TRAINING THE UNTRAINABLE: INTRODUCING INDUCTIVE BIAS VIA REPRESENTATIONAL ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We demonstrate that architectures which traditionally are considered to be ill-suited for a task can be trained using inductive biases from another architecture. Networks are considered untrainable when they overfit, underfit, or converge to poor results even when tuning their hyperparameters. For example, plain fully connected networks overfit on object recognition while deep convolutional networks without residual connections underfit. The traditional answer is to change the architecture to impose some inductive bias, although what that bias is, is unknown. We introduce guidance, where a guide network guides a target network using a neural distance function. The target is optimized to perform well and to match its internal representations, layer-by-layer, to those of the guide; the guide is unchanged. If the guide is trained, this transfers over part of the architectural prior and knowledge of the guide to the target. If the guide is untrained, this transfers over only part of the architectural prior of the guide. In this manner, we can investigate what kinds of priors different architectures place on a fully connected network. We demonstrate that this method overcomes the immediate overfitting of fully connected networks on vision tasks, makes plain CNNs competitive to ResNets, closes much of the gap between plain vanilla RNNs and Transformers, and can even help Transformers learn tasks which RNNs can perform more easily. We also discover evidence that better initializations of fully connected networks likely exist to avoid overfitting. Our method provides a mathematical tool to investigate priors and architectures, and in the long term, may demystify the dark art of architecture creation, even perhaps turning architectures into a continuous optimizable parameter of the network.

1 INTRODUCTION

When creating neural networks, as a community we follow recipes that select among a few architectures known to work for particular tasks (Ren et al., 2021; Cong & Zhou, 2023; Goodfellow et al., 2014). Architecture is critical. The gains made on tasks like object recognition had to do with imposing an inductive bias, i.e., a prior, by designing new architectures (Goyal & Bengio, 2022; Bachmann et al., 2024). Convolutional networks unlocked many vision problems (Krizhevsky et al., 2012; He et al., 2016a) and the recent advent of the Transformer (Vaswani, 2017; Devlin, 2018; Achiam et al., 2023) did the same for language. Despite this, finding new architectures and overcoming the limitations of existing architectures is a dark art. While an architecture imposes some prior, we often do not fully understand what that prior is. For example, there is discussion about exactly what the role of residual connections is (Jastrzebski et al., 2017). If we fully understood the priors that architectures imposed, we could translate between priors and architectures freely – either by creating the prior we want and then deriving the appropriate architecture or by dispensing with architectures and imposing the prior directly.

Recent theorems (Poggio & Fraser, 2024) state that for each function which is efficiently Turing computable, there exists a deep network that can approximate it well. Furthermore, a graph representing such a function is compositionally sparse, that is the nodes of the associated Directed Acyclic graph (DAG) represent constituent functions with a small effective dimensionality. A reasonable conjecture is that neural networks with an architecture which is similar to the DAG of the unknown target function are especially successful in learning it, as it is the case for convolutional networks for image recognition and similar tasks. Because we do not understand the relationship

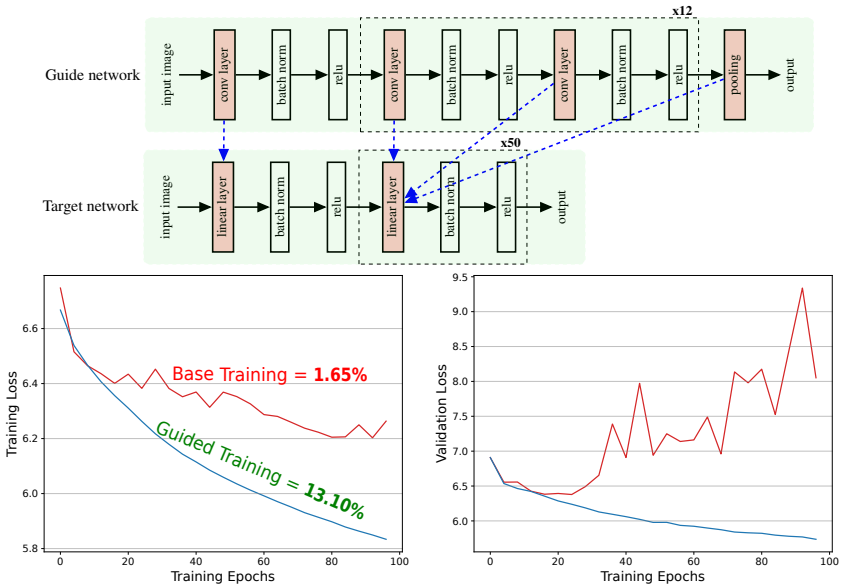

Figure 1: **Guidance between two networks makes untrainable networks trainable**. Given a target which cannot be trained effectively on a task, e.g., a fully connected network which immediately overfits on vision tasks, we guide it with another network. In addition to the target's cross-entropy loss, we encourage the network to minimize the representational similarity between target and guide activations layer by layer. The guide can be untrained, i.e., randomly initialized. This procedure transfers the inductive biases from the architecture of the guide to the target. The guide is never updated. The target undergoing guidance no longer immediately overfits can now be be trained. Here we show an untrained ResNet guiding a deep fully connected network to perform object classification. The FCN alone overfits, the guided version can now be optimized. It has gone from untrainable to trainable.

between the kinds of priors on the target functions that different architectures impose, even simple questions have no known answer, such as, does there exist an initialization of an FCN that makes it behave like a CNN, though the graphs of the function they represent are fundamentally different?

We introduce a new tool which we term *guidance* to study this problem. Given a *target network*, we guide it with a *guide network*. In addition to the target's original loss, the target attempts to match the representation of its intermediate layers to those of the guide. We use a measure of representational similarity (Kornblith et al., 2019; Cristianini et al., 2001; Cortes et al., 2012), also termed a neural distance function, to compute the distance between representations of two arbitrary layers. Neural distance functions are often used in neuroscience to compare activity in networks and brains (Schrimpf et al., 2018; Conwell et al., 2021a; Subramaniam et al., 2024). In light of recent work that shows that networks of very different architectures have internal activity that is extremely similar to one another (Han et al., 2023; Conwell et al., 2021a;b), we repurpose this distance function as a means to transfer priors between networks layer by layer.

We make the following contributions:

1. We develop guidance to transfer priors between networks using representational alignment, investigate one representational alignment method, centered kernel alignment, CKA (Kornblith et al., 2019), and apply it to several classical problems.
2. Image classification improvements: Deep or wide fully connected networks stop overfitting when guided by a ResNet. No-skip CNNs close much of the gap with ResNets when guided by a ResNet.
3. Sequence modeling improvements: RNNs significantly improve their copy and paste accuracy when guided by a Transformer. Transformers increase their parity accuracy when guided by an RNN. RNNs close much of the gap to a Transformer when guided by one.
4. Many of these gains exist when the guide network only transfers over its priors, not its knowledge, i.e., the guide is untrained and is never updated.
5. By disconnecting the guide early, we provide evidence for a better initialization regime for fully connected networks that avoids immediate overfitting.

Through the transfer of priors across architectures, guidance can serve as a new tool for investigating the role of architecture and the relationship between priors and architectures. It can also help discover new initialization methods, for which no known tool exists.

Our work has a number of limitations. We aimed for coverage of many tasks instead of maximal performance on any one task. This would have required us to carefully tune the hyperparameters involved. We preferred to show how guidance works in general rather than in cherry picked and carefully tuned settings. To that end, we also did not optimize networks to convergence, nor did we attempt to experiment with other optimizers. Once we reproduced a well-known problematic training phenomenon, we showed that it could be overcome. We consider a network trainable and a problem to be overcome, when the original problem disappears. For example, successfully training fully connected networks for object recognition was hopeless because they immediately overfit; using our guidance method they no longer do so. This does not mean that they are necessarily useful as object recognizers at present. In the case of fully connected networks, their present performance with guidance training is too low, but with additional work we believe their performance could be substantially increased now that their train and test loss are moving in the right direction. In some cases, by applying guidance, we do see large useful improvements, such as with deep CNNs, RNNs and Transformers, although much more remains to be exploited there too.

2 Related Work

Representational Distance: Our method builds on several metrics that measure distance between high-dimensional activations extracted from neural networks or activity in the brain (Klabunde et al., 2023). Some of these distance metrics make comparisons based on kernel matrices (Kornblith et al., 2019; Cristianini et al., 2001; Cortes et al., 2012) or relative distances (Kriegeskorte et al., 2008; Moschella et al., 2022) between sample representations in a set. Others compute linear (Wehbe et al., 2014; Schrimpf et al., 2018) or orthogonal projections (Beauducel, 2018) from one set of representations to another. These metrics are designed based on a set of desired invariant properties such as permutation invariance or invariance to linear transformations.

Such approaches have been commonly applied in neuroscience for measuring representational distance of activations from networks and activity in the brain to understand which neural networks are architecturally most similar to the brain (Wehbe et al., 2014; Conwell et al., 2021a; Subramaniam et al., 2024; Goldstein et al., 2020). Under this context, Han et al. (2023) has shown the inability of current representational distance metrics – specifically the metric used here, centered kernel alignment – to distinguish representations based on architecture. This paper gives the foundation for our intuition that networks may have similar representations that allow for transferring inductive biases from one network to another.

Untrainable Networks: This work examines FCNs and plain CNNs for image classification and RNNs for sequence modeling. Prior work explored similar approaches but performed poorly compared to the guide networks we leverage for improved training.

For image classification, small feed-forward networks with 3-5 hidden layers and fewer than 100 units per layer were trained on object recognition datasets (Ma & Khorasani, 2004; Bebis & Papadourakis, 1992; Khasnobish et al., 2012; Oh & Suen, 2002). These efforts prioritized training fit over generalization performance and achieved low results (Ma & Khorasani, 2004; Bebis & Papadourakis, 1992). Strategies to reduce overfitting, such as topological structure (Schittenkopf et al., 1997) or early stopping (Caruana et al., 2000), were hindered by complex designs and hyperparameter tuning, leading to poor training fits. Further methods used alignment between thin deep FCNs and wide shallow FCNs to prevent overfitting, an approach similar to our paper (Romero et al., 2014). Deep convolutional networks were also applied to image classification (Krizhevsky et al., 2012; Bengio et al., 1993) but struggled with vanishing gradients, limiting their depth.

In sequence modeling, classical RNNs (Schuster & Paliwal, 1997; Pearlmutter, 1990; Connor et al., 1994; Hammer, 2000) were constrained by vanishing and exploding gradients (Hochreiter, 1998), making them unsuitable for long sequence tasks requiring memorization (Graves, 2014). Gradient flow techniques were developed, but significant progress came from architectures like LSTMs (Hochreiter, 1997) and transformers (Vaswani, 2017). Transformers, however, have been found un-

trainable on formal language tasks requiring full-sequence reasoning, where RNNs succeed (Bhattamishra et al., 2020).

Model Distillation: Guidance shares a resemblance with model distillation (Hinton, 2015; Gou et al., 2021; Sanh, 2019; Hsieh et al., 2023). Distillation transfers knowledge from a teacher model to a student model by introducing a new component to the loss function that enforces the student model to behave like the teacher model (Kim et al., 2021; Zhou et al., 2021). This usually consists of penalizing the KL-divergence between the logit predictions of the student and teacher model.

Representation-based distillation (Tian et al., 2019; Chen et al., 2021; Lin et al., 2020) and alignment techniques have been proposed to improve alignment between two networks. Certain works have proposed correlation congruence or similarity preserving metrics (Ramos et al., 2023) for aligning two networks, particularly as a way to do architecture search between CNNs (Bashivan et al., 2019). 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 (Saha et al., 2022; Dapello et al., 2022).

We distinguish guidance from distillation. Guidance can use a smaller untrained guide instead of a larger trained teacher. This is due to guidance operating over intermediate activations of the network instead the output of the network probabilities or output features, like distillation does. Guidance also operates at many levels at the same time, aligning many layers at once. This helps address the credit assignment problem that gradient descent has when tuning weights early in a network. We also consider many more networks for guidance than is traditional for distillation including networks which have very different architectures like Transformers to RNNs. Distillation is usually carried out between two closely related architectures. We apply guidance to do the opposite.

3 METHODS

Guidance introduces a term in the loss of a target network, $\mathcal{N}^T$, to encourage representational alignment with a guide network, $\mathcal{N}^G$. Only the parameters, θ^T, of the target are updated — the guide's parameters, θ^G, remain fixed. Per minibatch, representational similarity is computed between the activations of i^Gth layer of the guide, $\mathbf{A}^G_{i^G}(\theta^G)$, and activations from a corresponding layer i^T of the target, $\mathbf{A}^T_{i^T}(\theta^T)$. We refer to the correspondence between layers of the guide $\{i^G\}$ and layers of the target $\{i^T\}$ as I. While this correspondence, I, could be complex as any two architectures can form a guide/target pair, here we choose architectures that make the correspondence obvious as is discussed later. For example, the stacked RNNs and Transformers have the same number of layers in our experiments.

The target and guide receive the same input. Per minibatch, we collect activations from intermediate layers of both networks. Layers of guide network are mapped to layers of the target network; see Figure 1. We formulate the loss in terms of minimizing the *representational dissimilarity*, $\bar{\mathcal{M}}$, i.e., the complement of a representational similarity metric, between guide and target activations layer by layer, summing the results. We only consider the centered kernel alignment, CKA, between the activations in this publication. Many other possibilities exist. Any representational similarity function which is differentiable could be used. Efficiency or incremental computation is much more important than it is in traditional applications since this operation happens for every minibatch. We discuss details in Appendix C.1.

Given $\mathcal{L}_T$ as the original loss of the target network, the guide network's original loss function is irrelevant. The guide need not even have been trained on the same task or the same dataset. It need not even have been trained at all. This latter setting is what allows transferring architectural priors without transferring knowledge from the guide to the target, as there is none in a randomly initialized guide. The final loss we optimize, $\mathcal{L}$ is:

$$\mathcal{L}(\theta^T) = \mathcal{L}_T(\theta^T) + \sum_{i \in I} \bar{\mathcal{M}}(\boldsymbol{A}^T_{i^T}(\theta^T), \boldsymbol{A}^G_{i^G}(\theta^G)) \tag{1}$$

This minimizes a task loss while increasing alignment between the target and guide networks given the mapping between them. The mapping may be sparse, not every layer needs to be used. This is important for guidance with transformers or stacked RNNs as will be explained later. We don't

incorporate any weight on the layer-wise similarity component of the loss. Note that the guide's parameters, θ^G, are constants, i.e., the guide is never updated.

Layerwise Mapping We design a simple method for mapping guide layers to target layers as part of providing supervision. The goal of this method is to make guide and target networks architecturally agnostic i.e. we can supervise any target network with any guide network.

As a simple approach, we evenly spread layer computations of our guide network over our target network. For example, if we consider ResNet-18 and a 50-layer FCN, we would map every convolutional ResNet layer to every second or third linear layer of the FCN. The intuition for this approach follows from the aim of discovering the same function of the guide network using a target network. Through the design of evenly spreading layers of our ResNet-18, we are guiding the FCN to find a function similar to the guide network.

For our mapping, we consider activations from all layers with tunable weights i.e. convolutional, linear, or LSTM/RNN based layers as well as activations from layer normalizations. For multiple stacked RNNs, LSTMs, or transformers, we extract feature representations from intermediate layers in the stack as well. Using all layers is useful for guidance as it provides a strong signal to induce alignment between the target and guide networks during training. We empirically find that more layers leads to stronger results. Skipping layers based on non-linear transformations reduces memory overhead associated with storing representations per batch.

4 Experiments

We design several settings with different target and guide networks to thoroughly test our approach. We include a range of image and sequence modeling tasks. In choosing target networks, we consider a broad range of designs for networks that are not traditionally applied (e.g., a FCN in image classification).

To systematically evaluate our approach, we

1. **Untrainable Architectures**: Experiments where the target networks are difficult to train due to architectural limitations, irrespective of the task. For example, overfitting in deep FCNs or memory incorporation in RNNs.
2. **Untrainable Tasks**: Experiments where certain tasks are inherently challenging for specific architectures, making them untrainable without additional supervision. For example, sequence classification with transformers.

Tasks We describe the task settings. For an image-based task, we focus on *image classification* and use the ImageNet-1K dataset (Deng et al., 2009) for training and testing. We use the splits defined by the dataset. We report accuracy on the validation set for all experiments.

We consider three sequence modeling tasks to allow for a broader range of architectural settings. We first consider a task called *copy-paste* (Graves, 2014). In this task, we generate a sequence of numbers in the range of $1 - 10$. The model is trained to recover the same sequence in the output. In our setting, we consider sequence lengths that range from 20 to 40 values total (internal sequence and padding). We generate a copy-paste dataset, sampling sequences containing numbers between 1 and 10. We generate a total of $100,000$ examples, training on $80,000$ examples, validating on $10,000$ examples, and testing on $10,000$ examples.

We also include the *parity* task, a binary classification task where a model is fed a bitstring and outputs 1 when there is an even number of ones in the bitstring and 0 otherwise. We generate a series of bitstrings with sequence lengths that range from 2 and 50 as done in prior work (Bhattamishra et al., 2020).

Finally, we consider a *language modeling* task using the WikiText-103 dataset (Merity et al., 2016) where models must predict the next token given some context. This uses the train, validation and testing splits defined by the WikiText dataset and for all experiments, we use a context length of 50. We tokenize the text data using the GPT-2 (Radford et al., 2019) tokenizer.

Tasks	Guide Networks	Target Networks
Image Classification	ResNet-18	Deep FCN Wide FCN
	ResNet-50	Deep ConvNet
Copy-Paste	Transformer	RNN
Parity	RNN	Transformer
Language Modeling	Transformer	RNN

Table 1: **Guide and target networks across tasks**. Our network designs include several untrainable target networks and corresponding trainable guide networks. We focus on FCNs and deep convolutional networks for image classification and recurrent networks for sequence modeling.

Architectures For all tasks, we describe our target untrainable architectures for each task separately as well as the guide networks that are employed to make the untrainable network trainable. We give an overview in table 1.

Image Classification: We use three target networks: Deep FCN, Wide FCN, and Deep ConvNet. Deep FCN is a fully-connected network with 50 blocks consisting of feedforward layers followed by non-linearities. This network is an untrainable architecture, lacking inductive biases to prevent overfitting and having vanishing gradients. Wide FCN is a fully connected network with 3 blocks with feedforward layers that have 8192 units. This is categorized as an untrainable task due to a saturation in the training performance. Deep ConvNet is the same architecture as ResNet-50 (He et al., 2016a), but without residual connections. This is categorized as an untrainable architecture due to the vanishing gradient problem. We use two guide networks: ResNet-18 and ResNet-50. ResNet-18/50 is a deep convolutional network with 18/50 convolutional blocks and residual connections. We refer to He et al. (2016a). We supervise the Deep FCN and Shallow FCN with ResNet-18 and supervise the Deep ConvNet with ResNet-50.

Sequence Modeling: For our copy-paste and language modeling tasks, we use a vanilla, 4-layer RNN as our target network. In copy-paste, architectural and algorithmic limitations make RNNs an untrainable architecture. For language modeling, vanishing gradients and limited context incorporation make RNNs an untrainable architecture as the training loss saturates. For the parity task, we use a 1-layer transformer encoder architecture, similar to prior work (Bhattamishra et al., 2020; Hahn & Rofin, 2024). For the copy-paste task, we train a guide network, 4-layer transformer decoder model which achieves 96.90% accuracy. Similarly, for language modeling, we train a 4-layer transformer decoder guide network with a context window of 50. Our final perplexity is 29.69. For the parity task, we train a 1-layer vanilla RNN as a guide network which achieves 100% accuracy as reported by (Bhattamishra et al., 2020).

Training For each setting, we train the base target network and perform an experiment where both a trained and untrained guide network supervises the base target network. All networks are trained with either cross-entropy loss or binary cross-entropy loss, although other loss functions could be applied as well.

When training networks for object detection using ImageNet-1K, we use the Adam (Kingma, 2014) optimizer. For all sequence modeling tasks, i.e. copy-paste, parity, and language modeling we use AdamW (Loshchilov, 2017). For language modeling, we also incorporate gradient clipping due to unstable training with long sequences.

To ensure consistency of comparisons across learning curves, we use a consistent batch size of 256. In general, representational similarity metrics are affected by the number of samples in the calculation. The more samples, the better the metric approximates representational distance. We use 256 as a proxy, dependent on GPU memory, although more memory would allow for bigger batch sizes with potentially better results. Due to the large number of training settings, we employ several different learning rates. We tune the learning rate carefully for baseline training to ensure maximal performance of base training. We sweep the parameter across 5 different values and choose the results with the lowest validation loss. This ensures we are choosing the training with the best performance.

Experiment	**ImageNet Validation Accuracy** (↑)
ResNet-18	69.41
Untrained ResNet-18	0.10 ± 0.003
ResNet-50	75.99
Untrained ResNet-50	0.10 ± 0.013
Deep FCN	1.65 ± 0.51
ResNet-18 → Deep FCN	7.50 ± 1.51
Untrained ResNet-18 → Deep FCN	**13.10** ± 0.72
Wide FCN	18.78 ± 0.91
ResNet-18 → Wide FCN	**23.79** ± 0.86
Untrained ResNet-18 → Wide FCN	21.76 ± 0.54
Deep ConvNet	54.58 ± 1.01
ResNet-50 → Deep ConvNet	**64.66** ± 2.31
Untrained ResNet-50 → Deep ConvNet	55.02 ± 2.52

Table 2: **Guidance improves performance for image classification**. Alignment with a ResNet dramatically improves a deep FCN, particularly with an untrained ResNet. Significant gains are seen with a wide FCN as well. Deep CNNs without residuals gain only with a trained ResNet. Across all settings, guidance can help train architectures that were otherwise considered unsuitable.

Experiment	**Copy-Paste Accuracy** (↑)	**Parity Accuracy** (↑)	**Language Modeling Perplexity** (↓)
RNN	14.35 ± 0.01	100	74.68 ± 2.81
Untrained RNN	—	2.32 ± 0.41	—
Transformer	96.98	71.98 ± 3.16	29.69
Untrained Transformer	1.04 ± 0.81	—	51948.8 ± 90.44
RNN → Transformer	—	**78.49** ± 2.16	—
Untrained RNN → Transformer	—	70.38 ± 4.17	—
Transformer → RNN	23.27 ± 1.02	—	**41.88** ± 4.52
Untrained Transformer → RNN	**42.56** ± 1.51	—	45.61 ± 5.27

Table 3: **Guidance improves performance for sequence modeling**. RNN performance improves dramatically when aligning with the representations of a Transformer for copy and paste, as well as for language modeling. RNNs close most of the gap to Transformers for language modeling. Transformers in turn, improve parity performance when aligning with an RNN. Guidance is able to transfer priors between networks.

After choosing the optimal learning rate, we then train all networks and settings for 100 epochs with 5 random seeds to compute error bars for our training loss, validation loss, and testing accuracy. Our error bars are associated with the standard error across each step across all seeds. We choose the seed-based average test accuracy associated with the epoch with the lowest seed-based average validation loss.

5 Results

Image Classification Across all our networks, we observe significant improvement from using a guide network to provide representational guidance; see table 2. We see significant accuracy gains of 5-10% on validation performance. We also observe significantly better loss curves from a better fit with the training loss and reduced overfitting with the validation loss. Most interestingly, we highlight that using a randomly initialized guide network can perform better than using a trained guide network. For example, the Deep FCN results in the top left of fig. 2 are significantly better with a randomly initialized ResNet-18 as the guide network instead of a trained ResNet-18. This trend also occurs with Wide FCN.

We note that this trend is not entirely consistent as indicated by the Deep ConvNet. One potential explanation is that the architectural prior is the same for both the Deep ConvNet and ResNet-50. This explanation provides an additional interpretation for the role of residual connections and their influence on the representation space. This indicates that residual connections must be trained to have an influence on the representation space. This aligns with prior studies of residual connections (Jastrzebski et al., 2017; He et al., 2016b).

Sequence Modeling On the copy-paste task, see fig. 2 and table 3, we see significant improvement when using a transformer as our guide network over an RNN target network. Prior work has observed the difficulty that RNNs have with copy-paste and usually attributes this to a fundamen-

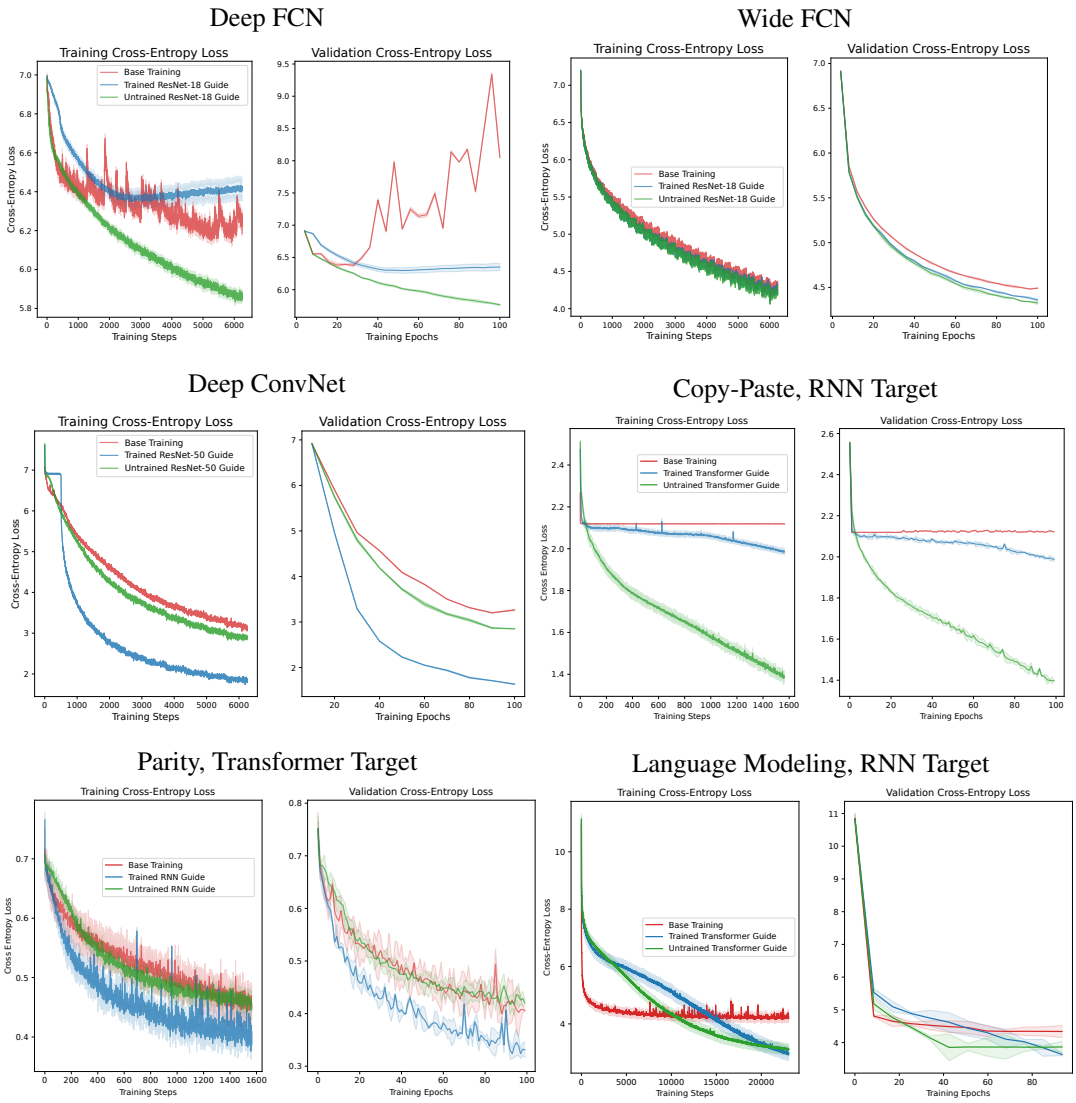

Figure 2: **Training and validation under guidance for all experiments reported in table 1**. For every result in Table 2 and Table 3 we show the training and validation loss with error bars across multiple runs although these are often too small to see. Note that often the best results occur with the untrained guide.

tal algorithmic limit on the capability of RNNs to memorize input sequences. Our results show a potential optimization scheme for RNNs that is applicable for sequence memorization. Similar to prior results with fully connected networks, this result improves when using a randomly initialized network. We explore this more thoroughly in further analyses but this is likely because optimization with randomly initialized networks is easier due to the degrees of freedom in CKA.

On the parity task, we similarly see improvement when using an RNN as our guide network over a transformer model as our target network. This is a complementary result to copy-paste and language modeling where the guide network was a transformer and our target network was a RNN. This improves over results from several prior papers that have pointed out fundamental limitations of transformers to perform certain formal language tasks.

Unlike the prior tasks, the performance improves when using a trained RNN as the guide network. This could be due to the wide gap in performance between an untrained RNN and trained RNN on parity. Trained RNNs achieve 100% on the task and could have meaningful positional information to complete the parity task. This information is likely crucial to the transformer which has limited sequence pooling capacity and fewer degrees of freedom unlike an RNN.

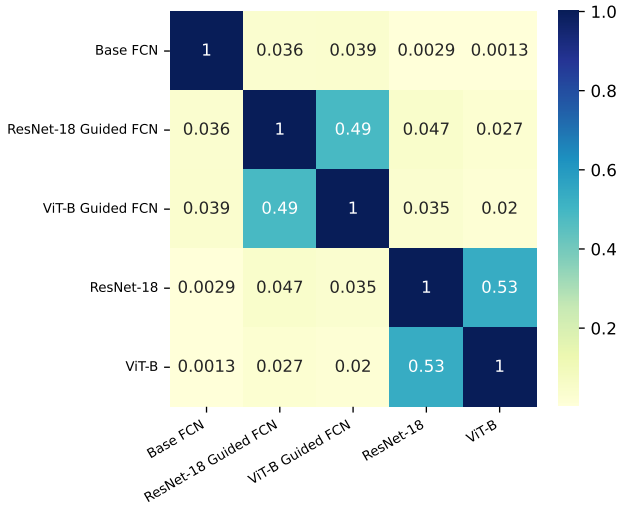

Figure 3: **Guidance aligns error consistency**. The relationship between the guide networks is mirrored in that of the guided networks even when the target is entirely unlike the guides initially. This is additional evidence that guidance doesn't just improve performance arbitrarily, the target becomes more like the guide.

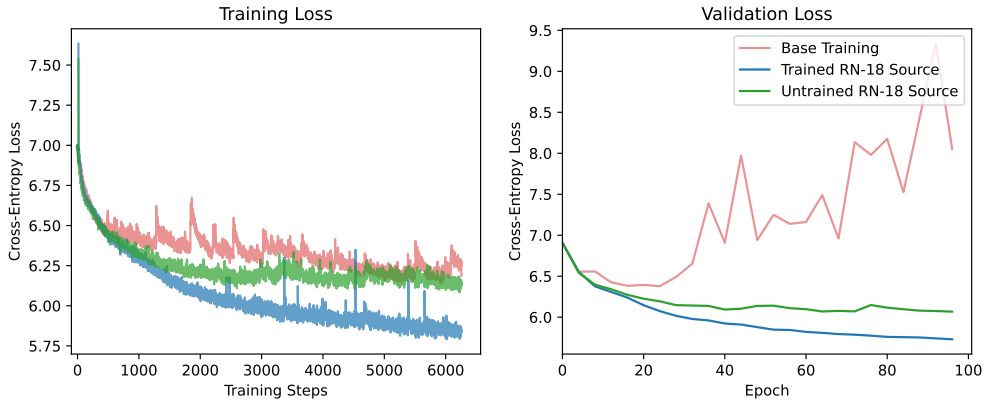

Figure 4: **Disconnecting the guide early can have long-term effects.** We remove the ResNet guide from the loss function of a Deep FCN after an arbitrary 150 steps. Surprisingly, this has little effect. The Deep FCN still does not diverge and the training and validation loss still mirror one another. Very likely, an initialization regime exists for Deep FCNs that prevents overfitting. We now have the tools to investigate this question.

On language modeling, similar to copy-paste, we see significant improvements with the RNN target network that is guided by a transformer guide network. While performance generally saturates for the 4-layer RNN, guide network guidance continuously improves the RNN performance by over 30 points for text perplexity for both trained and randomly initialized guide networks. This implies that information from the transformer can be transferred to the RNN. We see a similar trend with a randomly initialized transformer as the guide network implying that architectural priors in the transformer are driving improvement in guided network performance.

Error Consistency Given our guided networks, we can analyze the functional properties of the guided network to confirm whether networks adopt priors from their guide networks. Using Deep FCN as our target model, we guide it with a ResNet-18 or a ViT-B-16 (Dosovitskiy, 2020). We then measure the error consistency (Geirhos et al., 2020) between all of the networks; see fig. 3. The error consistency between the initial FCNs is entirely unlike the ResNet-18 or ViT-B. Guidance creates two FCNs which have the same relationship to one another. The ResNet-18-guided FCN and ViT-B-guided FCN have the same error consistency with respect to one another as ResNet-18 and ViT-B do. It's not just that the FCN gets generically better, it adopts a prior from the original architecture. We provide further details of the error consistency metric in appendix G.

Network Initialization Is guidance needed throughout training, or is the effect of the guide to move the target into a regime where the two are aligned and the target can be optimized further without reference to the guide? The answer to this question can shed light on whether the guide is telling us that better initializations are likely to exist for the target. To answer this question, we disconnect the guide from the target after an arbitrary and nominal number of training steps, 150. We did not investigate what the minimal number of steps needed was.

For FCNs, even this early disconnect avoids overfitting; see fig. 4. Furthermore, while preventing overfitting, we have lower training loss from guidance, indicating a better fit. This implies that there likely exists a better initialization for FCNs. In appendix K, we further explore how guidance can be used to find new initialization schemes in FCNs. In future work, we intend to investigate what the untrained guide is doing to the FCN weights to prevent validation loss from increasing dramatically.

6 Conclusion

We demonstrated that guidance eliminates the failure modes of networks previously considered unsuitable or ineffective for specific tasks. Aligning with another network overcomes these shortcomings by transferring inductive biases—either architectural and knowledge-based, or solely architectural when using an untrained guide. This opens the door to many applications.

Our method can be used to study representational and functional design of neural networks in new ways to reinforce or reanalyze prior theory of neural network optimization. For example, we can understand distances between architectural components based on which target networks are easier to guide with a particular guide network. We also refine this notion to include a narrow channel through which guidance can occur, the representational similarity. This can serve as a kind of probe. Different representational similarity methods enforce alignment between different properties of the activations of networks.

Characterizing architectural priors and untrainable networks can be made more precise using our methodology. For example, guidance clearly distinguishes between at least three cases. One where the guide does not help. Another where the architecture of the guide helps, but not the knowledge. And one where both the architecture and the knowledge of the guide help. These are different phenomena whose root causes are not understood at present but that could elucidate the relationship between priors and architectures.

Guidance also proved to be a tool with which to discover the possibility of new initializations. At the moment, no known method exists to find better initializations for networks. In some cases as with the FCNs for vision, guidance can be disconnected after a nominal number of steps, but still goes on to regularize the target network. This strongly implies that an initialization regime for that target with the same regularization exists. This is all that guidance could do in that case. We now need tools to go backwards, given networks which are correctly initialized and networks which are not, discover what that initialization is. This is a much better place to be in. A systematic sweep of targets and guides to look for better initializations should be carried out.

For some networks, like fully connected ones, we only overcame the initial obstacle. Further research is needed to refine these into effective vision models, as they avoid immediate overfitting. This may be a matter of scale, optimizers, learning rates, etc. Our results suggest practical applications by significantly narrowing the performance gap between vanilla stacked RNNs and Transformers, albeit in small-scale experiments. Given that stacked RNNs are equivalent to single-layer RNNs, most directly to delayed RNNs (Turek et al., 2020), this implies that complex modifications to RNNs may be unnecessary for language modeling. In the future, we hope to find methods for making these networks competitive with their respective guide networks.

Looking into the long-term future, it may be that architecture can be turned into a prior that we add to the loss of a generic fully-connected network, or any convenient and capable-enough network. Architecture would lose its prominence and the computational substrate could be disconnected from the priors we need for learning and inference. Architecture could potentially even be turned into a parameter to be optimized, rather than a hyperparameter.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling mlps: A tale of inductive bias. *Advances in Neural Information Processing Systems*, 36, 2024.

Pouya Bashivan, Mark Tensen, and James J DiCarlo. Teacher guided architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5320–5329, 2019.

André Beauducel. Recovering wood and mccarthy's erp-prototypes by means of erp-specific procrustes-rotation. *Journal of Neuroscience Methods*, 295:20–36, 2018.

George N Bebis and George M Papadourakis. Object recognition using invariant object boundary representations and neural network models. *Pattern Recognition*, 25(1):25–44, 1992.

Yoshua Bengio, Yann LeCun, and Donnie Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models. *Advances in neural information processing systems*, 6, 1993.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7096–7116, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.576. URL https://aclanthology.org/2020.emnlp-main.576.

Rich Caruana, Steve Lawrence, and C Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 13, 2000.

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*, pp. 16296–16305, 2021.

Shuang Cong and Yang Zhou. A review of convolutional neural network architectures and their optimizations. *Artificial Intelligence Review*, 56(3):1905–1969, 2023.

Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

Colin Conwell, David Mayo, Andrei Barbu, Michael Buice, George Alvarez, and Boris Katz. Neural regression, representational similarity, model zoology & neural taskonomy at scale in rodent visual cortex. *Advances in Neural Information Processing Systems*, 34:5590–5607, 2021a.

Colin Conwell, Jacob S Prince, George A Alvarez, and Talia Konkle. What can 5.17 billion regression fits tell us about artificial models of the human visual system? In *SVRHM 2021 Workshop@ NeurIPS*, 2021b.

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.

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

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*, pp. 2022–07, 2022.

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*, pp. 248–255. Ieee, 2009.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Mingyu Fan, Nannan Gu, Hong Qiao, and Bo Zhang. Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint arXiv:1002.2050*, 2010.

Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. *Advances in Neural Information Processing Systems*, 33:13890–13902, 2020.

Ariel Goldstein, Zaid Zada, Eliav Buchnik, Mariano Schain, Amy Price, Bobbi Aubrey, Samuel A Nastase, Amir Feder, Dotan Emanuel, Alon Cohen, et al. Thinking ahead: spontaneous prediction in context as a keystone of language in humans and machines. *BioRxiv*, pp. 2020–12, 2020.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

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

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 478(2266):20210068, 2022.

Alex Graves. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *arXiv preprint arXiv:2402.09963*, 2024.

Barbara Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4):107–123, 2000.

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*, pp. 12430–12444. PMLR, 2023.

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*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016b.

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

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

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.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

Anwesha Khasnobish, Arindam Jati, Garima Singh, Saugat Bhattacharyya, Amit Konar, DN Tibarewala, Eunjin Kim, and Atulya K Nagar. Object-shape recognition from tactile images using a feed-forward neural network. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2012.

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.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2023.

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

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

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

Liying Ma and Khashayar Khorasani. Facial expression recognition using constructive feedforward neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34 (3):1588–1595, 2004.

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

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. *arXiv preprint arXiv:2209.15430*, 2022.

Il-Seok Oh and Ching Y Suen. A class-modular feedforward neural network for handwriting recognition. *pattern recognition*, 35(1):229–244, 2002.

Barak A Pearlmutter. Dynamic recurrent neural networks. 1990.

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

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.

Patrick Ramos, Raphael Alampay, and Patricia Abu. Knowledge distillation with relative representations for image representation learning. In *International Conference on Computer Recognition Systems*, pp. 133–143. Springer, 2023.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

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.

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.

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

Christian Schittenkopf, Gustavo Deco, and Wilfried Brauer. Two strategies to avoid overfitting in feedforward networks. *Neural networks*, 10(3):505–516, 1997.

Martin Schrimpf, Jonas Kubilius, Ha Hong, Najib J Majaj, Rishi Rajalingham, Elias B Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Franziska Geiger, et al. Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, pp. 407007, 2018.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Vighnesh Subramaniam, Colin Conwell, Christopher Wang, Gabriel Kreiman, Boris Katz, Ignacio Cases, and Andrei Barbu. Revealing vision-language integration in the brain with multimodal networks. *arXiv preprint arXiv:2406.14481*, 2024.

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

Javier Turek, Shailee Jain, Vy Vo, Mihai Capotă, Alexander Huth, and Theodore Willke. Approximating stacked and bidirectional recurrent architectures with the delayed recurrent neural network. In *International Conference on Machine Learning*, pp. 9648–9658. PMLR, 2020.

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

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 233–243, 2014.

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

A Appendix Overview

We present additional details of guidance, experiments and analysis, as well as additional results. In Appendix B, we present some limitations of guidance as a method. In Appendix C, we provide additional details for our guidance approach, with a full explanation of centered kernel alignment. In Appendix D, we provide additional details on our architectural designs and network training. In Appendix E, we introduce a new experiment where we feed noise to our guide network rather than real data and compute representational alignment, leading to similarly improved results. This further establishes a transfer of a prior rather than knowledge. In Appendix F, we show visualizations of the representational dissimilarity loss over training to give context of dynamics over training and show additional explanations for results with trained and randomly initialized guides. In Appendix G, we provide further explanation of error consistency as a measure of functional similarity between networks. In Appendix H, we provide test accuracy metrics over training as a complementary of network performance over training outside of cross-entropy loss.

B Methodology Limitations

Our guide network supervision through representational alignment has one primary limitation due to increased memory usage during training. Due to saving activations across several layers of the two networks, GPU memory usage increases dramatically. Moreoever, our methodology works better as batch size increases since this allows for better approximation of representational similarity, increasing memory usage even more. Furthermore, including more layers for supervision leads to improved results.

In this paper, we introduce simple techniques to handle memory constraints such as gradient accumulation and gradient checkpointing (Lin et al., 2017). In practice, more memory optimization techniques may become necessary to consider larger untrainable networks. Further work could consider using stronger representational alignment strategies to reduce the number of samples necessary to achieve a strong fit.

C Methods Overview

We give an overview of guidance in algorithm 1 and highlight crucial changes to base neural network training in either red or blue. We use blue to indicate the collection of network activations and red to indicate the layerwise mapping and representational alignment using a distance metric. This gives an overview of our layer mapping between the target and guide network. Crucially, we find that the simplest layer mapping where we evenly distribute guide network layers across target network layers for supervision obtains strong results.

C.1 Centered Kernel Alignment

To compare representations, we use a representation similarity metric, $\mathcal{M}$, which corresponds to centered kernel alignment (CKA) (Kornblith et al., 2019; Cortes et al., 2012; Cristianini et al., 2001) 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 $\boldsymbol{R} \in \mathbb{R}^{b \times d_1}$ and $\boldsymbol{R}' \in \mathbb{R}^{b \times d_2}$. We first compute the Gram matrices for each set of representations

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

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

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

We compute the HSIC on the Gram matrices.

Algorithm 1 Guidance: Guide Network Representational Alignment

Require: Target network; $\mathcal{N}^T$ with parameters θ^T; Guide network $\mathcal{N}^G$; Dataset $\mathcal{D} = \{(x_i, y_i)\}_{j=1}^{N}$; Representational Distance Metric $\bar{\mathcal{M}}$; Loss function $\mathcal{L}^T$

for $j = 1 \rightarrow N$ **do**
  # Base training with vanilla loss function
  outputs $\leftarrow \mathcal{N}^T(x_j)$
  loss $\leftarrow \mathcal{L}_T(\text{outputs}, y_j \mid \theta^T)$
  # collect layer activations
  $\{\boldsymbol{A}_{i^T}^T\}_{i^T=1}^{t} \leftarrow \text{activations}(\mathcal{N}^T(x_j))$
  $\{\boldsymbol{A}_{i^G}^G\}_{i^G=1}^{l} \leftarrow \text{activations}(\mathcal{N}^G(x_j))$
  # Get step size between the number of layers between the two networks for layer mapping.
  if $l > 1$ **then**
    $step \leftarrow (t-1)/(l-1)$
  else
    $step \leftarrow 1$
  end if
  # Map the layers and add up layer-wise representational distance
  total $\leftarrow 0$
  for $i = 1 \rightarrow l$ **do**
    index $\leftarrow \min(\text{round}(i \times step), l-1)$
    rep $\leftarrow \mathcal{M}(\boldsymbol{A}_{\text{index}}^T, \boldsymbol{A}_{\text{index}}^G)$
    total $\leftarrow$ total + rep
  end for
  loss $\leftarrow$ loss + total
end for

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

Finally, we define our linear CKA metric as:

$$\mathcal{M}(\boldsymbol{R}, \boldsymbol{R}') := \text{CKA}(\boldsymbol{K}, \boldsymbol{L}) = \frac{HSIC(\boldsymbol{K}, \boldsymbol{L})}{\sqrt{HSIC(\boldsymbol{K}, \boldsymbol{K}) * HSIC(\boldsymbol{L}, \boldsymbol{L})}} \tag{5}$$

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:

$$\bar{\mathcal{M}}(\boldsymbol{R}, \boldsymbol{R}') = 1 - \mathcal{M}(\boldsymbol{R}, \boldsymbol{R}') \tag{6}$$

Linear CKA ranges from 0 (identical representations) to 1 (very different representations). Because of this, we take the complement by subtracting the linear CKA from 1 to represent dissimilarity.

D Architecture and Training Details

D.1 Architectural Design Details

For all tasks, we describe our target untrainable architectural designs for each task separately as well as the guide networks that are employed to make the untrainable network trainable.

D.1.1 Image Classification

Target Networks

Deep FCN: We design a fully-connected network consisting of 50 blocks. Each block contains a feedforward linear layer, a batch normalization, and a ReLU nonlinear activation. The intermediate feedforward linear layers contain 2048 units. This network is untrainable due to vanishing gradients since the network is very deep and due to overfitting.

Shallow FCN: We design a network similar to Deep FCN but only containing 3 blocks where each feedforward linear layer contains 8192 units. This network is considered untrainalbe due to a saturation on the training performance.

Deep ConvNet: We design a deep convolutional network with the same architecture as ResNet-50 (convolutional layers followed by batch normalization) but remove the residual connections. This makes the network untrainable due to the vanishing gradient problem as observed in He et al. (2016a), causing saturation of the loss.

Guide Networks

ResNet-18/50: A deep convolutional network with 18/50 convolutional blocks and residual connections. We refer to He et al. (2016a).

We supervise the Deep FCN and Shallow FCN with ResNet-18 and supervise the Deep ConvNet with ResNet-50.

D.1.2 COPY-PASTE

Target Networks

RNN: We design a 4-layer RNN with a hidden dimension of 768 units, followed by a fully connected layer. In copy-paste, architectural and algorithmic limitations make RNNs an untrainable architecture fortask. Specifically, RNNs must memorize the input sequence which is difficult, particularly with a padding token. RNNs are generally considered to be unapplicable to the copy-paste task.

Guide Networks

Transformer: We consider a 4 layer transformer decoder architecture with a hidden dimension of 768 units across 12 transformer heads. The transformer is well-suited for copy-paste as the attention mechanism can act as a routing mechanism for the sequence. We train the transformer guide from scratch, as with language modeling and achieve 96.90% accuracy on the task.

D.1.3 PARITY

Target Networks

Transformer: Similar to Bhattamishra et al. (2020), we design a 1 layer transformer encoder network with a hidden dimension of 64 units across 4 attention heads. Transformers have lower accuracy on formal language tasks that require reasoning over a sequence in comparison to traditional sequence models (Hahn & Rofin, 2024). Due to the enormous gap in performance and saturation of performance, we categorize the transformer as untrainable.

Guide Networks

RNN: We include a 1 layer vanilla RNN with a hidden dimension of 64 units. Similar to Bhattamishra et al. (2020), we achieve 100% accuracy on the task.

D.1.4 LANGUAGE MODELING

Target Networks

RNN: We design a 4 layer vanilla RNN with a hidden dimension of 512 and with a ReLU activation function. We train this with on sequences with a context length of 50. This makes the network untrainable due to problems associated with exploding and vanishing gradients during backpropagation through time.

Guide Networks:

Transformer: We design a 4 layer transformer decoder network with 16 attention heads and a hidden dimension of 512. We train the transformer on WikiText with a context length of 50 and achieve a final test perplexity of 29.69.

Tasks	Experiment	Learning Rate
Image Classification	Deep FCN	1×10^{-4}
	Wide FCN	1×10^{-4}
	Deep ConvNet	1×10^{-3}
	ResNet-18 $\rightarrow$ Deep FCN	5×10^{-5}
	ResNet-18$_0$ $\rightarrow$ Deep FCN	1×10^{-4}
	ResNet-18 $\rightarrow$ Wide FCN	1×10^{-4}
	ResNet-18$_0$ $\rightarrow$ Wide FCN	1×10^{-4}
	ResNet-50 $\rightarrow$ Deep ConvNet	1×10^{-3}
	ResNet-50$_0$ $\rightarrow$ Deep ConvNet	1×10^{-3}
Copy-Paste	RNN	1×10^{-4}
	Transformer	1×10^{-4}
	Transformer $\rightarrow$ RNN	1×10^{-4}
	Transformer$_0$ $\rightarrow$ RNN	1×10^{-4}
Parity	Transformer	1×10^{-3}
	RNN	1×10^{-2}
	RNN $\rightarrow$ Transformer	1×10^{-3}
	RNN$_0$ $\rightarrow$ Transformer	1×10^{-3}
Language Modeling	RNN	1×10^{-4}
	Transformer	1×10^{-4}
	Transformer $\rightarrow$ RNN	1×10^{-4}
	Transformer$_0$ $\rightarrow$ RNN	1×10^{-4}

Table 4: **Learning rates for network training**. For all networks, we sweep over 5 learning rate values before choosing the learning rate with the lowest validation loss for training. Our training does not use any learning rate scheduling such as a warm-up scheduler although such techniques may improve results.

D.2 Training

In Table 4, we show the different learning rate settings we converged to in each experiment. For each experiment, we did a grid search over 5 different learning rate parameters to ensure optimal learning rate setting. We did careful tuning of all training of target networks to ensure maximum performance.

For all image classification tasks, we used the Adam optimizer (Kingma, 2014), in-line with prior work (He et al., 2016a). For all sequence modeling tasks, we use AdamW (Loshchilov, 2017), which has been useful in training sequence models like RNNs and Transformers (Radford et al., 2019).

The training experiments in this paper were completed across 4 H100s and 4 A100 GPUs for 3 weeks in total. GPU optimization techniques were taken such as gradient accumulation and gradient checkpointing and some language modeling experiments used mixed-precision training.

E Representational Regularization

We also aim to understand the role of the guide network as in guidance. In all experiments, we use trained and untrained guide networks and see consistent improvements for training the target network. The success of untrained networks implies that our training method is not performing distillation but instead truly transferring a prior from the guide network to the target network. To more strictly test this theory, we include an experiment where we feed noise to the guide network instead of the same batch of data fed to the target network as implied by eq. (1).

We apply this experiment to the Deep FCN with an untrained ResNet-18 as the target network. At each training step, we pass a noisy batch which is sampled from a random Gaussian with mean of 0 and standard deviation of 1. We train for 100 epochs and report the learning curve results in fig. 5.

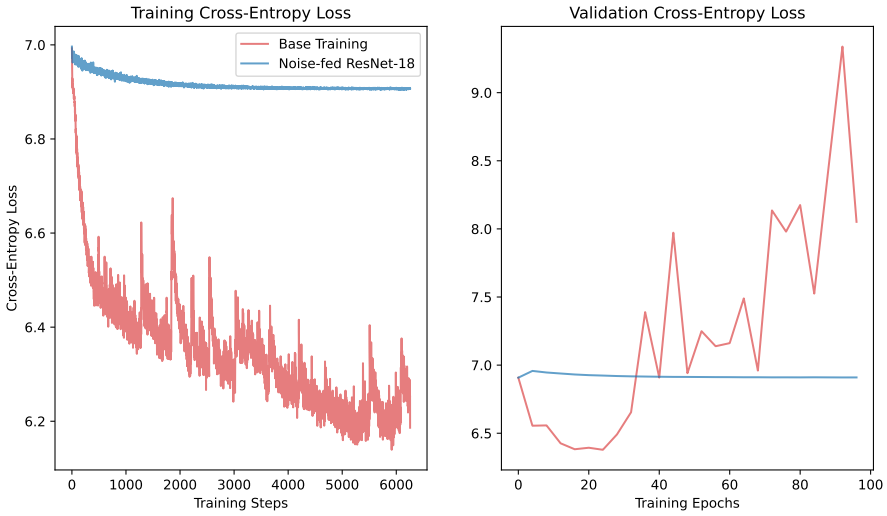

Figure 5: **Feeding noise prevents overfitting**. We introduce an additional experiment where we feed noise to our guide network rather than the same batch during each training step with guide network guidance. We sample noise from a Gaussian distribution with a mean of 0 and a standard deviation of 1. We find that despite having no information about the images in the batch, the guide network still provides an inductive bias to prevent overfitting. While this noise increase the training loss, this shows a true transfer of an inductive bias that is not driven by pure distillation of similar features.

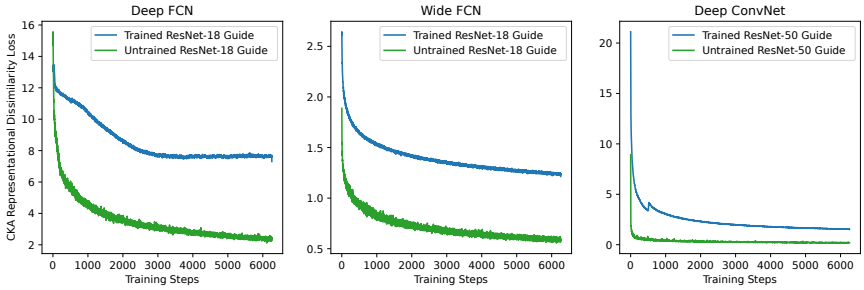

Figure 6: **CKA representational similarity loss for image classification**. We visualize the total CKA dissimilarity loss across all layers across training for image classification. The CKA dissimilarity loss represents the representational alignment between our guide network and target network. We can observe that for Deep FCN and Wide FCN, the target network aligns with a randomly initialized network more quickly. This corresponds with results where randomly initialized guide networks had superior performance to trained guide networks.

This result confirms our intuition about the role of guide network: as a guide on model priors rather than a pure distillation of information. While the overall cross-entropy loss magnitudes are higher and the overall accuracy is lower when passing noise to the guide network, our results are significantly better than applying vanilla training approaches to the Deep FCN.

F Representational Similarity Loss

We can view the representational alignment between the guide and target networks during training. This allows us to better understand how this representational alignment influences network performance. We show image classification results in fig. 6 and sequence modeling results in fig. 7.

We notice that across most tasks, reducing representational dissimilarity is easier with activations from randomly initialized networks rather than trained networks. This provides additional evidence of representational alignment for inductive bias transfer. We notice that for certain cases, such as

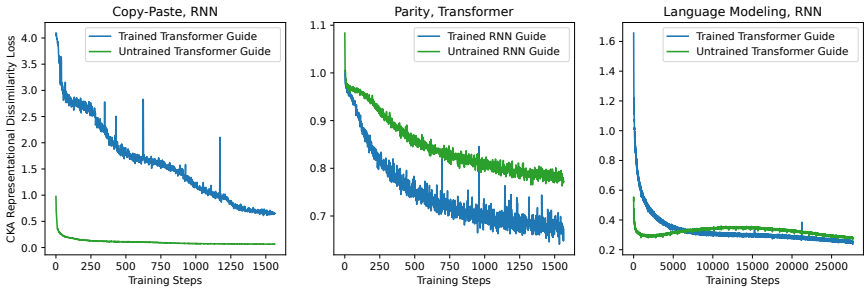

Figure 7: **CKA representational similarity loss for sequence modeling tasks.** We visualize the total CKA dissimilarity loss across all layers across training for all three sequence modeling tasks. The CKA dissimilarity loss represents the representational alignment between our guide network and target network. We can observe that for the copy-paste task and language modeling task, the target network aligns with a randomly initialized network more quickly. This could be because of special properties of RNNs.

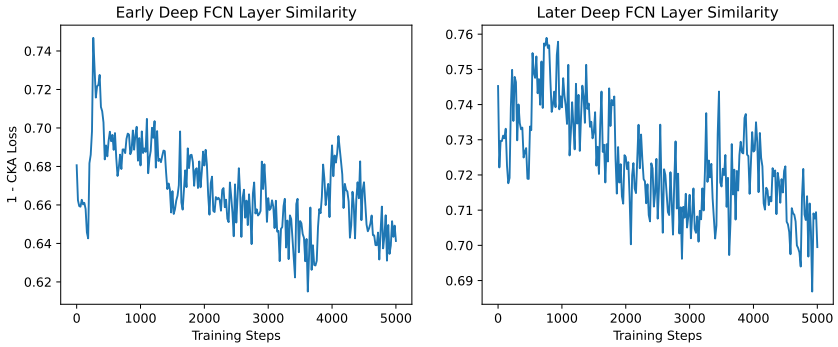

Figure 8: **CKA dissimilarity decreases more in earlier network layers than later layers.** When we separate the CKA dissimilarity across layers of the target network, we find that earlier layers optimize more and earlier. We take two layers from a Deep FCN during guidance with a randomly initialized ResNet-18. The early layer comes from the 15th FCN block. The later layer comes from the 43rd FCN block. We see that both layers are eventually optimized but the later layer receives less supervision and has a higher CKA at the end of training.

Parity, the randomly initialized guide network has higher representational dissimilarity loss than the trained guide network. This is matched with the Parity result in table 3 and fig. 2.

However, we can also observe more inconsistent results with the Deep ConvNet where the untrained guide network has lower representational dissimilarity loss than the trained guide network, even at the end of training. One possible explanation that the inductive bias was more similar for Deep ConvNet and ResNet-50. This means that trained features are more important for better Deep ConvNet results and representational alignment with a trained network is important. This result has interesting implications for understanding the role of residual connections. Since untrained ResNet-50 is easier to align with than a trained ResNet-50, this demonstrates that residual connections influence representation spaces during training. The untrained residual connections have little influence on the inductive biases of the network or the overall representation space. This demonstrates the strength of our method as a way to interpret neural network design choices and how they influence representation and functional aspects of a network.

These results are also potentially indicative of architectural properties of RNNs and FCNs which match randomly initialized networks more quickly. For instance, one potential explanation is that RNNs have more degrees of freedom (Bhattamishra et al., 2020) and therefore, only need inductive guidance rather than trained features. Transformers may require learned features indicating that the bottleneck for transformers on the parity is not algorithmic but feature-based. Much of the future work can use these results to design better networks with more informed designs.

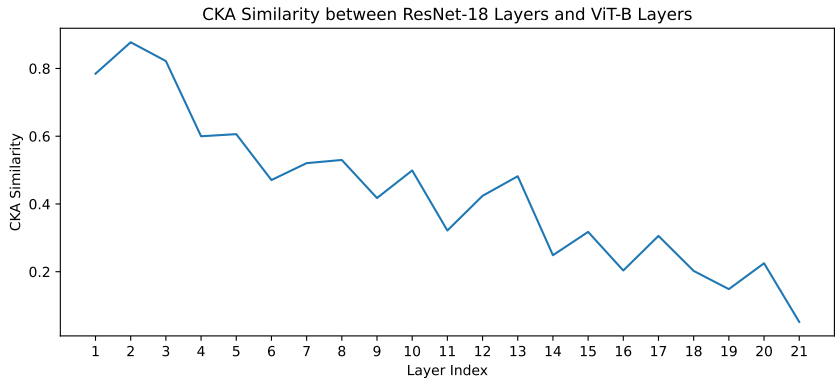

Figure 9: **Earlier layers of ResNet-18 and ViT-B are more similar.** We analyze the representational similarity between activations from layers ResNet-18 and ViT-B-16 via CKA. We find that earlier layers are more similar while later layers have divergent representations. We see that this manifests in distinct error consistency patterns when ResNet-18 and ViT-B are used as guide networks.

F.1 Layerwise Analysis

We provide a deeper analysis of patterns of the representational dissimilarity across different layers during guidance in fig. 8. We find that earlier layers generally have higher CKA similarities with their corresponding layer from the guide network and later layers have lower CKA similarities. Furthermore, these later layers optimize later in the training process.

G Error Consistency

We measure *error consistency* (κ) (Geirhos et al., 2020) between the guided target networks which indicates the error overlap between two networks based on the accuracy of the networks, i.e. do the two networks make similar class predictions? The measure first calculates the expected error overlap. Suppose a_1 is the accuracy of the first guided network and a_2 is the accuracy of the second. The expected error overlap is given by $c_{\text{exp}} = a_1 * a_2 + (1 - a_1) * (1 - a_2)$. Next, we measure the observed error overlap across each sample in the validation set as $c_{\text{obs}} =$ # of samples where both models agree / total trials. Finally, we can write κ as:

$$\kappa = \frac{c_{\text{obs}} - c_{\text{exp}}}{1 - c_{\text{exp}}} \tag{7}$$

κ ranges from -1 to 1, where 1 is perfect agreement, -1 is perfect disagreement and 0 is change agreement. When $\kappa > 0$, this implies that models make consistent error patterns, $\kappa < 0$ implies that models make inverse error patterns, and $\kappa \approx 0$ implies independent error patterns.

G.1 Guide Network Representation Comparison

We contextualize the findings in error consistency by comparing the representations of the guide networks, in this case ResNet-18 and ViT-B-16. We apply a layer mapping between layers of ResNet-18 and ViT-B-16 and compute the representational similarity over 1000 input images. Results are shown in fig. 9.

Our findings are useful for error consistency. If models are inheriting inductive biases from their guide network, then the models would have similar methods to process low-level image features are indicated by a stronger CKA in earlier layers between the ResNet-18 and ViT-B. This means that errors will be consistent for low level features but inconsistent for high level features collected in later layers.

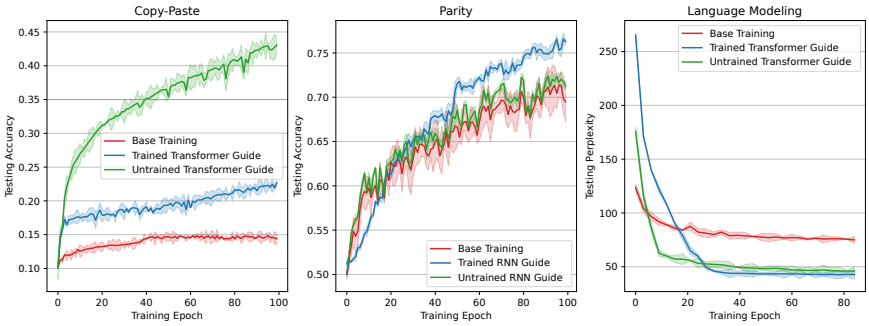

Figure 10: **Testing accuracy improves across guidance for sequence modeling**. We visualize the testing accuracy for sequence modeling as an example to demonstrate that guidance improves accuracy across training and this improvement is significantly better across training. This allows for another intepretation of the method outside of cross-entropy loss.

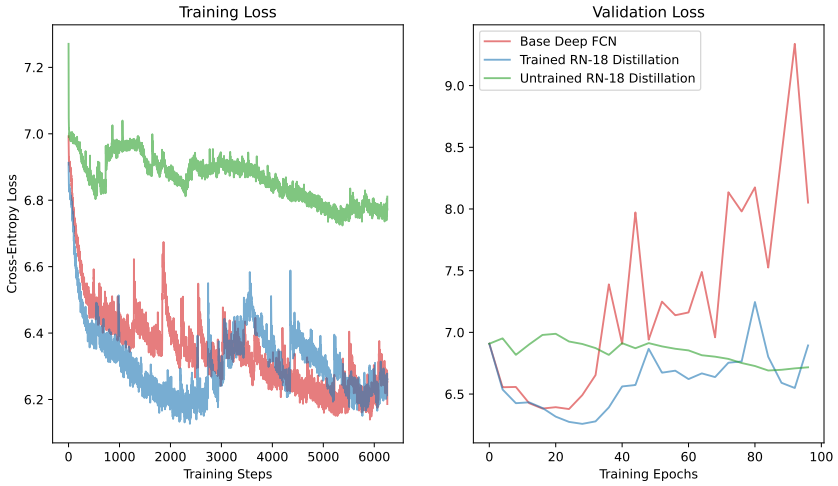

Figure 11: **Distillation does not prevent overfitting**. We compare basic distillation (Hinton, 2015) to see if we can prevent overfitting. We use Deep FCN as our student network. We find that distillation with a trained ResNet-18 teacher network leads to a small improvement in performance but still has some patterns of overfitting. Distillation with an untrained ResNet-18 teacher network hurts performance.

H Test Accuracy across Training

We plot accuracies over training as a complement to cross-entropy loss in fig. 10 for the sequence modeling experiments. We can use these experiments to de-couple our results from properties of cross-entropy loss that may lead to misleading improvements across training. We find that accuracies improve consitently across training, supporting the loss curve interpretation that guided training improves results.

I Basic Distillation Comparison

To show the effectiveness of guidance, we compare it with distillation from Hinton (2015). Distillation involves transferring knowledge from a performant teacher network to a less performant student network via maximizing the alignment of the output logits. This encourages the student to have similar predictions as the teacher network. This occurs via the following loss function. Assume $\boldsymbol{Q}$ is the logits extracted from the target (student) network and the $\boldsymbol{P}$ is the logits extracted from the guide (teacher) network.

Experiment	**ImageNet Validation Accuracy** (↑)
Deep FCN	1.65 ± 0.51
Trained ResNet-18 Teacher	3.45 ± 0.92
Untrained ResNet-18	1.41 ± 0.66

Table 5: **Distillation performs worse than guidance**. We include a distillation baseline over the Deep FCN with ResNet-18 as the teacher network. We find that the accuracy improvement is small with a trained teacher. And, accuracy decreases with an untrained teacher.

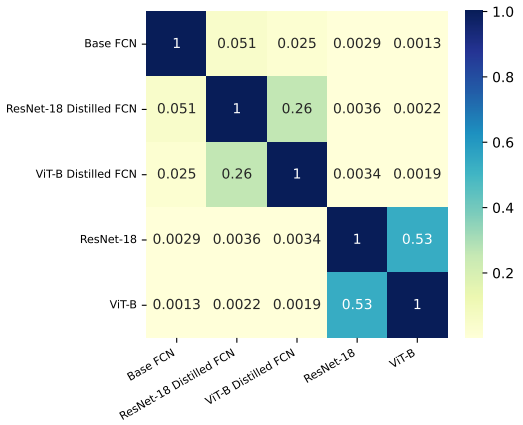

Figure 12: **Distillation lowers error consistency**. In general, we find that distillation results in error consistency patterns that are less consistent than what is reported with guidance.

$$\mathcal{L}_{\text{distill}} = \alpha * T^2 * \text{KL}(\sigma(\boldsymbol{Q}/T)||\sigma(\boldsymbol{P}/T)) + (1-\alpha) * \mathcal{L}_{\text{CE}}(\boldsymbol{Q}, y) \tag{8}$$

where y is the ground truth labels, T is the temperature to soften the logits, and α is the weighting factor between the distillation loss and cross-entropy loss. In this case, KL refers to the Kullback-Liebler divergence and σ corresponds with the softmax function. In practice, we set α to 0.5 and T to 2. We continue to track the full cross-entropy loss across training as well.

We show results in fig. 11 and table 5. Distillation with an trained network can improve performance but much less than guidance. Distillation with a untrained network reduces performance on average, although not by a significant amount.

I.1 Error Consistency

We show error consistency performance over distilled networks rather than guided networks in fig. 12.

J Guided Network Analysis and Interpretation

The results from guidance open many questions in order to explain why untrained guide networks can be better at improving target network performance. We provide an intuitive explanation as well as some geometric analysis of guided networks to see if there is a stronger interpretation.

J.1 Interpreting Guidance

The current results seem to indicate that guidance is finding a distinction between learned priors and architectural priors when preventing undesirable features of training target networks. For example, it seems architectural aspects of CNNs are useful for preventing overfitting in the Deep FCN. Similarly, memory incorporation improves in an RNN when guiding with a randomly initialized transformer as seen with improved copy-paste performance.

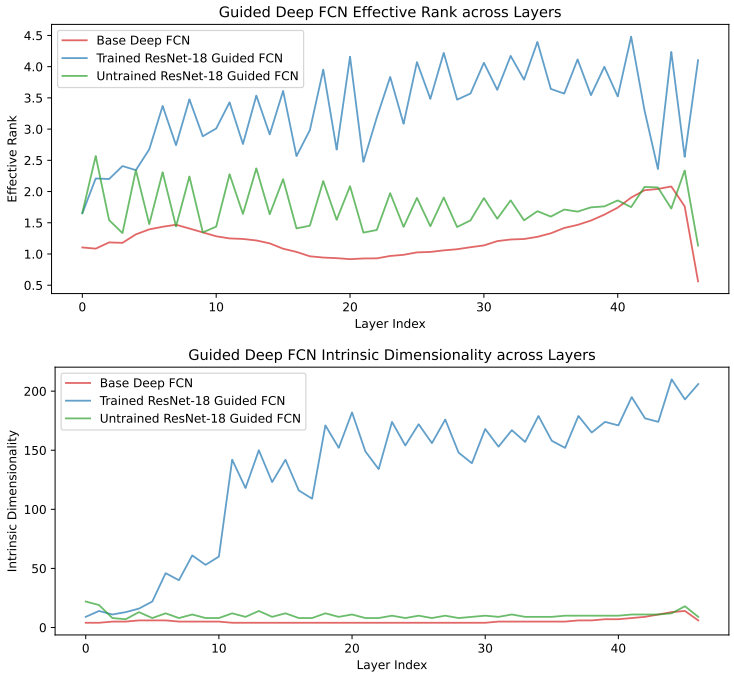

Figure 13: **Explaining guidance using effective rank and intrinsic dimensionality.** We compare the geometric properties of activations associated with target networks guided by untrained and trained guides as well the base target network. We observe that effective rank and intrinsic dimensionality seem unrelated to performance obtained by guidance using CKA. Guidance with trained networks leads to higher effective rank and intrinsic dimensionality than having no guidance. Yet, this property doesn't connect with findings that untrained guide networks have better results.

These architectural priors don't disappear when the guide network is trained but are more difficult to replicate due to the target network having to replicate both learned and architectural priors, assuming that the learned prior isn't as useful for overcoming the gap in performance. It's likely that some aspect of the ResNet is useful for preventing overfitting. This could be sparsity, the distribution of the eigenvalues, etc. We plan to dedicate further work to understanding these distinctions.

J.2 Geometric Analysis

We aim to understand how guidance with a randomly initialized guide network differs from a trained guide network. To do so, we compare the representation space of a target network guided by a trained guide network and a randomly initialized guide network. Our comparisons include (1) effective rank and (2) intrinsic dimensionality.

Effective rank measures analyzes the capacity of neural network activations by measuring the spread of singular values. When the singular values are evenly distributed for a given set of activations, this indicates that fewer directions of the representation space are utilized. Lower effective rank indicates stronger inductive biases. Effective rank is written as

$$\text{Effective Rank} = \exp\left(-\sum_i p_i \log(p_i)\right) \tag{9}$$

where $p_i = \frac{\sigma_i}{\sum_j \sigma_j}$ are the normalized activations associated with activaitons from a specific network layer.

Intrinsic dimensionality refers to the minimum number of dimensions required to capture the structure or variability in the input data. It represents the true complexity of the data manifold, ignoring noise or redundant dimensions. Following Fan et al. (2010), for a given threshold β, the intrinsic di-

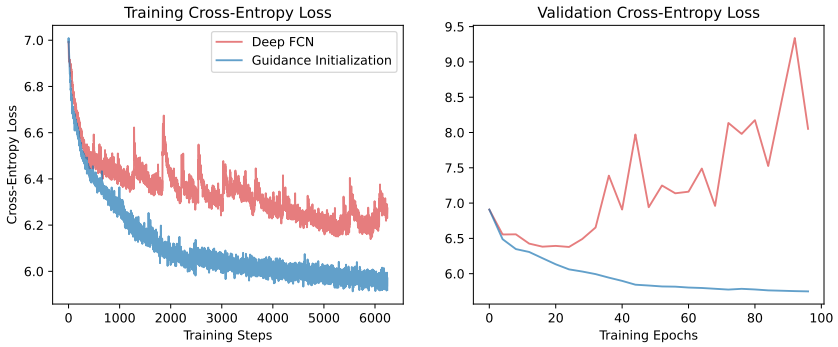

Figure 14: **Guidance finds initialization schemes in FCNs**. We introduce a stricter experiment for finding initialization in FCNs where we apply guidance separate from the task loss on noise before applying ImageNet training. We find that this prevents the overfitting regime and leads to significant improvements in network performance.

mension is the $d \in \mathbb{N}$ such that the ratio of explained variance for d dimensions of a N dimensional PCA is above β:

$$\frac{\sum_{i=1}^{d} var(y_i)}{\sum_{j=1}^{N} var(y_j)} > \beta \tag{10}$$

We measure the effective rank and intrinsic dimensionality for the activations of all linear layers in our Deep FCN guided by a untrained and trained ResNet-18 guide over 1000 images as well the trained Deep FCN. We show results in fig. 13. We find that guiding with an untrained guide network has little effect on effective rank and intrinsic dimensionality. Meanwhile, guiding with a trained guide leads to increases in effective rank and intrinsic dimensionality. Due to the discrepancy in results where guidance with an untrained network leads to the best result, this implies that guidance can't be explained with effective rank and intrinsic dimensionality.

K FCN Initialization

In fig. 4, we demonstrated that guidance does not need to be applied at every training step but can instead stop early. Continuing optimization of the target network on just the task afterwards prevented overfitting and lead to improved results.

We can introduce a stricter variant of this experiment to further demonstrate a new initialization scheme for FCNs. Our new experiment involves splitting guidance into two stages. The first stage involves only optimizing the target network to form representations that are similar to the guide network using CKA – so we only optimize representational similarity between the target and guide network. Crucially, we optimize using random noise, sampled from a Gaussian with mean 0 and standard deviation 1. This is important for network initialization, which should not be data dependent. We optimize the representational similarity for 150 steps.

Afterwards, we optimize the target network independently on the task. We apply this experiment to the Deep FCN, trained on ImageNet. We optimize the representational similarity with an untrained ResNet-18 using noise and apply ImageNet optimization afterwards. We find that this prevents overfitting and leads to an improved performance of the Deep FCN, as shown in fig. 14.

We believe this establishes the potential for new initialization schemes with FCNs.

L GUIDANCE WITH RSA AND RIDGE REGRESSION

L.1 REPRESENTATIONAL SIMILARITY ANALYSIS

We use the RSA formulation as described in Kriegeskorte et al. (2008). Specifically, RSA constructs representational dissimilarity matrices (RDMs) for two sets of representations and compares them using an outer similarity function.

Given two sets of representations, $\boldsymbol{R} \in \mathbb{R}^{b \times d_1}$ and $\boldsymbol{R}' \in \mathbb{R}^{b \times d_2}$, we first calculate RDMs for each set of representations using a distance function d. Formally, we define $\boldsymbol{D} \in \mathbb{R}^{b \times b}$ as

$$\boldsymbol{D}_{i,j} := s(\boldsymbol{R}_i, \boldsymbol{R}_j) \tag{11}$$

Each row $\boldsymbol{D}_i$ corresponds to the distance between the representations of input i and the representations of all inputs including itself. This is done per-batch, meaning that RSA is sensitive to batch size.

Given two RDMs $\boldsymbol{D}$ and $\boldsymbol{D}'$ constructed from sets of representations $\boldsymbol{R}$ and $\boldsymbol{R}'$ respectively, we vectorize the RDM matrices using a function v (since the RDMs are symmetric, we only need to compare the lower triangles), and compute the similarity between the two vectorized RDMs using a similarity function s.

$$\mathcal{M}(\boldsymbol{R}, \boldsymbol{R}') = s(v(\boldsymbol{D}), v(\boldsymbol{D}')) \tag{12}$$

As with CKA, we use the complement of the similarity to construct $\bar{\mathcal{M}}$. In practice, we define d to be the cosine distance between every pair of inputs and s to be the pearson correlation between the RDMs as done in previous work (Conwell et al., 2021a).

We apply guidance with RSA to Deep FCN as our target network and ResNet-18 as our guide network. Similar to our CKA results, we train for 100 epochs with a batch size of 256, as RSA is sensitive to the number of samples when comparing sets of representations.

L.2 RIDGE REGRESSION

We used similar ridge regression formulation as (Conwell et al., 2021a; Subramaniam et al., 2024) without cross-validation.

Given two sets of representations, $\boldsymbol{R} \in \mathbb{R}^{b \times d_1}$ and $\boldsymbol{R}' \in \mathbb{R}^{b \times d_2}$, we first apply a sparse random projection on the representations. Since the dimensionality of the representations is prohibitively large, the projection makes the ridge regression feasible to compute. We refer to the resulting representations as $\boldsymbol{P}$ and $\boldsymbol{P}'$ which correspond to the projected representations $\boldsymbol{R}$ and $\boldsymbol{R}'$ respectively. $\boldsymbol{P}$ and $\boldsymbol{P}$ have dimension d, where d is fixed using the Johnson-Lindenstrauss lemma (Johnson et al., 1986).

Afterwards, we mean-center the representations and apply ridge regressions using the original least-squared solution as follows. Our goal is to predict the representations $\boldsymbol{P}'$ using regressors over $\boldsymbol{P}$. We first split our representations into a training set i.e. $\boldsymbol{P}_{\text{train}}, \boldsymbol{P}'_{\text{train}}$ and testing set $\boldsymbol{P}_{\text{test}}, \boldsymbol{P}'_{\text{test}}$ where the training set contain half the representations and the testing set contains the other half. We first a set of regressors $\hat{\beta}$ as follows:

$$\hat{\beta} = ((\boldsymbol{P}_{\text{train}})^T \boldsymbol{P}_{\text{train}} + \lambda \boldsymbol{I}_d)^{-1} (\boldsymbol{P}_{\text{train}})^T \boldsymbol{P}'_{\text{train}} \tag{13}$$

where λ is the ridge penalty, which is a hyperparameter. The coefficients $\hat{\beta}$ are then used to predict the held out data where:

$$\hat{\boldsymbol{P}'_{\text{test}}} = \boldsymbol{P}_{\text{test}} \beta \tag{14}$$

We measure the cosine similarity between the predicted representations $\hat{\boldsymbol{P}'_{\text{test}}}$ and actual representations $\boldsymbol{P}_{\text{test}}$.

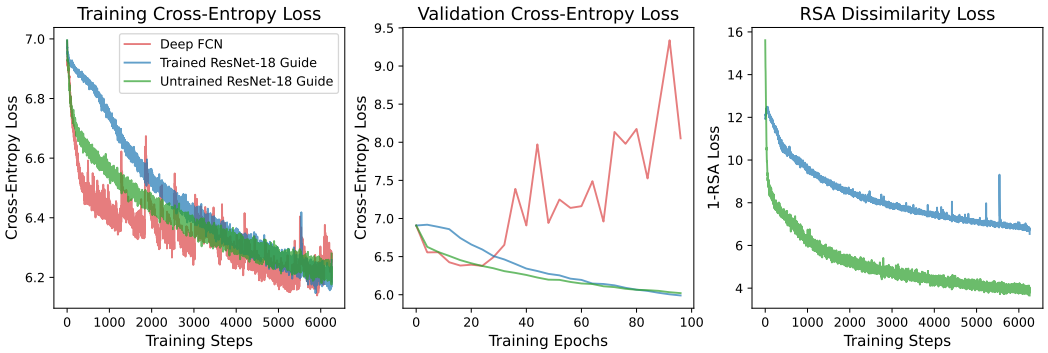

Figure 15: **Guidance with RSA as the representational similarity metric maintains similar performance to CKA**. We include a further experiment where we change the metric for representational alignment from CKA to RSA during guidance training. We apply this to the Deep FCN with ResNet-18 as a guide network. We see that, like CKA, RSA alignment also allows for transferring the prior from ResNet-18. However, unlike CKA, the untrained guide network only does marginally better than the trained network, potentially indicating the RSA is better at transferring trained features.

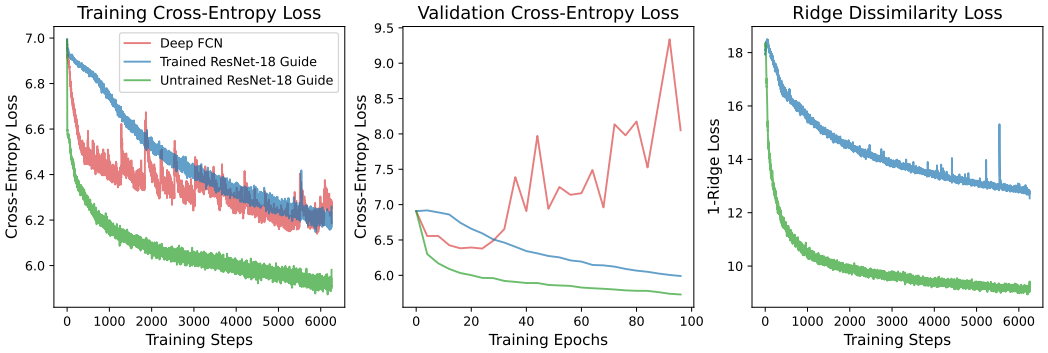

Figure 16: **Guidance with ridge regression as the representational similarity metric improves performance over CKA**. We change the metric for representational alignment from CKA to ridge regression during guidance training. We apply this to the Deep FCN with ResNet-18 as a guide network. We see that, like CKA, ridge regression alignment also allows for transferring the prior from ResNet-18. We find that this improves over CKA significantly

$$\mathcal{M}(\boldsymbol{R}, \boldsymbol{R}') = \text{cosine}(\hat{\boldsymbol{P}}'_{\text{test}}, \boldsymbol{P}'_{\text{test}}) \tag{15}$$

We apply guidance with RSA to Deep FCN as our target network and ResNet-18 as our guide network. Similar to our CKA results, we train for 100 epochs with a batch size of 256, as ridge regression is sensitive to the number of samples when comparing sets of representations. We manually tune the λ hyperparameter, finding that $\lambda = 10.0$ is optimal for the trained guide network and $\lambda = 100.0$ is optimal for the untrained guide network.

L.3 Results

RSA: We see results over the training, validation, and representational dissimilarity loss in fig. 15. The Deep FCN guided by a trained ResNet-18 achieves an accuracy of *11.02%* and the Deep FCN guided by a randomly initialized ResNet-18 achieves an accuracy of *11.74%*.

We can first observe that guided training improves over base training as noted in fig. 2 and table 2. This demonstrates the generality of our approach to other metrics. As long as a representational similarity metric is differentiable, we can optimize the metric for alignment between two networks as a method to transfer the prior of one network to another.

Experiment	**CIFAR-10 Test Accuracy** (↑)
Deep FCN	60.58
All layer Guidance	70.15
Last layer Guidance	67.18
Last two layers Guidance	68.03
Last five layers Guidance	67.50
Last ten layers	69.31
First ten layers	79.22
First five layers	**79.58**
First two layers Guidance	73.34
First layer Guidance	65.11
Multiple Guide Layers	68.14

Table 6: **Guiding earlier layers of deep networks leads to better results**. We apply an ablation experiment to identify which layers lead to stronger improvement when guided. We use a Deep FCN, guided by a randomly initialized ResNet-18 on CIFAR-10. We find that guiding earlier layers leads to strong improvement, even over guiding all layers. Guiding any layer leads to an improvement of performance.

We can also observe some minute differences between the results with CKA. Most notably, the trained guide network has similar performance to the untrained guide network. This is likely because less information about trained features are present in the RSA metric. RSA measures relative distance between input instances and imposes a constraint of placing these into relative distances. It could be possible that fewer degrees of freedom are useful for aligning target network with trained guides.

Ridge: We see results over the training, validation and representational dissimilarity loss in fig. 16. The Deep FCN guided by a trained ResNet-18 achieves an accuracy of *9.46%* and the Deep FCN guided by a randomly initialized ResNet-18 achieves an accuracy of *15.69%*. Similar to RSA and CKA, we can see the guided training improves over base training.

Similar to CKA, we observe that randomly initialized guide networks outperform trained guide networks. Furthermore, performance with ridge regression is better than with CKA. This is finding is intuitive. Ridge regression generally has more degrees of freedom than other similarity metrics because of fewer invariances imposed on the metric. This means that the solution search space is larger, leading to better results. We believe this provides a promising path forward for making target networks have better performance.

Furthermore, ridge regression has a desirable property because it can be probed. Since we are mapping representations from our target network to our guide network, we can probe predicted representations to measure similarity. We can use probing analyses on predicted representations to see what information the target has inherited from the guide network. This opens up many avenues for studying guidance in the future.

M Ablation Experiments

We analyze the current design of our layer mapping for guidance by experimenting with the number of layers used in guidance and whether more complex mappings exist like mapping several layers of the guide network to a single target network.

We run layer-wise ablation experiments the Deep FCN guided by an untrained ResNet-18 over CIFAR-10. Similarly, we experiment with RNNs guided by untrained transformers over the copy-paste task.

In table 6, we first show the effect of guiding over a subset of layers in the Deep FCN, evaluated over CIFAR-10. We find that earlier layers are much more impactful for a deep network. Intuitively, this could be due to guidance providing aiding with the credit assignment problem in deep networks: gradients don't propogate properly to earlier layers. However, table 7 shows that later layers in the RNN are more useful to apply guidance to when improving copy-paste performance.

Experiment	**Copy-Paste Accuracy** (↑)
RNN	14.35
All layer Guidance	**42.56**
Last layer Guidance	38.19
Last three layers Guidance	42.33
Last two layers Guidance	41.55
First layer Guidance	27.59
First two layers Guidance	28.15
First three layers Guidance	33.93
One guide layer	36.11

Table 7: **Guidance of later layers improves RNN performance**. We apply an ablation experiment over RNNs trained for copy-paste to see whether guiding certain layers lead to improved performance. We find that guiding later layers leads to stronger performance overall. Furthermore, RNN layers are guided by several guide network layers in the transformer such as the linear layer and layer-normalization in the transformer decoder. Including both of these leads to better results.

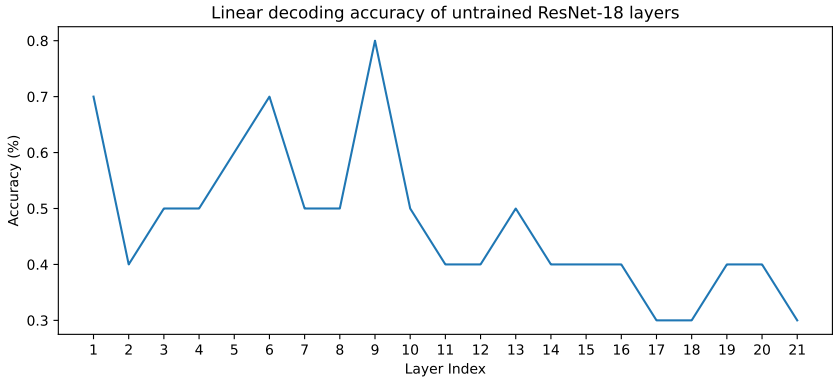

Figure 17: **ImageNet classes are barely decodable from a randomly initialized ResNet-18.** In order to assess the performance of our randomly initialized networks, we design a linear decoder to decode ImageNet classes from all layers of the network. Chance accuracy is 0.1% (1/1000) on the graph above. We find that we can decode ImageNet classes with an accuracy above chance the accuracy is still very low.

In general, we find that guiding any layer leads to improvements in results generally, showing the general applicability.

Furthermore, in table 6 and table 7, we consider new methods to map guide network layers to target network layers. When guiding a Deep FCN with ResNet-18, we only apply a 1-1 layer mapping for supervision i.e. each Deep FCN layer is guided by only one ResNet-18 layer. From table 6, introducing multiple sources of supervision from guide network layers by allowing a one-to-many mapping decreases performance. However, with the RNN, we guide with representations from the linear layer and layer normalization in the transformer decoder. One could consider that linear layers are redundant with layer normalization for guidance, so we remove its representations as a potential supervisory target. We find that this hurts performance, showing that RNNs benefit from multiple levels of supervision from its guide.

Understanding the dynamics of guidance supervision is interesting and could allow for understanding training dynamics of neural networks or allow us to form cross-architectural relationships.

N Linear Decoding

We assess whether object detection is decodable from internal representations of a randomly initialized ResNet-18. A potential explanation for improvements in the target network with an untrained

guide is the ability to linearly decode the ImageNet classes with significant accuracy at certain layers of the untrained guide network.

We train a linear decoder with 4000 ImageNet images from the train set and test on 1000 images from the validation set for each layer. We show results in fig. 17. We find that the linear decoder never achieves any accuracy greater than 0.8% for any of the layers. Furthermore, later layers contain little information that is useful to linearly decode ImageNet classes. This means that linearly decodable information isn't present in the guide network and this aspect isn't driving improvements in target networks.