMSE↓	$\mathrm{KL}_{P\|\|Q}\downarrow$	PPL↓	Acc↑	Acc↑	Acc↑	Params (M)↓	GFLOPs/token↓
Pythia-410M									
Pythia-410M (teacher)	–	–	–	9.02	0.580	–	–	405.3	0.708
Skip 1 layer (6)	0.052	0.180	0.257	11.44	0.561	0.577	0.579	392.7	0.682
Skip 2 layers (6–7)	0.307	0.266	0.529	15.27	0.552	0.571	0.577	380.1	0.657
Skip 3 layers (6–8)	1.157	0.307	0.933	22.59	0.523	0.569	0.564	367.5	0.632
Replacing Layer 6-12 (25% of layers, $\ln R = -35$*)*									
Regression (Chen et al., 2025)	0.042	0.010	0.305	12.45	**0.563**	0.563	0.561	350.0	0.597
LFT - SFM ($k = 1$)	0.076	0.015	0.409	14.04	0.553	0.557	0.561	350.0	0.597
LFT - SFM ($k = 3$)	0.076	0.015	0.407	13.98	0.554	0.558	0.562	350.0	0.678
LFT - FW ($k = 1$)	0.050	0.012	0.300	12.42	0.558	0.561	0.562	350.0	0.597
LFT - FW ($k = 3$)	**0.032**	**0.009**	**0.254**	**11.86**	0.560	**0.563**	**0.564**	350.0	0.678
Replacing Layer 6-18 (50% of layers, $\ln R = -35$*)*									
Regression (Chen et al., 2025)	0.156	0.024	0.864	21.97	0.521	0.532	0.535	274.4	0.446
LFT - SFM ($k = 1$)	0.282	0.040	1.216	31.35	0.511	0.529	0.528	274.4	0.446
LFT - SFM ($k = 3$)	0.295	0.042	1.232	31.91	0.505	0.528	0.527	274.4	0.527
LFT - FW ($k = 1$)	0.160	0.031	0.839	21.82	0.525	0.535	0.539	274.4	0.446
LFT - FW ($k = 3$)	**0.115**	**0.021**	**0.736**	**19.92**	**0.531**	**0.546**	**0.544**	274.4	0.527
Replacing Layer 3-9 (25% of layers, $\ln R = -3.67$*)*									
Regression (Chen et al., 2025)	0.073	**0.014**	0.483	15.17	0.546	–	–	350.0	0.597
LFT - SFM ($k = 1$)	0.143	0.027	0.642	17.71	0.552	–	–	350.0	0.597
LFT - SFM ($k = 3$)	0.141	0.029	0.628	17.42	**0.553**	–	–	350.0	0.678
LFT - FW ($k = 1$)	0.103	0.021	0.600	17.04	0.546	–	–	350.0	0.597
LFT - FW ($k = 3$)	**0.068**	**0.014**	**0.479**	**15.05**	0.551	–	–	350.0	0.678
Replacing Layer 14-20 (25% of layers, $\ln R = -3.67$*)*									
Regression (Chen et al., 2025)	0.100	0.020	**0.385**	**13.44**	**0.550**	–	–	350.0	0.597
LFT - SFM ($k = 1$)	0.194	0.033	0.523	15.36	0.536	–	–	350.0	0.597
LFT - SFM ($k = 3$)	0.202	0.034	0.522	15.32	0.536	–	–	350.0	0.678
LFT - FW ($k = 1$)	0.118	0.024	0.439	14.12	0.547	–	–	350.0	0.597
LFT - FW ($k = 3$)	**0.079**	**0.019**	0.405	13.72	0.548	–	–	350.0	0.678
Replacing Layer 0-6 (25% of layers, $\ln R = -0.55$*)*									
Regression (Chen et al., 2025)	0.215	0.097	1.288	32.49	0.529	–	–	350.0	0.597
LFT - SFM ($k = 1$)	3.695	0.974	3.199	276.19	0.475	–	–	350.0	0.597
LFT - SFM ($k = 3$)	12.474	3.269	3.392	347.95	0.463	–	–	350.0	0.678
LFT - FW ($k = 1$)	0.093	0.133	2.200	97.87	0.493	–	–	350.0	0.597
LFT - FW ($k = 3$)	**0.052**	**0.055**	**0.789**	**20.76**	**0.540**	–	–	350.0	0.678
Pythia-1.4B									
Pythia-1.4B (teacher)	–	–	–	7.26	0.628	–	–	1414.6	2.623
Replacing Layer 6-12 (25% of layers, $\ln R = -35$*)*									
Layer skipping (Gromov et al., 2025)	14.370	0.146	3.027	183.57	0.467	0.592	0.593	1112.5	2.019
Regression (Chen et al., 2025)	$\mathbf{1.4\times10^{-4}}$	**0.025**	**0.301**	10.07	0.594	0.599	0.601	1192.8	2.179
LFT - SFM ($k = 1$)	3.6×10^{-4}	0.040	0.419	11.39	0.589	0.592	0.601	1192.8	2.179
LFT - SFM ($k = 3$)	2.5×10^{-4}	0.041	0.414	11.32	0.588	0.592	0.600	1192.8	2.500
LFT - FW ($k = 1$)	5.0×10^{-4}	0.033	0.349	10.57	0.584	0.599	**0.605**	1192.8	2.179
LFT - FW ($k = 3$)	1.6×10^{-4}	0.026	0.302	**10.05**	**0.595**	**0.604**	0.603	1192.8	2.500
Replacing Layer 6-18 (50% of layers, $\ln R = -35$*)*									
Layer skipping (Gromov et al., 2025)	14.375	0.469	7.286	11264.30	0.386	0.538	0.535	810.3	1.415
Regression (Chen et al., 2025)	0.005	0.060	0.929	19.87	0.539	0.571	0.573	890.6	1.575
LFT - SFM ($k = 1$)	0.010	0.096	1.340	31.85	0.525	0.560	0.560	890.6	1.575
LFT - SFM ($k = 3$)	0.010	0.101	1.375	33.23	0.522	0.558	0.560	890.6	1.896
LFT - FW ($k = 1$)	0.008	0.101	1.218	26.95	0.539	0.569	0.573	890.6	1.575
LFT - FW ($k = 3$)	**0.003**	**0.056**	**0.863**	**18.30**	**0.546**	**0.575**	**0.574**	890.6	1.896

Table 4: **Layer replacement, model repairing, and model efficiency: full results. Boldface** marks the best method within each replaced span.