Network of Theseus (Like the ship)

Vighnesh Subramaniam[1]*, **Colin Conwell**[2], **Boris Katz**[1],
Andrei Barbu[1], **Brian Cheung**[3]
[1]MIT CSAIL [2]Department of Cognitive Science, Johns Hopkins University [3]UCSF
[1]{vsub851,boris,abarbu}@mit.edu
[2]cconwel2@jhu.edu
[3]bcheung@ucsf.edu

Abstract

A standard assumption in deep learning is that the inductive bias introduced by a neural network architecture must persist from training through inference. The architecture you train with is the architecture you deploy. This assumption constrains the community from selecting architectures that may have desirable efficiency or design properties due to difficulties with optimization. We challenge this assumption with the Network of Theseus (NoT), a method for progressively converting a trained, or even untrained, guide network architecture part-by-part into an entirely different target network architecture while preserving the performance of the guide network. At each stage, components in the guide network architecture are incrementally replaced with target architecture modules and aligned via representational similarity metrics. This procedure preserves most of the functionality of the guide network even under substantial architectural changes—for example, converting a convolutional network into a multilayer perceptron, or GPT-2 into a recurrent neural network. By decoupling optimization from deployment, NoT expands the space of viable inference-time architectures, opening opportunities for better accuracy–efficiency tradeoffs and enabling more directed exploration of the architectural design space.

1 Introduction

In machine learning research, we tend to assume that training is coupled with inference. If a specific architecture is trained to solve a task with specific inductive biases and computational mechanisms, that same architecture should be used at test time. This assumption is embedded in common practice. Neural architecture search (NAS) discovers an efficient inference-time architecture that is then trained and deployed [62, 40, 29]. Compression pipelines prune and quantize a trained model to meet deployment budgets while preserving the trained structure [16, 24]. This assumption applies even in settings like distillation, where teacher networks train student networks designed explicitly for inference [20, 15, 41, 22]. A few lines of work partially relax this coupling by training a supernet and selecting subnets for deployment [6, 54], but the resulting inference models remain architectural subgraphs or variants of the original training network rather than cross-architectural conversions.

We challenge this premise that the architecture used for training must be the architecture used for inference. Decoupling the architecture used for training from that used for inference would enable models to be trained with large, optimization friendly architectures and converted into lighter architectures for efficient inference on edge devices. This would also enable controlled exploration of inductive biases by comparing architectures without confounding optimization difficulty.

*Corresponding author. Code: https://github.com/vsubramaniam851/net-convert

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Large-scale analyses demonstrate that very different architectures (e.g. CNNs vs Transformers) converge toward similar internal representations as they scale [23, 17, 27, 9]. Furthermore, classic observations of representational alignment across independently trained networks [39, 26] suggest that sufficiently expressive, distinct architectures can implement the same input–output functions. This statement of universality is also supported by theoretical work that shows that functions that are Turing-computable can be approximated by any neural network [36]. However, the real challenge is not whether such functions exist, but how to reach them through optimization. This view implies that architectural priors are not constraints that must persist at inference, but are primarily training biases – scaffolds designed to guide and stabilize optimization.

Motivated by these perspectives and potential benefits, we introduce *Network of Theseus* (NoT), a part-by-part conversion procedure that starts from a given guide architecture and progressively replaces its components (e.g. layers) with target modules. Our name for this procedure is a reference to Plutarch's Ship of Theseus paradox [35], which asks: once all the decaying planks of a ship are replaced (and none of the original remain), is it still the same ship?

At each replacement stage, we instantiate a target module and optimize it to match the activations of the guide network using a representational distance function [26, 11, 10]. This optimization transfers priors between networks at a lower level, e.g. at the layer level rather than at the architectural level [45]. After the conversion is complete, the resultant target architecture is trained end-to-end on the downstream task. This staged alignment decouples training from deployment: we can train with one architecture and convert it into an entirely different one for test-time use, aiming to preserve the original performance as closely as possible.

Using NoT, we show broad architectural conversions including convolutional (CNN) models to fully connected MLPs with low-rank linear layers, vision transformers with multihead attention to token-wise MLPs, and transformer language models to Elman RNNs. Interestingly, we observe even untrained guide networks contain useful inductive biases; NoT transfers these effectively, yielding performance comparable to using trained guides. Furthermore, we find that NoT can convert from larger to smaller networks and that converting from a larger to a smaller architecture (e.g., a deeper ResNet to a shallower one) can at minimum preserve performance. We show our conversions using centered kernel alignment (CKA) and further validate our findings by introducing a new kernel-based similarity metric, differentiable mutual nearest neighbors (D-MNN). Designed from the metric introduced in Huh et al. [23], D-MNN exploits local geometric structure for comparing representations.

NoT has both practical and theoretical implications for architectural transfer and design. With trained or untrained guides, we can convert to architectures that are more efficient or better aligned with deployment constraints or theoretical desiderata. Because NoT aims to preserve performance during conversion, it enables discovery of inference architectures with accuracy–efficiency trade-offs not previously attained, including transfers from larger (even untrained) guides to smaller targets. NoT implies that designing and training an architecture with simple optimization dynamics is not necessary: we can now design and deploy neural networks where constraints matter, expanding architectural choice at inference time without resolving the full supervised learning problem from scratch for every candidate design.

Our contributions are as follows:

1. We introduce the Network of Theseus (NoT), a method to part-by-part convert from one architecture to another using representational alignment.
2. We demonstrate NoT across a wide variety of architectural conversions such as converting convolutions in ResNet-18 to linear layers to create an MLP or converting attention in GPT-2 to RNN layers to create a Deep RNN. We find that across all conversions, we preserve performance of the original architecture despite dramatic architectural shifts.
3. Surprisingly, we find that we can perform the same conversion starting from an untrained network architecture, demonstrating that we can transfer inductive biases to our target networks.
4. We validate our findings using representational similarity metrics like CKA and introduce a new metric based on a differentiable version of mutual nearest neighbors, D-MNN, to further validate our findings as general and invariant to metric.

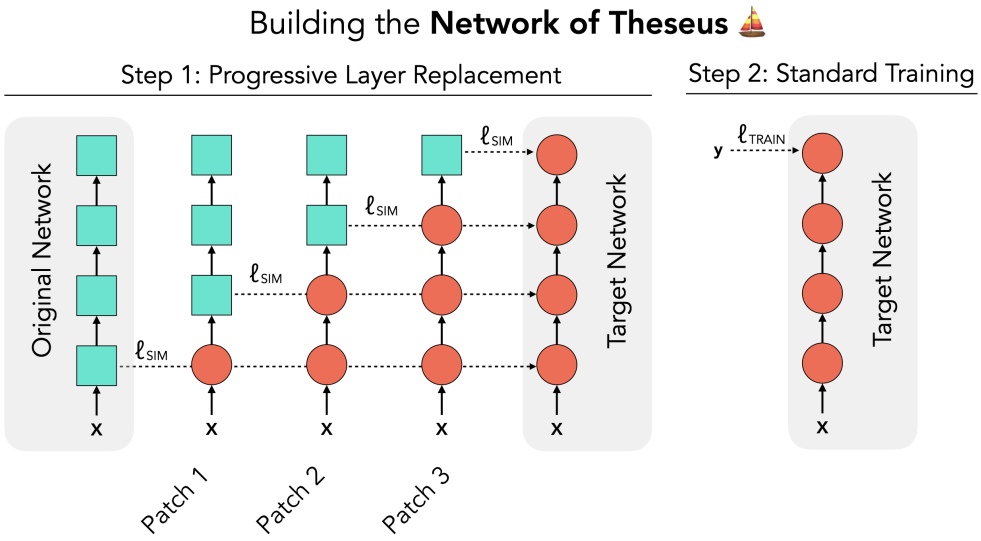

Figure 1: **Network of Theseus**. A network can be converted to any desired target network by replacing each piece of the original network incrementally, part-by-part. Each original part is replaced by optimizing the representational alignment, ℓ_{SIM}, of the target part to the original part. After all original parts are replaced, only the target network remains and can be trained on any downstream task (i.e. standard training).

2 Related Work

Architectural Transfer via Linearizing Transformers: NoT is a general method that builds on work applying cross-architectural distillation for linearizing transformers [32, 57, 58, 4]. These works focus on specifically distilling multihead attention to either SSMs [52, 5] or linear attention. Most approaches involve training linear attention layers or SSMs to approximate softmax attention via MSE. Some show that linearizing can specifically scale to very large transformers [57] e.g. 7B - 72B LLMs without considerable effect on performance.

These papers are significantly different from NoT, both in scope and approach. We greatly expand architectural transfer with linear attention or SSMs to transfers that are not dependent on having the same computations in order to use MSE. Our representational distance functions allow for any architectural pairing as well, making our conversions much richer. While previous work linearized with only linear attention, we can now focus on other layer replacements such as converting convolutional layers to linear layers, etc. NoT also makes methodological changes to how layer swaps are applied, considering changes that are staged and progressive rather than all-at-once. Our comparisons show that this progressive replacement is important, especially when architectures are different.

Model Distillation and Compression: Distillation [20, 15, 43, 21, 46, 8, 28] transfers knowledge from a teacher model to a student model by introducing a new component to the loss function that forces the student model to behave like the teacher model [25, 61]. This usually consists of penalizing the KL-divergence between the logit predictions of the student and teacher model. Methods have been proposed that use CKA as an alignment approach between representations of two networks or with representations in the brain with notable improvement in network performance [42, 12].

Other works also transfer knowledge across models while compressing information. For example, BERT-of-Theseus [53] uses the "Ship-of-Theseus" framing to compress BERT into a smaller architecture using stochastic replacement schedule during supervised training. NoT significantly expands on BERT-of-Theseus to include multiple replacement schedules, a wider range of architectures, and untrained starting architectures. Similarly, other papers gradually convert layers to identity functions in order to shrink the given architecture [7], where adjacent convolutions are merged for inference. This uses a within-architecture training trick to prune non-linearities and fuse layers.

We distinguish NoT from distillation and compression. NoT aligns computations and can transfer inductive biases from one architecture to another. Like prior work [45, 48, 60], this allows us to use completely untrained guide networks as part of our conversion, although our replacement strategy in

NoT is distinct from this previous work. Doing the same with distillation would lead to much worse results [45]. NoT is significantly different from previous work using feature kernels to convert across architectures [45, 18]. To our knowledge, the stage-wise replacement of NoT has not been proposed in previous work. These works do not use representational alignment as an initialization method.

Representational Convergence: Our paper builds on prior investigations into representational alignment across different network architectures. Prior work has seen both behavioral and representational convergence across diverse neural network architectures. For example, prior work has found that networks trained with self-supervised and supervised have similar errors, indicating behavioral alignment [14]. Many other prior works have discovered similar representational alignment through different similarity metrics such as kernel alignment [23, 17] and model stitching [3]. These prior works have discovered representational convergence across different architectures, learning tasks, and even modalities. This representational convergence across architectures and learning tasks have been exploited in prior work to improve network performance. For example, representational alignment is used to fix architectures that are traditionally ill-suited for certain tasks due to optimization failures or diffusion models with inconsistent representations [55]. Similar to these, we use representational convergence in NoT to convert across architectures progressively.

3 Methods: Network of Theseus (NoT)

Network of Theseus (NoT) provides a general procedure for replacing parts of a source ("guide") network with parts of a target architecture while preserving internal representations. At any time, a subset of components is replaced; we then train the replaced subset to *match* the guide's activations, stage by stage, according to a **replacement schedule**. Figure 1 illustrates our default schedule, **progressive replacement**, which we find works best empirically. Figure 2 compares alternative schedules. NoT is task-agnostic. After the replacement completes, we finetune end-to-end for the downstream objective (e.g., image classification or language modeling).

Let $f^G(\cdot)$ be a fixed guide network with k layers and let $f^T(\cdot;\theta)$ denote the under-construction target network obtained by replacing some of the guide's layers with new modules (e.g., Conv2d→Linear, attention variants, or block-level substitutions). We assume that both networks share input/output interfaces so that they can be run on the same inputs $\boldsymbol{x} \in \mathbb{X}$. For a mini-batch from train, let $\boldsymbol{A}_i^G(\boldsymbol{x})$ and $\boldsymbol{A}_i^T(\boldsymbol{x};\theta)$ be the activations extracted at layer i from f^G and f^T, respectively. In practice, for similarity computation, we flatten spatial/token dimensions.

We write $g(\cdot,\cdot) \in [0,1]$ for a representational similarity (e.g., linear CKA) and use its complement $\Delta(\boldsymbol{A},\boldsymbol{B}) \triangleq 1 - g(\boldsymbol{A},\boldsymbol{B}) \in [0,1]$ as a **dissimilarity** to be minimized.

3.1 Part-by-part matching objective

For illustrative purposes, we show how to apply NoT for a single layer i. The matching loss is $\mathcal{L}_i(\boldsymbol{\theta}_i) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\Delta\big(\boldsymbol{A}_i^T(\boldsymbol{x};\boldsymbol{\theta}_i), \boldsymbol{A}_i^G(\boldsymbol{x})\big)\right]$ where θ_i are the parameters introduced at location i by the replacement. At a training *stage*, we optimize a set $I \subseteq \{1,\cdots,k\}$ of replaced layers jointly:

$$\mathcal{L}_I(\boldsymbol{\theta}_I) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{1}{|I|}\sum_{i\in I}\Delta\big(\boldsymbol{A}_i^T(\boldsymbol{x};\boldsymbol{\theta}_I), \boldsymbol{A}_i^G(\boldsymbol{x})\big)\right]. \tag{1}$$

Layers not in I are frozen to prevent drift in the guide pathway.

Replacement Modules: NoT is agnostic to the specific replacement, requiring only shape compatibility. For Conv2d, we use an *equivalent low-rank linear layer* that applies to a flattened input and reshapes back to the expected output. Because of a large parameter count for the target linear layer, we first apply a linear layer to project the input to a lower dimensionality and then use another linear layer to project the desired output dimensionality, both of which are tunable. For attention, we support replacements with basic tokenwise MLPs, without any token-mixing, or RNNs. The framework also accommodates block-level replacements e.g., ResNet-50→ ResNet-18 basic blocks.

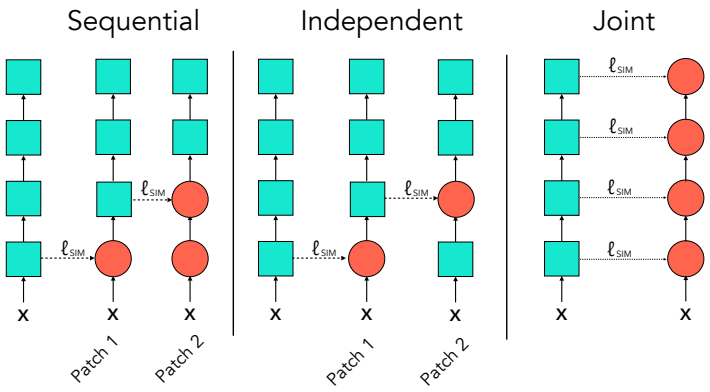

Figure 2: **Alternative replacement schedules**: Sequential freezes earlier replacements. Independent fits each layer using guide inputs. Joint trains all target layers at once.

3.2 Replacement Schedules

A **replacement schedule** is a sequence $\mathbb{S} = (I_1, \cdots, I_T)$ of index sets indicating which layers are trainable at each stage. NoT supports several schedules; we adopt **progressive replacement** by default due to its favorable stability–accuracy tradeoff.

Progressive: At stage t, we replace the t-th layer and set $I_t = \{1, \cdots t\}$ jointly optimizing all previously replaced layers together with the new one. This minimizes error accumulation while x feature distributions calibrated across the replaced prefix (Figure 1).

Sequential: Identical to progressive in the order of replacement, but we *freeze* all previously replaced layers and optimize only the newest one, i.e., $I_t = \{t\}$ with earlier replacements fixed. This can suffer from error accumulation because misalignment in earlier layers propagates forward (Figure 2).

Independent: Each layer i is optimized *in place* using the guide's input distribution at that layer, without coupling to other replacements. After fitting all layers independently, we assemble the full target. This avoids sequential error accumulation but introduces distribution shift once the independently trained layers are composed (Figure 2).

Joint: Replace all k layers at once and set $I_1 = \{1, \cdots, k\}$. This removes schedule bias but can be harder to optimize because later layers must adapt to rapidly changing early layers (Figure 2). This is closest to [45] although our application is closest to an initialization.

Group-wise variants: Beyond scalar layer indices, I_t can denote *groups* (e.g., a ResNet group). Our approach supports progressive group introduction (e.g., ResNet-50 $\rightarrow$ ResNet-18) as well as progressive *alignment* where a smaller student unfreezes groups over time to match a fixed teacher.

3.3 Similarity choices and Tasks

We instantiate g with **linear CKA** or **differentiable mutual nearest neighbors (D-MNN)**, both of which are scale-invariant across rows (mini-batch samples). D-MNN is a representational similarity metric designed and introduced in this paper based on the nearest neighbor metric used in Huh et al. [23]. In this work, we were inspired by the locality property of the metric and aimed to make it differentiable in the hope that local structure in the representation space has structure that can be exploited and aligned. We refer to Appendix B.1 and Appendix B.2.

4 Experiments

We apply NoT across a range of architectures and replacements to measure how well we can use our staged replacement across different replacement schedules. For all experiments, we compare using a trained and untrained guide network and separate training for representational similarity and task training into two stages.

Tasks: We consider image classification and language modeling in this paper. For testing image classification, we use the ImageNet dataset [13], measuring Top-1 performance on the pre-defined

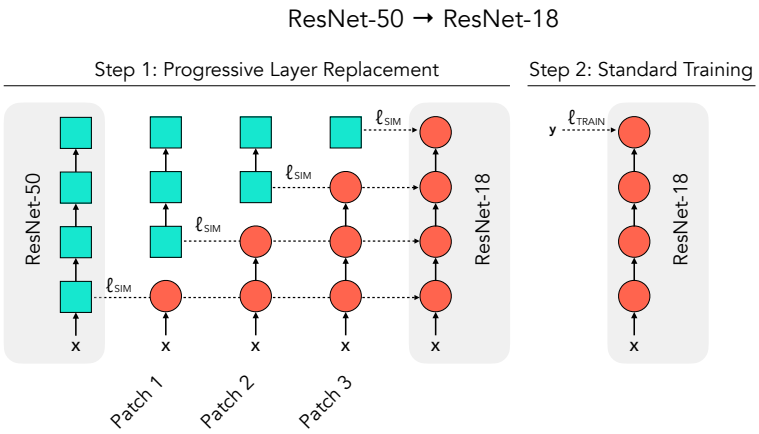

Figure 3: **NoT makes any-to-any architecture conversion possible.** Without the requirement of functional intermediate models, the target network can be any architecture. For instance, we convert the deeper ResNet-50 to the shallower ResNet-18. Blue squares are a set of multiple ResNet-50 blocks. These are replaced with a single ResNet-18 block. ResNet-18 blocks are added until we have a full ResNet-18 network.

validation set. For language modeling, we use the Wikitext-103 dataset [33] where models must predict the next token given some context. We use a sequence length of 128 for all models and use the training, validation and testing sets defined by the dataset. We tokenize the text using the GPT-2 [38] tokenizer.

Architectures and Replacements: We study cross-family and within-family conversions spanning convolutions, attention, and recurrent computation. In all cases, layers are progressively replaced and optimized to match the guide's intermediate activations via representational similarity.

ResNet-18→MLP: We convert ResNet-18 [19] to a fully-connected MLP by replacing each convolutional layer with a linear layer. This removes the spatial priors of convolutional layers such as receptive fields or translational equivariance. We use two low-rank linear layers (rank 256-1024) to keep the parameter size of the linear layers reasonable, as discussed in Section 3.1. Batch normalization recalibration details are in Appendix B.

DINOv2→Patch-MLP: We convert DINOv2 [34] to a patch-wise MLP by replacing each multihead attention with a token-wise MLP that applies feedforward layers independently to each visual token. This eliminates cross-token communication, increasing efficiency at the cost of token interaction.

GPT-2→RNN: We replace GPT-2 [38, 49] multihead attention layers with two Elman RNN layers per attention block. This tests whether sequential memory-based processing can substitute for parallel attention mechanisms.

ResNet-50→ResNet-18: We study architectural transfer rather than direct replacements that require matching input and output shapes. We map every four ResNet-50 blocks to one ResNet-18 block (blocks follow [19]), see Figure 3. We rebuild ResNet-18 block by block, independent of the guide architecture. At each stage the new block learns to match the representations of its four source blocks, testing whether NoT transfers depth based expressivity to a shallower model. This setting shows NoT can operate across any architectural conversion. See Appendix C.2 for details.

GPT-2 Large→GPT-2 Small: We conduct the same architectural transfer for language models. We map every three GPT-2 Large blocks to one GPT-2 Small block (blocks follow [38, 49] and correspond to transformer decoder layers). We rebuild GPT-2 block by block, independent of the guide architecture. At each stage, the new block learns to match the representations of its three blocks.

Training: For each setting, we train NoT with a guide network and the target replacements. We include a baseline with *naive replacement* where all guide network layers are replaced with the target modules from scratch for a fair comparison to understand what representational similarity provides to performance. The baseline is FLOP matched so this is trained with tasked training for the same number of optimization steps as progressive replacement, including alignment and standard training. All replacement schedules are also trained for the same total optimization steps as well, so they are

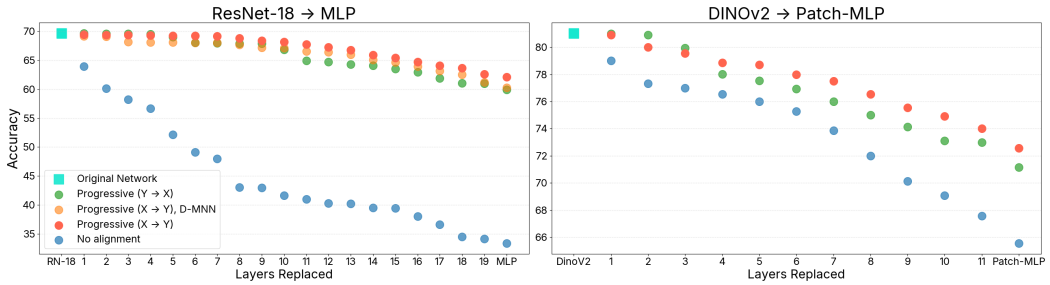

Figure 4: **Progressive layer replacement preserves performance.** Across patches, NoT aligns each replacement before finetuning the hybrid network, far exceeding naive replacement. Forward and reverse replacement perform similarly, and D-MNN is consistent with CKA in ResNet-18→MLP.

Original (Guide) → Target	Similarity Metric	Guide	NoT	Baseline
			ImageNet Top-1 Accuracy (↑)	
ResNet-18 → MLP	CKA	69.66	62.12±0.42 (↑22.11)	40.01±1.44
ResNet-18 → MLP	D-MNN	69.66	60.28±0.39 (↑20.27)	40.01±1.44
DINOv2 → Patch-MLP	CKA	81.03	72.56±0.20 (↑3.84)	68.72±0.55
DINOv2 → Patch-MLP	D-MNN	81.03	73.02±0.84 (↑4.30)	68.72±0.55
ResNet-50 → ResNet-18	CKA	76.13	73.81±0.26 (↑4.91)	68.90±0.33
ResNet-50 → ResNet-18	D-MNN	76.13	73.02 ±0.84 (↑4.12)	68.90±0.33
			Perplexity (↓)	
GPT-2 → RNN	CKA	37.50	50.58±0.89 (↓68.86)	119.44±3.01
GPT-2 → RNN	D-MNN	37.50	47.44±1.91 (↓70.00)	119.44±3.01
GPT-2 Large → GPT-2 Small	UCKA	22.05	24.98±1.52 (↓3.03)	28.01±1.02
GPT-2 Large → GPT-2 Small	D-MNN	22.05	26.02±1.22 (↓1.99)	28.01±1.02

Table 1: **Network of Theseus vastly improves over naive replacement and preserves performance**. We compare NoT with a FLOP-matched standard training and the original guide network performance. We find that NoT vastly outperforms naive replacement, close to 25% on ImageNet and 70 points on Wikitext-103.

FLOP-matched. During task training, all networks are trained with cross-entropy loss, without loss of generality. For all task training and representational similarity optimization, we use the AdamW [30] optimizer. When optimizing representational similarity, we also incorporate gradient clipping due to unstable training. We use CKA and D-MNN for all settings.

We use batch size 256 throughout because representational similarity estimates improve with more samples. Alignment learning rates and epochs are tuned per stage (15–100 epochs), and image-classification runs use warmup plus cosine scheduling [31]. Task training runs for 100 epochs with swept learning rates and 5–10 warmup epochs under the same scheduler; full hyperparameters are in Appendix C. After tuning, each setting is run with 5 seeds and we report standard errors. For Wikitext-103, test perplexity is taken from the epoch with the lowest seed-averaged validation loss.

5 Results

NoT significantly outperforms naive replacement: We apply NoT to all of our previously described networks and summarize the results in Table 1. We find that across all settings, NoT improves performance over naive replacement by up to 22%. For example, we find that fully-connected MLPs can only be trained to achieve 40% accuracy. With NoT, we improve by 22% and identify a fully-connected MLP that is competitive with ResNet-18. Similarly, we are able to replace attention in vision transformers with Patch-MLPs while preserving accuracy, meaning that token communication is not necessary for downstream image classification, achieving competitive results on DINOv2. This holds for Elman RNNs, which become effective replacements for attention computations. Most excitingly, we find that NoT can be applied across similar architectures like ResNet-18. These results far exceed naive replacement and standard training. We also find consistent results with D-MNN as with CKA.

Figure 4 tracks progressive replacement for ResNet-18 and DINOv2 against the guide and naive target. Forward and reverse schedules both outperform naive training. Reverse replacement is slightly

Original → Target	Similarity Metric	Progressive	Joint	Independent	Sequential
		ImageNet Top-1 Accuracy (↑)			
ResNet-18 → MLP	CKA	62.12±0.42	57.98±0.28	45.10±0.23	30.73±0.14
ResNet-18 → MLP	D-MNN	60.28±0.39	57.66±0.22	45.81±0.06	31.90±0.33
DINOv2 → Patch-MLP	CKA	72.56±0.20	70.42±0.33	66.02± 0.47	47.15±0.32
DINOv2 → Patch-MLP	D-MNN	73.02±0.84	71.69±0.94	69.88±0.13	51.94±1.55
ResNet-50 → ResNet-18	CKA	73.81±0.26	70.38±0.35	61.51±0.46	61.08±0.15
ResNet-50 → ResNet-18	D-MNN	73.02 ±0.84	69.12±1.10	60.87±0.11	62.04±0.92
		Perplexity (↓)			
GPT-2 → RNN	CKA	50.58±0.89	57.69±0.58	100.94±1.15	121.36±2.23
GPT-2 → RNN	D-MNN	47.44±1.91	56.66±1.55	115.96±1.65	136.98±0.54
GPT-2 Large → GPT-2	UCKA	24.98±1.52	25.04±1.33	33.66±0.94	29.62±2.36
GPT-2 Large → GPT-2	D-MNN	26.02±1.22	27.20±1.43	33.96±2.65	34.91±3.41

Table 2: **Progressive replacement outperforms other replacement schedules**: We compare progressive replacement with previously discussed replacement schedules, finding that our staged, progressive schedule is extremely useful. All replacement schedules are trained for the same FLOP budget. Joint replacement is the closest but likely requires significantly longer training.

Original → Target	Similarity Metric	**Guide**		Baseline
		Trained	Untrained	
		ImageNet Top-1 Accuracy (↑)		
ResNet-18 → MLP	CKA	62.12±0.42	60.85±0.48	40.01±1.44
ResNet-18 → MLP	D-MNN	60.28±0.39	60.90±0.69	40.01±1.44
DINOv2 → Patch-MLP	CKA	72.56±0.20	70.39±0.25	68.72±0.55
DINOv2 → Patch-MLP	D-MNN	73.02±0.84	68.15±0.96	68.72±0.55
ResNet-50 → ResNet-18	CKA	73.81±0.26	71.61±0.42	68.90±0.33
ResNet-50 → ResNet-18	D-MNN	73.02±0.84	70.22±1.18	68.90±0.33
		Wikitext Perplexity (↓)		
GPT-2 → RNN	CKA	50.58±0.89	58.26±0.43	119.44±3.01
GPT-2 → RNN	D-MNN	47.44±1.91	49.71±2.56	119.44±3.01
GPT-2 Large → GPT-2	UCKA	24.98±1.52	25.10±1.97	28.01±1.02
GPT-2 Large → GPT-2	D-MNN	26.02±1.22	26.15±2.43	28.01±1.02

Table 3: **NoT with untrained guide networks improves over naive replacement**. Can we apply NoT with untrained guide networks to transfer inductive bias? We find that untrained guide networks contain useful inductive biases and can improve over naive replacement with no alignment, given by baseline. NoT with untrained guide networks is competitive with NoT with trained guide networks.

weaker, likely from representational drift but still achieves strong performance. Drops at layers such as layer 13 of ResNet-18 indicate bottlenecks associated with downsampling layers.

NoT significantly outperforms training from scratch: Training MLPs on ImageNet remains difficult: Bachmann et al. [2] report 51.5% with heavy training and test-time augmentation. NoT achieves higher accuracy without such augmentation, highlighting the gap between functions an MLP can represent and those reached by direct optimization.

Alignment becomes more challenging at the end: Figure 5 shows CKA loss during ResNet-18→MLP alignment. Bottleneck layers such as 8 and 13, associated with downsampling in ResNet-18, are harder to align, and later layers are generally harder across reverse replacement, DINOv2, and GPT-2. Improved late-stage alignment is a clear target for future work.

CKA alignment predicts accuracy and improves with rank: Using checkpoints from the final progressive stage, Figure 6 shows that higher average CKA predicts higher task accuracy. Larger low-rank linear layers reduce CKA loss and improve downstream accuracy, suggesting further capacity or optimization can improve NoT.

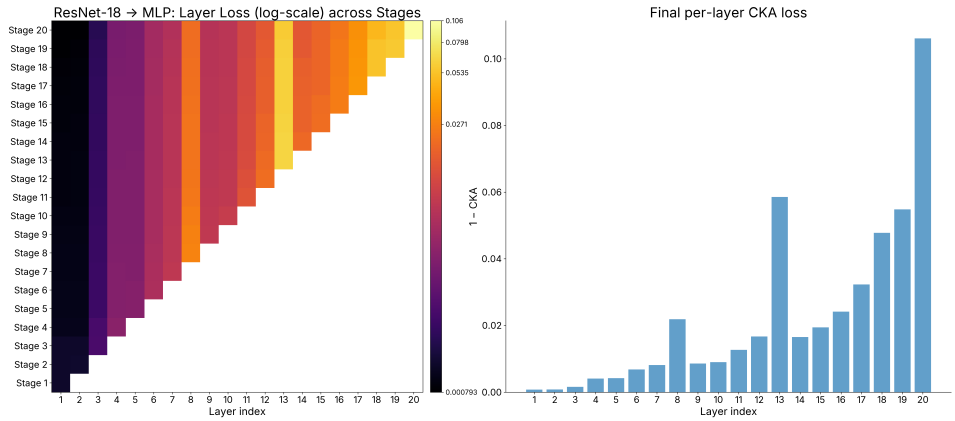

Figure 5: **Representational similarity across stages reveals difficult layers**: (left) We show CKA similarity losses (log-scaled) across all stages of progressive replacement. We see that CKA loss decreases across all stages for all layers. (right) We plot the final CKA loss for the last stage across all layers. We can identify bottle[illegible] higher loss.

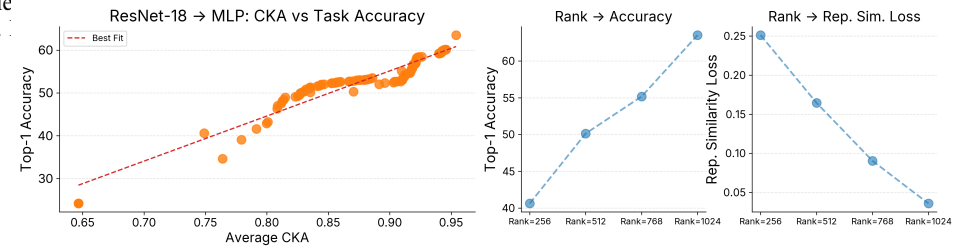

Figure 6: **NoT performance improves with larger CKA and larger ranks**: We compare final task performance of the ResNet-18→MLP setting over different final CKA alignment scores (left) and linear layer ranks (right). We find that stronger average alignment for the final stage leads to better performance. Similarly, larger rank leads to better loss and stronger task performance.

Progressive replacement outperforms other schedules: Table 2 compares progressive, joint, independent, and sequential schedules. Progressive is strongest across settings; joint is closest but likely needs more training, independent suffers composition-time distribution shift, and sequential accumulates error.

Untrained guide networks improve over naive replacements: Table 3 shows that NoT transfers useful priors from untrained guides [45, 48, 60]. This distinguishes NoT from distillation, which is weak with untrained teachers. For example, untrained ResNet-50→ResNet-18 improves the target by 3%, suggesting depth and connectivity can be transferable even without learned weights.

6 Conclusion

Training and inference are usually coupled, but Network of Theseus (NoT) breaks this coupling by progressively replacing a guide network with target modules while aligning intermediate representations; therefore, the trained function can be carried into a different inference architecture. Across ImageNet and Wikitext-103, this staged conversion preserves much of the guide's performance for large cross-family changes, specifically ResNet-18→MLP, DINOv2→Patch-MLP, ResNet-50→ResNet-18, GPT-2→RNN, whereas naive replacement collapses. Therefore, representational alignment is the key mechanism.

Regardless of whether the guide architecture was trained for a downstream task or not, we find that such staged replacement allows for an inductive bias transfer, allowing us to discover target architectures that preserve performance in comparison to the guide architecture. Our results raise concrete scientific questions: What defines a "discoverable representation", one that a different target architecture can reliably realize, and how can training objectives be designed to increase discoverability? Given that NoT decouples training from deployment, how should we jointly choose guide architectures (for favorable optimization geometry) and target architectures (for deployment desiderata such as latency, memory, and parallelism) to trace the optimal alignment–efficiency frontier?

We believe NoT reframes how we think about architectural design. If representational alignment can be used to carry performance across network architecture families, then the new goal of an architecture during training is to construct discoverable representations rather than dictating the deployed inference architecture.

Limitations: We have not fully explored the space of architectures in our current work and, more importantly, have not reached the boundary of the performance of our alignment procedure for two reasons. First, we have not reached a point of overfitting in Figure 6 that increased target network capacity continues to improve performance, indicating that further training is necessary to see such convergence. Second, this lack of coverage is due to computational limitations as the progressive process requires more resources that simply fitting all layers at once. This gap between our performance and the possible best performance is driven by certain bottleneck layers, e.g. optimizing the last layers as shown in Figure 5. Further training methods can be explored to improve these results such as more data augmentation.

In most cases, Network of Theseus does not drive target network to match or beat guide network performance. The remaining gap is likely driven by the capacity of the target network and what can learned. This gap is interesting because it is indicative of what is special about the current networks we are designing. Further research will be dedicated to tightening this gap.

Acknowledgments

This work was supported by the DARPA Mathematics for the DIscovery of ALgorithms and Architectures (DIAL) program, the DARPA Knowledge Management at Scale and Speed (KMASS) program, the NSF award 2124052, the MIT-IBM Watson AI Lab, the Department of the Air Force Artificial Intelligence Accelerator under Cooperative Agreement Number FA8750-19-2-1000, and the Air Force Office of Scientific Research (AFOSR) under award number FA9550-21-1-0399. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

We would like to thank Chris Wang, David Mayo, Phillip Isola and the rest of the Isola lab for their broad feedback on this work.

References

[1] Shaden Alshammari, John Hershey, Axel Feldmann, William T Freeman, and Mark Hamilton. I-con: A unifying framework for representation learning. *arXiv preprint arXiv:2504.16929*, 2025.

[2] Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling mlps: A tale of inductive bias. *Advances in Neural Information Processing Systems*, 36:60821–60840, 2023.

[3] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

[4] Aviv Bick, Kevin Li, Eric Xing, J Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *Advances in Neural Information Processing Systems*, 37:31788–31812, 2024.

[5] Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *arXiv preprint arXiv:2502.14458*, 2025.

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[7] Hanting Chen, Yunhe Wang, Jianyuan Guo, and Dacheng Tao. Vanillanet: the power of minimalism in deep learning, 2023. URL https://arxiv.org/abs/2305.12972.

[8] Liqun Chen, Dong Wang, Zhe Gan, Jingjing Liu, Ricardo Henao, and Lawrence Carin. Wasserstein contrastive representation distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16296–16305, 2021.

[9] Colin Conwell, Jacob S Prince, Kendrick N Kay, George A Alvarez, and Talia Konkle. A large-scale examination of inductive biases shaping high-level visual representation in brains and machines. *Nature communications*, 15(1):9383, 2024.

[10] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

[11] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

[12] Joel Dapello, Kohitij Kar, Martin Schrimpf, Robert Geary, Michael Ferguson, David D Cox, and James J DiCarlo. Aligning model and macaque inferior temporal cortex representations improves model-to-human behavioral alignment and adversarial robustness. *bioRxiv*, pages 2022–07, 2022.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Robert Geirhos, Kantharaju Narayanappa, Benjamin Mitzkus, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. On the surprising similarities between supervised and self-supervised models. *arXiv preprint arXiv:2010.08377*, 2020.

[15] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[16] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[17] Yena Han, Tomaso A Poggio, and Brian Cheung. System identification of neural systems: If we got it right, would we know? In *International Conference on Machine Learning*, pages 12430–12444. PMLR, 2023.

[18] Bobby He and Mete Ozay. Feature kernel distillation. In *International conference on learning representations*, 2022.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[21] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

[22] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. *Advances in Neural Information Processing Systems*, 35:33716–33727, 2022.

[23] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[24] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[25] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. *arXiv preprint arXiv:2105.08919*, 2021.

[26] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[27] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

[28] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33: 2351–2363, 2020.

[29] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[30] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[32] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

[33] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[34] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[35] Plutarch. *Parallel Lives*. 100–125. Composed ca. 100–125 CE.

[36] Tomaso Poggio and Maia Fraser. Compositional sparsity of learnable functions. *Bulletin of the American Mathematical Society*, 2024.

[37] PyTorch Contributors. ReduceLROnPlateau — PyTorch 2.9 documentation. https://docs.pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html, 2025. Accessed: 2025-11-28.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[39] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

[40] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[41] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[42] Aninda Saha, Alina Bialkowski, and Sara Khalifa. Distilling representational similarity using centered kernel alignment (cka). In *Proceedings of the the 33rd British Machine Vision Conference (BMVC 2022)*. British Machine Vision Association, 2022.

[43] V Sanh. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[44] Le Song, Alex Smola, Arthur Gretton, Karsten M Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, pages 823–830, 2007.

[45] Vighnesh Subramaniam, David Mayo, Colin Conwell, Tomaso Poggio, Boris Katz, Brian Cheung, and Andrei Barbu. Training the Untrainable: Introducing Inductive Bias via Representational Alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019.

[47] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272, 2021.

[48] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[49] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[50] Shashanka Venkataramanan, Amir Ghodrati, Yuki M Asano, Fatih Porikli, and Amirhossein Habibian. Skip-attention: Improving vision transformers by paying less attention. *arXiv preprint arXiv:2301.02240*, 2023.

[51] Jiahao Wang, Songyang Zhang, Yong Liu, Taiqiang Wu, Yujiu Yang, Xihui Liu, Kai Chen, Ping Luo, and Dahua Lin. Riformer: Keep your vision backbone effective but removing token mixer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14443–14452, 2023.

[52] Junxiong Wang, Daniele Paliotta, Avner May, Alexander Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. *Advances in Neural Information Processing Systems*, 37:62432–62457, 2024.

[53] Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.633.

[54] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[55] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[56] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):896–912, 2023.

[57] Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*, 2024.

[58] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

[59] Shuoxi Zhang, Hanpeng Liu, Stephen Lin, and Kun He. You only need less attention at each stage in vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6057–6066, 2024.

[60] Ziqian Zhong and Jacob Andreas. Algorithmic capabilities of random transformers. *Advances in Neural Information Processing Systems*, 37:104357–104382, 2024.

[61] Wangchunshu Zhou, Canwen Xu, and Julian McAuley. Bert learns to teach: Knowledge distillation with meta learning. *arXiv preprint arXiv:2106.04570*, 2021.

[62] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Appendix Overview

> The ship wherein Theseus and the youth of Athens returned had thirty oars, and was preserved by the Athenians down even to the time of Demetrius Phalereus, for they took away the old planks as they decayed, putting in new and stronger timber in their place, insomuch that this ship became a standing example among the philosophers, for the logical question of things that grow; one side holding that the ship remained the same, and the other contending that it was not the same.
>
> –Plutarch

We present additional details of NoT, experiments and analysis, as well as additional results. In Appendix B, we provide further details of NoT, providing an overview of the method and covering the similarity metrics applied in this paper, CKA and D-MNN in detail. In Appendix C, we provide additional details on our experiments, going into further detail on training and evaluation. We cover our batch normalization recalibration as well as general details on our ResNet-50→ResNet-18 result. In Appendix E, we recreate Figure 5 with D-MNN for further analysis with D-MNN. In Appendix I, we compare hidden-state adapters and logit distillation. In Appendix J, we report inference latency measurements. In Appendix K, we provide further analysis of DINOv2 to investigate why we can remove cross-token calculations.

B Methods Overview

We give an overview of NoT in Algorithm 1 and cover more methodological details here.

Algorithm 1 Network of Theseus (NoT): staged representational alignment and conversion

Require: Guide $f^G(\cdot;\theta^G)$ (frozen), target f^T (init as f^G), replacement mapping $\mathcal{R}$ (shape-compatible), schedule $\mathcal{S} = (I_1, \ldots, I_T)$ (default progressive), unlabeled inputs $\mathcal{X}$, labeled data $\mathcal{D} = \{(x, y)\}$, similarity g (e.g., CKA); define $\Delta(\boldsymbol{A}, \boldsymbol{B}) = 1 - g(\boldsymbol{A}, \boldsymbol{B})$

for $t = 1$ to T **do** ▷ Alignment stage t
  # Replace layer t, assume previous layers have been replaced
  $J_t \leftarrow I_t \setminus I_{t-1}$; **replace** $\{i \in J_t\}$ in f^T with $\{\mathcal{R}(i)\}$ (initialize)
  freeze $\theta^T_{I_t}$ and **unfreeze** $\theta^T_{I_t}$ ▷ no drift outside replaced set
  for mini-batches $x \sim \mathcal{X}$ **do**
    For each $i \in I_t$: $\boldsymbol{A}_i^G \leftarrow$ activations of layer i in $f^G(x)$; $\boldsymbol{A}_i^T \leftarrow$ activations of layer i in $f^T(x)$
    $L_{\text{align}} \leftarrow \Delta(\boldsymbol{A}_i^T, \boldsymbol{A}_i^G)$
    Update $\theta^T_{I_t}$ to minimize L_{align}
  end for
end for
Fine-tune: unfreeze all θ^T and minimize $L_{\text{task}}(f^T(x), y)$ over $(x, y) \in \mathcal{D}$
return converted target f^T

B.1 Centered Kernel Alignment

To compare representations, we use a representation similarity metric, $\mathcal{M}$, which corresponds to centered kernel alignment (CKA) [26, 10, 11] in our setting. We specifically consider linear CKA.

CKA uses kernel functions on mean-centered representations to compute representational similarity matrices, which are then compared via the Hilbert-Schmidt Independence Criterion (HSIC). More specifically, suppose we have two sets of representations $\boldsymbol{A} \in \mathbb{R}^{b \times d_1}$ and $\boldsymbol{B} \in \mathbb{R}^{b \times d_2}$. We first compute the Gram matrices for each set of representations

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{A}^T, \boldsymbol{L} = \boldsymbol{B}'\boldsymbol{B}'^T \tag{2}$$

We center the Gram matrices by introducing a matrix, H, where $H = \boldsymbol{I}_b - \frac{1}{b}\mathbf{1}\mathbf{1}^T$.

$$\tilde{\boldsymbol{K}} = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}, \tilde{\boldsymbol{L}} = \boldsymbol{H}\boldsymbol{L}\boldsymbol{H} \tag{3}$$

We compute the HSIC on the Gram matrices.

$$HSIC(\boldsymbol{K}, \boldsymbol{L}) = \text{tr}(\tilde{\boldsymbol{K}}, \tilde{\boldsymbol{L}}) \tag{4}$$

Finally, we define our linear CKA metric as:

$$g_{\text{CKA}}(\boldsymbol{A}, \boldsymbol{B}') := \text{CKA}(\boldsymbol{K}, \boldsymbol{L}) = \frac{HSIC(\boldsymbol{K}, \boldsymbol{L})}{\sqrt{HSIC(\boldsymbol{K}, \boldsymbol{K}) * HSIC(\boldsymbol{L}, \boldsymbol{L})}} \tag{5}$$

In our setting, we consider representational *dissimilarity* and aim to minimize the dissimilarity between representations from our target network and guide network. We define this as:

$$\Delta_{\text{CKA}}(\boldsymbol{A}, \boldsymbol{B}) = 1 - g(\boldsymbol{A}, \boldsymbol{B}) \tag{6}$$

Linear CKA ranges from 0 to 1 (very different representations). Because of this, we take the complement by subtracting the linear CKA from 1 to represent dissimilarity. Most importantly, before computing g, rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ are ℓ_2-normalized.

B.1.1 Unbiased CKA

We use an unbiased estimator for HSIC for our GPT-2 Large→GPT-2 Small experiments, motivated by findings covered in Appendix D. We cover the unbiased HSIC here.

We start with our Gram matrices, $\boldsymbol{K}, \boldsymbol{L}$. An unbiased estimator for HSIC then involves forming zero-diagonal copies of both matrices.

$$\tilde{\boldsymbol{K}} \coloneqq \boldsymbol{K} - \text{diag}(\boldsymbol{K}), \qquad \tilde{\boldsymbol{L}} \coloneqq \boldsymbol{L} - \text{diag}(\boldsymbol{L}), \tag{7}$$

so that $\tilde{\boldsymbol{K}}_{ii} = \tilde{\boldsymbol{L}}_{ii} = 0$ and $\tilde{\boldsymbol{K}}_{ij} = \boldsymbol{K}_{ij}$, $\tilde{\boldsymbol{L}}_{ij} = L_{ij}$ for $i \neq j$. Let $\mathbf{1} \in \mathbb{R}^m$ be the all-ones vector. The unbiased estimator of HSIC is

$$HSIC_U(K, L) = \frac{1}{m(m-3)} \left[\text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}) + \frac{\text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}})}{(m-1)(m-2)} - \frac{2}{m-2} \text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}) \right], \tag{8}$$

which is well-defined for $m \geq 4$. The last term corresponds to the unbiased correction.

B.2 Differentiable Mutual Nearest Neighbors

For Differentiable MNN (D-MNN), we use Huh et al. [23] and Alshammari et al. [1] to reformulate nearest neighbors into a differentiable form.

Let b be the mini-batch size and anchors $i \in \{1, \ldots, b\}$. Two encoders f_1, f_2 produce features $\phi_i = f_1(x_i) \in \mathbb{R}^d$, $\psi_i = f_2(y_i) \in \mathbb{R}^d$. Define $\mathcal{J}_i = \{1, \ldots, b\} \setminus \{i\}$. For $k \in \mathbb{N}$, let $s_{f_1}(i), s_{f_2}(i) \subset \mathcal{J}_i$ be the index sets of the k nearest neighbors of i under $\{\phi_j\}$ and, $\{\psi_j\}$, respectively.

k-Nearest Neighbors: As defined by Huh et al. [23], the overlap score for anchor i is

$$g_{NN}(\phi_i, \psi_i) = \frac{|s_{f_1}(i), s_{f_2}(i)|}{k} \tag{9}$$

and the batch score is the average over i.

From overlap to probabilities: We introduce hard conditional distributions uniform on the kNN sets:

$$p_{f_1}(j \mid i) = \frac{\mathbf{1}\{j \in s_{f_1}(i)\}}{k}, \qquad q_{f_2}(j \mid i) = \frac{\mathbf{1}\{j \in s_{f_2}(i)\}}{k}, \quad j \in \mathcal{J}_i. \tag{10}$$

Then

$$\sum_{j \in \mathcal{J}_i} p_{f_1}(j \mid i)\, q_{f_2}(j \mid i) = \frac{|s_{f_1}(i) \cap s_{f_2}(i)|}{k^2}, \quad \Rightarrow \quad g_{NN}(\phi_i, \psi_i) = k \sum_{j \in \mathcal{J}_i} p_{f_1}(j \mid i)\, q_{f_2}(j \mid i). \tag{11}$$

Differentiable conditional neighborhoods: Let $\tau > 0$ be a temperature. Define pairwise similarities either by

$$s_{ij}^f = -\frac{\|\phi_i - \phi_j\|_2^2}{\tau} \quad \text{or} \quad s_{ij}^f = \frac{\phi_i^\top \phi_j}{\tau}, \qquad j \in \mathcal{J}_i, \tag{12}$$

and analogously s_{ij}^g from $\{\psi_j\}$. Let $T_k^f(i) \subset \mathcal{J}_i$ and $T_k^g(i) \subset \mathcal{J}_i$ denote top-k index selections under s_{ij}^f and s_{ij}^g, respectively. We define masked, normalized conditionals over the top-k:

$$P_{ij} \equiv p_{f_1}(j \mid i) = \frac{\exp(s_{ij}^{f_1})\,\mathbf{1}\{j \in T_k^{f_1}(i)\}}{\sum_{j' \in T_k^{f_1}(i)} \exp(s_{ij'}^{f_1})}, \qquad Q_{ij} \equiv q_{f_2}(j \mid i) = \frac{\exp(s_{ij}^{f_2})\,\mathbf{1}\{j \in T_k^{f_2}(i)\}}{\sum_{j' \in T_k^{f_2}(i)} \exp(s_{ij'}^{f_2})}. \tag{13}$$

(Any differentiable soft-top-k operator may be used; with an exact mask and $\tau \to 0$, P_{ij} and Q_{ij} converge to the hard uniforms.)

D-MNN alignment and limit: We define the differentiable alignment for anchor i as the inner product

$$g_{\text{soft}}(\phi_i, \psi_i) \;=\; \sum_{j \in \mathcal{J}_i} P_{ij}\, Q_{ij}, \qquad \bar{g}_{\text{soft}} \;=\; \frac{1}{b} \sum_{i=1}^{b} m_{\text{soft}}(\phi_i, \psi_i). \tag{14}$$

Under the exact top-k mask and $\tau \to 0$, $P_{ij}, Q_{ij} \to \frac{1}{k}\mathbf{1}\{j \in s_{f_1}(i)\}, \frac{1}{k}\mathbf{1}\{j \in s_{f_2}(i)\}$, so that $k\, g_{\text{soft}}(\phi_i, \psi_i) \to g_{NN}(\phi_i, \psi_i)$ by equation 11.

Training objective: Excluding the anchor, we write $P_{i\setminus *} = \{P_{ij}\}_{j \in \mathcal{J}_i}$ and similarly for Q. When P (from f) guides Q (from g), we minimize

$$\Delta_{\text{D-MNN}} \;:=\; \mathcal{L}_{KL} \;=\; \frac{1}{b} \sum_{i=1}^{b} D_{KL}\Big(P_{i\setminus *} \,\Big\|\, Q_{i\setminus *}\Big). \tag{15}$$

This KL aligns conditional neighborhood distributions. Most importantly, before computing g, rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ are ℓ_2-normalized. For our setting in this paper, we have to tune the temperature for D-MNN.

B.3 Methodology Limitations

NoT has a number of limitations as covered previously. We go deeper into methodological limitations here. NoT can be expensive memory-wise and runtime-wise. Each stage of NoT requires independent tuning. We also find that training can take a significant amount of time. For some conversions, we ran some conversions for up to 50 epochs of training for the best conversion. We believe these limitations may be overcome with further training optimizations and techniques such as more rich data augmentation and stronger label smoothing as used in previous papers [2].

C Experimental Details

All training experiments were done on 8 H100 GPUs over 2 months in total. GPU optimization techniques were taken such as gradient accumulation and gradient checkpointing and some experiments with converting ResNet-18 to an MLP used mixed-precision training.

In Table 4, we show a table of learning rates used at each stage of training for our replacements. We also show our task training details in Table 5.

C.1 Handling BatchNorm

A difficulty with NoT is handling networks which use batch normalization, a common design choice in convolutional networks. However, in NoT, we find that batch normalization requires additional design considerations to ensure proper architectural transfer. For example, batch-feature statistics may shift for the replaced layers, meaning that batch normalization may need changes. We find that recalibrating batch normalization during representational similarity optimization has adverse effects on downstream performance.

To mitigate this, we freeze batch normalization calculations in the guide network during replacement. This also makes the replacement more difficult for the target modules. After representational similarity optimization, we recalibrate the batch normalization layers in the resultant target architecture before task training. The recalibration only focuses on tuning the batch normalization layers before doing target training. We find that this separation is useful, changing performance by a significant amount.

C.2 ResNet-50 Replacement

Beyond direct layer/block replacements, we generalize NoT across architectures to convert from one architecture to another more generally. To do this, we relax the constraint requiring replaced components of our guide network to functionally fit into the guide network i.e. the replaced target module must accept the correct input and return the correct output. Having this constraint was necessary for Figure 4 to train different hybrid networks as we replace layers in the guide network.

See Figure 3. ResNet-50 blocks can be replaced with ResNet-18 blocks without incorporation into the full architecture. A full hybrid architecture with ResNet-18 and ResNet-50 blocks is unnecessary in this setting since the main goal is to reconstruct ResNet-18 while optimizing representational similarity with ResNet-50. Our findings shown in Table 1 and Table 3 are exciting because it demonstrates that we can make conversion across architectures general. For example, in Figure 3, ResNet-50 can be replaced with any architecture like GPT-2, and this can be converted to a smaller transformer or a smaller RNN.

D Biased vs Unbiased CKA: GPT-2 Large→GPT-2 Small

In our GPT-2 Large→GPT-2 Small setting, we found that using biased CKA as our alignment metric between GPT-2 Small transformer blocks and three GPT-2 Large transformer blocks did not have significant improvement. This is explained by inflated biased CKA values between GPT-2 Large blocks and GPT-2 Small blocks. We found that starting CKA values between randomly initialized GPT-2 Small blocks and trained GPT-2 Large blocks started at 0.98, leading to very small optimization gains from optimizing CKA and very small improvements on downstream performance with Wikitext-103.

To better understand this trend, we plotted the distribution of values in our Gram matrices as a histogram in Figure 7. These plots correspond with visualizing the Gram matrices for 5 batches of 256 examples. The first two columns correspond to a Gram matrix as described by Equation (2) and the third columns corresponds to the product across the Gram matrices of the two models used by HSIC. We can see that the Gram matrices are heavily skewed with some values appearing to be much higher. These values correspond with the self-similarity in the networks, leading to inflated CKA scores. To mitigate this, we removed computing the HSIC with the diagonal using unbiased CKA [44], which we discussed in Appendix B.1.1.

We applied UCKA in other experimental settings but found either degraded or comparable results to using CKA.

E D-MNN Staged Alignment

We recreate Figure 5 for our new metric, D-MNN (Appendix B.2) to analyze loss dynamics. We show results in Figure 8. We find that the quantitative loss for D-MNN is much higher and farther from 0, indicating little saturation. This is likely because D-MNN prioritizes locality over global geometry when comparing representations. We believe this demonstrates the benefit of our method.

F Reversing NoT Architecture Replacement

A potential concern is that representational alignment is harmful when the guide network is poorly chosen. In order to test target network sensitivity to a specific guide network, we add two additional experiments. In the first, we replace linear layers in an MLP with convolutional layers from ResNet-18, effectively reversing our ResNet-18→MLP experiment. In the second, we repeat this with ResNet-18 with ResNet-50, where we replace a ResNet-18 with several ResNet-50 block. Our

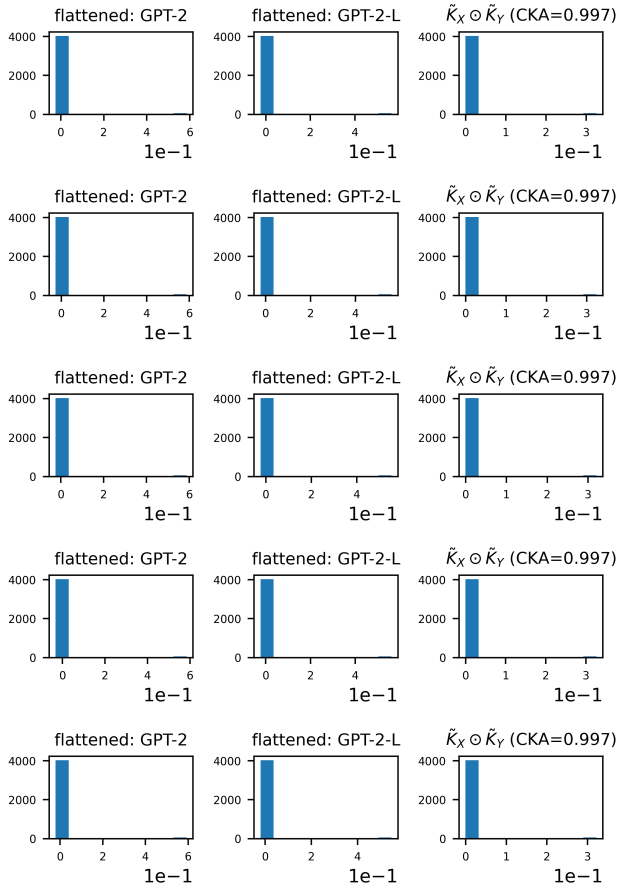

Figure 7: **Biased CKA inflated similarity scores between GPT-2 Small and GPT-2 Large.** When plotting values in the Gram matrices of a random pair of layers from GPT-2 Small and GPT-2 Large, we find that the Gram matrix values are very skewed. This is lead by inflated diagonal scores and skewed Gram matrix comparisons. We fix this using an unbiased HSIC estimator.

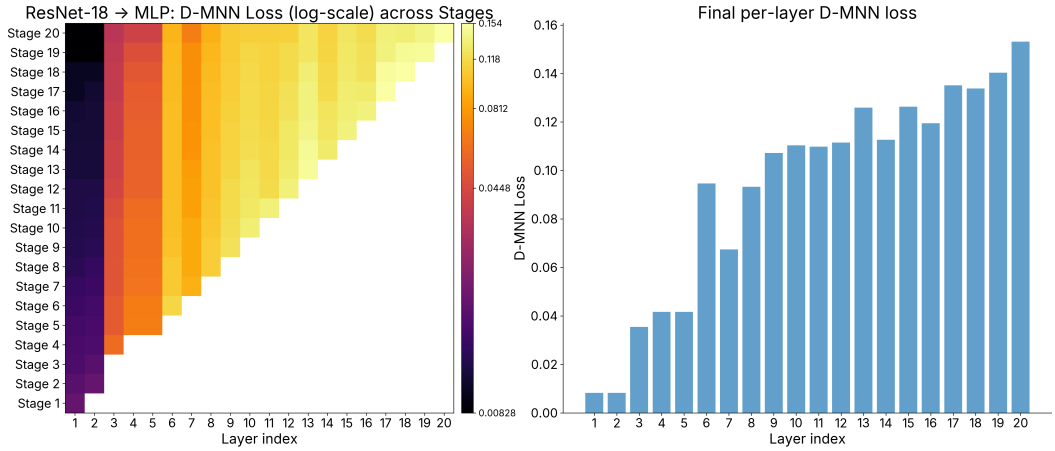

Figure 8: **D-MNN finds similar layer bottlenecks as CKA.** We show the per-stage losses for D-MNN, as with CKA. We find that D-MNN is far from saturating unlike CKA, which had loss values closer to 0. We find that D-MNN is also more consistent across layers, indicating that further optimization can be pushed for better alignment in D-MNN. We believe this shows even better correlation between D-MNN and final task accuracy and optimizing D-MNN further may lead to a task accuracy closer to the guide network.

goal is to see whether this hurts performance of the resultant ResNet-18 or ResNet-50, specifically by changing geometric aspects of the architecture, leading us to believe that we are more sensitive to guide network architecture.

We report results in Table 6. We find that when the guide network is ill-suited for the given task, this does not affect the performance of the target network on the same task, given that the target network is well-suited for the downstream task. This demonstrates that NoT is not overly sensitive to guide network choices and a given CKA target won't derail downstream performance.

G Comparison with Progressive Distillation

While in this work, we mainly use representational methods to align our converted target architecture to the guide architecture, another method to perform such alignment is knowledge distillation [20]. We introduce knowledge distillation as part of our progressive replacement schedule as a comparison to NoT, which we refer to as *progressive distillation*.

At each replacement stage, we convert a guide network layer to a target layer and perform knowledge distillation from the guide network to the newly converted architecture. We train the converted layers using the distillation. Due to needing to pass activations to the final logit layer, we only test settings where the activation shapes are the same across the converted layers in the guide network.

We show results in Table 7. We find that progressive distillation performs competitively with NoT with CKA when the guide architecture is trained but performs worse than the target architecture baseline when the guide architecture is untrained. This is intuitive because distillation doesn't provide access to the internal geometry of the architecture which we exploit with CKA. This further establishes that using representational similarity metrics explains our improvement with untrained architectures.

One interesting finding we would like to highlight is that we find a significant improvement with NoT in our GPT-2→RNN setting than over distillation. This implies that NoT overcomes a training problem with RNNs that could not be overcome with distillation. Using the output of a teacher model was not sufficient to prevent stability problems in training an Elman RNN. However, using internal representations of a guide architecture could prevent said problems and achieve stronger results. We believe this has significant implications for RNN training and demonstrates that RNN training stability can be improved by using untrained architectures.

H Automating NoT

One major component we aim to address is that NoT seems to rely on a hand-designed schedule where every layer has the learning rate and number of epochs of training tuned for optimal representational similarity. To overcome this limitation, we also discuss further methodology for automating the learning rate and number of epochs of training applied per stage of representational similarity optimization.

Our automation applies an exponential moving average (EMA) over the representational similarity loss at every stage to measure the average change over a 500 step window. We then use this EMA as a learning rate condition and stopping condition. For our learning rate, we start with a consistent setting for all stages. We reduce the learning rate when our EMA plateaus over 4 epochs [37]. We then stop our optimization when we saturate the EMA of the representational similarity loss i.e. we reach a CKA loss of less than 0.01 or train for 100 epochs. We find that this threshold is sufficient to get a strong result.

We apply this to our cross-network architecture conversions including ResNet→MLP, DINOv2→Patch-MLP, and GPT-2→RNN in Table 8. We convert GPT-2 to an RNN with a starting learning rate of 1e-4 at every stage, allowing the scheduler to tune the learning rate, and using an EMA over 500 steps to measure when to apply the scheduler and when to stop the training. For each stage, we either train until the similarity loss for CKA is less than 0.01 or stop training after 100 epochs. We find that our automatic tuning leads to a slightly stronger result than what was reported for NoT. This reduces the expensive nature of NoT.

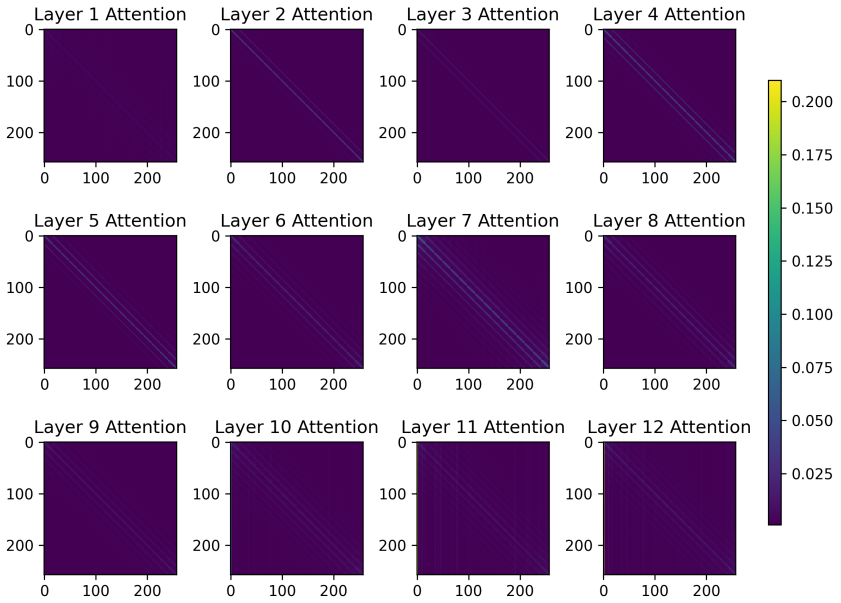

Figure 9: **DINOv2 has simple attention maps**: We visualize attention maps for DINOv2 over a set of ImageNet images to better understand how we are able to remove cross-token communication via NoT. We find attention is fairly sparse in DINOv2, when focusing on image classification.

I Hidden-State Adapters

We evaluate two direct alternatives to NoT: hidden-state conversion with adapters and standard logit distillation. For hidden-state conversion, we insert a learned linear adapter that maps target-layer representations to the corresponding guide-layer representations, using either a trained (T) or untrained (U) guide. Due to memory constraints, we did not run the adapter baseline for the ResNet-18→MLP setting. For logit distillation, we train the target with standard teacher-logit supervision from a trained guide. Results are shown in Table 9. Hidden-state adapters are competitive with NoT in the tested settings, which supports the broader motivation of NoT: representational alignment can transfer computations across architecture families. Unlike a fixed adapter baseline, NoT progressively replaces guide modules with target modules, leaving only the target architecture at inference.

J Accuracy-Efficiency Tradeoff

We provide inference-time measurements for a representative NoT conversion in Table 10. The main paper already includes settings with concrete accuracy–efficiency tradeoffs, most notably ResNet-50→ResNet-18 and GPT-2 Large→GPT-2 Small, where the target is shallower while improving over the baseline target accuracy. We also measure wall-clock latency for the DINOv2→Patch-MLP conversion. Using an H100 and an RTX-3090, we measure one forward pass over a single image and report the mean and standard deviation over ten runs. The Patch-MLP target is faster on both devices while preserving much of the guide accuracy reported in the main results.

These measurements complement the accuracy results by showing that NoT-converted target architectures can yield practical inference-time gains. More broadly, NoT opens a space of architecture-transfer experiments in which targets can be selected for deployment constraints and initialized through representational alignment rather than trained from scratch.

K Analyzing DINOv2

A surprising result we found was that we could replace attention in DINOv2 with a Patch-MLP. This Patch-MLP has no cross-token communication, with linear layers applying transformations

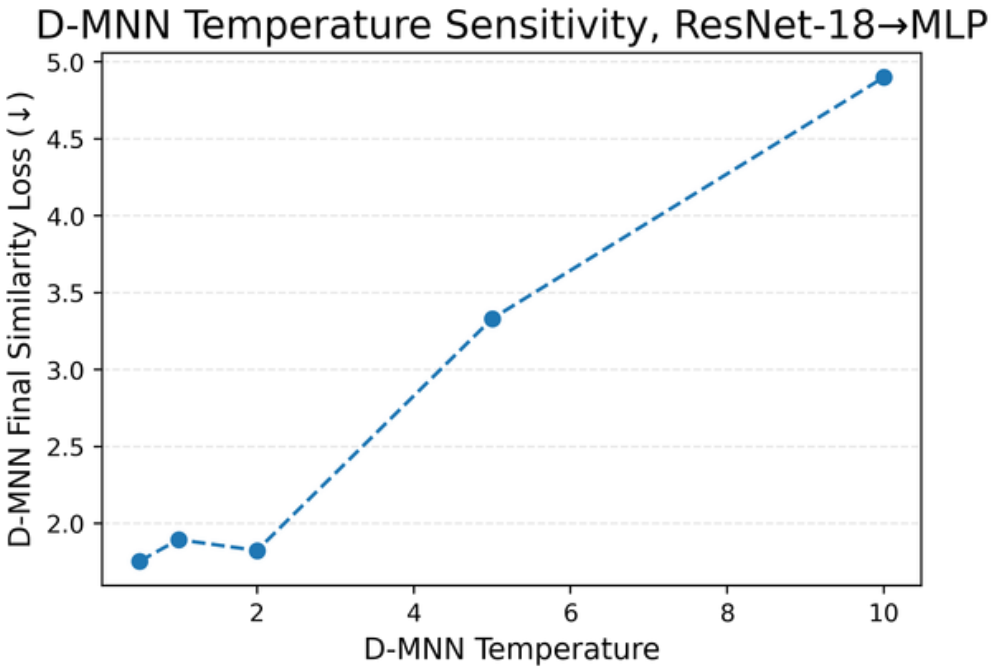

Figure 10: **Temperature Variations across D-MNN**: We analyze how D-MNN similarity loss varies across different temperature values from 0.5 to 10.0. We see that close temperature values like 0.5 to 2.0 have similarity loss. As we increase similarity we see worse similarity.

per-token instead. We investigated why we could convert from attention to Patch-MLPs in DINOv2 without much loss in performance. First, we found that the first layer of DINOv2 is a convolution and the last layer aggregates over tokens for a label prediction, meaning that cross-token prediction is not entirely eliminated.

In Figure 9, we show attention map visualizations for all layers in DINOv2 for a set of ImageNet images. We see that for most DINOv2 layers, attention is highly sparse. Image tokens only focus on tokens in a nearby neighborhood. Most of the other focus goes to the original token itself. Cross-token communication is sparse. We believe this makes attention in DINOv2 much easier to replace with a Patch-MLP.

This matches intuition in prior findings that attention matrices in ViTs are repetitive across layers [59, 50] when analyzing attention patterns when completing tasks like image classification. Intuitively, this makes sense. In image classification, there is a central image and cross-token communication may not be entirely necessary. We are not the first to propose removing cross-token computation in ViTs. Other works such as MetaFormer [56] or RIFormer [51] have proposed methods to replace cross-token mixing in ViTs or MLP-Mixers [47] due to sparse cross-token communication.

We believe such findings demonstrate the benefits of NoT. We were able to match findings in prior work by directly training a replacement for attention with strong performance. This is indicative that image classification is sparse, requiring a network to focus on a specific token. However, we also believe that such findings may not transfer to other tasks. For example, if we were to focus on semantic segmentation, then we may find that we cannot use NoT for strong performance given the likelihood that semantic segmentation requires cross-token communication.

L D-MNN Analysis

We analyze our new similarity metric, D-MNN in Figure 10, starting with analyzing how temperature in Equation (12) affects our final similarity loss for our ResNet-18→MLP conversion. We find that close temperature values have similar final similarity losses but as we make larger changes in temperature, we see lower D-MNN similarity between our original and converted network. This is

intuitive due to our design of D-MNN, where larger temperatures flatten the probability distribution of the target and guide network representations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We support our claim with detailed experiments across a large set of vision and language networks and large scale datasets like ImageNet-1K and Wikitext-103.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitations section in the conclusion and a limitations section in the Appendix to further elaborate.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: There are no theoretical results provided in this paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: This work uses open-source datasets and models. We report all learning rates, number of training epochs for alignment as well as standard training, optimizers, batch size, and alignment objective clearly. For our automatic procedure, we cleanly describe our EMA tuning and the requirements for changing stages.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is uploaded as part of the submission. The code will be made available to the community publicly upon acceptance of this paper. The data and networks used in these experiments are open-source and available to the community.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Many of these are covered in the main paper. Section 4 covers optimizer and batch size choices. Appendix C covers all hyperparameter choices related to learning rates. Appendix H also contains information on using EMA as part of tuning similarity in NoT.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use five random seeds on every single experiment and report the average performance along with standard error.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We cover our compute requirements in Appendix C along with training settings and GPU requirements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our paper studies conversions across network architectures. This conforms with the code of ethics as wed do not involve human participants, have any data related concerns, and our impact will be with respect to network efficiency.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: We believe our work does not have negative societal impacts and aims to only further advanced in deep learning. However, we will include an impact statement upon acceptance of the paper in order to emphasize the goals of this work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our method uses open source networks and datasets. We will not release models that achieve extremely high accuracy and have a high risk of misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We use open-source networks and datasets and properly cite these assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are introduced in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects is used in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: This work does not involve LLMs as an important, original, non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

Model	Stage	Trained Learning Rate	Untrained Learning Rate	D-MNN Learning Rate	Number of Epochs
ResNet-18→MLP	1	2×10^{-5}	2×10^{-5}	1×10^{-5}	10
	2	2×10^{-5}	2×10^{-5}	1×10^{-5}	65
	3	2×10^{-5}	2×10^{-5}	7.5×10^{-6}	45
	4	1.5×10^{-5}	1.5×10^{-5}	7.5×10^{-6}	45
	5	1.5×10^{-5}	1.5×10^{-5}	7.5×10^{-6}	45
	6	1×10^{-5}	1×10^{-5}	7.5×10^{-6}	65
	7	1×10^{-5}	1×10^{-5}	7.5×10^{-6}	45
	8	1×10^{-5}	1×10^{-5}	5×10^{-6}	45
	9	1×10^{-5}	1×10^{-5}	5×10^{-6}	50
	10	7.5×10^{-6}	7.5×10^{-6}	5×10^{-6}	50
	11	7.5×10^{-6}	7.5×10^{-6}	5×10^{-6}	50
	12	7.5×10^{-6}	7.5×10^{-6}	5×10^{-6}	50
	13	7.5×10^{-6}	7.5×10^{-6}	5×10^{-6}	65
	14	7.5×10^{-6}	7.5×10^{-6}	1×10^{-6}	50
	15	5×10^{-6}	5×10^{-6}	1×10^{-6}	50
	16	5×10^{-6}	5×10^{-6}	1×10^{-6}	50
	17	2.5×10^{-6}	2.5×10^{-6}	1×10^{-6}	75
	18	2.5×10^{-6}	2.5×10^{-6}	1×10^{-6}	75
	19	1×10^{-6}	1×10^{-6}	1×10^{-6}	75
	20	1×10^{-6}	1×10^{-6}	1×10^{-6}	100
DINOv2→Patch-MLP	1	1×10^{-4}	1×10^{-4}	1×10^{-4}	10
	2	1×10^{-4}	1×10^{-4}	1×10^{-4}	35
	3	1×10^{-4}	1×10^{-4}	1×10^{-4}	35
	4	1×10^{-4}	1×10^{-4}	7.5×10^{-5}	35
	5	1×10^{-4}	1×10^{-4}	7.5×10^{-5}	50
	6	1×10^{-4}	1×10^{-4}	7.5×10^{-5}	50
	7	1×10^{-4}	1×10^{-4}	5×10^{-5}	50
	8	1×10^{-4}	1×10^{-4}	5×10^{-5}	50
	9	7.5×10^{-5}	7.5×10^{-5}	5×10^{-5}	60
	10	7.5×10^{-5}	7.5×10^{-5}	5×10^{-5}	75
	11	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	75
	12	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	100
GPT-2→RNN	1	1×10^{-4}	1×10^{-4}	1×10^{-4}	10
	2	1×10^{-4}	1×10^{-4}	1×10^{-4}	35
	3	1×10^{-4}	1×10^{-4}	1×10^{-4}	35
	4	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	5	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	6	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	7	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	8	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	9	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	10	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	11	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
	12	1×10^{-4}	1×10^{-4}	1×10^{-4}	50
ResNet-50→ResNet-18	1	1×10^{-3}	1×10^{-3}	1×10^{-3}	5
	2	1×10^{-3}	1×10^{-3}	1×10^{-3}	15
	3	1×10^{-3}	1×10^{-3}	1×10^{-3}	25
	4	1×10^{-3}	1×10^{-3}	1×10^{-3}	35
	5	1×10^{-3}	1×10^{-3}	1×10^{-3}	35
	6	1×10^{-3}	1×10^{-3}	1×10^{-3}	35
	7	1×10^{-3}	1×10^{-3}	1×10^{-3}	75
	8	1×10^{-3}	1×10^{-3}	1×10^{-3}	100
GPT-2 Large → GPT-2 Small	1	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	2	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	3	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	4	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	5	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	6	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	7	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	8	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	9	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	10	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	11	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15
	12	7.5×10^{-5}	7.5×10^{-5}	1×10^{-5}	15

Table 4: **Learning rates and epochs of training across NoT representational similarity optimization**: We report learning rates and number of epochs of training used for training our model replacements for NoT representational similarity matching. We find that different stages can have significantly different learning rate requirements.

Model	Learning Rate	Epochs	Warm-up	Scheduler	Notes
ResNet-18→MLP	5×10^{-4}	100	10	Linear+Cosine	Gradient Clipping to 1.0
DINOv2→Patch-MLP	1×10^{-4}	100	5	Linear+Cosine	N/A
ResNet-50→ResNet-18	1×10^{-3}	100	0	None	N/A
GPT-2→RNN	1×10^{-4}	100	5	Linear+Cosine	Gradient Clipping to 1.0

Table 5: **Learning rates, epochs, and scheduler settings for NoT task training.** We report learning rates, number of epochs of training, number of warmup epochs, and scheduler details for our settings along with miscellaneous details like using gradient clipping.

Original (Guide) → Target	Similarity Metric	Guide	NoT	Baseline
		ImageNet Top-1 Accuracy (↑)		
MLP (Untrained) → ResNet-18	CKA	0.0911	68.06	68.90
MLP (Trained) → ResNet-18	CKA	32.18	67.54	68.90
RNN (Untrained) → ResNet-18	CKA	0.1134	67.22	68.90
ResNet-18 → ResNet-50	CKA	69.66	76.54	76.12

Table 6: **Network of Theseus is robust to reverse replacement**. We can reverse replacement in NoT by converting from an MLP to ResNet-18, converting from an RNN designed for image classification to ResNet-18, or converting from ResNet-18 to ResNet-50. We find that this has little effect on results, indicating that NoT is robust. We emphasize the impact of this finding: we do not find impact on network performance when the original guide network is poorly designed.

Setting	Guide	NoT Target	Distillation Target	Original
ResNet-18→MLP	69.16	62.55	59.13	40.01
Untrained ResNet-18→MLP	0.10	60.28	31.11	40.01
GPT-2→RNN (↓)	37.50	50.58	75.49	119.44
Untrained GPT-2→RNN (↓)	51948.4	58.26	135.55	119.44

Table 7: **Progressive distillation**: We include a comparison to knowledge distillation as a method for aligning our converted architectures to our original architectures. We progressively convert and distill ResNet-18 into our MLP and GPT-2 into our RNN. We only apply this method for conversion where the guide and target have the same layer shapes to allow propagation to the final logits. We find competitive results with NoT when the guide is trained and much worse results when the guide is untrained. This emphasizes that NoT can exploit untrained networks.

	Original	NoT (Trained-EMA)	NoT (Trained-Original)
ResNet-18 →MLP	69.66	60.96	62.12
DINOv2 →Patch-MLP	81.03	71.45	72.56
GPT-2→RNN	37.50	48.60	50.58

Table 8: **Automating NoT with EMA and new learning rate schedulers**: We automate NoT by tuning the learning rate and stopping criterion with an EMA-based scheduler.

Task	NoT (T/U)	Adapter (T/U)	Logit-T
ResNet→MLP (IN-1K)	× / ×	× / ×	46.11
DINOv2→Patch-MLP (IN-1K)	71.45 / 72.56	69.11 / 67.91	69.47
GPT-2→RNN (WT)	48.60 / 50.58	45.18 / 44.45	58.19

Table 9: **Hidden-state adapters and logit distillation.** We compare NoT with a linear hidden-state adapter and standard logit distillation from a trained guide. T/U denotes trained/untrained guide results. IN-1K reports ImageNet Top-1 accuracy (higher is better), and WT reports Wikitext perplexity (lower is better).

Setting	Guide Time (H100)	Target Time (H100)	Guide Time (RTX-3090)	Target Time (RTX-3090)
DINOv2→Patch-MLP	0.69 ms/image (0.05)	0.61 ms/image (0.03)	1.03 ms/image (0.055)	0.87 ms/image (0.031)

Table 10: **Inference latency for DINOv2→Patch-MLP.** We report mean wall-clock latency for one forward pass, with standard deviation in parentheses, over ten runs on each GPU.