Causal Distillation for Language Models

Zhengxuan Wu*¶, Atticus Geiger*¶, Joshua Rozner, Elisa Kreiss, Hanson Lu

Thomas Icard, Christopher Potts, Noah D. Goodman

Stanford University
{wuzhengx, atticusg}@stanford.edu

Abstract

Distillation efforts have led to language models that are more compact and efficient without serious drops in performance. The standard approach to distillation trains a student model against two objectives: a task-specific objective (e.g., language modeling) and an imitation objective that encourages the hidden states of the student model to be similar to those of the larger teacher model. In this paper, we show that it is beneficial to augment distillation with a third objective that encourages the student to imitate the *causal* dynamics of the teacher through a *distillation interchange intervention training objective* (DIITO). DIITO pushes the student model to become a *causal abstraction* of the teacher model – a faithful model with simpler causal structure. DIITO is fully differentiable, easily implemented, and combines flexibly with other objectives. Compared against standard distillation with the same setting, DIITO results in lower perplexity on the WikiText-103M corpus (masked language modeling) and marked improvements on the GLUE benchmark (natural language understanding), SQuAD (question answering), and CoNLL-2003 (named entity recognition).

1 Introduction

Large pretrained language models have improved performance across a wide range of NLP tasks, but can be costly due to their large size. *Distillation* seeks to reduce these costs while maintaining performance by training a simpler student model from a larger teacher model (Hinton et al., 2015; Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2019).

Hinton et al. (2015) propose model distillation with an objective that encourages the student to produce output logits similar to those of the teacher while also supervising with a task-specific objective (e.g., sequence classification). Sanh et al. (2019), Sun et al. (2019), and Jiao et al. (2019) adapt this method, strengthening it with additional supervision to align internal representations between the two models. However, these approaches may push the student model to match all aspects of internal states of the teacher model irrespective of their *causal* role in the network's computation. This motivates us to develop a method that focuses on aligning the *causal* role of representations in the student and teacher models.

We propose augmenting standard distillation with a new objective that pushes the student to become a *causal abstraction* (Beckers and Halpern, 2019; Beckers et al., 2020; Geiger et al., 2021a) of the teacher model: the simpler student will faithfully model the causal effect of teacher representations on output. To achieve this, we employ the *interchange intervention training* (IIT) method of Geiger et al. (2021b). The *distillation interchange intervention training objective* (DIITO) aligns a high-level student model with a low-level teacher model and performs *interchange interventions* (swapping of aligned internal states); during training the high-level model is pushed to conform to the causal dynamics of the low-level model.

Figure 1 shows a schematic example of this process. Here, hidden layer 2 of the student model (bottom) is aligned with layers 3 and 4 of the teacher model. The figure depicts a single interchange intervention replacing aligned states in the left-hand models with those from the right-hand models. This results in a new network evolution that is shaped both by the original input and the interchanged hidden states. It can be interpreted as a certain kind of counterfactual as shown in Figure 1: what would the output be for the sentence "I ate some ⟨MASK⟩." if the activation values for the second token at the middle two layers were set to the values they have for the input "The water ⟨MASK⟩ solid."? DIITO then pushes the student model to output the same logits as the teacher, i.e., matching the teacher's output distribution under

*Equal contribution. ¶Correspondence authors.

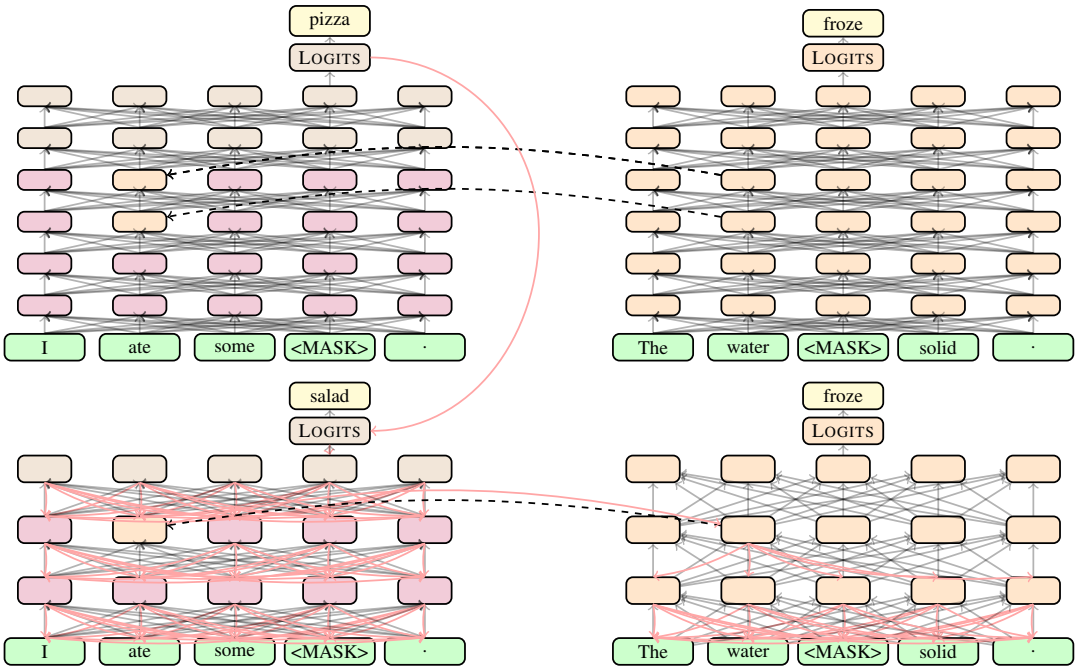

Figure 1: An IIT update in the context of masked language modelling (MLM). The teacher network (top) has 6 layers and the student (bottom) has 3 layers, and we align layer 2 in the student with layers 3-4 in the teacher. Solid lines are feed-forward connections, red lines show the flow of backpropagation, and dashed lines indicate interchange interventions. In this case, the student originally predicted the token "salad" under the interchange intervention, while the teacher predicted the token "pizza" under an aligned interchange intervention. DIITO trains the student to minimize the divergence between the student logits and the teacher logits under the interchange intervention. This updates the student to conform to causal dynamics of the teacher.

the counterfactual setup.

To assess the contribution of distillation with DIITO, we begin with $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019a) and distill it under various alignments between student and teacher while pretraining on the WikiText-103M corpus (Merity et al., 2016) (WikiText) achieving -2.24 perplexity on the MLM task compared to standard DistilBERT trained on the same data. We then fine-tune the best performing distilled models and find consistent performance improvements compared to standard DistilBERT trained with the same setting on the GLUE benchmark (+1.77%), CoNLL-2003 name-entity recognition (+0.38% on F1 score), and SQuAD v1.1 (+2.46% on EM score).[1]

2 Related Work

Distillation was first introduced in the context of computer vision (Hinton et al., 2015) and has since been widely explored for language models (Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2019). For example, Sanh et al. (2019) propose to extract information not only from output probabilities of the last layer in the teacher model, but also from intermediate layers in the fine-tuning stage. Recently, Rotman et al. (2021) adapt causal analysis methods to estimate the effects of inputs on predictions to compress models for better domain adaptation. In contrast, we focus on imbuing the student with the causal structure of the teacher.

Interventions on neural networks were originally used as a structural analysis method aimed at illuminating neural representations and their role in network behavior (Feder et al., 2021; Pryzant et al., 2021; Vig et al., 2020; Elazar et al., 2020; Giulianelli et al., 2020; Geiger et al., 2020, 2021a). Geiger et al. (2021b) extend these methods to network optimization. We contribute to this existing research by adapting intervention-based optimization to the task of language model distillation.

3 Causal Distillation

Here, we define our distillation training procedure. See Algorithm 1 in the Appendix for a summary.

GETVALS. The GETVALS operator is an activation-value retriever for a neural model. Given a neural model $\mathcal{M}$ containing a set of neurons $\mathbf{N}$

[1]We release our code at https://github.com/frankaging/Causal-Distill.

Model	Layers	Pretraining Tokens	WikiText Perplexity	GLUE Score	CoNLL-2003 acc	CoNLL-2003 F1	SQuAD v1.1 EM	SQuAD v1.1 F1
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019b) (Wikipedia+BookCorpus)	12	3.3B	10.27 (–)†	82.75 (–)	96.40 (–)	92.40 (–)	80.80 (–)	88.50 (–)
DistilBERT (Sanh et al., 2019) (Wikipedia+BookCorpus)	6	3.3B	17.48 (–)†	79.59 (–)	98.39 (–)†	93.10 (–)†	77.70 (–)	85.80 (–)
DistilBERT (WikiText)	3	0.1B	29.51 (0.32)	67.42 (1.10)	97.88 (0.04)	88.89 (0.29)	26.04 (0.93)	68.38 (0.77)
$\text{DIITO}_{\text{MIDDLE}}$ (WikiText)	3	0.1B	26.04 (0.93)	69.30 (1.08)	98.03 (0.04)	89.69 (0.18)	58.74 (0.69)	70.23 (0.57)
$\text{DIITO}_{\text{LATE}}$ (WikiText)	3	0.1B	25.97 (0.63)	69.01 (1.69)	98.03 (0.03)	89.82 (0.18)	58.75 (0.49)	70.21 (0.41)
$\text{DIITO}_{\text{FULL}}$ (WikiText)	3	0.1B	24.85 (0.58)	69.36 (0.87)	98.02 (0.03)	89.67 (0.16)	58.72 (0.67)	70.50 (0.56)
DistilBERT (WikiText)	6	0.1B	15.69 (1.51)	75.80 (0.42)	98.48 (0.03)	92.12 (0.23)	70.23 (0.75)	79.99 (0.55)
$\text{DIITO}_{\text{MIDDLE}}$ (WikiText)	6	0.1B	14.32 (0.12)	76.71 (0.47)	98.56 (0.04)	92.47 (0.19)	71.93 (0.31)	81.32 (0.23)
$\text{DIITO}_{\text{LATE}}$ (WikiText)	6	0.1B	14.93 (0.23)	76.80 (0.34)	98.51 (0.02)	92.36 (0.27)	71.47 (0.28)	81.01 (0.23)
$\text{DIITO}_{\text{FULL}}$ (WikiText)	6	0.1B	13.59 (0.25)	76.67 (0.21)	98.53 (0.04)	92.35 (0.24)	71.96 (0.29)	81.33 (0.25)
$\text{DIITO}_{\text{FULL}}$+Random (WikiText)	6	0.1B	13.95 (0.18)	76.84 (0.29)	98.54 (0.03)	92.41 (0.24)	71.90 (0.54)	81.27 (0.39)
$\text{DIITO}_{\text{FULL}}$+Masked (WikiText)	6	0.1B	13.99 (0.16)	76.80 (0.32)	98.55 (0.03)	92.45 (0.18)	71.77 (0.59)	81.09 (0.42)
$\text{DIITO}_{\text{FULL}}$+$\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ (WikiText)	6	0.1B	13.45 (0.19)	77.14 (0.37)	98.54 (0.04)	92.35 (0.24)	71.94 (0.31)	81.35 (0.23)

Table 1: Performance on the development sets of the WikiText, GLUE benchmark, CoNLL-2003 corpus for the name-entity recognition task, and SQuAD v1.1 for the question answering task. The score is the averaged performance scores with standard deviation (SD) for all tasks across 15 distinct runs. †Numbers are imputed from released models on Hugging-face (Wolf et al., 2020).

(an internal representations) and an appropriate input $\mathbf{x}$, $\textsc{GetVals}(\mathcal{M}, \mathbf{x}, \mathbf{N})$ is the set of values that $\mathbf{N}$ takes on when processing $\mathbf{x}$. In the case that $\mathbf{N}$ represents the neurons corresponding to the final output, $\textsc{GetVals}(\mathcal{M}, \mathbf{x}, \mathbf{N})$ is the output of model $\mathcal{M}$ when processing $\mathbf{x}$ (i.e., output from a standard forward call of a neural model).

SetVals. The SetVals operator is a function generator that defines a new neural model with a computation graph that specifies an intervention on the original model $\mathcal{M}$ (Pearl, 2009; Spirtes et al., 2001). $\textsc{SetVals}(\mathcal{M}, \mathbf{N}, \mathbf{v})$ is the new neural model where the neurons $\mathbf{N}$ are set to constant values $\mathbf{v}$. Because we overwrite neurons with $\mathbf{v}$ in-place, gradients can back-propagate through $\mathbf{v}$.

Interchange Intervention. An interchange intervention combines GetVals and SetVals operations. First, we randomly sample a pair of examples from a training dataset $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2) \in \mathcal{D}$. Next, where $\mathbf{N}$ is the set of neurons that we are targeting for intervention, we define $\mathcal{M}_{\mathbf{N}}^{\mathbf{x}_1}$ to abbreviate the new neural model as follows:

$$\textsc{SetVals}\big(\mathcal{M}, \mathbf{N}, \textsc{GetVals}(\mathcal{M}, \mathbf{x}_1, \mathbf{N})\big) \quad (1)$$

This is the version of $\mathcal{M}$ obtained from setting the values of $\mathbf{N}$ to be those we get from processing input $\mathbf{x}_1$. The interchange intervention targeting $\mathbf{N}$ with $\mathbf{x}_1$ as the source input and $\mathbf{x}_2$ as the base input is then defined as follows:

$$\textsc{IntInv}(\mathcal{M}, \mathbf{N}, \mathbf{x}_1, \mathbf{x}_2) \stackrel{\text{def}}{=} \textsc{GetVals}(\mathcal{M}_{\mathbf{N}}^{\mathbf{x}_1}, \mathbf{x}_2, \mathbf{N}^{\mathbf{y}}) \quad (2)$$

where $\mathbf{N}^{\mathbf{y}}$ are the output neurons. In other words, $\textsc{IntInv}(\mathcal{M}, \mathbf{N}, \mathbf{x}_1, \mathbf{x}_2)$ is the output state we get from $\mathcal{M}$ for input $\mathbf{x}_2$ but with the neurons $\mathbf{N}$ set to the values obtained when processing input $\mathbf{x}_1$.

DIITO. DIITO employs $\mathcal{T}$ as the teacher model, $\mathcal{S}$ as the student model, $\mathcal{D}$ as the training inputs to both models, and Π as an alignment that maps sets of student neurons to sets of teacher neurons. For each set of student neurons $\mathbf{N}_{\mathcal{S}}$ in the domain of Π, we define DIITO loss as:

$$\mathcal{L}_{\text{CE}}^{\text{DIITO}} \stackrel{\text{def}}{=} \sum_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}} \text{CE}_{\text{S}}\Big(\textsc{IntInv}(\mathcal{S}, \mathbf{N}_{\mathcal{S}}, \mathbf{x}_1, \mathbf{x}_2), \textsc{IntInv}(\mathcal{T}, \Pi(\mathbf{N}_{\mathcal{S}}), \mathbf{x}_1, \mathbf{x}_2)\Big) \quad (3)$$

where CE_{S} is the smoothed cross-entropy loss measuring the divergences of predictions, under interchange, between the teacher and the student model.

Distillation Objectives. We adopt the standard distillation objectives from DistilBERT (Sanh et al., 2019) (defined formally in Appendix A.1): $\mathcal{L}_{\text{MLM}}$ for the task-specific loss for the student model, $\mathcal{L}_{\text{CE}}$ for the loss measuring the divergence between the student and teacher outputs on masked tokens, and $\mathcal{L}_{\text{Cos}}$ for the loss measuring the divergence between the student and teacher contextualized representations on masked tokens in the last layer. Our final training objective for the student is a linear combination of the four training objectives reviewed above: $\mathcal{L}_{\text{MLM}}$, $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{Cos}}$, and $\mathcal{L}_{\text{CE}}^{\text{DIITO}}$. In a further experiment, we introduce a fifth objective $\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ which is identical to $\mathcal{L}_{\text{Cos}}$, except the teacher and

student are undergoing interchange interventions (see Appendix A.2 for details).

4 Experimental Set-up

Student and Teacher Models. Our two students have the standard BERT architecture, with 12 heads with a hidden dimension of 768. The larger student has 6 layers, the smaller 3 layers. Our pretrained teacher has the same architecture, except with 12 layers. Following practices introduced by Sanh et al. (2019), we initialize our student model with weights from skipped layers (one out of four layers) in the teacher model. We use WikiText for distillation to simulate a practical situation with a limited computation budget. We leave the exploration of our method on larger datasets for future research.

Alignment. Our teacher and student BERT models create columns of neural representations above each token with each row created by the feed-forward layer of a Transformer block, as in Figure 1. We define $L_{\mathcal{T}}$ and $L_{\mathcal{S}}$ to be the number of layers in the student and teacher, respectively. In addition, we define $\mathcal{S}_i^j$ and $\mathcal{T}_i^j$ to be the representations in the ith row and jth column in the student and teacher, respectively. An alignment Π is a partial function from student representations to sets of teacher representations. We test three alignments:

FULL Π is defined on all student representations:
$\Pi(\mathcal{S}_i^j) = \{\mathcal{T}_{4i+k}^j : 0 \leq k < L_{\mathcal{T}}/L_{\mathcal{S}}\}$

MIDDLE Π is defined for the row $L_{\mathcal{S}} \mathbin{/\!\!/} 2$:
$\Pi(\mathcal{S}_{L_{\mathcal{S}} \mathbin{/\!\!/} 2}^j) = \{\mathcal{T}_{L_{\mathcal{T}} \mathbin{/\!\!/} 2}^j\}$

LATE Π is defined on the student representations in the first and second rows:
$\Pi(\mathcal{S}_1^j) = \{\mathcal{T}_{L_{\mathcal{T}}-2}^j\}$ and $\Pi(\mathcal{S}_2^j) = \{\mathcal{T}_{L_{\mathcal{T}}-1}^j\}$

For each training iteration, we randomly select one aligned student layer to perform the interchange intervention, and we randomly select 30% of token embeddings for alignment for each sequence. We experiment with three conditions with the **FULL** alignment: consecutive tokens (**DIITO$_{\texttt{FULL}}$**), random tokens (**DIITO$_{\texttt{FULL}}$+Random**) and masked tokens (**DIITO$_{\texttt{FULL}}$+Masked**). We also include $\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ to the **FULL** alignment (**DIITO$_{\texttt{FULL}}$+$\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$**).

5 Results

Language Modeling. We first evaluate our models using perplexity on the held-out evaluation data from WikiText. As shown in Table 1, DIITO brings performance gains for all alignments. Our best result is from the **FULL** alignment with the $\mathcal{L}_{\text{Cos}}$ (**DIITO$_{\texttt{FULL}}$+$\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$**), which has -2.24 perplexity compared to standard DistilBERT trained with the same amount of data.

GLUE. The GLUE benchmark (Wang et al., 2018) covers different natural language understanding tasks. We report averaged GLUE scores on the development sets by fine-tuning our distilled models in Table 1. Individual task performance score of each GLUE task is included in Table 2 in the Appendix. The results suggest that distilled models with DIITO lead to consistent improvements over standard DistilBERT trained under the same setting, with our best result (**DIITO$_{\texttt{FULL}}$+$\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$**) being +1.77% higher.

Named Entity Recognition. We also evaluate our models on the CoNLL-2003 Named Entity Recognition task (Tjong Kim Sang and De Meulder, 2003). We report accuracy and Macro-F1 scores along with precision and recall on the development sets. We fine-tune our models for three epochs. Our best performing model (**DIITO$_{\texttt{MIDDLE}}$**) numerically surpasses not only standard DistilBERT (+0.38% on F1 score) trained under the same setting, but also its teacher, BERT$_{\text{BASE}}$ (+0.05% on F1 score). Though these improvements are small, in this case distillation produces a smaller model with *better performance*.

Question Answering. Finally, we evaluate on a question answering task, SQuAD v1.1 (Rajpurkar et al., 2016). We report Exact Match and Macro-F1 on the development sets as our evaluation metrics. We fine-tune our models for two epochs. DIITO again yields marked improvements (Table 1). Our best result is from the vanilla **FULL** alignment (**DIITO$_{\texttt{FULL}}$**), with +2.46% on standard DistilBERT trained under the same setting.

6 Conclusion

In this paper, we explored distilling a teacher by training a student to capture the *causal dynamics* of its computations. Across a wide range of NLP tasks, we find that DIITO leads to improvements, with the largest gains coming from the models that use the richest alignment between student and teacher. Our results also demonstrate that DIITO performs on-par (maintaining 97% of performance on GLUE tasks) with standard DistilBERT (Sanh et al., 2019) while consuming 97% less training data. These findings suggest that DIITO is a

promising tool for effective model distillation.

References

Sander Beckers, Frederick Eberhardt, and Joseph Y. Halpern. 2020. Approximate causal abstractions. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 606–615, Tel Aviv, Israel. PMLR.

Sander Beckers and Joseph Y. Halpern. 2019. Abstracting causal models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2678–2685.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. 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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2020. Amnesic probing: Behavioral explanation with amnesic counterfactuals. In *Proceedings of the 2020 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. 2021. CausaLM: Causal Model Explanation Through Counterfactual Language Models. *Computational Linguistics*, pages 1–54.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021a. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*.

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah D. Goodman, and Christopher Potts. 2021b. Inducing causal structure for interpretable neural networks. ArXiv:2112.00826.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*, 2nd edition. Cambridge University Press, USA.

Reid Pryzant, D. Card, Dan Jurafsky, Victor Veitch, and Dhanya Sridhar. 2021. Causal effects of linguistic properties. In *NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Guy Rotman, Amir Feder, and Roi Reichart. 2021. Model compression for domain adaptation through causal effect estimation. *arXiv preprint arXiv:2101.07086*.

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

Peter Spirtes, Clark N Glymour, and Richard Scheines. 2001. *Causation, Prediction, and Search*, 2nd edition. MIT Press.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4314–4323.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Appendix

A.1 Standard Distillation Objectives

In our setting, our teacher model $\mathcal{T}$ is a BERT model, and our student model $\mathcal{S}$ is a shallower BERT model with fewer layers.

Assume that we randomly draw a training example $(\mathbf{x}_1, \mathbf{y}_1) \in \mathcal{D}$, where $\mathbf{x}_1$ is the input to our models and $\mathbf{y}_1$ is the corresponding ground truth (the token prediction at each masked position). We denote the model predictions (output logits) as $\mathcal{T}(\mathbf{x}_1)$ and $\mathcal{S}(\mathbf{x}_1)$. Additionally, we denote the contextualized representation for tokens for $\mathbf{x}_1$ at the last layer as $\text{BERT}_{\mathcal{T}}(\mathbf{x}_1)$ and $\text{BERT}_{\mathcal{S}}(\mathbf{x}_1)$.

We adopt the three standard distillation objectives of Sanh et al. (2019):

$\mathcal{L}_{\text{MLM}}$ The masked language modeling loss of the student model calculated over all examples using the cross-entropy loss as follows:

$$\sum_{\{\mathbf{x}_1, \mathbf{y}_1\} \in \mathcal{D}} \text{CE}(\mathcal{S}(\mathbf{x}_1), \mathbf{y}_1) \tag{4}$$

$\mathcal{L}_{\text{CE}}$ Following Hinton et al. (2015), the smoothed cross-entropy loss measuring the divergence between the student and teacher outputs as follows:

$$\sum_{\mathbf{x}_1 \in \mathcal{D}} \text{CE}_{\text{S}}(\mathcal{S}(\mathbf{x}_1), \mathcal{T}(\mathbf{x}_1)) \tag{5}$$

$\mathcal{L}_{\text{Cos}}$ The cosine embedding loss defined in terms of the final hidden states of the teacher and the student as follows:

$$\sum_{\mathbf{x}_1 \in \mathcal{D}} \text{Cos}(\text{BERT}_{\mathcal{S}}(\mathbf{x}_1), \text{BERT}_{\mathcal{T}}(\mathbf{x}_1)) \tag{6}$$

As a result, comparing to standard DistilBERT, DIITO essentially adds a new type of objective by pushing the student model to become a *causal abstraction* of the teacher model.

A.2 Causal Distillation Objectives

In addition to our causal loss $\mathcal{L}_{\text{CE}}^{\text{DIITO}}$, we also propose a new loss $\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ which is identical to $\mathcal{L}_{\text{Cos}}$ with interchange interventions. In this section, we provide a formal definition for $\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$.

We denote our teacher and student models as $\mathcal{T}$ and $\mathcal{S}$ respectively. Using the notational conventions from Section 3, we use $\mathbf{N}_{\mathcal{T}}^{\mathbf{y}}$ and $\mathbf{N}_{\mathcal{S}}^{\mathbf{y}}$ to represent the neurons corresponding to the final output for each model. Likewise, we use $\mathbf{N}_{\mathcal{T}}^{L_{\mathcal{T}}}$ and $\mathbf{N}_{\mathcal{S}}^{L_{\mathcal{S}}}$ to represent the neurons representing contextualized representation for each token after the final BERT layer.

Assuming we randomly sample a pair of examples from a training dataset $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2) \in \mathcal{D}$, we can then rewrite our causal loss $\mathcal{L}_{\text{CE}}^{\text{DIITO}}$ by rearranging Eqn. 2 and Eqn. 3 as follows:

$$\sum_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}} \text{CE}_{\text{S}}\Big(\begin{array}{c}\textsc{GetVals}(\mathcal{M}_{\mathbf{S}}^{\mathbf{x}_1}, \mathbf{x}_2, \mathbf{N}_{\mathcal{S}}^{\mathbf{y}}), \\ \textsc{GetVals}(\mathcal{M}_{\mathbf{T}}^{\mathbf{x}_1}, \mathbf{x}_2, \mathbf{N}_{\mathcal{T}}^{\mathbf{y}})\end{array}\Big) \tag{7}$$

where $\mathcal{M}_{\mathbf{S}}^{\mathbf{x}_i}$ and $\mathcal{M}_{\mathbf{T}}^{\mathbf{x}_i}$ are derived as in Eqn. 1 for each model respectively. Crucially, Eqn. 7 can be regarded as the *causal* form of the standard smoothed cross-entropy loss with interchange intervention. Likewise, we can further define the $\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ as:

$$\sum_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}} \text{COS}\Big(\begin{array}{c}\textsc{GetVals}(\mathcal{M}_{\mathbf{S}}^{\mathbf{x}_1}, \mathbf{x}_2, \mathbf{N}_{\mathcal{S}}^{L_{\mathcal{S}}}), \\ \textsc{GetVals}(\mathcal{M}_{\mathbf{T}}^{\mathbf{x}_1}, \mathbf{x}_2, \mathbf{N}_{\mathcal{T}}^{L_{\mathcal{T}}})\end{array}\Big) \tag{8}$$

with adjusted interchange alignments for $\mathbf{N}_{\mathcal{T}}^{L_{\mathcal{T}}}$ and $\mathbf{N}_{\mathcal{S}}^{L_{\mathcal{S}}}$.

A.3 Distillation Set-up

We adapt the open-source Hugging-face implementation for model distillation (Wolf et al., 2020).[2]

[2] https://github.com/huggingface/transformers

Model	Layers	Pretraining Tokens	General Language Understanding Evaluation (GLUE) CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019b) (Wikipedia+BookCorpus)	12	3.3B	56.30	84.70	88.60	91.80	89.60	69.30	92.70	89.00
DistilBERT (Sanh et al., 2019) (Wikipedia+BookCorpus)	6	3.3B	51.30	82.10	87.50	89.20	88.50	59.90	91.30	86.90
DistilBERT (WikiText)	3	0.1B	22.78	71.55	82.51	82.12	82.16	55.43	86.47	56.33
$\text{DIITO}_{\text{MIDDLE}}$ (WikiText)	3	0.1B	23.21	72.97	82.81	83.15	82.83	55.98	86.52	66.93
$\text{DIITO}_{\text{LATE}}$ (WikiText)	3	0.1B	24.12	72.80	82.16	82.88	82.85	57.29	87.31	62.65
$\text{DIITO}_{\text{FULL}}$ (WikiText)	3	0.1B	25.01	72.85	82.71	83.05	82.85	55.37	86.92	66.15
DistilBERT (WikiText)	6	0.1B	40.43	78.95	87.45	84.76	84.96	60.10	89.38	80.40
$\text{DIITO}_{\text{MIDDLE}}$ (WikiText)	6	0.1B	43.97	79.47	87.57	85.45	85.21	60.72	89.97	81.33
$\text{DIITO}_{\text{LATE}}$ (WikiText)	6	0.1B	43.93	79.49	87.70	85.79	85.22	60.14	90.31	81.79
$\text{DIITO}_{\text{FULL}}$ (WikiText)	6	0.1B	43.43	79.66	88.17	85.57	85.28	59.95	90.01	81.26
$\text{DIITO}_{\text{FULL}}$+Random (WikiText)	6	0.1B	44.27	79.70	88.06	85.63	85.34	60.89	89.76	81.08
$\text{DIITO}_{\text{FULL}}$+Masked (WikiText)	6	0.1B	43.39	79.63	87.88	85.61	85.30	61.06	89.97	81.58
$\text{DIITO}_{\text{FULL}}$+$\mathcal{L}_{\text{Cos}}^{\text{DIITO}}$ (WikiText)	6	0.1B	45.17	79.68	88.18	85.83	85.31	60.94	90.32	81.69

Table 2: Model performance results on the development sets of the GLUE benchmark. The GLUE score is the averaged performance scores across 15 distinct runs with precision aligned for a fair comparison. Following the evaluation for BERT (Devlin et al., 2019b), we exclude WNLI for evaluation.

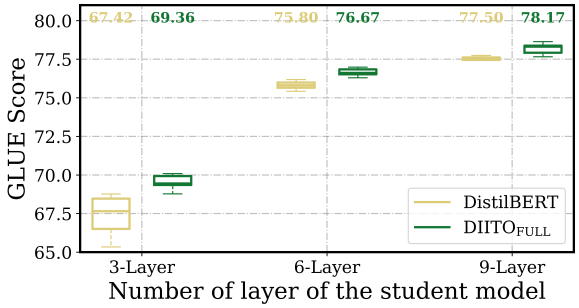

Figure 2: GLUE score distribution across 15 distinct runs of students in different sizes. Following the evaluation for BERT (Devlin et al., 2019b). we exclude WNLI for evaluation.

We distill our models on the MLM pretraining task (Devlin et al., 2019b). We use large gradient accumulations over batches as in Sanh et al. (2019) for better performance. Specifically, we distill all models for three epochs for an effective batch size of 240. In contrast to the setting of 4K per batch in the BERT (Devlin et al., 2019b) and DistilBERT Sanh et al. (2019) models, we found that small effective batch size works better for smaller dataset. We weight all objectives equally for all experiments. With our new objectives, the distillation takes approximately 9 hours on 4 NVIDIA A100 GPUs.

A.4 Evaluation Set-up

GLUE We fine-tune for 25 epochs for the smaller datasets (RTE and CoLA) and 3 epochs for the others. Following Devlin et al. (2019b) and Sanh et al. (2019), we use Matthew's Correlation for CoLA, F1 for MRPC and QQP, Spearman correlation for STS-B, and accuracy for all the other tasks in GLUE.

A.5 Reproducibility

To foster reproducible and provide a fair comparison between methods, we distill BERT for each condition with three distinct random seeds. We then fine-tune each model with five distinct random seeds. Consequently, we report results aggregated from three distinct runs for the language modeling task, and 15 distinct runs for others.

Named Entity Recognition We follow the experiment set-up as in Hugging-face (Wolf et al., 2020) repository for evaluation for the CoNLL-2003 Named Entity Recognition task (Tjong Kim Sang and De Meulder, 2003). For fine-tuning, we set the learning rate to $5e^{-5}$ with an effective batch size of 32 for three epochs.[3]

Question Answering We follow the experiment set-up as in Sanh et al. (2019) for evaluation for the question answering task, SQuAD v1.1 (Rajpurkar et al., 2016). For fine-tuning, we set the learning rate to $3e^{-5}$ with an effective batch size of 48 for two epochs. We set the stride to 128.

[3]For DistilBERT performance in Table 1on CoNLL-2003, we evaluate with a publicly avaliable model downloaded from https://huggingface.co/delpart/distilbert-base-uncased-finetuned-ner.

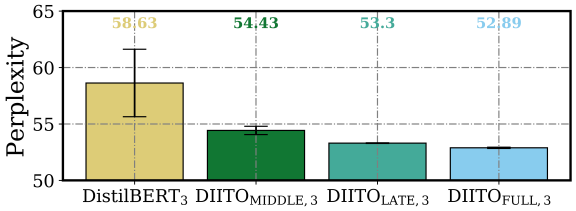

Figure 3: Perplexity score distribution for the development set of WikiText of models trained in a low-resource setting. The best model is the one with the richest alignment structure.

A.6 Low-Resource Model Distillation

We experiment with an extreme case in a low-resource setting where we only distill with 15% of WikiText by keeping other experiment set-up constant. Our results suggest that DIITO training is also beneficial in extremely low-resource settings as shown in Figure 3.

A.7 Layer-wise Ablation

We further study the effect of DIITO training with respect to the size of the student model through a layer-wise ablation experiment. As shown in Figure 2, we compare GLUE performance for models trained with standard distillation pipeline and with DIITO training (**DIITO$_{\texttt{FULL}}$**). Specifically, we compute the averaged GLUE scores following the same procedure described in Section A.5. Our results suggest that DIITO training brings consistent improvements over GLUE tasks with smaller models marking the greatest gains.

Algorithm 1 Causal Distillation via Interchange Intervention Training

Require: Student model $\mathcal{S}$, teacher model $\mathcal{T}$, student output neurons $\mathbf{N}_{\mathcal{S}}^{\mathbf{y}}$, alignment Π, shuffled training dataset $\mathcal{D}$.

$\mathcal{S}$.train()
$\mathcal{T}$.eval()
$\mathcal{D}'$ = random.shuffle($\mathcal{D}$)
$\mathbf{N}_{\mathcal{T}}^{\mathbf{y}} = \Pi(\mathbf{N}_{\mathcal{S}}^{\mathbf{y}})$
while not converged **do**
  for $\{\mathbf{x}_1, \mathbf{y}_1\}$, $\{\mathbf{x}_2, \mathbf{y}_2\}$ **in** iter($\mathcal{D}$, $\mathcal{D}'$) **do**
    $\mathbf{N}_{\mathcal{S}}$ = sample_student_neurons()
    $\mathbf{N}_{\mathcal{T}} = \Pi(\mathbf{N}_{\mathcal{S}})$
    with no_grad:
      $\mathcal{T}_a = \textsc{SetVals}($
        $\mathcal{T}, \mathbf{N}_{\mathcal{T}}, \textsc{GetVals}(\mathcal{T}, \mathbf{x}_1, \mathbf{N}_{\mathcal{T}}))$
      $o_{\mathcal{T}} = \textsc{GetVals}(\mathcal{T}_a, \mathbf{x}_2, \mathbf{N}_{\mathcal{T}}^{\mathbf{y}})$
    $\mathcal{S}_a = \textsc{SetVals}($
      $\mathcal{S}, \mathbf{N}_{\mathcal{S}}, \textsc{GetVals}(\mathcal{S}, \mathbf{x}_1, \mathbf{N}_{\mathcal{S}}))$
    $o_{\mathcal{S}} = \textsc{GetVals}(\mathcal{S}_a, \mathbf{x}_2, \mathbf{N}_{\mathcal{S}}^{\mathbf{y}})$
    $\mathcal{L}^{\text{DIITO}} =$ **get_loss**$(o_{\mathcal{T}}, o_{\mathcal{S}})$
    Calculate $\mathcal{L}_{\text{MLM}}$, $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{Cos}}$
    $\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{Cos}} + \mathcal{L}^{\text{DIITO}}$
    $\mathcal{L}$.backward()
    Step optimizer
end while