Transferring Architecture as an Optimization Prior

Vighnesh Subramaniam[1]*, **Boris Katz**[1], **Brian Cheung**[2]
[1]MIT CSAIL, [2]UCSF
[1]{vsub851,boris}@mit.edu
[2]bcheung@ucsf.edu

Abstract

Initialization determines the optimization "fate" of a model, going far beyond just setting the scale of its weights. It establishes a fundamental initial geometry over the input data, dictating certain aspects such as which examples are considered close and which representations gradient descent will naturally refine first. In this paper, we view the initial geometry of a network as an *optimization prior* and ask whether a network can change its optimization prior by referencing a completely different architecture. To investigate this question, we use an initialization procedure where a target network is aligned to the representational geometry of a randomly initialized guide network before any downstream training occurs. Crucially, alignment to the guide network cannot transfer learned features, never receives gradients, and does not use labels. We demonstrate that this cross-architecture transfer of "fate" benefits the target network in two key domains: (1) instilling the optimization benefits of depth into shallow networks by aligning them to deeper networks, and (2) transferring representational granularity by using fine-resolution networks to shape the geometry of networks with coarser representations. In both cases, the alignment results in improved downstream performance of the target network. Ultimately, this demonstrates that a model's architectural destiny can be decoupled from its physical structure and explicitly modified at initialization.

1 Introduction

Classical initialization methods traditionally focus on parameter scale to stabilize signal and gradient propagation [10, 12, 2, 24]. But an initialization also defines something richer, notably, an initial representation of the data. Before a network takes any gradient steps, its random weights already determine geometric relationships among data points such as which examples are close and which relationships govern earliest stages of training. We refer to the geometric bias that an initialization induces on subsequent learning as an *optimization prior*. This perspective raises a question about initialization beyond signal stability, namely what initial geometry a network should induce.

Architectural choices such as depth and patch resolution change the computations applied to inputs and therefore, the induced geometry. These choices indicate that certain network architectures have better priors than others. We ask whether a network can change its optimization prior by referencing a completely different architecture. This question is motivated by the principle that the initial geometry need not be treated as fixed. In fact, this geometry could be reshaped by referencing the geometry of another architecture via optimization of representation similarity objectives. Representational similarity methods such as SVCCA [29] and CKA [16] were originally developed to compare hidden states across networks, layers, and training runs. More recent work has motivated this perspective by transferring architectural inductive biases across networks [31, 37, 36]. If we can transfer architecture and reshape geometry, networks with better downstream performance, i.e., ones that use more computations via depth, width, or tokenization, could transfer their optimization priors to networks that are shallower or narrower, leading to downstream performance improvements. More

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

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

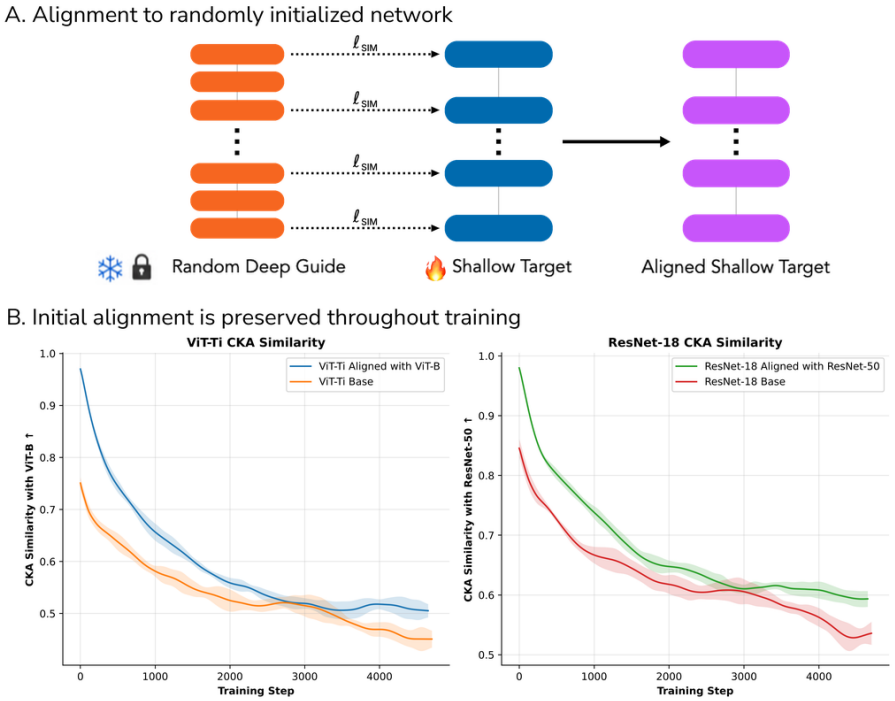


Figure 1: **The Optimization Prior.** (A) A frozen, randomly initialized guide defines a representational geometry. We align a target network to this guide at initialization, discard the guide, and then train the aligned target normally. (B) During supervised training, aligned ResNet and ViT targets remain closer to their deep guides than standard targets, suggesting that the initialization persists beyond the alignment stage.

importantly, this has scientific implications for initialization and its connection with architecture. Could architectures with increasingly desirable properties, e.g., deeper, wider, etc., transfer *more* useful geometry to shallow or narrow networks?

As stated, we formulate the transferring architecture as an optimization prior using representational similarity, designing an initialization-only procedure for choosing the starting geometry of a target network. The procedure is simple. First, we choose a target network, which is the model that will ultimately be trained and deployed. Next, we choose a guide network, which is sampled from a standard random initialization and then frozen. The guide is never trained on the downstream task. Using an unlabeled alignment set, we optimize the target so that its representations become similar to the guide's representations under a representational dissimilarity objective such as CKA or DiffKNN [37]. Only the target parameters are updated during this stage. Finally, after alignment, we discard the guide and train the aligned target normally on the supervised downstream objective. Thus, the guide does not provide labels, logits, predictions, or a trained function, which distinguishes it from standard distillation. It only provides an initial representational geometry that is absorbed into the target initialization.

We apply the optimization on two settings, *depth transfer* and *representational granularity transfer*. First, for depth transfer, shallow target networks are aligned to deeper randomly initialized guides and then trained on the downstream task. We evaluate this procedure across several architecture families, including ResNets, vision transformers, ConvNeXtV2 models, and GPT-2 language models, showing downstream improvement across all settings. Second, for representational granularity transfer, ViT-B/32, a network with coarse patch tokens, is aligned to ViT-B/16, a network with fine-grained patch tokens, at initialization and then train the ViT-B/32 on the downstream task. Similarly, this leads to downstream improvement.

We summarize our contributions below:

1. We propose transferring architecture as an *optimization prior*, changing the optimization trajectory of a network.
2. We show that depth can be transferred as an optimization prior by aligning shallow targets to deeper guides across vision and language architectures.

3. We extend this idea to representational granularity by aligning coarse-grained target representations to fine-grained guide representations, focusing on coarse-token ViT-B/32 targets and fine-token ViT-B/16 guides.
4. Critically, we find that our results lead to improved downstream performance across all settings. Moreover, even aligning to guide networks with data sampled Gaussian noise leads to improvements in downstream performance.
5. We analyze the mechanism of transfer by measuring guide proximity during training, demonstrating chained transfer, and deriving how kernel alignment at initialization explains similar early supervised training trajectories.

2 Related Work

We study representational similarity as an initialization-only optimization prior for architectural transfer. This connects to initialization methods that shape trainability before learning and distillation methods that use teachers—including random teachers—to guide students. Our setting differs in that the guide is randomly initialized and discarded after shaping the target initialization.

Network Initialization. Initialization is central to trainability. Classical schemes choose weight distributions to preserve activation and gradient statistics, including Xavier initialization [10], He initialization [12], and orthogonal initializations motivated by deep linear dynamics [32]. Our work is closest to data-dependent initialization, where weights are chosen using data statistics rather than only layer fan-in, fan-out, or nonlinearities. Examples include data-dependent normalization for CNNs [18], LSUV initialization [23], and PCA- or variance-based initialization procedures [17, 34]. A related theoretical literature studies how random initialization shapes signal propagation, expressivity, and trainability in deep networks [27, 33, 26, 40, 44]. We present a complementary form of data-dependent initialization: the data supply the alignment set, while a random guide architecture supplies the representational geometry.

Self-Supervised Learning. Learning favorable geometry for downstream learning has been a key goal of self-supervised learning (SSL), particularly in computer vision where the goal is to reconstruct representations of image patches. Contrastive methods such as SimCLR align representations of different views of the same image [4] while masked autoencoders reconstruct missing image patches [14]. DINOv2 and DINOv3 combine image-level DINO and patch-level iBOT objectives to build strong self-supervised representations [25, 35]. DINOv3 further preserves relationships between relationships between through Gram anchoring. In language settings, recent work has explored unsupervised on-policy self-distillation, using a model's own generations to provide its training signal [19]. In this paper, we apply alignment at initialization, a slightly simpler variant than SSL. Our goal is to show that such alignment reshapes an optimization prior depending on the guide network, implying that such findings could extend to using SSL methods for alignment.

Distillation with Randomly Initialized Networks. Knowledge distillation trains a student to match signals from a teacher, usually a trained high-capacity model [15]. Later work distills intermediate representations or attention maps, including FitNets [30], attention transfer [43], and contrastive representation distillation [38]. Self-distillation and related methods show that the teacher need not be a larger external model; students can learn from identical or evolving teachers, as in Born-Again Networks [9], BYOL [11], and DINO [3]. Several recent works further show that randomly initialized networks can provide useful learning signals [31, 1, 6], and representational alignment methods show that random guides can transfer architectural priors without trained task knowledge [36, 37].

Our work builds on this view but differs from distillation in two ways. First, the guide provides no logits, labels, predictions, or trained solution. Second, the guide is absent during downstream supervised training. Thus, the transferred object is an initial representational geometry rather than an ongoing teacher signal.

3 Representational Alignment Reshapes an Optimization Prior

Given the goal of transferring architectural properties, we first introduce our procedure for changing an optimization prior using representational alignment objectives and then describe some simple derivations to show that optimizing for alignment changes downstream training.

3.1 Optimization Prior Procedure

Let S_θ denote the target network whose optimization prior we want to shape, and let D_ϕ denote a guide network. The guide is sampled from its standard random initialization, frozen, and never trained on labels. Given an unlabeled alignment set U, we define representation readouts

$$R_S(u;\theta), \qquad R_D(u;\phi),$$

which extract the target and guide representations to be compared on an input minibatch u. These readouts may correspond to final features, intermediate features, or a collection of internal activations; the specific choice is an experimental design decision. The optimization prior is obtained by updating only the target parameters to minimize a representational dissimilarity,

$$\mathcal{L}_{\text{prior}}(\theta) = \mathbb{E}_{u\sim U}\left[\Delta_{\text{rep}}\left(R_S(u;\theta), R_D(u;\phi)\right)\right],$$

where Δ_{rep} can be a global similarity objective such as CKA or a local-neighborhood objective such as DiffKNN. We denote the resulting aligned target parameters by θ_0^S. After this alignment stage, the guide D_ϕ is discarded and only the initialized target $S_{\theta_0^S}$ is trained on the downstream supervised task. Thus the guide does not provide labels, logits, or a trained solution; it only defines a representational geometry that is absorbed into the target initialization.

3.2 Why initialization alignment changes optimization

Representational alignment changes the representations with which the target begins supervised training. To see why this matters, consider a supervised batch (X, Y) used after alignment. Let

$$H_0^S(X) = H_{\theta_0^S}(X), \qquad H_0^D(X) = H_\phi(X)$$

denote the target and guide feature matrices on this batch, and define their feature kernels

$$K_0^S(X) = H_0^S(X)H_0^S(X)^\top, \qquad K_0^D(X) = H_0^D(X)H_0^D(X)^\top.$$

Although alignment is performed on the unlabeled set U, downstream optimization is controlled by the geometry induced on the supervised batch X. We therefore measure the relevant mismatch by

$$\varepsilon_0(X) = \|K_0^S(X) - K_0^D(X)\|_{\text{op}}.$$

This distinction is important. Representational alignment metrics like CKA may only encourage similar representational geometry, but the first supervised updates depend on the raw feature kernel, or equivalently the initial NTK.

The mechanism is clearest when the classifier head is initialized at zero. Write the logits as

$$F = H_\theta(X)V,$$

with $V_0 = 0$. Then $F_0 = 0$, and the first supervised step does not move the backbone because the gradient through the backbone is multiplied by V_0. Only the readout moves. If

$$G_0 = \nabla_F \mathcal{L}(F_0, Y),$$

then the first logit update is

$$F_1 = -\eta K_0(X)G_0.$$

Thus the initial feature kernel determines which examples influence one another in the first gradient step. If the aligned target and guide have similar kernels on X, then their first supervised logit updates are close:

$$\|F_1^S - F_1^D\|_F \le \eta\, \varepsilon_0(X)\, \|G_0\|_F.$$

In words, the optimization prior works by changing the similarity map through which training examples communicate at the start of learning. The guide is no longer present during supervised training, but its geometry remains encoded in the target initialization.

4 Experiments

We evaluate representational similarity as an initialization-only optimization prior for architectural transfer. In all settings, the guide is random and frozen. The target is aligned on unlabeled data, the guide is discarded, and the target is trained with the same supervised recipe as the standard-initialization baseline. We study two settings: depth transfer from deeper guides to shallower targets, and granularity transfer from fine-resolution ViT guides to coarser ViT targets.

Initialization-only alignment via NoT: In order to test alignment as a prior, we incorporate a mechanism for optimizing alignment. Our alignment procedure follows the stagewise replacement principle of Network of Theseus [37]. We use NoT only as a practical alignment operator and to validate our scientific findings. Let r_ϕ denote a frozen, randomly initialized reference network, and let s_θ denote the target network whose initialization we optimize. Both networks are decomposed into ordered stages. For an input minibatch $u \in U$, let $R_\ell(u)$ and $Z_j(u;\theta)$ denote the activations of reference stage ℓ and target stage j, respectively. Because the reference and target may have different depths or widths, we use coarse correspondence via a monotone stage map π that assigns each aligned target stage j to a reference stage $\pi(j)$. We specify the stage mapping across architectures in Section B.

For a set of target stages A, we optimize

$$\mathcal{L}_{\text{align}}(\theta_A) = \mathbb{E}_{u\sim U}\left[\frac{1}{|A|}\sum_{j\in A}\Delta_{\text{sim}}\left(Z_j(u;\theta),\left(R_{\pi(j)}(u)\right)\right)\right],$$

During each alignment phase, only the target parameters in the active stage set A are updated. After alignment, all target parameters are unfrozen and trained normally on the supervised objective. We use two similarity objectives. The first is linear CKA, which compares global pairwise geometry in a minibatch. The second is DiffKNN, a differentiable nearest-neighbor similarity objective that compares local neighborhood structure rather than only global kernel alignment. Both objectives operate through batchwise relational structure and therefore allow alignment between stages with different hidden dimensions. More details on DiffKNN are provided in Section B.

Unlike the full progressive conversion setting of Subramaniam et al. [37], we do not use all stages in every architecture. We find that aligning the upper half of the target network is more effective in our setting. This suggests that, for architectures within the same broad family, much of the transferable representational geometry emerges in later stages. Unless stated otherwise, all experiments align only the upper-half target stages for one epoch per alignment phase. We compare these choices in Section F.

Instilling depth: We first study whether depth can be transferred as an optimization prior. For each architecture family, we choose a randomly initialized guide and align a target from the same family. The aligned target is trained from scratch on the downstream task using the standard supervised objective. Our experiments yields three types of comparisons: deeper-to-shallower transfer, same-architecture alignment, and shallower-to-deeper transfer. The same-architecture setting controls for the effect of alignment alone, while the shallower-to-deeper setting tests whether the direction of transfer matters. For each target architecture, we report the change in downstream performance relative to training the same target from a standard initialization. For vision models, positive values indicate higher ImageNet top-1 accuracy. For language models, negative values indicate lower perplexity. This produces a transfer matrix whose rows are target models and whose columns are guide models.

We evaluate depth transfer across four architecture families. For ResNets, we use ResNet-10, ResNet-18, ResNet-50, ResNet-101, and ResNet-152 [13]; ResNet-10 follows the standard ResNet design with a convolutional stem, residual stages, progressive downsampling, and global average pooling. For vision transformers, we use ViT-Tiny, ViT-Small, ViT-Base, and ViT-Large [7], all with 16×16 patches unless otherwise stated. ViT-Tiny has six transformer encoder blocks with hidden size 384, and ViT-Small has eight blocks with hidden size 512. For ConvNeXtV2, we use Atto, Nano, Tiny, Base, and Large variants following the original depth and width configurations [39]. For language modeling, we use GPT-2-Nano, GPT-2-Tiny, GPT-2-Base, GPT-2-Medium, and GPT-2-Large [28]. GPT-2-Nano and GPT-2-Tiny are defined for this work with four and eight transformer decoder

blocks, respectively, and all variants share the same tokenizer, positional encoding scheme, and transformer block structure.

Instilling detail: We next ask whether the same initialization prior can transfer representational granularity. In vision transformers, patch size controls token granularity. A ViT-B/16 represents an image with a finer spatial grid than a ViT-B/32, while ViT-B/32 uses larger patches and therefore forms coarser token representations. We use this contrast to test whether fine-token geometry can shape the initialization of a coarse-token model.

The guide is a randomly initialized ViT-B/16 and the target is a ViT-B/32. We align the target to the guide, then train the ViT-B/32 target normally. Improvement over a standard training baseline indicates that the fine-token guide provides useful initial geometry even though the final deployed target remains a coarse-token model. We compare ViT-B/16 and ViT-B/32 by flattening token representations and aligning class-token representations, allowing comparison of their random global geometry. As in the depth-transfer setting, we add controls for same-tokenization and for the reverse direction, from coarse tokenization to fine tokenization.

Datasets, training and evaluation protocol: For vision experiments, we train on ImageNet-1K and report top-1 accuracy on the validation set [5]. For language experiments, we train on WikiText-103 and report test set perplexity [22]. We use the standard training, validation, and test splits for each dataset.

All alignment phases are optimized with AdamW [21]. Each alignment phase runs for one epoch over the alignment set. After alignment, the target is trained with standard supervised training for 100 epochs with AdamW. The baseline for each target architecture is the same model trained from a standard initialization for 101 epochs with the same optimizer, scheduler, and learning rate used after alignment. Thus, aligned and baseline targets receive the same training budget, optimizer, scheduler, augmentations, and hyperparameters. For all supervised training runs, we use cosine learning-rate decay with 5% warmup [20]. Hyperparameters are discussed in Section B. For ImageNet models, we use the same preprocessing and augmentation policy for aligned and baseline targets. For WikiText-103, we use the same tokenizer, context length, batching procedure, and evaluation protocol for aligned and baseline targets.

5 Results

We separate our results into findings over transferring depth, transferring detail, and general analyses. For each case, aligned performance is compared to the corresponding baseline target trained from a standard initialization. All reported deltas are averaged over five random seeds, and raw aligned and baseline performances with standard deviations are reported in Appendix C.

Instilling Depth: We first measure our ability to instill the geometry of deeper architectures into shallower architectures by optimizing representational alignment at initialization. We show results for vision networks in Figure 2 and for language networks in Figure 3. Across both settings and with both similarity metrics, we observe that deeper guide networks lead to better target performance. In vision models, deeper ResNets, ViTs, and ConvNeXtV2 networks improve ImageNet accuracy after alignment at initialization. In language models, deeper GPT-2 guides improve shallow GPT-2 perplexity on WikiText-103. The strongest improvements reach roughly four top-1 accuracy points for vision or four perplexity points for language. Generally, CKA and DiffKNN give similar improvements for depth transfer. When the guide is shallower than the target, performance usually stays close to the baseline or degrades slightly. These reverse-direction controls show that alignment is a partial constraint on target geometry, not an architectural projection. A shallow guide can perturb a deeper target, but it does not force the target to collapse to shallow-model behavior, because the guide is removed before supervised training and the target retains its own parameterization. When the guide and target have the same architecture, alignment also has little effect, with only small improvements or degradations in accuracy or perplexity.

The strongest pattern is that target performance improves as guide depth increases. For example, aligning ResNet-10 to a randomly initialized ResNet-152 gives stronger downstream performance than aligning ResNet-10 to ResNet-50 or ResNet-101. Similarly, the shallow GPT-2 variants benefit most from the largest GPT-2 guides, with GPT-2-Large and GPT-2-Medium producing the strongest

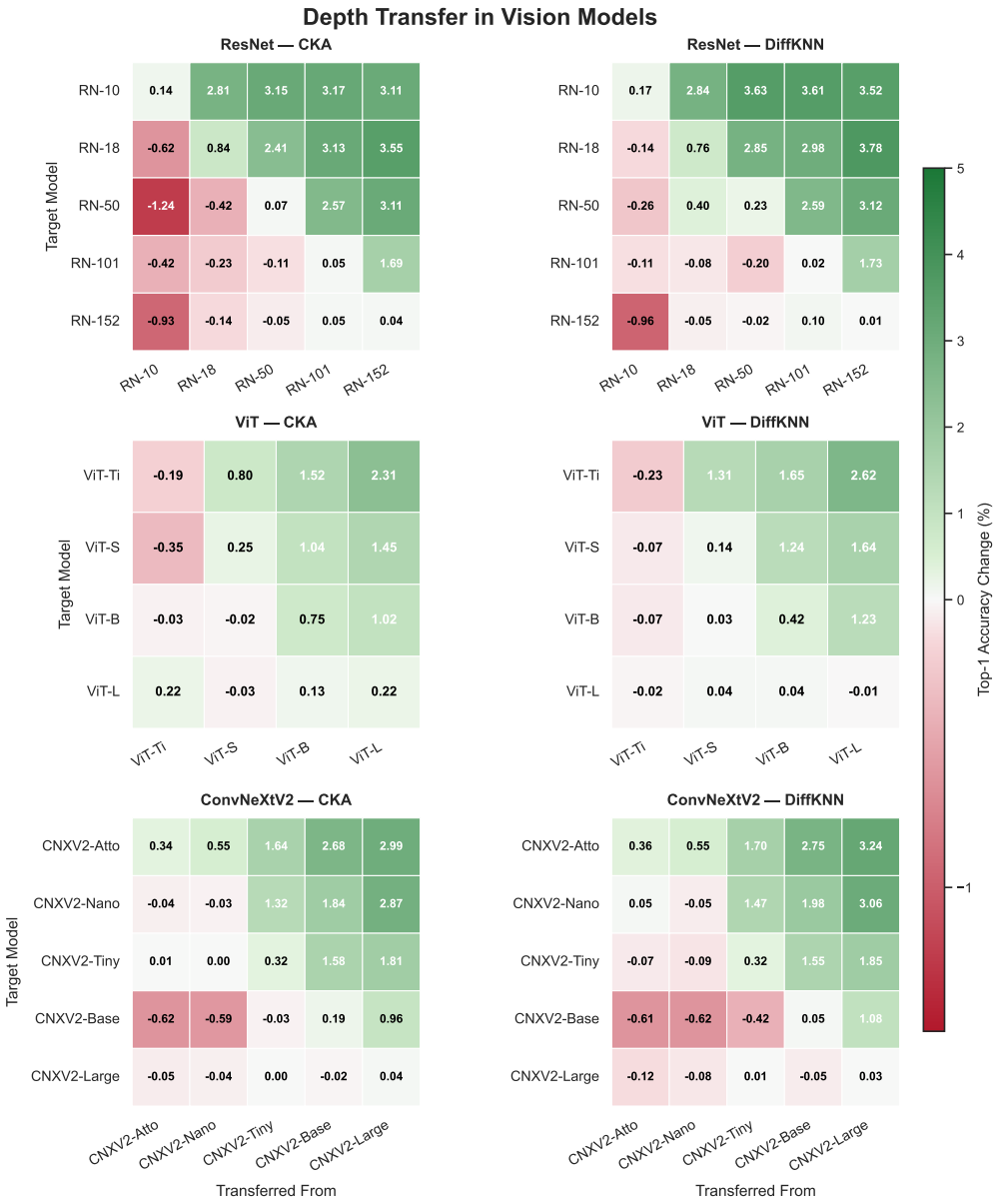


Figure 2: **Depth transfers to shallow vision networks.** We align different vision networks across different depths at initialization. Rows are targets and columns are random guides used only during alignment. Each cell reports the average ImageNet top-1 change relative to the same target trained from standard initialization. Green is better. Across ResNets, ViTs, and ConvNeXtV2 models, deeper guides consistently improve shallower targets under both CKA and DiffKNN. Raw performance with error bars is reported in Section C.

perplexity improvements. This suggests a possible initialization scaling law, where guide networks that apply more computation induce more useful initial geometry for the target.

Instilling detail. We next test whether fine-resolution ViT geometry can initialize a coarser ViT. Figure 4 aligns ViT-B/32 targets to ViT-B/16 guides, with same-tokenization and reverse-direction controls. CKA gives little improvement, while DiffKNN substantially improves the ViT-B/32 target. The same-tokenization and reverse-direction controls are much weaker, suggesting that fine-token geometry provides a useful initialization signal for coarse-token representations.

This setting also clarifies the difference between CKA and DiffKNN. The initial CKA between random ViT-B/32 and ViT-B/16 is already about 0.95, leaving little room for optimization. DiffKNN is less saturated because it compares local neighborhoods, allowing it to distinguish tokenizations

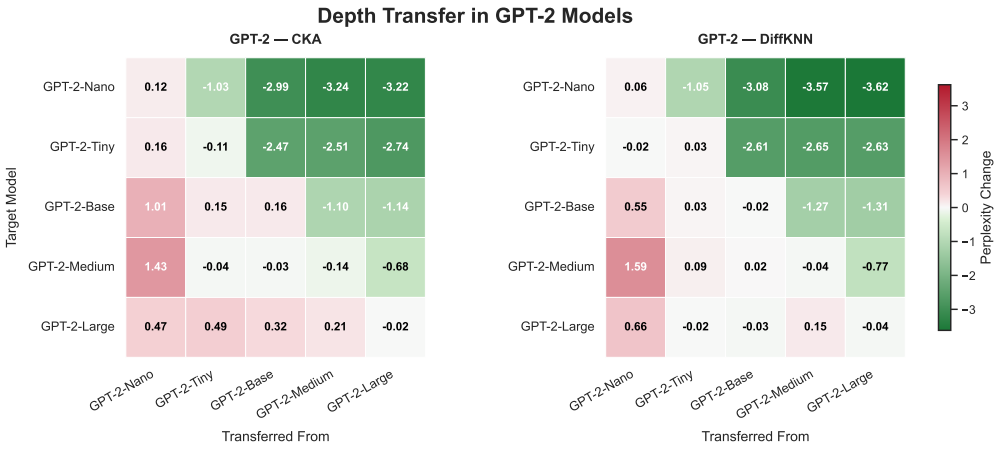


Figure 3: **Depth transfers to shallow language models.** We align different GPT-2 networks across different depths at initialization. Rows are GPT-2 targets and columns are random guides. Each cell reports WikiText-103 perplexity change relative to standard initialization. Negative is better. Deeper guides reduce perplexity for shallower targets under both CKA and DiffKNN. Raw performance with error bars is reported in Section C.

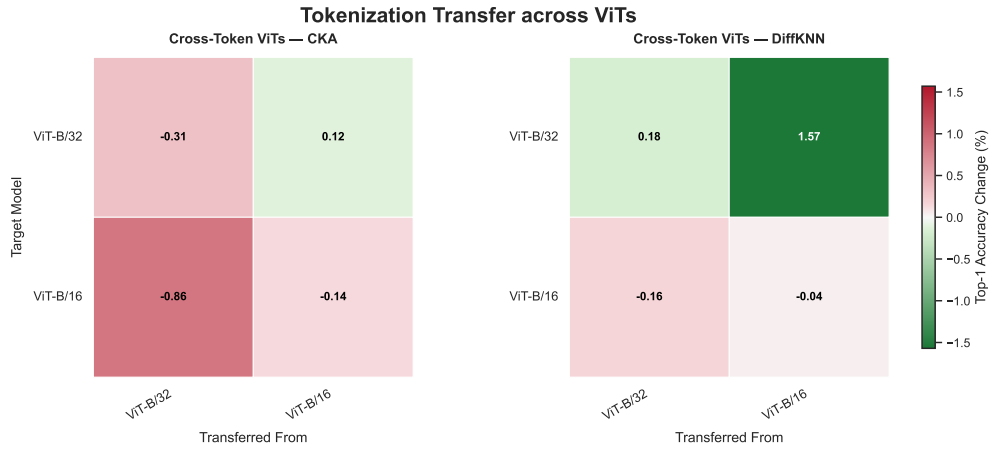


Figure 4: **Detail can transfer to coarse vision transformers.** Rows are target tokenizations, and columns are randomly initialized guide tokenizations. Each cell reports ImageNet top-1 accuracy change relative to the same target trained from standard initialization; green indicates improvement. The top-right cell tests fine-to-coarse transfer by aligning a coarse-token ViT-B/32 target to a fine-token ViT-B/16 guide. DiffKNN gives the clearest improvement for transferring fine-token geometry into ViT-B/32, while CKA provides a weaker signal when the random ViT-B/16 and ViT-B/32 representations are already highly aligned.

that have similar global kernel structure. Thus local geometry can be more informative than global kernel shape when transferring representational granularity.

Chained transfer. We also ask whether transferred geometry can pass through intermediate targets. Figure 5 first aligns an intermediate model to a deep guide, then uses the aligned intermediate model as the guide for a smaller target. For example, ViT-S is aligned to ViT-B and then used to align ViT-Ti. This suggests that guide-induced geometry is partially compositional: it can be absorbed by an intermediate target and then passed to a smaller model.

Alignment without data: Currently, our alignment procedure relies on real data for optimization. This begs the question of whether real, natural images or text are necessary to see the gains. If we can establish that target networks can improve by aligning to a deeper network without using real data, this motivates our approach as a method closer to weight initialization. To test this, target networks are optimized to align to guide networks using synthetic data. To generate synthetic data, we can sample data from a Gaussian. Either images or token IDs are sampled from Gaussian distributions as input data to the image or language models we align. Results are shown in Figure 6. We show that target networks still improve even when alignment does not use real data. This is significant, demonstrating that gains are not driven by data statistics.

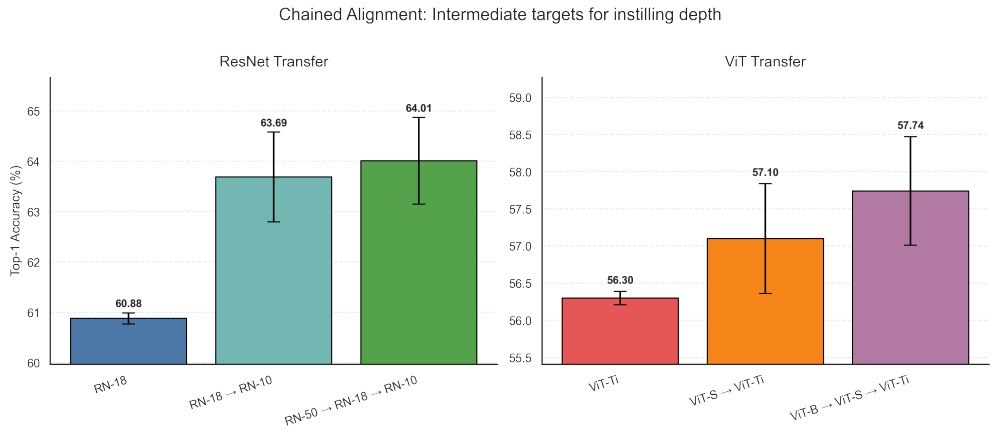


Figure 5: **Chained transfer improves shallow targets through intermediate networks.** We first align an intermediate target to a deep guide, then use that aligned intermediate model as the guide for a smaller target. For example, ViT-S is aligned to ViT-B, and the resulting ViT-S is then used to align ViT-Ti. Chained transfer improves over using a standard intermediate guide, suggesting that guide-induced geometry can be passed through successive target networks.

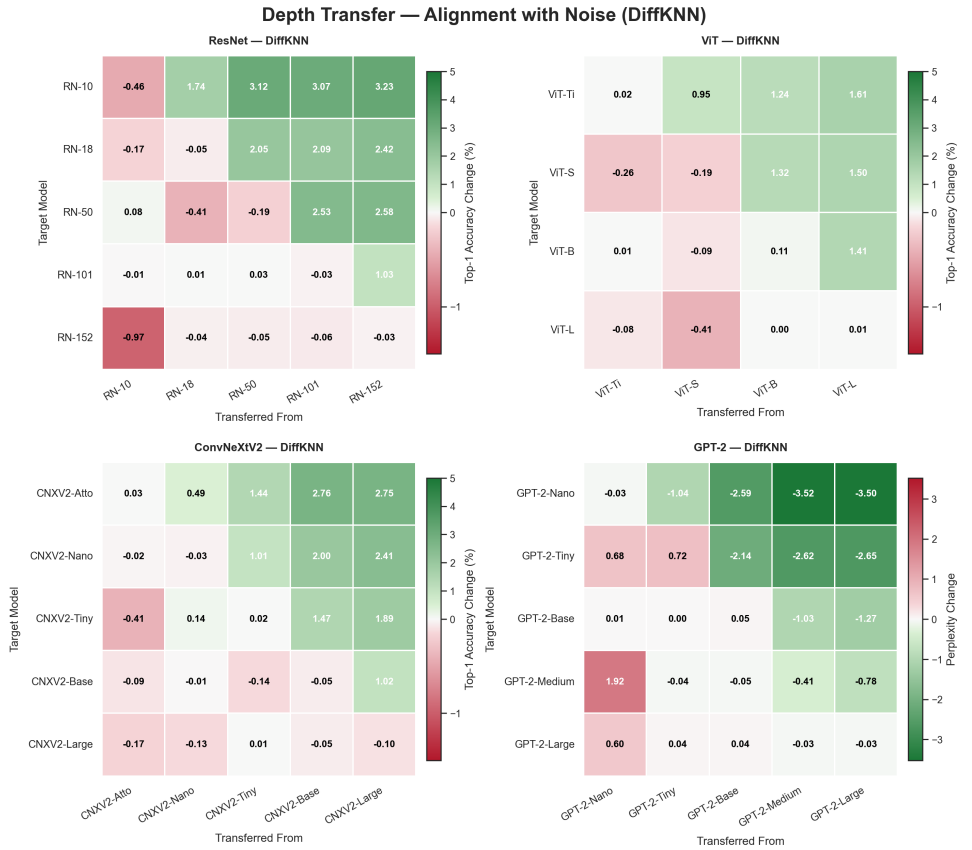


Figure 6: **Transferring architecture without data**. We align target networks to guide networks without relying on real data. Data is sampled from a Gaussian distribution for both images and text tokens. Gains over baseline training are observed for all target networks. This demonstrates that target networks can have their optimization priors reshaped just by fitting the geometric distribution of the guide. Gains were not driven by data statistics.

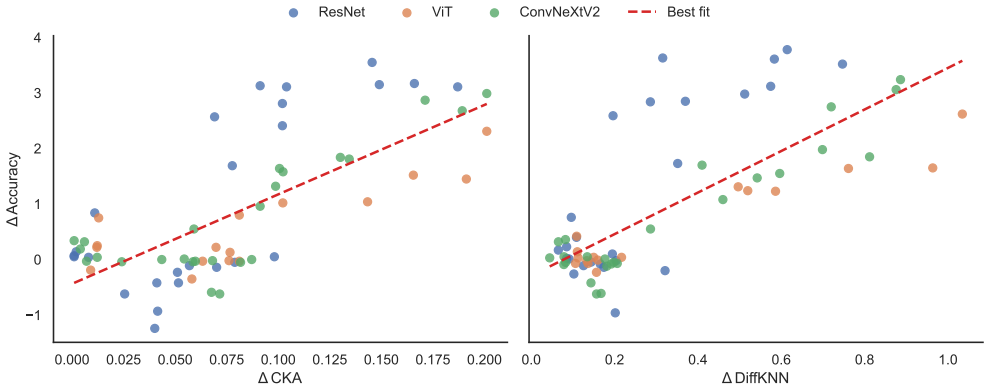


Figure 7: **Alignment movement predicts downstream gains.** For each guide-target pair, we compare the change in alignment during the initialization stage, $\Delta\text{sim} = \text{sim}_{\text{end}} - \text{sim}_{\text{start}}$, with the final ImageNet top-1 improvement of the aligned target. Larger alignment movement generally corresponds to larger downstream gains. When CKA is already saturated, the target moves little and gains are smaller; DiffKNN remains informative by measuring local neighborhood structure.

Alignment movement vs. final accuracy: As suggested by the detail-transfer results, one factor that affects transfer is how much the target geometry actually moves during initialization alignment. For each guide-target pair, we measure the starting alignment a_{start}, the ending alignment a_{end}, and the alignment movement $\Delta a = a_{\text{end}} - a_{\text{start}}$. We compare Δa to downstream improvement in Figure 7. Larger alignment movement generally corresponds to larger accuracy gains, suggesting that the optimization prior is strongest when the alignment objective induces a nontrivial change in target geometry. When CKA is already close to saturated, as in the ViT-B/16 to ViT-B/32 setting, alignment moves little and provides a weaker signal. DiffKNN is less saturated because it compares local neighborhood structure, allowing it to move the target in settings where global kernel geometry is already similar. Together, these results suggest that CKA and DiffKNN are complementary, and representational alignment is a partial constraint on geometry rather than an architectural projection.

6 Conclusion

We show that the geometry of a network at initialization, i.e. its *optimization prior*, can be changed through representational alignment with another, potentially more desirable architecture. Across vision and language settings, randomly initialized guides transfer useful architectural structure without labels or trained task knowledge. In particular, depth can be instilled into shallow targets, and fine-resolution representational geometry can be transferred into coarser ViT representations. Aligned targets remain closer to their guides during training, transfer can be chained through intermediate networks, and the induced initial kernel explains the earliest supervised dynamics.

Our results suggest a broader view of initialization and architecture. Rather than treating architecture only as a fixed computation graph, we show that some of its geometric consequences can be used transiently to shape how another network begins learning. This gives random architectures a scientific role as probes for optimization biases such as depth, locality, recurrence, and representational granularity. It also suggests new initialization strategies based on composing guide networks, chaining intermediate targets, and combining global and local similarity objectives such as CKA and DiffKNN. More broadly, representational alignment offers a way to separate the optimization benefits of architectural structure from the requirement that this structure persist in the final model.

Limitations. Our study does not exhaust the space of possible architectures, datasets, or alignment objectives. The improvements are consistent but modest, with the stron9gest gains around three to four accuracy points for vision or perplexity points for language in our tested settings. The tokenization experiments also show that transfer can be sensitive to the starting similarity between guide and target, especially when CKA is already saturated. Future work should study stronger local alignment objectives, combinations of CKA and DiffKNN, and additional architectural properties such as width, scale, recurrence, and locality.

Acknowledgements

This work was supported by the DARPA Mathematics for the DIscovery of ALgorithms and Architectures (DIAL) 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 all reviewers and the area chair of this NeurIPS submission for their thoughtful feedback. We would also like to thank Chris Wang, Kushagra Tiwary, David Mayo, Abdulrahman Alabdulkareem, the Isola Lab at MIT CSAIL, and the Efros Lab at UC Berkeley for their extremely useful input on this work.

References

[1] Ehsan Amid, Rohan Anil, Manfred K Warmuth, et al. Learning from randomly initialized neural network features. *arXiv preprint arXiv:2202.06438*, 2022.

[2] Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[5] 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.

[6] Peijie Dong, Lujun Li, and Zimian Wei. Diswot: Student architecture search for distillation without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11898–11908, 2023.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

[9] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pages 1607–1616. PMLR, 2018.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] 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.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 15979–15988. IEEE, 2022.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] 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.

[17] Saiprasad Koturwar and Shabbir Merchant. Weight initialization of deep neural networks (dnns) using data statistics. *arXiv preprint arXiv:1710.10570*, 2017.

[18] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[19] Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision. *arXiv preprint arXiv:2608.06296*, 2026.

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

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

[23] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[24] Nima Mohajerin and Steven L Waslander. State initialization for recurrent neural network modeling of time-series data. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2330–2337. IEEE, 2017.

[25] 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.

[26] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

[27] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

[28] 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.

[29] 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.

[30] Adriana Romero. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[31] Felix Sarnthein, Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Random teachers are good teachers. In *International Conference on Machine Learning*, pages 30022–30041. PMLR, 2023.

[32] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[33] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[34] Yeonjong Shin and George Em Karniadakis. Trainability of relu networks and data-dependent initialization. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

[35] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[36] Vighnesh Subramaniam, David Mayo, Colin Conwell, Tomaso Poggio, Boris Katz, Brian Cheung, and Andrei Barbu. Training the untrainable: Introducing inductive bias via representational alignment. *Advanced in neural information processing systems*, 39, 2025.

[37] Vighnesh Subramaniam, Colin Conwell, Boris Katz, Andrei Barbu, and Brian Cheung. Network of theseus (like the ship). *Advances in neural information processing systems*, 40, 2026.

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

[39] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16133–16142, 2023.

[40] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International conference on machine learning*, pages 5393–5402. PMLR, 2018.

[41] Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. *arXiv preprint arXiv:2410.01405*, 2024.

[42] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.

[43] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[44] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

A Appendix Overview

We cover methodological and experimental details, analyses, results, and ablations. In Section B, we cover additional details for applying NoT [37] in our work. We describe the alignment metrics used in more detail, our approach for mapping layers across networks, and training details. In Section C, we provide raw performance values across our settings to better ground our findings. In Section D, we expand on our theoretical derivations introduced in the paper. We explore a new transfer setting, recursive transfer, in Section E, where we can simulate depth using a single network module applied recursively. In Section F, we provide ablation experiments to justify experimental design choices made in the paper.

B Methodological and Experimental Details

B.1 Similarity Metrics

CKA: We use linear CKA as our global representational similarity objective. CKA uses kernel functions on centered representations to compute representational similarity matrices, which are then compared through 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 $\boldsymbol{B} \in \mathbb{R}^{b \times d_2}$, where b is the minibatch size. Before computing CKA, we ℓ_2-normalize each row of $\boldsymbol{A}$ and $\boldsymbol{B}$. We then compute the linear Gram matrices

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{A}^\top, \qquad \boldsymbol{L} = \boldsymbol{B}\boldsymbol{B}^\top. \tag{1}$$

We center the Gram matrices using the centering matrix $\boldsymbol{H}$, where

$$\boldsymbol{H} = \boldsymbol{I}_b - \frac{1}{b}\mathbf{1}\mathbf{1}^\top.$$

This gives

$$\tilde{\boldsymbol{K}} = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}, \qquad \tilde{\boldsymbol{L}} = \boldsymbol{H}\boldsymbol{L}\boldsymbol{H}. \tag{2}$$

We compute HSIC as the Frobenius inner product between the centered Gram matrices,

$$\mathrm{HSIC}(\boldsymbol{K}, \boldsymbol{L}) = \langle \tilde{\boldsymbol{K}}, \tilde{\boldsymbol{L}} \rangle_F = \mathrm{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}). \tag{3}$$

Finally, we define linear CKA as

$$g_{\mathrm{CKA}}(\boldsymbol{A}, \boldsymbol{B}) := \mathrm{CKA}(\boldsymbol{K}, \boldsymbol{L}) = \frac{\mathrm{HSIC}(\boldsymbol{K}, \boldsymbol{L})}{\sqrt{\mathrm{HSIC}(\boldsymbol{K}, \boldsymbol{K})\,\mathrm{HSIC}(\boldsymbol{L}, \boldsymbol{L})}}. \tag{4}$$

In our setting, we optimize representational *dissimilarity*. Since linear CKA is a similarity score, with larger values indicating more similar representations, we minimize the negative-log CKA:

$$\Delta_{\mathrm{CKA}}(\boldsymbol{A}, \boldsymbol{B}) = -\log(g_{\mathrm{CKA}}(\boldsymbol{A}, \boldsymbol{B})). \tag{5}$$

Thus, minimizing Δ_{CKA} encourages the target and guide representations to have similar centered kernel geometry.

DiffKNN: We also use a differentiable nearest-neighbor objective, DiffKNN, to compare local neighborhood structure. The goal of DiffKNN is to make the target network assign high similarity to the same local neighbors as the guide network.

Suppose we have target representations $\boldsymbol{A} \in \mathbb{R}^{b \times d_1}$ and guide representations $\boldsymbol{B} \in \mathbb{R}^{b \times d_2}$. We first compute pairwise similarity matrices over examples in the minibatch. For cosine similarity, we normalize rows and compute

$$\boldsymbol{S}^A_{ij} = \left\langle \frac{\boldsymbol{A}_i}{\|\boldsymbol{A}_i\|_2}, \frac{\boldsymbol{A}_j}{\|\boldsymbol{A}_j\|_2} \right\rangle, \quad \boldsymbol{S}^B_{ij} = \left\langle \frac{\boldsymbol{B}_i}{\|\boldsymbol{B}_i\|_2}, \frac{\boldsymbol{B}_j}{\|\boldsymbol{B}_j\|_2} \right\rangle. \tag{6}$$

We mask the diagonal entries so that an example cannot select itself as a nearest neighbor.

For each anchor example i, we compute the guide's top-k nearest-neighbor set

$$\mathcal{N}^B_k(i) = \mathrm{TopK}_{j \neq i}\ \boldsymbol{S}^B_{ij}. \tag{7}$$

This set is computed from the frozen guide network and is treated as a fixed target during the update. We then construct a binary neighborhood target matrix $\boldsymbol{T} \in \{0,1\}^{b\times b}$:

$$\boldsymbol{T}_{ij} = \mathbb{1}\{j \in \mathcal{N}_k^B(i)\}, \qquad \boldsymbol{T}_{ii} = 0. \tag{8}$$

The target similarities $\boldsymbol{S}^A$ are used as logits. Before applying the loss, we row-standardize the non-diagonal logits. Let

$$\mu_i = \frac{1}{b-1}\sum_{j\neq i} \boldsymbol{S}_{ij}^A, \;\; \sigma_i^2 = \frac{1}{b-1}\sum_{j\neq i}(\boldsymbol{S}_{ij}^A - \mu_i)^2. \tag{9}$$

For $j \neq i$, we define

$$\boldsymbol{Z}_{ij}^A = \frac{\boldsymbol{S}_{ij}^A - \mu_i}{\tau(\sigma_i + \epsilon)}, \tag{10}$$

where τ is a temperature parameter. The diagonal entries are masked out and do not contribute to the loss.

We optimize a binary cross-entropy objective that encourages the target to assign high probability to the guide's nearest neighbors and low probability to other examples. For each pair (i, j), define

$$\ell_{ij} = -\boldsymbol{T}_{ij}\log\sigma(\boldsymbol{Z}_{ij}^A) - (1-\boldsymbol{T}_{ij})\log\left(1-\sigma(\boldsymbol{Z}_{ij}^A)\right), \tag{11}$$

where $\sigma(\cdot)$ is the logistic sigmoid. For each anchor i, we average the positive and negative terms separately:

$$\ell_i^+ = \frac{1}{k}\sum_{j\neq i}\boldsymbol{T}_{ij}\ell_{ij}, \;\; \ell_i^- = \frac{1}{b-1-k}\sum_{j\neq i}(1-\boldsymbol{T}_{ij})\ell_{ij}. \tag{12}$$

The DiffKNN dissimilarity is then

$$\Delta_{\text{DiffKNN}}(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{b}\sum_{i=1}^{b}\frac{\ell_i^+ + \lambda\ell_i^-}{1+\lambda}, \tag{13}$$

where λ controls the weight of negative examples. In our experiments, the guide network is frozen, so gradients are taken only through the target logits $\boldsymbol{Z}^A$. Thus, DiffKNN provides a differentiable objective for matching the guide's local neighborhood structure, even though the guide neighborhoods themselves are computed using a discrete top-k operation.

Unless otherwise stated, we use cosine similarity with row-standardized logits, $\lambda = 1$, and fixed values of k and τ across each experiment. In all experiments, we set k to 10 and τ to 1.0.

B.2 Stage Mapping

We describe how stages and layers are mapped across target and guide networks. As covered in Section 4, we only used the top half of all networks as justified later in Section F. In each setting, we first take a linear mapping from the target network to the layers of the guide network. This follows a simple linear interpolation where the target network layers are spread out over the guide network layers, similar to the mapping used in Subramaniam et al. [36]. We then drop the bottom half of target network layers to get the final NoT stages.

For certain settings, we align different layers. In ResNets, batch normalization can inflate alignment. Therefore, we only consider alignment from the last convolutional layer in each block. Similarly, we take the pre-layer-normalization layers in ViTs and GPT-2. This corresponds with the second feedforward layer. In ConvNeXt-V2, this corresponds to the last 1x1 convolutional in a ConvNeXt-V2 block. Each layer represents a stage and is mapped to the corresponding guide network layer.

B.3 Training Details

We cover training hyperparameters in Table 1. As stated in Section 4, our optimization via NoT uses one epoch per progressive alignment stage. In our setting, we have anywhere between 3-6 alignment stages, making our alignment much shorter than used in previous work.

Setting	Alignment Epochs per Stage	Optimizer	Learning Rate (Alignment)	Learning Rate (Standard Training)	Warmup (Alignment)	Warmup (Standard Training)	Weight Decay	Gradient Clipping
ResNets	1	AdamW [21]	1×10^{-3}	1×10^{-3}	3%	5%	0.1	1.0
ViTs	1	AdamW	1×10^{-3}	1×10^{-3}	3%	5%	0.1	1.0
ConvNeXtV2	1	AdamW	1×10^{-3}	1×10^{-3}	3%	5%	0.1	1.0
GPT-2	1	AdamW	5×10^{-4}	5×10^{-4}	3%	5%	0.1	1.0

Table 1: **Training details for depth and detail transfer.** We report all hyperparameters for our optimizer and scheduler. All alignment stages use 1 full epoch of training with 3% warmup per stage. We use the AdamW optimizer with the same learning rate for alignment and standard training. The learning rate is chosen for best reproduction of baseline performance.

ImageNet Top-1 Accuracy (↑): ResNet

Target	Baseline	CKA guide → target					DiffKNN guide → target				
		RN-10	RN-18	RN-50	RN-101	RN-152	RN-10	RN-18	RN-50	RN-101	RN-152
RN-10	60.88 ±0.59	61.02±0.92	63.69 ±0.96	64.03 ±0.99	64.05 ±0.56	63.99 ±0.49	61.05±0.54	63.72±0.61	64.51±0.51	64.49±0.89	64.40±0.92
RN-18	68.16±0.11	67.54 ±0.81	69.00 ±0.44	70.57 ±0.89	71.29±0.65	71.71±0.74	68.02±0.47	68.92±0.93	71.01±0.26	71.14±0.46	71.94±0.54
RN-50	74.43±0.14	73.19±0.75	74.01±0.72	74.50±0.61	77.00±0.696	77.54±0.78	74.17±0.86	74.83±0.81	74.66±0.62	77.02±0.71	77.55±0.86
RN-101	76.11±0.16	75.69±0.78	75.88±0.71	76.00±0.51	76.16±0.72	78.80±0.69	76.00±0.61	76.03±0.54	75.91±0.65	76.13±0.52	78.84±0.78
RN-152	77.05±0.12	76.12±0.91	76.91±0.96	77.00±1.01	77.10±0.85	77.09±1.02	76.09±0.56	77.00±0.67	77.03±1.00	77.15±0.99	77.06±0.87

ImageNet Top-1 Accuracy (↑): ViT

Target	Baseline	CKA guide → target				DiffKNN guide → target			
		ViT-Ti	ViT-S	ViT-B	ViT-L	ViT-Ti	ViT-S	ViT-B	ViT-L
ViT-Ti	56.30±0.24	56.11±0.54	57.10±0.44	57.82±0.37	58.61±0.79	56.07±0.62	57.61±0.61	57.95±0.74	58.92±0.74
ViT-S	65.55±0.17	65.20±0.57	65.80±0.38	66.59±0.44	67.00±0.79	65.48±0.83	65.69±0.89	66.79±0.91	67.19±0.57
ViT-B	74.14±0.25	74.11±0.24	74.12±0.36	74.89±0.39	75.16±0.31	74.07±0.56	74.17±0.79	74.56±0.42	75.37±0.56
ViT-L	77.69±0.19	77.91±0.68	77.66±0.77	77.82±0.53	77.91±0.74	77.67±0.85	77.73±0.82	77.73±0.74	77.68±0.76

ImageNet Top-1 Accuracy (↑): ConvNeXtV2

Target	Baseline	CKA guide → target					DiffKNN guide → target				
		CNXV2-Atto	CNXV2-Nano	CNXV2-Tiny	CNXV2-Base	CNXV2-Large	CNXV2-Atto	CNXV2-Nano	CNXV2-Tiny	CNXV2-Base	CNXV2-Large
CNXV2-Atto	71.25±0.11	71.59±0.56	71.80±0.49	72.89±0.51	73.93±0.47	74.24±0.51	71.61±0.67	71.80±0.92	72.95±0.83	74.00±0.42	74.49±0.82
CNXV2-Nano	73.14±0.16	73.10±0.57	73.11±0.34	74.46±0.39	74.98±0.56	76.01±0.61	73.19±0.65	73.09±0.36	74.61±0.41	75.12±0.38	76.20±0.57
CNXV2-Tiny	79.41±0.19	79.42±0.77	79.41±0.39	79.73±0.42	80.99±0.36	81.22±0.47	79.34±0.82	79.32±0.76	79.73±0.77	80.96±0.82	81.26±0.77
CNXV2-Base	81.63±0.15	81.01±0.57	81.04±0.58	81.60±0.48	81.82±0.47	82.59±0.76	81.02±0.74	81.01±0.51	81.21±0.46	81.68±0.47	82.71±0.87
CNXV2-Large	83.76±0.18	83.71±0.26	83.72±0.54	83.76±0.55	83.74±0.79	83.80±0.68	83.64±0.33	83.68±0.45	83.77±0.37	83.71±0.75	83.79±0.91

Perplexity (↓): GPT-2

Target	Baseline	CKA guide → target					DiffKNN guide → target				
		GPT-2-Nano	GPT-2-Tiny	GPT-2-Base	GPT-2-Medium	GPT-2-Large	GPT-2-Nano	GPT-2-Tiny	GPT-2-Base	GPT-2-Medium	GPT-2-Large
GPT-2-Nano	69.11±0.96	69.23±1.17	68.08±1.15	66.12±1.05	65.87±1.07	65.89±1.06	69.17±0.89	68.06±1.01	66.03±0.92	65.54±0.96	65.49±0.99
GPT-2-Tiny	51.16±0.91	51.32±1.06	51.05±1.07	48.69±1.03	48.65±1.07	48.42±1.05	51.14±1.10	51.19±0.93	48.55±0.96	48.51±1.00	48.53±1.01
GPT-2-Base	28.01±0.93	29.02±1.02	28.16±1.01	28.17±1.07	26.91±1.08	26.87±1.02	28.56±0.95	28.04±1.04	27.99±1.05	26.74±1.06	26.70 ±0.97
GPT-2-Medium	22.55±0.98	23.98±1.01	22.51±1.08	22.52±1.07	22.41±1.00	21.87±1.00	24.14±0.98	22.64±0.97	22.57±1.04	22.51±1.05	21.78±1.10
GPT-2-Large	16.44±0.90	16.91±0.96	16.93±1.10	16.76±1.09	16.65±1.02	16.42±1.01	17.10±0.92	16.42±0.93	16.41±1.01	16.59±1.02	16.40±0.91

Table 2: **NoT transfers across model families and scales.** Each row is a target network, and each guide column reports the resulting NoT performance when transferring from that guide into the target. The diagonal entries correspond to self-transfer. Baseline denotes the target-only model performance. Vision results report ImageNet top-1 accuracy, where higher is better; language results report perplexity, where lower is better.

Our alignment uses the AdamW optimizer with a tuned learning rate. In general, we find one epoch sufficient to get the best alignment optimization. We also use a cosine scheduler [20] with 3% warmup. When applying standard training, we find that the same learning rate is sufficient to find similar results as what is reported in previous works. We apply 100 epochs of standard training with the AdamW optimizer, a cosine scheduler, and 5% warmup. Across all training we use gradient clipping to a norm of 1.0 to stabilize training. Similarly, across all settings, we use a batch size of 256. For this work, we trained all networks on eight H200s over the period of one month. A single conversion can be completed with one H200 in a single day.

C Network Performance

We show raw network performance across our settings in Table 2. This includes the raw performance, accuracy or perplexity, along with the standard error for measuring performance.

D Theoretical Results

This section explains why an initialization-only alignment procedure can affect early supervised training. After alignment, the guide is removed, so the only object that can influence later learning is the geometry encoded in the target initialization. On a supervised batch (X, Y), this geometry is captured by the target feature kernel

$$K_0^S(X) = H_0^S(X)H_0^S(X)^\top.$$

We compare it to the corresponding guide kernel

$$K_0^D(X) = H_0^D(X)H_0^D(X)^\top$$

through

$$\varepsilon_0(X) = \|K_0^S(X) - K_0^D(X)\|_{\mathrm{op}}.$$

This derivation necessarily is conditional on this quantity being small because alignment is optimized on the unlabeled set U, while supervised learning happens on batches X.

Connecting alignment to kernel similarity. CKA alignment controls the normalized centered kernel shape. In particular, if two nonzero kernels A and B satisfy

$$\left\langle \frac{A}{\|A\|_F}, \frac{B}{\|B\|_F} \right\rangle_F = 1 - \delta,$$

then

$$\left\| \frac{A}{\|A\|_F} - \frac{B}{\|B\|_F} \right\|_F = \sqrt{2\delta}.$$

Thus, high CKA means that the two kernels have similar shape, up to scale. If their norms are also close, then the raw kernels are close. This clarifies what CKA does and does not guarantee: it aligns normalized geometry, while supervised dynamics depend on the raw kernel or NTK.

First supervised step. Consider logits

$$F = H_\theta(X)V$$

with a zero-initialized head $V_0 = 0$. Let

$$G_0 = \nabla_F \mathcal{L}(F_0, Y).$$

Since $F_0 = 0$, the first backbone update is zero. The gradient through the backbone is multiplied by V_0. The readout update is

$$V_1 = -\eta H_0^\top G_0,$$

and therefore

$$F_1 = H_0 V_1 = -\eta H_0 H_0^\top G_0 = -\eta K_0 G_0.$$

Applying this identity to the aligned target S and the guide D gives

$$\|F_1^S - F_1^D\|_F \le \eta \|K_0^S - K_0^D\|_{\mathrm{op}} \|G_0\|_F = \eta \varepsilon_0(X) \|G_0\|_F.$$

Thus, if alignment makes the target and guide kernels close, then their first supervised logit updates are close.

Initial NTK interpretation. At zero head, the empirical NTK of

$$f_{\theta,V}(x) = V^\top h_\theta(x)$$

is exactly

$$\Theta_0 = K_0 \otimes I_C.$$

The backbone contribution vanishes because $V_0 = 0$, and the head contribution is the feature kernel replicated across classes. Therefore, matching the initial feature kernels is the same as matching the initial NTKs in this setting.

Short-time dynamics. The same conclusion holds for the beginning of gradient flow. With $V_0 = 0$,

$$V_t = -tH_0^\top G_0 + o(t), \qquad \theta_t = \theta_0 + o(t), \qquad F_t = -tK_0G_0 + o(t)$$

as $t \downarrow 0$. Therefore the initial kernel controls not only the first discrete step, but also the leading-order continuous-time trajectory.

Robustness and finite time. The zero-head assumption is mainly for clarity. If the head is small, $\|V_0\|_F \le \alpha$, then the initial backbone gradient is small and the NTK is only a small perturbation of $K_0 \otimes I_C$:

$$\|\nabla_\theta \mathcal{L}(F_0, Y)\| \le B_X \alpha \|G_0\|_F, \qquad \|\Theta_0 - K_0 \otimes I_C\|_{\mathrm{op}} \le B_X^2 \alpha^2.$$

For longer times, one can compare the aligned target trajectory to a hypothetical guide trajectory. If both NTKs remain close to their initial values and the loss gradient is Lipschitz in the logits, then the logit distance satisfies a Gronwall bound:

$$\delta_t \le e^{\kappa L t}\delta_0 + G \int_0^t e^{\kappa L(t-s)} \left(\epsilon_0 + \rho_s^S + \rho_s^D\right) ds.$$

This bound says that when the initial kernels are close and the NTKs do not drift too quickly, the early supervised trajectories remain close.

Takeaway. The optimization prior does not transfer labels, logits, or trained knowledge. It changes the target initialization so that the target begins supervised learning with a feature kernel closer to that of the guide. This kernel determines the first supervised updates, and under mild stability assumptions it also explains the early training trajectory.

E Recursive Transfer

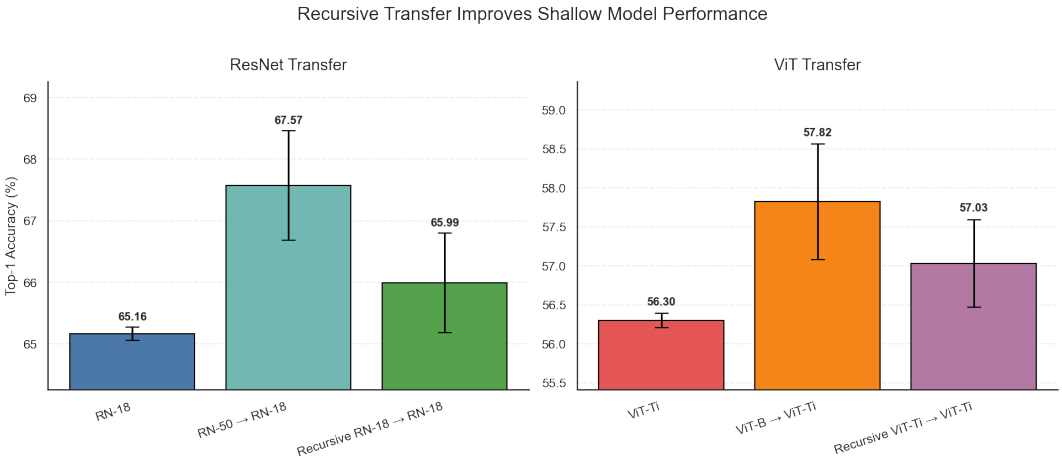


Figure 8: **Recursive guide networks can improve shallow network performance.** We construct a randomly initialized recursive guide by repeating a single block from the target architecture, matching the depth of a larger guide such as ResNet-50 or ViT-B. Recursive guides improve ResNet-18 and ViT-Ti over standard initialization, although they are weaker than the original deep guides. This suggests that additional guide computation helps, but width, spatial structure, and other architectural factors also matter.

To test whether additional guide computation is sufficient to improve the target, we introduce guide networks that apply recursive computation. This follows previous work [42, 41, 8] showing that recursive or looped computations can improve model performance. We take a single block of the target network and repeat it several times, applying the same block recursively to its own activations. For ResNets, we repeat a single ResNet-18 block from the first stage to construct a recursive guide with the same depth as ResNet-50. For ViTs, we repeat a single ViT-Ti block to construct a recursive guide with the same depth as ViT-B. We show results in Figure 8. We find that randomly initialized recursive guides improve shallow target performance, although not as strongly as the original deep

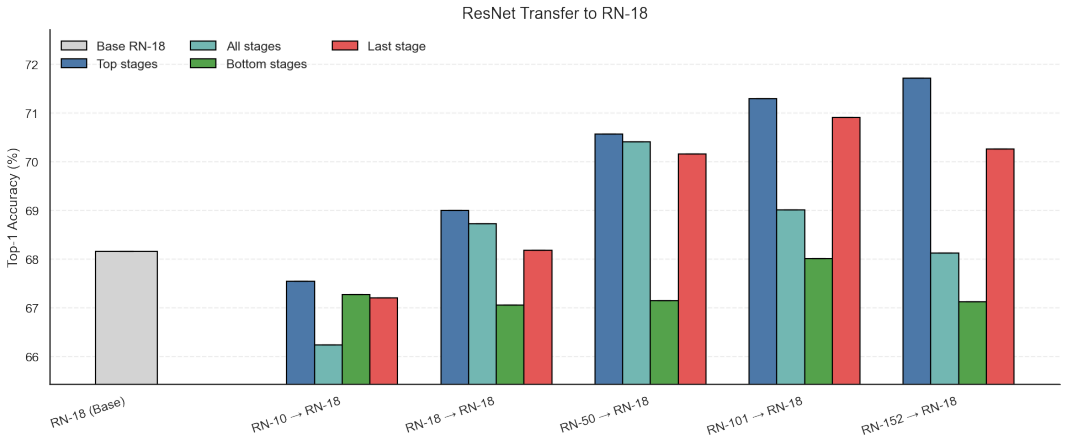


Figure 9: **Selecting the top half of NoT stages yields the best results.** Rather than aligning all stages across our target and guide network, in this work, we only align the top half of stages in the target network to the randomly initialized guide network. Above, we compare this choice with aligning all the stages, the bottom half of stages, or just the last stage. Only the last stage achieves comparable results but falls short when transferring ResNet-152 to ResNet-18. This establishes that the top half of stages have enough rich geometry to differentiate across networks. A similar trend exists for instilling detail to coarse representations.

guide networks. This indicates that additional computation in the guide can help induce better target initializations. At the same time, the gap between recursive guides and full deep guides suggests that depth alone is not the whole story. Other architectural factors, including width and spatial structure, also contribute to the geometry transferred by the guide.

F Ablation Experiments

As discussed in Section 4, we use a budgeted version of NoT in comparison to what is reported in prior work [37]. Early experiments established that using all stages in NoT led to noisy results in most settings with instilling depth and instilling detail. Upon further inspection, we found that most networks had very similar representations at early stages. This was not found in previous works as representations were fairly different across the target and guide network settings.

In this work, settings like transferring ResNet-101 to ResNet-18 would fail when using all stages for alignment. To test this thoroughly, we ran a systematic ablation where we compared using all stages of NoT with using half the stages and using only the last stage for alignment. We report results in Figure 9 where we show our results with ResNet-18 as the target network and vary the ResNet depth as the guide network. We find that results are noisy and inconsistent when using all stages of NoT, so early layers of ResNet-18 are aligned to early layers of the randomly initialized guide network. For example, we can faithfully improve when aligning ResNet-18 to ResNet-50 but do not succeed when aligning to ResNet-101. This is confirmed when using only the bottom half of stages for alignment, which achieves much worse results when aligning ResNet-18 to ResNet-101. When aligning based on only the last stage, we achieve slightly worse results than using the top half of the stages. We speculate that in cases with less compute, using the last stage would be sufficient in certain settings. We find a similar trend when analyzing our setting aligning ViT-B/32 to ViT-B/16, indicating that the early layers of two networks are often very similar.

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 carefully scope contributions in the abstract and introduction and run experiments and provide theoretical derivations to demonstrate the correctness of our findings. Our findings are on using representational alignment as an optimization prior at initialization and we demonstrate that this leads to improvement over standard training.

   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: Limitations are discussed in the conclusion. We are transparent about limitations.

   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: [Yes]

Justification: Our proof includes lemmas and remarks to intuitively cover the underlying assumptions and we are up front about the connection of our proof to the findings.

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: We include training details for all settings including hyperparameters necessary, the exact networks we define and use, the optimizer used, the number of training steps, the learning rate scheduler, etc. These are covered in Section 4 and Section B.3. We use open-source models and datasets in this work.

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 will be released upon acceptance of the paper. We will open-source our code for the community to build on top of.

   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: We cover training and testing details as well as hyperparameters, optimizers, etc. Our method follows standard training protocols and is transparent about choices made.

   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 to get error bars and report standard error on our final accuracy and perplexity values across the random seeds.

   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 describe our compute resources in Section B.3. These include resources necessary for preliminary and failed experiments.

   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: This work does not involve human subjects and does not have data concerns. We do not have societal impacts to discuss deeply.

   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: This work studies neural networks and deep learning. We believe that there won't be strong negative impacts of this work but will include a statement about this upon acceptance of the paper.

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: This work does not require safeguards.

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: [N/A]

Justification: This work uses and cites existing assets that are already open source and available for use by the community.

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: This paper does not introduce any new assets.

    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: This work does not incorporate with human subjects.

    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: This work does not incorporate 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 any important, original, or 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.