Causal Distillation for Language Models

Anonymous ACL submission

Abstract

Distillation efforts have led to language models that are more compact and efficient without serious drops in performance. The standard distillation method trains a student model against two objectives: a task-specific objective (e.g., language modeling) and an imitation objective that seeks to make the hidden states of the student model more like those of the larger teacher model. In this paper, we show that it is effective to augment these objectives with a third objective by optimizing the student model to imitate the *causal* computation process of the teacher model through *interchange intervention training* (IIT). IIT pushes the student model to become a *causal abstraction* of the teacher model – a simpler model with the same causal structure. IIT is fully differentiable, easily implemented, and combines flexibly with other objectives. Compared with standard distillation of BERT, distillation via IIT results in lower perplexity on Wikipedia (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 contributed to impressive performance improvements across a wide range of tasks in NLP, but they have also brought increased costs due to their very large size. Recent *distillation* efforts have sought to reduce these costs while maintaining the benefits that these models can bring (Hinton et al., 2015; Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2019).

Hinton et al. (2015) propose model distillation by training the sudent model (i.e., a shallow network) with an additional objective to match output behavior (e.g., output logits) with the teacher model (i.e., a dense network) while supervised by a task-specific objective (e.g.,sequence classification). Sanh et al. (2019); Sun et al. (2019); Jiao et al. (2019) adapt this method, and try to strength it by adding additional supervised signals to match internal representations between two models. However, these approaches may push the student model to match all aspects of internal states of the teacher model even if these aspects have no *causal* role in the computation performed by the teacher model. We thus introduce a method that aims to match the *causal* role of representations between the student and the teacher model.

We propose to augment previous objectives with a new one that pushes the student model to become a *causal abstraction* of the teacher model – a simpler model that has approximately the same overall *causal* structure (Beckers and Halpern, 2019; Beckers et al., 2020; Geiger et al., 2020). To achieve this, we employ the *interchange intervention training* (IIT) method of **?**. In IIT, one aligns a high-level *causal* model with a low-level neural model and then performs *interchange interventions* (swapping of aligned internal states) on the neural model during training, guided by the output behavior of the high-level model. This has the effect of pushing the low-level model to conform to the *causal* dynamics of the high-level model.

Adapting IIT to model distillation is straightforward: in addition to standard distillation objectives, we use IIT to push the student model to match the *causal* dynamics of the teacher model. Figure 1 shows a schematic example of how this happens. 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 creates two new examples that are shaped in part by their original inputs and in part by these other hidden states. Note that this can be interpreted as a certain kind of counterfactual: what would the output have been for the sentence "I ate some ⟨MASK⟩." had the activations in the middle two layers, second token, been what they would have been for the sentence "The water ⟨MASK⟩ solid."? The IIT distillation objective then pushes the student model to output the same logits as the teacher – matching the counterfactual outputs.

To assess the contribution of distillation with IIT, we begin with BERT-base (Devlin et al., 2019a) and distill it under various alignments between student and teacher, and we assess the results in a language modeling task (Wikipedia), the GLUE benchmark, SQuAD, and CoNLL-2003 name-entity recognition. All of the alignments we explore lead to marked improvements across all these tasks as compared to standard distillation, and we see the best results with the richest alignment we explore.[1]

2 Related Work

Pretrained Model Compression Numerous methods have been developed for compressing large-scale pretrained language models, including architecture pruning (Cui et al., 2019; McCarley, 2019), weight sharing and compression (Dehghani et al., 2018; Ma et al., 2019), knowledge distillation (Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2019), and quantization (Shen et al., 2020).

Distillation 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 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 adaption. In contrast, we focus on learning the *causal* structure of the teacher through interventions on hidden representations.

Causal Interventions on Neural Networks Causal interventions on neural networks were originally developed 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). **?** extend these methods to the optimization process. The central contribution of the current paper is adapting this optimization method to language model distillation.

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

3 Causal Distillation via Interchange Intervention Training

We first describe the standard distillation method for BERT compression, adapted from Sanh et al. (2019), and then we present our new distillation objective with IIT.

3.1 Standard Distillation

In our settings, our teacher model $\mathcal{T}$ is s a BERT model, and our student model $\mathcal{S}$ is a shallow BERT model with fewer layers. Assuming we randomly draw a training example $\{\mathbf{x}_i, \mathbf{y}_i\}$ for $i \in [1, |\mathcal{D}|]$ from the training dataset $\mathcal{D}$, where $\mathbf{x}_i$ is the i-th input to our models and $\mathbf{y}_i$ is the corresponding ground-truth (e.g., token prediction at each masked position), we denote the model predictions (i.e., output logits) as $\mathcal{T}(\mathbf{x}_i)$ and $\mathcal{S}(\mathbf{x}_i)$ for two models respectively. Additionally, we denote the contextualized representation for tokens for $\mathbf{x}_i$ at the last layer as $\text{BERT}_{\mathcal{T}}(\mathbf{x}_i)$ and $\text{BERT}_{\mathcal{S}}(\mathbf{x}_i)$ for two models respectively.

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:

$$\sum_{\{\mathbf{x}_i,\mathbf{y}_i\}\in\mathcal{D}} \text{CE}(\mathcal{S}(\mathbf{x}_i), \mathbf{y}_i) \tag{1}$$

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

$$\sum_{\mathbf{x}_i\in\mathcal{D}} \text{CE}_{\text{S}}(\mathcal{S}(\mathbf{x}_i), \mathcal{T}(\mathbf{x}_i)) \tag{2}$$

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

$$\sum_{\mathbf{x}_i\in\mathcal{D}} \text{COS}(\text{BERT}_{\mathcal{S}}(\mathbf{x}_i), \text{BERT}_{\mathcal{T}}(\mathbf{x}_i)) \tag{3}$$

3.2 Interchange Intervention Training

In this section, we formally define our distillation training procedure and provide a simplified implementation in Algorithm 1. Our GET and DO operators are inspired by those of Pearl (2001).

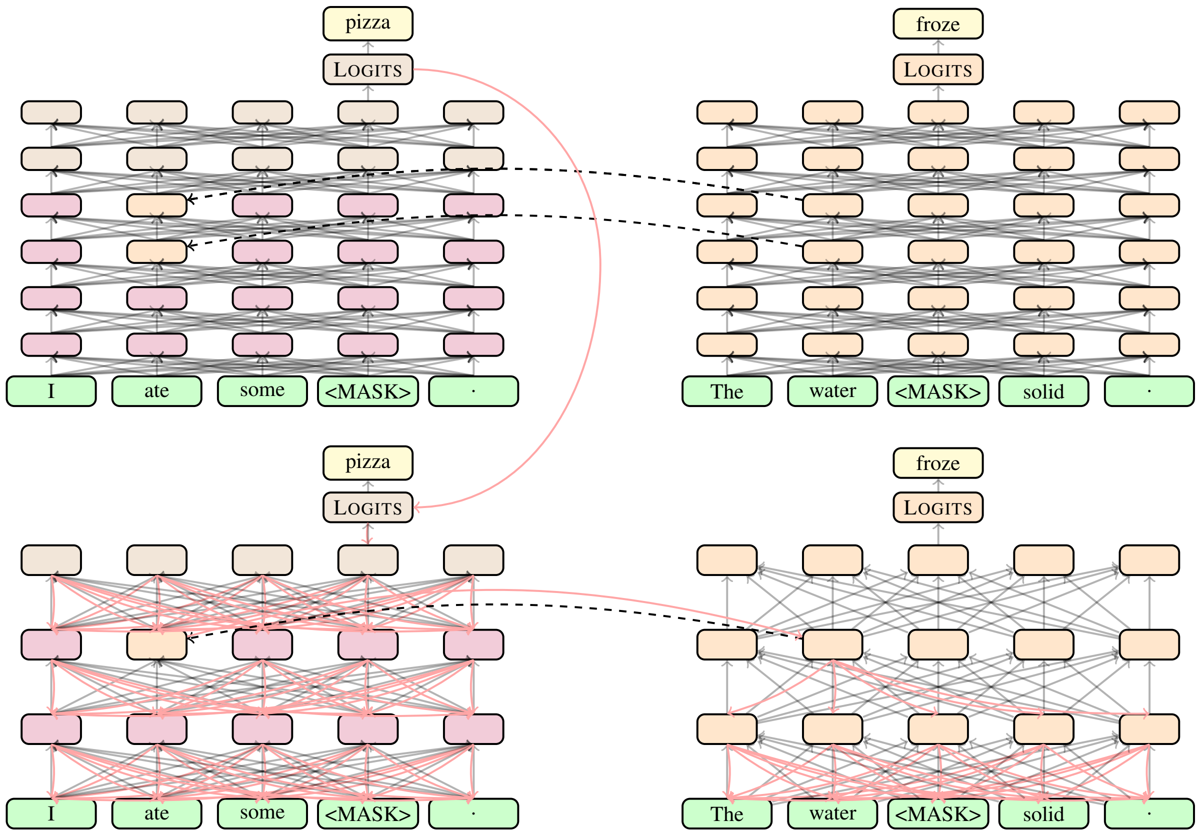

Figure 1: A IIT update in the context of masked language modelling (MLM). The teacher network (top) has 6 layers and the student (bottom) has 3 layers, and we align layer 2 in the student with layers 3-4 in the teacher. Solid lines are feed-forward connections, red lines show the flow of backpropagation, and dashed lines indicate interchange interventions. In this case, we can suppose that the student originally predicted some logits $\mathbf{y}$ and, in turn, some other token (say, "lettuce") after the intervention. IIT replaces $\mathbf{y}$ with the logits from the teacher model after the intervention, which has the effect of updating the student weights to conform to these teacher outputs. The weights below the interchange intervention are updated twice (i.e, once for each input).

The $\texttt{GET}$ Operator The $\texttt{GET}$ operator is an activation value retriever for a neural model. Consider a neural model $\mathcal{M}$ containing a set of neurons $\mathbf{N}$ (a set of internal representations or subparts of representation) and an appropriate input $\mathbf{x}_i$, $\texttt{GET}(\mathcal{M}, \mathbf{x}_i, \mathbf{N})$ is the set of values that $\mathbf{N}$ takes on when processing $\mathbf{x}_i$. In the case that $\mathbf{N}^{\mathbf{y}}$ represents the neurons corresponding to the final output, $\texttt{GET}(\mathcal{M}, \mathbf{x}_i, \mathbf{N}^{\mathbf{y}})$ is the output of model $\mathcal{M}$ when processing $\mathbf{x}_i$ (i.e., output from a standard forward call of any neural model).

The $\texttt{DO}$ Operator The $\texttt{DO}$ operator is a function generator that defines a new neural model with a computation graph that specifies an intervention on the original model $\mathcal{M}$. $\texttt{DO}(\mathcal{M}, \mathbf{N}, \mathbf{v})$ is the new neural model where the neurons $\mathbf{N}$ are set to constant values $\mathbf{v}$.

Interchange Intervention An interchange interventions is a simple combination of $\texttt{GET}$ and $\texttt{DO}$ operations. First, we randomly draw two training examples $\{\mathbf{x}_i, \mathbf{y}_i\}$ and $\{\mathbf{x}_j, \mathbf{y}_j\}$ for $i, j \in [1, |\mathcal{D}|]$ from the training dataset $\mathcal{D}$ independently. Next, where $\mathbf{N}$ is the set of neurons that we are targeting for intervention, we use $\mathcal{M}_{\mathbf{N}}^{\mathbf{x}_i}$ to abbreviate the new neural model as follows:

$$\texttt{DO}\big(\mathcal{M}, \mathbf{N}, \texttt{GET}(\mathcal{M}, \mathbf{x}_i, \mathbf{N})\big) \tag{4}$$

This is the version of $\mathcal{M}$ obtained from setting the values of $\mathbf{N}$ to be those we get from processing input $\mathbf{x}_i$. The interchange intervention targeting $\mathbf{N}$ with $\mathbf{x}_i$ as the source input and $\mathbf{x}_j$ as the base input[2] is then defined as follows:

$$\textsc{IntInv}(\mathcal{M}, \mathbf{N}, \mathbf{x}_i, \mathbf{x}_j) \stackrel{\text{def}}{=} \texttt{GET}(\mathcal{M}_{\mathbf{N}}^{\mathbf{x}_i}, \mathbf{x}_j, \mathbf{N}^{\mathbf{y}}) \tag{5}$$

where $\mathbf{N}^{\mathbf{y}}$ is the output neurons for $\mathcal{M}$. In other words, $\textsc{IntInv}(\mathcal{M}, \mathbf{N}, \mathbf{x}_i, \mathbf{x}_j)$ is the output state

[2] For simplicity, standard distillation objectives are not calculated over the base input.

Algorithm 1 Causal Distillation via Interchange Intervention Training

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

$\mathcal{S}$.train()
$\mathcal{T}$.eval()
$\mathcal{D}'$ = random.shuffle($\mathcal{D}$)
$\mathbf{N}_\mathcal{T} = \Pi(\mathbf{N}_\mathcal{S})$
$\mathbf{N}^\mathbf{y}_\mathcal{T} = \Pi(\mathbf{N}^\mathbf{y}_\mathcal{S})$
while not converged **do**
  // input and output pairs
  for $\{\mathbf{x}_i, \mathbf{y}_i\}$, $\{\mathbf{x}_j, \mathbf{y}_j\}$ **in** iter($\mathcal{D}$, $\mathcal{D}'$) **do**
    with no_grad:
      $\mathcal{T}_a = \texttt{DO}(\mathcal{T}, \mathbf{N}_\mathcal{T}, \texttt{GET}(\mathcal{T}, \mathbf{x}_i, \mathbf{N}_\mathcal{T}))$
      $o_\mathcal{T} = \texttt{GET}(\mathcal{T}_a, \mathbf{x}_j, \mathbf{N}^\mathbf{y}_\mathcal{T})$
    $\mathcal{S}_a = \texttt{DO}(\mathcal{S}, \mathbf{N}_\mathcal{S}, \texttt{GET}(\mathcal{S}, \mathbf{x}_i, \mathbf{N}_\mathcal{S}))$
    $o_\mathcal{S} = \texttt{GET}(\mathcal{S}_a, \mathbf{x}_j, \mathbf{N}^\mathbf{y}_\mathcal{S})$
    $\mathcal{L}_\text{Causal} =$ **get_loss**$(o_\mathcal{T}, o_\mathcal{S})$
    Calculate losses for other objectives
    $\mathcal{L} = \mathcal{L}_\text{MLM} + \mathcal{L}_\text{CE} + \mathcal{L}_\text{Cos} + \mathcal{L}_\text{Causal}$
    $\mathcal{L}$.backward()
    Update model parameters with gradients
end while

we get from $\mathcal{M}$ for example $\mathbf{x}_j$ but with the neurons $\mathbf{N}$ set to the values obtained when processing $\mathbf{x}_i$.

IIT Objective The IIT objective 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 $\mathbf{N}_S$ in the domain of Π, we define the IIT objective as follows:

$$\mathcal{L}_\text{Causal} \stackrel{\text{def}}{=} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}} \text{CE}_\text{S}\Big(\textsc{IntInv}(\mathcal{S}, \mathbf{N}_\mathcal{S}, \mathbf{x}_i, \mathbf{x}_j), \textsc{IntInv}(\mathcal{T}, \Pi(\mathbf{N}_\mathcal{S}), \mathbf{x}_i, \mathbf{x}_j)\Big) \quad (6)$$

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

Alignment Our teacher and student BERT models can both be understood as having columns of neural representations above each token, with L rows (layer) and M columns (sequence length), as in Figure 1.[3] For alignment Π, we map student representations at selected row $a \in [1, L_\mathcal{S}]$ to the teacher representations at selected rows $b \in [1, L_\mathcal{T}]$ through $c \in [1, L_\mathcal{T}]$. Additionally, we may select multiple rows in the student model for different alignments in the teacher model. In case of multiple alignments, we randomly select one alignment at each training iteration for intervention.

We experiment with three different alignments:

`FULL` Each layer a in the student is aligned with layers $(a-1) \times 4 + 1$ to $a \times 4 + 1$ in the teacher.

`MIDDLE` The single middle layer $a = L_\mathcal{S}$ // 2 in the student is aligned with the single middle layer $b = L_\mathcal{T}$ // 2 in the teacher.

`LATE` The first layer in the student is aligned with layer $b = L_\mathcal{T} - 2$ in the teacher, and the second layer in the student is aligned with the second to last layer $c = L_\mathcal{T} - 1$ in the teacher.

For each, we align neurons after the feed-forward layer at each Transformer block.[4] For each training iteration, we randomly select one aligned student layer to perform interchange intervention, and we randomly select 30% of token embeddings for alignment for each sequence. For simplicity, we select consecutive tokens.

Distillation Objectives Our final training objective for the student is a linear combination of four training objectives including $\mathcal{L}_\text{MLM}$, $\mathcal{L}_\text{Actual}$, $\mathcal{L}_\text{Cos}$, and $\mathcal{L}_\text{Causal}$.[5] We further discuss formal connections between our IIT objective and standard distillation objectives in Appendix A.1.

4 Experimental Set-up

Student Model Our student has the standard BERT architecture, with 3 layers and 12 heads with a hidden dimension of 768.[6] Following practices introduced by Sanh et al. (2019), we initialize our student model with weights from skipped layers (i.e., one out of four layers) in the teacher model.

Distillation We adapt the open-source Huggingface implementation for model distillation (Wolf

[3]We use subscripts to differentiate the rows and columns for the teacher (i.e., $L_\mathcal{T}$ and $M_\mathcal{T}$) and the student (i.e., $L_\mathcal{S}$ and $M_\mathcal{S}$) models.

[4]Different mappings may result in different computational costs for new gradient computation graph.

[5]Note that all objectives listed here can be expressed by our `GET` and `DO` operators as in Appendix A.1.

[6]The standard distilBERT introduced by Sanh et al. (2019) has the same architecture but with 6 layers. Here, we experiment with a more extreme case with a smaller model.

Model	**Score**	CoLA	MNLI	MNLI-mm	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
Standard	67.81	22.78	71.55	72.75	78.17	82.12	84.27	55.43	86.47	56.73	**24.23**
MIDDLE	69.63	23.21	**72.97**	**73.98**	**78.75**	**83.15**	84.85	55.98	86.52	**67.23**	23.94
LATE	69.35	24.12	72.80	73.85	77.96	82.88	**84.88**	**57.29**	**87.31**	63.03	21.41
FULL	**69.66**	**25.01**	72.85	73.78	78.59	83.05	84.85	55.37	86.92	66.51	21.50

Table 1: Model performance results on the development sets of the GLUE benchmark. The GLUE score is the average of all performance scores excluding WNLI. Numbers with the best aggregated performance are bolded.

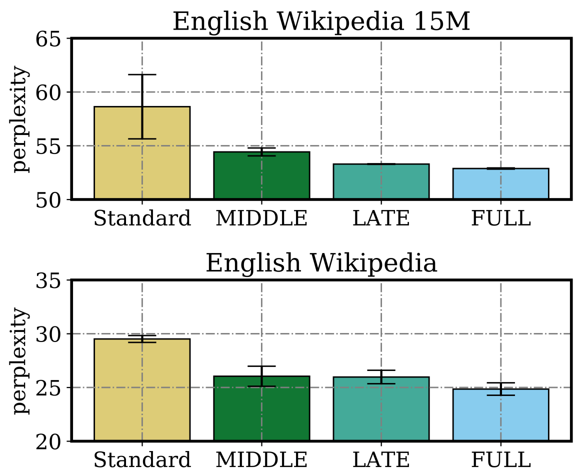

Figure 2: Perplexity scores with standard deviations for the development set of the `English Wikipedia` corpus after three training epochs. The best models are the ones with the richest alignment structure (FULL), in the full and low-resource distillation settings.

et al., 2020).[7] 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. We use the `English Wikipedia` corpus for distillation. Additionally, we experiment with a low-resource case where we only distill with 15% of the `English Wikipedia` corpus. For fair comparison, we distill all models for three epochs. We weight all objectives equally for all experiments. With our new objectives, the distillation takes about 90 hours on 4 NVIDIA Titan 12G GPUs.

5 Experiments

In this section, we compare our IIT distilled BERT with standard distilled BERT across multiple benchmarks. To ensure 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

[7] `https://github.com/huggingface/transformers`

Model	CoNLL-2003 acc./F1	SQuAD EM/F1
Standard	97.93/89.15	56.23/68.26
MIDDLE	98.01/89.75	58.64/70.33
LATE	97.98/89.21	58.79/70.56
FULL	**98.01/89.85**	**59.33/71.05**

Table 2: Model performance results on the development sets of the CoNLL-2003 corpus for the name-entity recognition task, and SQuAD v1.1 for the question answering task. Numbers with the best performance are bolded.

distinct runs for the language modeling task, and 15 distinct runs for others.

Language Modeling We first evaluate our models using perplexity on the held-out evaluation data from the `English Wikipedia` corpus. As shown in Figure 2, our IIT objective brings performance gains for all alignments. IIT training is also beneficial in a low-resource setting (top panel). Additionally, we find that more complete alignments result in lower perplexity (2.85% for the low-resource setting and 13.7% for the full corpus). This suggests that even richer alignments might lead to even larger gains.

GLUE The GLUE benchmark (Wang et al., 2018) covers nine different NLP tasks. We report scores on the development sets for each task by fine-tuning our distilled models. We fine-tune for 15 epochs for the smaller datasets (WNLI, RTE and CoLA) and 3 epochs for the others. We use Matthew's Correlation for CoLA, the mean of accuracy and F1 for MRPC and QQP, the mean of Pearson and Spearman correlation for STS-B, and accuracy for all the other datasets.

Our GLUE results are summarized in Table 1 along with the macro-score (average of individual scores, with WNLI left out to allow comparisons with Sanh et al. 2019). The results suggest that distilled models with the IIT leads to consistent improvements over standard distillation, except for the WNLI task. Overall, IIT with the `FULL` mapping brings an average of 2.72% improvement.

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. Overall, IIT brings small but consistent improvements, as see in Table 2.

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 metrices. We fine-tune our models for two epochs. IIT again yields marked improvements (Table 2).

6 Conclusion

In this paper, we explored distilling a teacher by training a student to capture the *causal* structure of its computations. Across a wide range of NLP tasks, we find that IIT training leads to improvements, with the largest gains coming from the models that use the richest alignment between student and teacher. These findings suggest that IIT 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.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune bert with sparse self-attention mechanism. 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 3548–3553.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *International Conference on Learning Representations*.

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, 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.

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*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in Neural Information Processing Systems*, 32:2232–2242.

J Scott McCarley. 2019. Pruning a bert-based question answering model. *arXiv preprint arXiv:1910.06360*, 142.

Judea Pearl. 2001. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page

411–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

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 Distillation Objectives

In this section, we show that standard distillation objectives can be written using our two operators defined in section 3.2. We denote our teacher and student models as $\mathcal{T}$ and $\mathcal{S}$ respectively. Following our notations in section 3.2, we use $\mathbf{N}^{\mathbf{y}}_{\mathcal{T}}$ and $\mathbf{N}^{\mathbf{y}}_{\mathcal{S}}$ to represent the neurons corresponding to the final output for each model respectively. Likewise, we use $\mathbf{N}^{L_{\mathcal{T}}}_{\mathcal{T}}$ and $\mathbf{N}^{L_{\mathcal{S}}}_{\mathcal{S}}$ to represent the neurons representing contextualized representation for each token after the final BERT layer.

Assuming we randomly draw a training example $\{\mathbf{x}_i, \mathbf{y}_i\}$ for $i \in [1, |\mathcal{D}|]$ from the training dataset $\mathcal{D}$, where $\mathbf{x}_i$ is the i-th input to our models and $\mathbf{y}_i$ is the corresponding ground-truth (e.g., token prediction at each masked position), we can then write the three standard distillation objectives of Sanh et al. (2019) as follows:

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

$$\sum_{\{\mathbf{x}_i,\mathbf{y}_i\}\in\mathcal{D}} \text{CE}(\texttt{GET}(\mathcal{S}, \mathbf{x}_i, \mathbf{N}^{\mathbf{y}}_{\mathcal{S}}), \mathbf{y}_i) \tag{7}$$

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

$$\sum_{\mathbf{x}_i\in\mathcal{D}} \begin{aligned}\text{CE}_{\text{S}}(&\texttt{GET}(\mathcal{S}, \mathbf{x}_i, \mathbf{N}^{\mathbf{y}}_{\mathcal{S}}), \\ &\texttt{GET}(\mathcal{T}, \mathbf{x}_i, \mathbf{N}^{\mathbf{y}}_{\mathcal{T}}))\end{aligned} \tag{8}$$

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

$$\sum_{\mathbf{x}_i\in\mathcal{D}} \begin{aligned}\text{COS}(&\texttt{GET}(\mathcal{S}, \mathbf{x}_i, \mathbf{N}^{L_{\mathcal{S}}}_{\mathcal{S}}), \\ &\texttt{GET}(\mathcal{T}, \mathbf{x}_i, \mathbf{N}^{L_{\mathcal{T}}}_{\mathcal{T}}))\end{aligned} \tag{9}$$

We can further substantiate our new objective as in Eqn. 6 with our operators as follows:

$$\sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}} \text{CE}_{\text{S}}\Big(\begin{aligned}&\texttt{GET}(\mathcal{M}^{\mathbf{x}_i}_{\mathbf{S}}, \mathbf{x}_j, \mathbf{N}^{\mathbf{y}}_{\mathcal{S}}), \\ &\texttt{GET}(\mathcal{M}^{\mathbf{x}_i}_{\mathbf{T}}, \mathbf{x}_j, \mathbf{N}^{\mathbf{y}}_{\mathcal{T}})\end{aligned}\Big) \tag{10}$$

where $\mathcal{M}_{\mathbf{S}}^{\mathbf{x}_i}$ and $\mathcal{M}_{\mathbf{T}}^{\mathbf{x}_i}$ are derived as in Eqn. 4 for each model respectively. Crucially, Eqn. 10 can be regarded as the *causal* form of the standard smoothed cross-entropy loss as in Eqn. 8 with intervened computation graph for two original models. Likewise, our causal distillation objective can be applied to any other losses. We left this for future research.