TRANSFERRING INDUCTIVE BIASES THROUGH KNOWLEDGE DISTILLATION

Samira Abnar[1], **Mostafa Dehghani**[2], **Willem Zuidema**[1]
[1]University of Amsterdam, [2]Google Brain
s.abnar@uva.nl, dehghani@google.com, w.h.zuidema@uva.nl

ABSTRACT

Having the right inductive biases can be crucial in many tasks or scenarios where data or computing resources are a limiting factor, or where training data is not perfectly representative of the conditions at test time. However, defining, designing and efficiently adapting inductive biases is not necessarily straightforward. In this paper, we explore the power of knowledge distillation for transferring the effect of inductive biases from one model to another. We consider families of models with different inductive biases, LSTMs vs. Transformers and CNNs vs. MLPs, in the context of tasks and scenarios where having the right inductive biases is critical. We study the effect of inductive biases on the solutions the models converge to and investigate how and to what extent the effect of inductive biases is transferred through knowledge distillation, in terms of not only performance but also different aspects of converged solutions.

1 INTRODUCTION

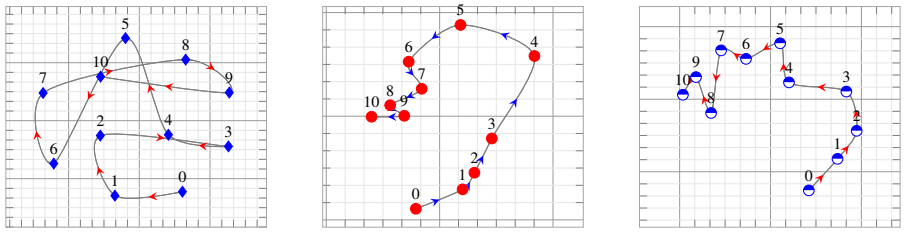
Inductive biases are the characteristics of learning algorithms that influence their generalization behaviour, independent of data. They are one of the main driving forces to push learning algorithms toward particular solutions (Mitchell, 1980). Having the right inductive biases is especially important for obtaining high performance when data or compute is a limiting factor, or when training data is not perfectly representative of the conditions at test time. Moreover, in the absence of strong inductive biases, a model can be equally attracted to several local minima on the loss surface; and the converged solution can be arbitrarily affected by random variations like the initial state or the order of training examples (Sutskever et al., 2013; McCoy et al., 2020; Dodge et al., 2020).

There are different ways to inject inductive biases into learning algorithms, for instance through architectural choices, the objective function, the curriculum, or the optimization regime. In this paper, we exploit the power of *Knowledge Distillation* (KD) to transfer the effect of inductive biases between neural networks. KD refers to the process of transferring knowledge from a teacher model to a student model, where the logits from the teacher are used to train the student. KD is best known as an effective method for model compression (Buciluǎ et al., 2006; Hinton et al., 2015; Sanh et al., 2019) which allows to take advantage of a huge number of parameters during training, while having an efficient smaller model during inference.

The advantage of KD goes beyond model compression and it can be used to combine strengths of different learning algorithms (Kuncoro et al., 2019; 2020). Different algorithms vary in terms of the speed at training/inference or the inductive biases to learn particular patterns. This makes them better at solving certain problems and worse at others, i.e., there is no "one size fits all" learning algorithm. Hence, it is important to explore the potential of KD for finding better trade-offs.

The question that we ask in this paper is: "*In KD, are the preferences of the teacher that are rooted in its inductive biases, also reflected in its dark knowledge*[1]*, and can they thus be transferred to the student?*". We are interested in cases where the student model can realize functions that are realizable by the teacher, i.e., the student model is efficient with respect to the teacher model (Cohen et al., 2016), while the teacher has a *preference inductive bias* so that the desired solutions are easily *learnable* for the teacher (Seung et al., 1991).

[1]Dark knowledge refers to the information encoded in the output logits of a neural network (Hinton et al., 2015).



(a) MLP (b) CNN (c) CNN → MLP

Figure 1: Training paths of different models on the Translated MNIST task. Different points represent the state of the model at different epochs, from the initial state to the convergence. The visualization is based on a 2D projection of the representational similarity of the activations from the penultimate layer for the examples from the validation set, i.e. Translated MNIST (more details in Appendix B).

We consider two scenarios where the teacher and the student are neural networks with heterogeneous architectures, hence have different inductive biases. We train the models, both independently and using KD, on tasks for which having the right inductive biases is crucial. In the first test case, we study RNNs vs. Transformers (Vaswani et al., 2017), on the subject-verb agreement prediction task (Linzen et al., 2016). In this task, we use LSTMs (Hochreiter & Schmidhuber, 1997) as the most widely used RNN variant. LSTMs are shown to perform better than vanilla Transformers in this task and their superior performance is attributed to their so-called "recurrent" inductive bias (Tran et al., 2018). First, we identify the sources of the recurrent inductive bias of LSTMs: *sequentiality*, *memory bottleneck*, and *recursion*, and design experiments to show the benefits of each. Then, we show that through distilling knowledge of LSTMs to Transformers, the solutions that the Transformer models learn become more similar to the solution learned by LSTMs.

In the second test case, we study CNNs vs. MLPs, in the context of the MNIST-C (Corrupted MNIST) benchmark (Mu & Gilmer, 2019), which is designed to measure out-of-distribution robustness of models. We train our models on MNIST and evaluate it on the Translated/Scaled MNIST. The particular form of parameter sharing in CNNs combined with the pooling mechanism makes them equivariant to these kinds of transformations (Goodfellow et al., 2016), which leads to better generalization in these scenarios compared to MLPs.

In our experiments and analysis on these two test cases;[2] **First**, we demonstrate the importance of having the right inductive biases for solving these tasks (§2.1 and §3.1). **Second**, we show that KD is a powerful technique in which the teacher model drives the student toward a particular set of solutions that is more restricted compared to the set of possible solutions that a student can converge to when it learns directly from data (§2.2, §3.2, and Appendix C). **Third**, we demonstrate that, when distilling the knowledge from a model with stronger inductive bias (that suits the task at hand) into a model with weaker inductive bias, the student model converges to a solution that has similar characteristics to its teacher's:

- We show the performance of the student model increases, not only on in-distribution test sets (§2.2), but also on out-of-distribution data (§3.2). We demonstrate that this happens when the teacher has the right inductive bias and not necessarily otherwise.
- Besides performance, we show that the solution that a student model converges to shares similar characteristics to the teacher's, for instance per sample behaviour of the model (Appendix E), in terms of qualitative metrics like perplexity (§2.2), or confidence calibration (§2.2 and §3.2).
- We demonstrate that although the student model is merely exposed to the final logits of the teacher, the structure of the latent space of the student model becomes similar to the teacher, i.e. relational similarity of the internal representations from the student and its teacher increases (§2.2 and §3.2).

As an example, in our second test case (MNIST-C), when training an MLP model with KD using a CNN teacher, the student model explores the solution space in ways more similar to its teacher. Figure 1 visualizes and compares the path that an MLP takes during training (Figure 1a), compared to a CNN (Figure 1b). The CNN model explores the surface in a completely different manner than the MLP, while the path of a student MLP distilled from the CNN model as the teacher (Figure1c) is more similar to the CNN.

[2]The codes for the input pipelines, models, analysis, and the details of the hyper-parameters used in our experiments is available at `https://github.com/samiraabnar/Reflect`.

2 DISTILLING LSTMS INTO TRANSFORMERS

LSTMs and Transformers are the basic building blocks of many state-of-the-art models for sequence modelling and natural language processing. Transformers are an expressive class of models that do extremely well on many tasks where the training data is adequate in quantity (Devlin et al., 2019; Keskar et al., 2019; Radford et al., 2019; Brown et al., 2020). Several studies, however, have shown that LSTMs can perform better than Transformers on tasks requiring sensitivity to (linguistic) structure, especially when the data is limited (Tran et al., 2018; Dehghani et al., 2019).

We chose the subject-verb agreement prediction task, introduced by Linzen et al. (2016), as the test case, as it yields a meaningful difference between LSTMs and Transformers (Tran et al., 2018). We compare these two families of models and conduct experiments to emphasize the differences between them when trained independently and through KD.

Recurrent Inductive Bias. Among sequence modelling architectures, models with recursion are in particular powerful for natural language processing due to their adequacy to model hierarchical structures (Linzen et al., 2016). The recursion in a model can be implemented in various ways, like in Recurrent Neural Networks (Elman, 1990), Recursive Neural Networks (Socher et al., 2010; Le & Zuidema, 2014) and Universal Transformers (Dehghani et al., 2019; Hao et al., 2019). While theoretically, both recurrent neural networks (RNNs) and Transformers can deal with finite hierarchical structures, empirical results indicate the superiority of RNNs over Transformers (Tran et al., 2018; Dehghani et al., 2019; Hahn, 2020).

In the literature (Sutskever et al., 2013; Dehghani et al., 2019), the inductive bias of RNNs is referred to as the *recurrent inductive bias*. Here, we distinguish between three main sources of this bias:

1. **Sequentiality**: There is an inherent notion of order in the architecture that forces the model to access next tokens in the input one by one and process them sequentially.
2. **Memory bottleneck**: The model has no direct access to the past tokens and has to compress all the information from the past in a hidden state, which is accessible when processing a new token.
3. **Recursion**: The model recursively applies the same function on the varying input at every step.

Transformers (Vaswani et al., 2017), in contrast, process the input in parallel. Although a weak notion of order is encoded by positional embeddings, no explicit assumption is made in the connectivity structure of the architecture. Moreover, they have a global receptive field and can access all tokens through self-attention. Finally, standard Transformers are not recursive. However, the standard Transformer can be modified to have an architecture with specifications that are similar to RNNs. We provide empirical results to demonstrate the benefits of these different sources of inductive biases of RNNs. For this purpose, we design experiments with variants of Transformers in which we attempt to approximate some of the RNNs' assumptions.

Task and Models. We study the performance of LSTMs and variants of Transformers on the task of predicting number-agreement between subjects and verbs in English sentences. We investigate the quality of the solutions they converge to when they are trained independently and when they are trained through distillation. We use the subject-verb agreement dataset of Linzen et al. (2016), for which the size of the training set is $\sim$121k examples and the size of the test set is $\sim$1m. Succeeding at this task is a strong indicator that a model can learn syntactic structures and is therefore proposed by Linzen et al. (2016) as a proxy for assessing the ability of models to capture hierarchical structure in natural language. It is shown that RNNs have better inductive biases to learn this compared to standard Transformers (Tran et al., 2018; Dehghani et al., 2019). In this task, examples are grouped into different levels of difficulty based on the number of "agreement attractors"[3], and distance between the verb and its subject. Hence, we report both micro accuracy ($\mu-$Accuracy) and macro accuracy over different groups in terms of distance ($\mathcal{D}-$Accuracy) and numbers of attractors ($\mathcal{A}-$Accuracy).

Similar to Tran et al. (2018), we follow two setups: 1) when the learning objective is next word prediction, i.e., language modelling (LM); 2) when we directly optimize for predicting the verb number, singular or plural, i.e., classification. In the LM setup, we look at the probabilities predicted when the target of the prediction is the verb of interest, and see whether the probability of the correct form of the verb is higher than the other form (singular vs plural). In the classification setup, the input to the model is a sentence up to the position of the verb of interest and the model predicts whether the verb at that position is singular or plural.

[3] Agreement attractors are intervening nouns with a different number than the number of the subject. E.g., given the input "`The **keys** to the cabinet (is?/are?).`", the word "cabinet" is an agreement attractor.

Table 1: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the LM objective.

Model	Perplexity ↓	$\mathcal{D}-$Accuracy ↑	$\mathcal{A}-$Accuracy ↑
Transformer	57.50 ± 0.1199	0.9417 ± 0.0017	0.9191 ± 0.0018
Small Transformer	**55.31** ± 0.0847	0.9467 ± 0.0012	0.9261 ± 0.0020
LSTM	56.68 ± 0.0906	**0.9510** ± 0.0012	**0.9400** ± 0.0024
Small LSTM	58.05 ± 0.1141	0.9491 ± 0.0006	0.9366 ± 0.0015

Table 2: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the classification objective.

Model	$\mu-$Accuracy ↑	$\mathcal{D}-$Accuracy ↑	$\mathcal{A}-$Accuracy ↑
Transformer	0.954 ± 0.0016	0.901 ± 0.0037	0.717 ± 0.0244
Transformer-seq	0.964 ± 0.0010	0.909 ± 0.0037	0.742 ± 0.0121
UniversalTransformer-seq	0.969 ± 0.0004	0.932 ± 0.0055	0.806 ± 0.0153
LSTM	**0.977** ± 0.0001	**0.970** ± 0.0003	**0.928** ± 0.0007

In the LM setup, we employ two unidirectional LSTMs with different sizes, *LSTM* and *Small LSTM*, and two Transformers, *Transformer* and *Small Transformer*. In this setup, corresponding LSTMs and Transformers have roughly the same number of parameters. In the classification setup we compare the following models: (1) a standard unidirectional LSTM (*sequentiality* + *memory bottleneck* + *recursion*) (2) Transformer: Transformer encoder with a class token (`CLS`) for classification, BERT (Devlin et al., 2019) style, (3) Transformer-seq: Transformer encoder with future masking where the classification is done using the representation of the last token[4] (*sequentiality*), (4) UniversalTransformer-seq: Universal Transformer (Dehghani et al., 2019) encoder, in which the parameters are shared in depth, with future masking (*sequentiality* + *recursion*). Appendix F provides more details on the architectures.

2.1 The Importance of Recurrent Inductive Bias

In this section, we report results without distillation that illustrate the merits of the recurrent inductive bias. Table 1 shows the performance of the models when trained with the LM objective. A first important observation, in line with the results of Tran et al. (2018), is that LSTMs achieve better accuracy on the subject-verb agreement task compared to Transformers. Even for instances of Transformer language models that achieve better (lower) perplexity, the accuracy on this task is worse compared to LSTM instances. Since both models achieve good scores on the training set (Appendix D), this suggests that LSTMs better capture relevant patterns, such as the hierarchical structure of the input, which leads to better generalization on this task.

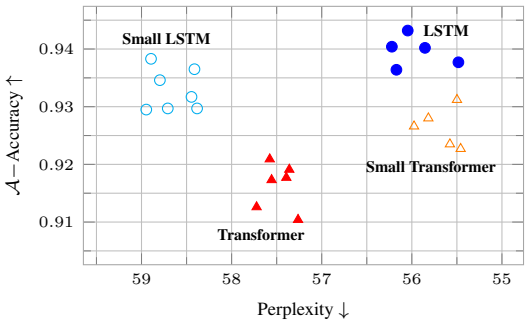
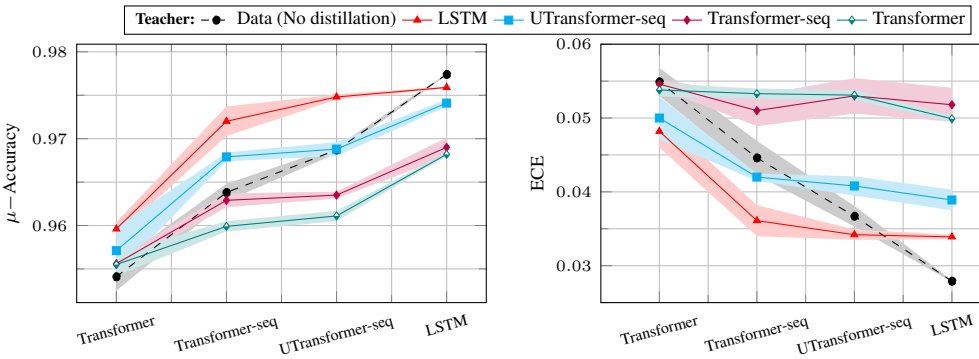


Figure 2: $\mathcal{A}-$Accuracy vs perplexity (high to low from left to right) for language models of different architectures and sizes.

Figure 2 illustrates the accuracy versus perplexity of several instances of each model, in the LM setup. Note that although perplexity is an indicator of how well the model is optimized given the objective function, the accuracy is the metric that matters and shows models' generalization in subject-verb agreement task. The plot in Figure 2 also shows different bias-variance trade-offs of Transformers and LSTMs, each with a large and a small variant (as measured by the number of trainable parameters). The richer hypothesis space of the Transformers hurts their generalization, as the variance increases and becomes a source of error. In contrast, adding more capacity leads to slightly better accuracy in LSTMs as their stronger inductive biases control the generalization error.

In Table 2 we show the results of models trained on the classification objective. We compare LSTM with variants of Transformers with different inductive biases. The table shows that similar to the LM results, LSTM achieves the best performance. Interestingly, comparing all four models, we find that the performance steadily increases as more aspects of the recurrent inductive bias are included. This is illustrated in Figure 3a, with the filled circles on the black, dashed line (no distillation).

[4]Note that future tokens are masked out by default when using a transformer in the decoder mode, e.g., in LM setup.



(a) Accuracy ↑ (b) Expected Calibration Error ↓

Figure 3: Performance (mean±std over 4 trials) of models with different inductive biases trained independently or using KD with different teachers.

Table 3: Performance (mean±std over 4 trials) of different LSTM and Transformer models with LM objective when we apply pure distillation with $\tau = 1$.

Student Model		Teacher Model **LSTM**	**Small LSTM**	**Transformer**	**Small Transformer**
LSTM	$\mathcal{A}-$Accuracy ↑	0.9275 ± 0.0017	0.9310 ± 0.0013	**0.9083** ± 0.0044	**0.9257** ± 0.0027
	perplexity ↓	59.45 ± 0.0191	60.92 ± 0.0185	60.01 ± 0.0328	58.65 ± 0.0036
Small LSTM	$\mathcal{A}-$Accuracy ↑	0.9224 ± 0.0024	0.9272 ± 0.0026	0.8985 ± 0.0057	0.9157 ± 0.0020
	perplexity ↓	62.52 ± 0.1071	63.44 ± 0.0272	63.45 ± 0.0644	61.62 ± 0.0619
Transformer	$\mathcal{A}-$Accuracy ↑	**0.9296** ± 0.0029	**0.9316** ± 0.0012	0.8956 ± 0.0018	0.9195 ± 0.0015
	perplexity ↓	**57.03** ± 0.0092	**59.09** ± 0.0126	**57.67** ± 0.0091	**56.64** ± 0.0352
Small Transformer	$\mathcal{A}-$Accuracy ↑	0.9201 ± 0.0018	0.9233 ± 0.0011	0.8827 ± 0.0027	0.9131 ± 0.0014
	perplexity ↓	57.84± 0.0269	59.73 ± 0.0166	58.44 ± 0.0354	57.16 ± 0.0087

As another indicator of the quality of the solutions that different models converged to in the classification setup, we look into their confidence calibration. Confidence calibration captures how well likelihood (confidence) of the prediction of the model predicts its accuracy (Guo et al., 2017). For a well-calibrated model, if we bin the confidence scores and compute the accuracy for each bin, the accuracies are perfectly correlated with the confidence values. The Expected Calibration Error (ECE) is computed as the distance between the calibration curve of the model and the perfect calibration curve (DeGroot & Fienberg, 1983). In Figure 3b, we plot the ECE (Guo et al., 2017) of the models in the classification setup, with the filled circles on the black dashed line (no distillation). In line with the trends in the performances of these models, the expected calibration error decreases as we move from Transformer toward LSTM.

In summary, the results from this section support the importance of recurrence for solving this task (Tran et al., 2018; Dehghani et al., 2019). Additionally, as shown in Figures 3a and 3b, we find a decreasing trend in the variance of the models, i.e., adding more inductive biases to the models decreases their variance. This is empirical evidence that supports the relation between variance of the solutions a model converges to and its inductive biases.

2.2 Transferring the Effect of Recurrent Inductive Bias

In this section, we show distilling knowledge from LSTM to Transformer can close the gap between their performance by pushing the Transformer to converge to solutions more similar to LSTM's.

Table 3 and Table 4 summarize the distillation results, when the training objective is language modeling and classification respectively. A first general observation is that, for these tasks and setups, distilling a model into an identical model could result in a decrease in the performance. Note that whether self-distillation results in improved performance could potentially depend on

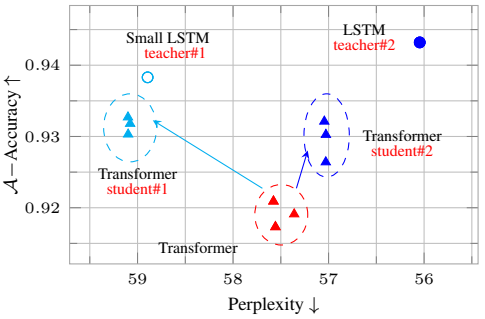


Figure 4: $\mathcal{A}-$Accuracy ↑ vs perplexity ↓ (high to low from left to right) for student Transformer with LM objective

Table 4: $\mu-$Accuracy ↑ (mean±std over 4 trials) of different LSTM and Transformer models with classification objective when we apply pure distillation with $\tau = 1$.

Student Model	Teacher Model			
	Transformer	**Transformer-seq**	**UTransformer-seq**	**LSTM**
Transformer	0.9555 ± 0.0013	0.9556 ± 0.0006	0.9571 ± 0.0027	0.9596 ± 0.0008
Transformer-seq	0.9599 ± 0.0006	0.9629 ± 0.0008	0.9679 ± 0.0005	0.9720 ± 0.0017
UTransformer-seq	0.9611 ± 0.0006	0.9635 ± 0.0004	0.9688 ± 0.0008	0.9748 ± 0.0003
LSTM	**0.9682** ± 0.0002	**0.9690** ± 0.0011	**0.9741** ± 0.0004	**0.9759** ± 0.0001

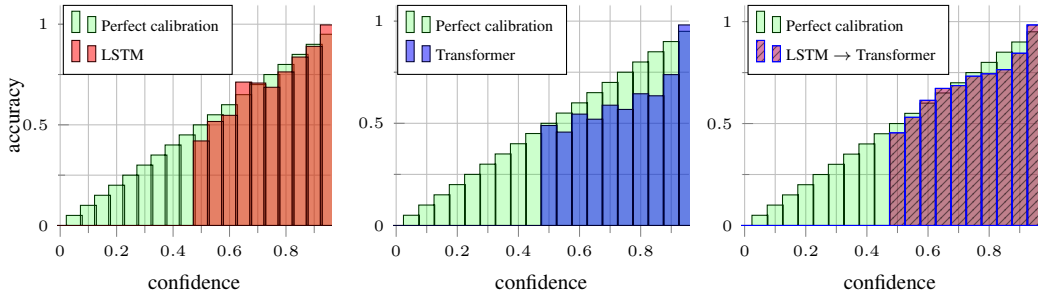


Figure 5: Calibration plots for independent and distilled Transformer for the classification setup. Note that since the task is binary classification, accuracy for confidences lower than 0.5 is not defined.

many different factors such as the architecture
of the model, optimization algorithm and details of the distillation process (Furlanello et al., 2018; Mobahi et al., 2020). Despite no significant changes in the performance with self-distillation, we can improve the performance of the Transformers through distillation from LSTM teachers.

To check whether this improvement is due to the transfer of the effect of inductive biases through distillation and whether distillation helps students to converge to solutions similar to their teachers, we run a series of analyses. In Figure 4 we see how teacher LSTMs pull student Transformers toward solutions with higher accuracy on the subject-verb agreement task in the LM setup. This happens even when the perplexity of the student Transformer is higher (worse) than the independent Transformer.

Figure 3, also shows the effects of distillation on each of the four models we study in the classification setup. In Transformer-based models, we get the most significant improvement both in accuracy and ECE when the teacher is an LSTM. As the recurrent inductive biases of the teacher get weaker, the amount of improvement in the performance of student models decreases. Figure 5 shows the effect of KD on the calibration, given a student Transformer and an LSTM teacher.

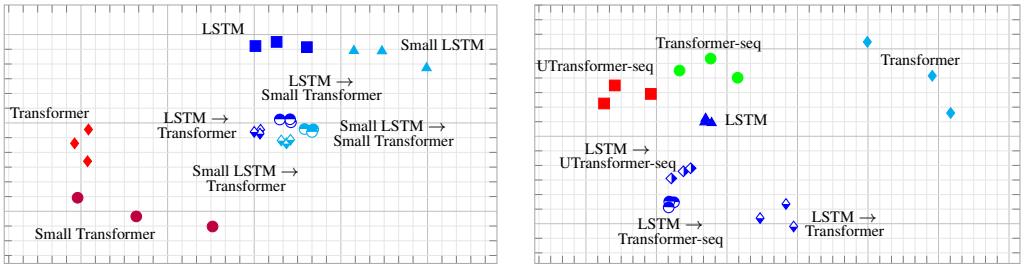
Is the improvement in calibration merely the product of using soft targets? Mueller et al. (2019) shows training neural networks with soft targets (e.g. through label smoothing) results in models that are better calibrated. On the other hand, KD has a regularization effect similar to label smoothing (Yuan et al., 2019; Tang et al., 2020). Given the lack of significant improvement in ECE in the self-distillation experiments (Figure 3b), it is more likely that the cause of the improvement in ECE when distilling LSTMs into Transformers is beyond the label smoothing effect of KD.

To further explore and better understand the effects of KD, we compare the internal representations of these models besides their final output. Figure 6 shows the 2D projection of the relational similarity of representations[5] (Laakso & Cottrell, 2000) from the penultimate layer of the models (see Appendix B for details). We see that, in the LM setup, the internal representations of student Transformers that are distilled from LSTMs are structured differently compared to independent Transformers and are more similar to the LSTM models. For the classification objective, we also see that the distilled models are further away from their independent versions. This supports the idea that the effect of distillation goes beyond the output of the models and their final performances.

3 Distilling CNNs into MLPs

To evaluate the robustness of our findings on the transfer of inductive biases through KD, we performed a second case study, using different neural architectures and a different task. We use

[5]Note that the relational similarity captures the similarity of the structures, not the absolute values.



(a) Language Modelling (b) Classification

Figure 6: 2D projection of representational similarity of the activations from the penultimate layers for 1000 examples from the validation set (check Appendix B for more details). We use the notation of $a \rightarrow b$ to refer to the student model b distilled from teacher model a.

Table 5: Accuracy and Expected Calibration Error (mean±std over 4 trials) of CNN and MLP trained independently on MNIST and evaluated on MNIST, MNIST-Scaled and MNIST-Translated.

(a) Accuracy

Model	MNIST	Scaled	Translated
CNN	**0.992** ± 0.0009	**0.962** ± 0.0021	**0.981** ± 0.0003
MLP	0.985 ± 0.0011	0.794 ± 0.0154	0.373 ± 0.0151

(b) Expected Calibration Error

Model	MNIST	Scaled	Translated
CNN	**0.011** ± 0.0006	**0.060** ± 0.0044	**0.028** ± 0.0016
MLP	0.015 ± 0.0006	0.175 ± 0.0081	0.564 ± 0.0091

convolutional neural networks (CNN) vs. multilayer perceptrons (MLP) as two families of models with different inductive biases. CNNs are the de facto choice for processing data with grid-like topology. Sparse connectivity and parameter sharing in CNNs make them an effective and statistically efficient architecture. The particular form of parameter sharing in the convolution operation makes CNNs equivariant to translation (Goodfellow et al., 2016). Note that, we can view CNNs as MLPs with an infinitely strong prior over their weights, which says that first of all the weights for each hidden unit are identical to the weights of its neighbour with a shift in space, second, the weights out of the spatially continues receptive field assigned to each hidden unit are zero.

Task and Models. We study CNNs and MLPs in the context of the Corrupted-MNIST dataset (MNIST-C) (Mu & Gilmer, 2019), which aims at benchmarking out-of-distribution robustness. We train the models on the original MNIST training set and evaluate them on the Translated and Scaled MNIST test sets from MNIST-C. In this scenario, the inductive biases of CNNs help them generalize better than MLPs. Our CNN architecture is a stack of convolutions and pooling layers. Combining convolution and pooling over spatial regions results in invariance to translation. To have CNNs that can learn to be invariant to other transformations like changes in the scale, we can use cross-channel pooling (Goodfellow et al., 2013), where we pool over separately parametrized convolutions that have learned to detect different transformed versions of the same underlying features. Our MLP is simply a stack of fully-connected layers. More details on the architectures are in Appendix F.

3.1 The Importance of Translation Equivariance.

Table 5 presents the accuracy and ECE of CNNs and MLPs when trained independently. All models are trained on the original MNIST training set and tested on the *Scaled* and *Translated* sets from MNIST-C. Even though CNNs' accuracy and ECE on the original MNIST test set are only slightly better than MLPs (.992 vs .985), there is a rather large gap between their performances on the Scaled (.962 vs. .794) and Translated (.981 vs. .373) test sets. This is as expected since the inductive biases of CNNs make them suitable for these types of generalizations. Moreover, the variance of the results from the CNNs is much less compared to MLPs. This is due to the fact different instances of a model with stronger inductive biases are more likely to converge to solutions that belong to the same basin in the loss landscape (Neyshabur et al., 2020) (See Appendix C for more analysis).

3.2 Better Out of Distribution Generalization with KD.

Table 6 shows that distillation from a CNN into an MLP improves both accuracy and ECE for all three test sets, almost closing the gap for the Scaled test set (.904 vs. .794 without KD), and much improving performance on the Translated test set (.510 vs. .373 without KD). We also see a lower variance in the performance of MLP models that are trained through KD with CNN teachers.

We further compare the results of all possible pairs of models as teachers and students, to take into account different effects of KD that can potentially improve the performance of the student

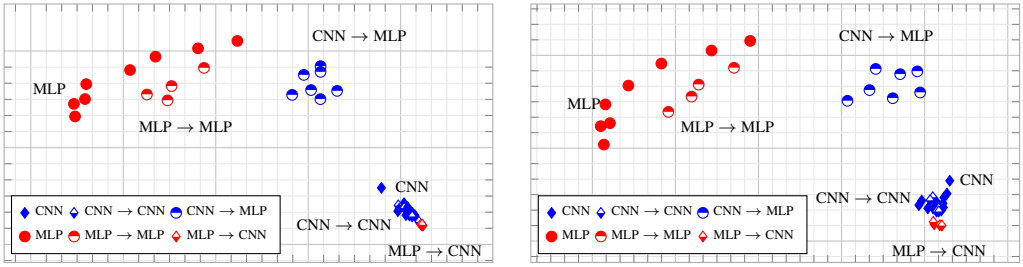
Table 6: Accuracy and Expected Calibration Error (mean±std over 4 trials) of CNN and MLP trained with pure distillation with $\tau = 5$, on MNIST and evaluated on MNIST, MNIST-Scaled and MNIST-Translated.

(a) Accuracy

Student Model	MNIST		Scaled		Translated	
	CNN	**MLP**	**CNN**	**MLP**	**CNN**	**MLP**
CNN	**0.991** ± 0.0004	**0.990** ± 0.0007	**0.951** ± 0.0046	**0.955** ± 0.0065	**0.978** ± 0.0003	**0.976** ± 0.0012
MLP	0.988 ± 0.0005	0.985 ± 0.0015	0.904 ± 0.0073	0.839 ± 0.0096	0.510 ± 0.0148	0.395 ± 0.0069

(b) Expected Calibration Error

Student Model	MNIST		Scaled		Translated	
	CNN	**MLP**	**CNN**	**MLP**	**CNN**	**MLP**
CNN	0.014 ± 0.0004	**0.013** ± 0.0005	**0.068** ± 0.0043	**0.054** ± 0.0063	**0.033** ± 0.0006	**0.030** ± 0.0016
MLP	**0.013** ± 0.0004	0.015 ± 0.0012	0.109 ± 0.0053	0.155 ± 0.0079	0.432 ± 0.0136	0.555 ± 0.0038



(a) Scaled MNIST (b) Translated MNIST

Figure 7: 2D projection of representational similarity of the activations from the penultimate layers for all examples from the test set (check Appendix B for more details). We use the notation of $a \to b$ to refer to the student model b distilled from teacher model a.

model. Although self-distillation results in a slightly better performance in MLPs, perhaps due to the regularization effect of distillation (Mobahi et al., 2020; Tang et al., 2020), the improvement in the performance of MLPs with an MLP teacher is much less compared to when the teacher is a CNN. Regardless of the teacher (MLP or CNN), KD results in slightly lower performances in student CNNs compared to CNNs trained independently (similar to results of an LSTM student in test case 1).

Furthermore, in Figure 7, we compare the relational similarity of the representations from penultimate layers of independently trained CNNs and MLPs as well as their distilled ones. First of all, as expected based on our assumptions about the inductive biases of these models, MLPs have more variance than CNNs. Second, distilling from a CNN to an MLP results in representations that are more similar to the representations learned by CNNs, while this is not the case with MLPs as teachers and CNNs as students. Moreover, for both CNNs and MLPs, self-distillation does not significantly change the representations they learn.

Finally, we compare the paths the models follow during training until they converge to a solution. To plot the training path of a model, we compute the pairwise representational similarity between different stages of training of the model. Figure 1, illustrates the training path for an independent MLP, an independent CNN, and an MLP that is distilled from a CNN. While MLP and CNN seem to have very different behaviour during training, the student MLP with a CNN as its teacher behaves differently than an independent MLP and more similar to its teacher CNN. This is interesting, in particular, since the student model is only exposed to the final solution the teacher has converged to and no information about the intermediate stages of training is provided in the offline KD.

4 Conclusions

The *no free lunch theorem* states: for any learning algorithm, any improvement on performance over one class of problems is balanced out by a decrease in the performance over another class (Wolpert & Macready, 1997). Neural networks with different architectures have different inductive biases and this is reflected in their performance across different tasks. In this paper, we investigate the power of KD to enable benefiting from the advantages of different models at the same time. We first demonstrate having the right inductive bias can be crucial in some tasks and scenarios. We further show that when a model has the right inductive bias, we can transfer its knowledge to a model that lacks the needed inductive bias and indicate that solutions that the student model learns are not only quantitatively but also qualitatively reflecting the inductive biases of the teacher model.

ACKNOWLEDGMENTS

We are grateful for the thorough feedback we got from Wilker Aziz on this project. Moreover, we would like to thank Rianne van den Berg, Justin Gilmer, Jessica Yung, Raquel Alhama, and Ece Takmaz for reviewing and commenting on the paper. The work presented here is funded by the Netherlands Organization for Scientific Research (NWO), through a Gravitation Grant 024.001.006 to the Language in Interaction Consortium.

REFERENCES

Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. Blackbox meets blackbox: Representational similarity & stability analysis of neural language models and brains. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019. URL https://www.aclweb.org/anthology/W19-4820.

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'19, 2019. URL https://arxiv.org/abs/1904.05835.

Rohan Anil, Gabriel Pereyra, Alexandre Tachard Passos, Robert Ormandi, George Dahl, and Geoffrey Hinton. Large scale distributed neural network training through online distillation. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18, 2018. URL https://openreview.net/pdf?id=rkr1UDeC-.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, 2006. URL https://dl.acm.org/doi/10.1145/1150402.1150464.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pp. 698–728, 2016.

Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems 8*, NeurIPS'95, Cambridge, MA, USA, 1995. MIT Press. URL https://papers.nips.cc/paper/1152-extracting-tree-structured-representations-of-trained-networks.pdf.

Mark William Craven. *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, University of Wisconsin-Madison, 1996. URL https://www.biostat.wisc.edu/~craven/papers/thesis.pdf.

Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 1983.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19, 2019. URL https://arxiv.org/abs/1807.03819.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'19, 2019. URL https://arxiv.org/abs/1810.04805.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv: 2002.06305*, 2020. URL https://arxiv.org/abs/2002.06305.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990. URL https://crl.ucsd.edu/~elman/Papers/fsit.pdf.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19, 2019. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2019.html#FrankleC19.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. Ensemble distillation for neural machine translation. *CoRR*, abs/1702.01802, 2017. URL http://arxiv.org/abs/1702.01802.

Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence*, 2017. URL https://arxiv.org/abs/1711.09784.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, ICML'18, 2018. URL https://arxiv.org/abs/1805.04770.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 2017. URL https://arxiv.org/abs/1706.04599.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8, 2020.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. Modeling recurrence for transformer. *arXiv preprint arXiv:1904.03092*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. URL https://dl.acm.org/doi/10.1162/neco.1997.9.8.1735.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019. URL https://arxiv.org/abs/1909.05858.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP'16, 2016. URL https://www.aclweb.org/anthology/D16-1139.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://arxiv.org/pdf/1412.6980.pdf.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. Scalable syntax-aware language models using knowledge distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3472–3484, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1337. URL https://www.aclweb.org/anthology/P19-1337.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. Syntactic structure distillation pretraining for bidirectional encoders. *arXiv preprint arXiv:2005.13482*, 2020.

Aarre Laakso and Garrison Cottrell. Content and cluster analysis: Assessing representational similarity in neural systems. *Philosophical Psychology*, 13(1):47–76, 2000. doi: 10.1080/09515080050002726. URL https://doi.org/10.1080/09515080050002726.

Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, 2014. URL https://www.aclweb.org/anthology/D14-1081.pdf.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4(0), 2016. URL https://transacl.org/ojs/index.php/tacl/article/view/972.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *ArXiv*, abs/1904.09482, 2019. URL https://arxiv.org/abs/1904.09482.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR'17, 2017. URL https://arxiv.org/abs/1608.03983.

Sihui Luo, Xinchao Wang, Gongfan Fang, Yao Hu, Dapeng Tao, and Mingli Song. Knowledge amalgamation from heterogeneous networks by common feature learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, IJCAI'19, 2019. URL https://arxiv.org/abs/1906.10546.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in neural information processing systems 32*, NeurIPS'19, 2019. URL https://arxiv.org/abs/1907.08549.

R. Thomas McCoy, Robert Frank, and Tal Linzen. Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *CoRR*, abs/2001.03632, 2020. URL https://arxiv.org/abs/2001.03632.

Tom M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980. URL http://dml.cs.byu.edu/~cgc/docs/mldm_tools/Reading/NeedforBias.pdf.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020.

Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

Rafael Rodrigo Mueller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems 32*, NeurIPS'19, 2019. URL https://arxiv.org/abs/1906.02629.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR'19, 2019. URL https://arxiv.org/abs/1904.05068.

Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. volume 97 of *Proceedings of Machine Learning Research*, pp. 5142–5151, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/phuong19a.html.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. URL https://www.ceid.upatras.gr/webpages/faculty/zaro/teaching/alg-ds/PRESENTATIONS/PAPERS/2019-Radford-et-al_Language-Models-Are-Unsupervised-Multitask-Learners.pdf.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. URL https://arxiv.org/abs/1910.01108.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18, 2018. URL https://openreview.net/forum?id=ry_WPG-A-.

H. S. Seung, H. Sompolinsky, and N. Tishby. Learning curves in large neural networks. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, 1991. URL https://dl.acm.org/doi/10.5555/114836.114847.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *In Proceedings of the NeurIPS'10 Deep Learning and Unsupervised Feature Learning Workshop*, 2010. URL https://ai.stanford.edu/~ang/papers/nipsdlufl10-LearningContinuousPhraseRepresentations.pdf.

Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *Proceedings of the 26th British Machine Vision Conference*, BMVC'15, 2015. URL https://www.semanticscholar.org/paper/Data-free-Parameter-Pruning-for-Deep-Neural-Srinivas-Babu/b0bd441a0cc04cdd0d0e469fe4c5184ee148a97d.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. In *EMNLP/IJCNLP*, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, ICML'13, 2013. URL https://dl.acm.org/doi/10.5555/3042817.3043064.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461*, 2019. URL https://arxiv.org/abs/1902.10461.

Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H Chi, and Sagar Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2020.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. *ArXiv*, abs/1903.12136, 2019. URL https://arxiv.org/abs/1903.12136.

Ke Tran, Arianna Bisazza, and Christof Monz. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP'18, 2018. URL https://www.aclweb.org/anthology/D18-1503.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. *ArXiv*, abs/1907.09682, 2019. URL https://arxiv.org/abs/1907.09682.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems 30*, NeurIPS'17, pp. 5998–6008, 2017. URL https://arxiv.org/abs/1706.03762.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997. ISSN 1089-778X. doi: 10.1109/4235.585893. URL https://doi.org/10.1109/4235.585893.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. *arXiv preprint arXiv:1909.11723*, 2019.

A KNOWLEDGE DISTILLATION IN NEURAL NETWORKS

Knowledge Distillation is a technique that transfers knowledge from one model to another (Hinton et al., 2015). Hinton et al. (2015) suggest that the power of KD is mostly in being able to transfer the useful information that is embedded in the soft targets of the teacher model, e.g., the relation between the output classes as captured by the teacher model. This is often referred to as *dark knowledge*. Phuong & Lampert (2019) studies KD from a theoretical point of view in a simplified setting where the task is a binary classification, and teacher and student are linear models. They attribute the success of distillation to three main factors: (1) data geometry, (2) optimization bias, and (3) strong monotonicity. And more recently Tang et al. (2020), conduct extensive analysis and identify three sources for why KD helps: (1) label smoothing, (2) example re-weighting based on teacher's confidence, and (3) prior knowledge of optimal output layer geometry.

The most well-known use of KD is to compress a large, unwieldy model or an ensemble model into a smaller model. Empirically, many people have found that bigger models are easier to train (often explained with the 'lottery ticket hypothesis' (Frankle & Carbin, 2019)); KD makes it possible to distil the knowledge in the large model into a much smaller model, and thus in some sense offer the best of both worlds (Buciluǎ et al., 2006; Hinton et al., 2015; Srinivas & Babu, 2015). Distilling knowledge from a very big model or an ensemble of models with similar or heterogeneous architectures that are trained on the same or different tasks into a single model with much fewer parameters can lead to similar or sometimes even better performance compared to the teachers (Luo et al., 2019; Liu et al., 2019; Hinton et al., 2015; Tan et al., 2019; Kim & Rush, 2016).

Previous work has examined the effectiveness of KD in different settings: where the teacher is bigger than the student, but both have similar building blocks (Hinton et al., 2015; Kim & Rush, 2016; Sanh et al., 2019); where teacher and student are of similar size and architecture (Furlanello et al., 2018; Freitag et al., 2017); or where the student and teacher have fundamentally different architectures (Frosst & Hinton, 2017; Tang et al., 2019; Luo et al., 2019; Ahn et al., 2019).

KD has also been proposed as an interpretation technique, where the knowledge of a big complex model is distilled into a more interpretable model (Craven & Shavlik, 1995; Craven, 1996; Frosst & Hinton, 2017); Or as a method to compare the capacity and expressiveness of different models (Maheswaranathan et al., 2019; Saxe et al., 2018).

Offline Distillation In most cases, KD is applied in an offline setting, i.e., we first train the teacher network and use the trained teacher to train the student, while the parameters of the teacher are fixed. This is the standard distillation process introduced by Buciluǎ et al. (2006); Hinton et al. (2015). We apply this setup in our experiments since it is the most common approach. There are other possible settings for KD, e.g. online distillation, where teacher and student models are trained simultaneously.

Distillation Loss There are several different ways of computing the distillation loss: using only the output of the teacher or taking intermediate layers into account as well (Anil et al., 2018; Ahn et al., 2019; Sun et al., 2019; Park et al., 2019; Tung & Mori, 2019; Buciluǎ et al., 2006; Hinton et al., 2015). Potentially, using these alternative losses could lead to transferring different kinds of knowledge depending on the tasks and the configurations of the models. While it is worth doing a thorough comparison of all these techniques, in this paper we have focused on the most commonly used loss introduced by Hinton et al. (2015), which is based on the Kullback-Leibler divergence between output distributions of the teacher, i.e., soft targets, and the output distributions of the student. The output distributions of the teacher and student model, P_t and P_s, are computed similarly, with Equation 1.

$$\frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}, \tag{1}$$

where $\tau > 1$ is the softmax temperature and z is the logits from the model.

The distillation loss is: $\mathcal{H}(P_t, P_s)$, where $\mathcal{H}$ is the cross entropy loss and is computed as:

$$\mathcal{H}(P_t, P_s) = -\sum_x P_t(x) \log P_s(x) \tag{2}$$

When KD is applied as a means for model compression, it is common to compute the total loss as a mixture of distillation loss and actual loss. Since, our focus in this paper is on how much the student model can learn from the teacher model, in our experiments we use pure distillation.

B VISUALISATION OF REPRESENTATIONAL SIMILARITY OF THE ACTIVATIONS FROM THE PENULTIMATE LAYER

To compare and visualize the state of m different models to each other (at convergence or any stage of training), we propose using representational similarity (Laakso & Cottrell, 2000; Abnar et al., 2019) of the activations from their penultimate layer.

Note that representational similarity measures how similar two models learn to represent the data in terms of the global "relations" between all the data points, not local example-by-example similarity. In fact, the "direct" similarity between the activations of the penultimate layers of two models can be quite low, while having high representational similarity. This is because models can keep the relations between data points similar while embedding data into completely different representational spaces.

This is particularly useful when these models do not have the same architecture and their parameter space is not directly comparable. To do so, given a sample set of size n from the validation/test set (e.g. 1000 examples), we feed them to the forward pass of each model to obtain the representation from the penultimate layer of the models. Then, for each model, we calculate the similarity of the representations of all pairs from the sample set using dot product which leads to a matrix of size $n \times n$. We use the samples similarity matrix associated with each model to compute the similarity between all pairs of models. To do this, we compute the dot product of the corresponding rows of these two matrices after normalization and average all the similarity of all rows, which leads to a single scalar. Given all possible pairs of models, we then have a model similarity matrix of size $m \times m$. We then apply a multidimensional scaling algorithm[6] to embed all the models in a 2D space based on their similarities.

The code for projecting the representational similarity of the activations from the penultimate layer to a 2D space can be found in `https://github.com/samiraabnar/Reflect/tree/master/notebooks/viz`.

C DO THE DISTILLED MODELS CONVERGE TO THE SAME BASIN IN THE LOSS LANDSCAPE?

To gain a better understanding of the effect of KD and inductive biases of the models from an optimization point of view, we looked into how different models relate in terms of the solutions they converged to in the loss landscape.

To do so, inspired by the discussion in (Neyshabur et al., 2020), we look into different pairs of models and check if their final solution belong to the same flat basin[7] of the loss landscape or they converged to completely different optima. To do so, given two models,m_1 and m_2, we take their parameters, θ_1 and θ_2, and evaluate a series of models obtained by linearly interpolating θ_1 and θ_2, with different coefficient, i.e., the parameters of model m_i is computed as $\theta_i = \lambda_i\theta_1 + (1 - \lambda_i)\theta_2$. It has been shown (Neyshabur et al., 2020) that if the converged solutions of m_1 and m_2 belong to the same flat basin of the loss landscape, the models obtained by linearly interpolating their parameters are well-behaved because they also remain in that basin. However, for two models that converge to different optima and don't share the flat basin of the loss landscape, the liner interpolations do not lead to well behave models.

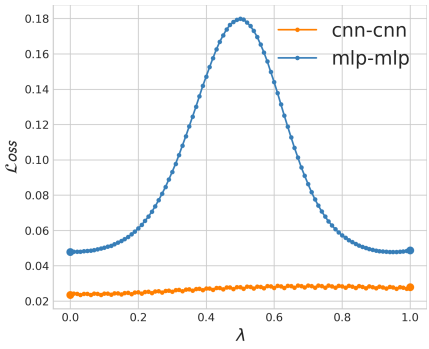


Figure 8: Performance barriers between different instances of MLPs and CNNs (with the same initialization), in terms of loss on the test.

Here, we first, compare different instances of MLPs and CNNs. We train two instances of the same architecture with the same initial state but different random seeds (which would lead to different ordering of training examples, and different dropouts). Figure 8 shows the loss on the test set (y

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html`

[7] Basin refers to areas in the parameter space where the loss function has relatively low values.

axis) for the two trained instances, as well as models obtained by linear interpolation of the two models with different λs (x axis). In the case of MLPs, there is a large barrier between the two instances, showing that these models, even with the same initialization, will converge to solutions in different basins of the loss landscape. In contrast, for CNNs, their strong inductive biases drive them to converge to the solutions in the same basin, regardless of the stochasticity of the training process. This also supports the higher variance in the results we report for models with weaker inductive biases in §2.2 and §3.2.

Next, we look into the effect distillation on the diversity of the basins different instances of models converge to. Figure 9 shows the performance barriers of different pairs of MLPs (MLP#1 and MLP#2), when they are trained independently (i.e. when the teacher is data), as well as trained through KD, with an MLP and a CNN model as teachers.

First of all, we observe that two models, initialized similarly but with different random seeds, trained through distillation with the same teacher are likely to converge to the same area in the loss surface (plots (c) and (f)). This happens regardless of the inductive bias of the teacher and student models. Comparing the plots in the diagonal of Figure 9, we can see that for both $CNN \rightarrow MLP$ (plot f) and $MLP \rightarrow MLP$ (plot c) the performance barrier is rather small in contrast to the large barrier between two independently trained MLPs (plot a). This indicates the power of KD to narrow down the search space of the student model and drive it to a particular set of solutions.

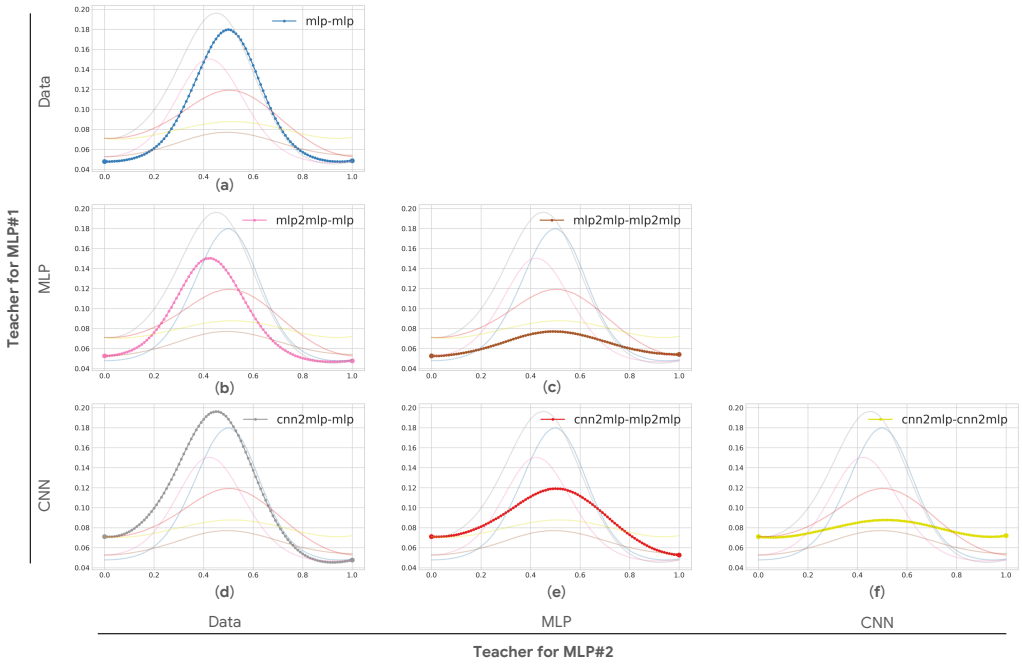


Figure 9: Performance barriers between different instances of MLPs with the same initialization trained independently or through knowledge distillation. Here y-axis on each subplot is the value of the loss on the test set and x-axis is the value of interpolation coefficient, λ. The rows in the figure correspond to the teacher of the instance on the left side (MLP#1) and the columns correspond to the teacher of the instance on the right side of the plots (MLP#2).

Moreover, comparing the distilled instance of a model with an independently trained instance with the same initialization and different random seeds, the first column of Figure 9 (plots (a), (b), and (d)), we see that the distilled instances and independent instances are not in the same basin, regardless of the teacher but the barrier is larger (larger bump in the plots) when the teacher has a stronger inductive bias ($CNN \rightarrow MLP$). Similarly, as depicted in the second and third columns of Figure 9, while models distilled from the same teacher seem to be close in the loss surface (plots (c) and (f)), models distilled from different teachers (plot (e)) seem to be further away (have a larger barrier in between).

D PERFORMANCE SCORES ON THE TRAINING DATA

In the paper, for our first test case, we report the performance of LSTM and different Transformer models on the test set, when trained independently and with knowledge distillation. We observe that LSTMs achieve better accuracy on test set compared to Transformers due to their inductive biases. Here, we also report the performance of all the models, for both classification and LM setup, on the training set, which confirms that Transformer models have enough capacity to achieve good scores on the training data.

This solidifies the narrative that the inductive bias of LSTMs is helping with generalization and rules out, for example, the possibility that LSTMs have a higher capacity or are trained better.

Model	**Perplexity** $\downarrow$	$\mathcal{D}-$**Accuracy** $\uparrow$	$\mathcal{A}-$**Accuracy** $\uparrow$
Transformer	29.62 ± 0.10	0.956 ± 0.001	0.936 ± 0.004
Small Transformer	33.02 ± 0.05	0.959 ± 0.001	0.948 ± 0.005
LSTM	28.92 ± 0.08	0.964 ± 0.003	0.955 ± 0.003
Small LSTM	31.03 ± 0.11	0.964 ± 0.001	0.952 ± 0.006

Table 7: Performance (mean$\pm$std over 4 trials) of different LSTM and Transformer models trained independently with the LM objective on the training set.

Model	**Train** $\mu-$**Accuracy** $\uparrow$
Transformer	99.57
Transformer-seq	99.57
UniversalTransformer-seq	99.66
LSTM	98.62

Table 8: Performance (mean$\pm$std over 4 trials) of different LSTM and Transformer models trained independently with the classification objective on the training set.

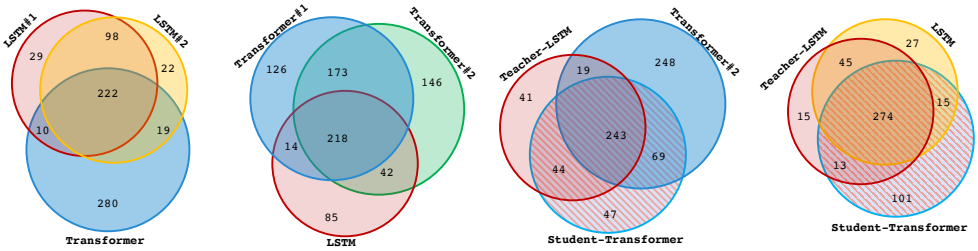
E PER-SAMPLE BEHAVIOUR

To compare the models with each other and better understand how distillation affects the student models, we take a closer look at their per sample behaviour and investigate if the errors a student model makes are more similar to its teacher's errors. Here, we look into the error overlap of the students and teachers, which reflects their similarity in terms of their behaviour per data example. This similarity can be another proxy to measure the similarity of the solutions learned by the models, with and without distillation. Figures 10, 11, and 12 illustrate the error overlap between different models as Venn diagrams when they are trained independently and when we use distillation.

In Figure 10, we observe that when the Transformer and LSTM models are trained independently, two independent LSTMs behave more similarly compared to two Transformers (Figures 10b and 10a). Given a similar number of trainable parameters, i.e., similar capacity for LSTMs and Transformers, this again supports the claim that models with stronger inductive biases converge to more similar solutions (Also shown in Figure 3a).

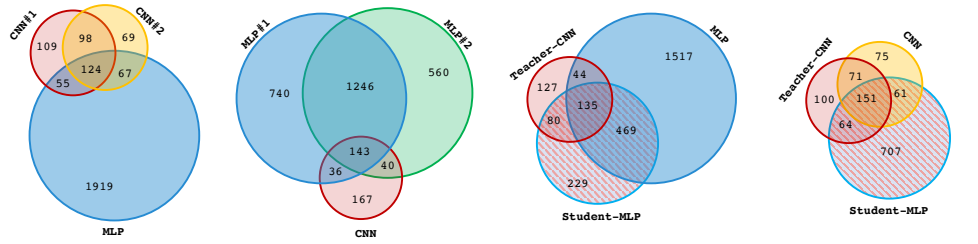
When we apply KD in a cross-architecture setting, with an LSTM teacher and a student Transformer, Figures 10d and Figure 10c, the student Transformer behaves more similarly to the LSTM teacher and an independent LSTM, compared to the independent version of itself. This confirms that through distillation the way the student model solves the task becomes more similar to the way the teacher model solves the task.

We have similar observations in Figures 11, and 12; where errors of a student MLP are less and more similar to the errors the teacher CNN compared to an independently trained MLP.



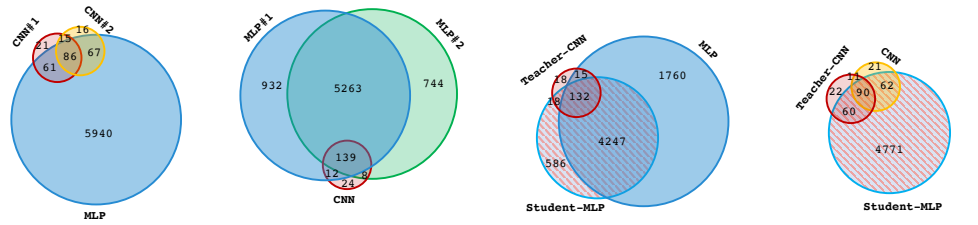
(a) (b) (c) (d)

Figure 10: Error overlap for LSTM and Transformer models trained with the classification objective on SVA task. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent LSTMs (LSTM#1 and LSTM#2) and an independent Transformer; in (b) we compare two independent Transformers (Transformer#1 and Transformer#2) and an independent LSTM; in (c) we compare a student Transformer and a teacher LSTM with an independent Transformer; and in (d) we compare a student Transformer and a teacher LSTM with an independent LSTM.



(a) (b) (c) (d)

Figure 11: Error overlap for CNN and MLP models trained on MNIST and tested on Scaled-MNIST set from MNIST-C dataset. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent CNN (CNN#1 and CNN#2) and an independent MLP; in (b) we compare two independent MLP (MLP#1 and MLP#2) and an independent CNN; in (c) we compare a student MLP and a teacher CNN with an independent MLP; and in (d) we compare a student MLP and a teacher CNN with an independent CNN.



(a) (b) (c) (d)

Figure 12: Error overlap for CNN and MLP models trained on MNIST and tested on Translated-MNIST set from MNIST-C dataset. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent CNN (CNN#1 and CNN#2) and an independent MLP; in (b) we compare two independent MLP (MLP#1 and MLP#2) and an independent CNN; in (c) we compare a student MLP and a teacher CNN with an independent MLP; and in (d) we compare a student MLP and a teacher CNN with an independent CNN.

F Detailed Models Architectures and Training Setup

For the subject-verb agreement task, we study Transformers and LSTMs. In the LM setup, we use two sizes for each architecture: LSTM: two-layer uni-direction LSTM, with a hidden size of 1024. Small LSTM: two-layer uni-direction LSTM, with a hidden size of 512. Transformer: six-layer

Transformer decoder with a hidden size of 512 and 8 heads. Small Transformer: Transformer: six-layer Transformer decoder with a hidden size of 256 and 8 heads.

In the classification setup, we employ an LSTM and three variants of Transformer, where the LSTM has a two-layer with a hidden size of 256, and the Transformers have 6 layers, 8 heads and a hidden size of 128. We use a hidden size of 256 for the UniversalTransformer-seq since its parameters are shared in depth and with the same hidden size as other Transformers, it will have fewer parameters.

On the MNIST-C dataset, we study CNNs and MLPs. Our CNN has two 3×3 convolutions, followed by a max-pooling layer over spatial dimensions, followed by another 3×3 convolution and a maxout (max-pooling over channel dimension) layer (Goodfellow et al., 2013). Finally a global averaging is done over spatial dimensions, before the projection layer. The MLP model simply has three fully connected layers.

For training the independent models we use the Adam optimizer (Kingma & Ba, 2014) with exponential decay learning rate scheduler and for the student models in the distillation process, we use Adam optimizer with cosine decay restart (Loshchilov & Hutter, 2017) learning rate scheduler. The hyperparameters related to the regularization and learning rate schedulers are tuned separately for each model/experiment. For each model, we report the set of hyper-parameters that gives the best average performance across multiple trials with different random seeds for initialization.

G CODE

The code for all the analysis and experiments including the input pipelines, models, the details of the hyper-parameter sets used in our experiments are available at `https://github.com/samiraabnar/Reflect`, to facilitate the replication of all the experiments.