Gradient Sparsification For *Masked Fine-Tuning* of Transformers

Anonymous ACL submission

Abstract

Fine-tuning masked language models is widely adopted for transfer learning to downstream tasks and can be achieved by (1) freezing gradients of the pretrained network or only updating gradients of a newly added classification layer or (2) performing gradient updates on all parameters. Gradual unfreezing trades off between the two by gradually unfreezing gradients of whole layers during training. We propose to extend this to *stochastic gradient masking* to regularize pretrained language models for improved fine-tuning performance. We introduce *GradDrop* and variants thereof, a class of gradient sparsification methods that mask gradients prior to gradient descent. Unlike gradual unfreezing which is non-sparse and deterministic, GradDrop is sparse and stochastic. Experiments on the multilingual XGLUE benchmark with XLM-R_{Large} show that *GradDrop* outperforms standard fine-tuning and gradual unfreezing, while being competitive against methods that use additional translated data and intermediate pretraining. Lastly, we identify cases where largest zero-shot performance gains are on less resourced languages.

1 Introduction

Fine-tuning pretrained transformer models for downstream tasks has been the defacto standard in natural language processing due to the recent successes of large-scale masked language modeling (Radford et al., 2018; Devlin et al., 2019; Lample and Conneau, 2019; Conneau et al., 2020a). This is usually achieved in one of two ways: (1) freeze the gradients of the pretrained portion of the network and perform stochastic gradient descent (SGD) on a newly added task-specific layer/s or (2) perform SGD on both the pretrained and newly added layer/s. However, freezing all gradients of the pretrained layers can be too restrictive, particularly when the downstream task is dissilmilar to the task of language modeling used during pretraining (Peters et al., 2019). In contrast, unfreezing all layers may lead to negative transfer whereby irrelevant features are tuned for a downstream task or stability issues may arise when performing SGD for a large number of parameters (Liu et al., 2020). While gradual unfreezing (Howard and Ruder, 2018) reduces training time (i.e less gradient updates) by consecutively unfreezing k layers from top to bottom during fine-tuning, it is deterministic and turns off the gradient flow in whole layers which is a strong constraint. Gradual unfreezing could benefit from sparse gradient dropout alternatives that allow at least a subset of weights of all layers to be tuned at each epoch. Concretely, instead of freezing gradients for *whole* layers, we mask a percentage of gradients in *all* layers to increase gradient flow through the whole network. Thus, in this paper we propose *gradient dropout*, which we refer to as *GradDrop*, for *stochastically masking gradients* to regularize pretrained language fine-tuning. We find two GradDrop variants significantly improve the fine-tuning of pretrained models, namely *GradDrop-Epoch* (where weight masks are fixed over the whole epoch) and *Layer-GradDrop* (stochastically masks out gradients of whole layers). GradDrop and its variants are *simple to implement* and are easily used as a default operation for LM fine-tuning. We provide a comprehensive analysis of the how masking and fine-tuning can be used to improve cross-lingual transfer to downstream tasks without any *task-specific cross-lingual alignment* or *translate-train* training schemes using the XLM-R_{Large} (Conneau et al., 2020b) model, given its wide adoption and success in transfer learning to various languages.

2 Related Research

Adapters fine-tune relatively small linear layers that are placed between pretrained frozen layers and generally only account for a small percentage (e.g., 2-5%) of the overall number of parameters. There are variants whereby some adapters

are placed only on the output of each self-attention block, within each self-attention block, or combining adapters that have been independently trained for specific tasks and languages (Pfeiffer et al., 2020b; Houlsby et al., 2019; Pfeiffer et al., 2020a).

Gradual Unfreezing. Howard and Ruder (2018) proposed gradually turning on gradients layer by layer for LM pretraining and fine-tuning, leading to a reduction in training time due to a reduction in gradient updates. Peters et al. (2019) have further explored which tasks benefit from fine-tuning when all gradients are active, when only the newly added fine-tuning layer gradients are active and when using gradual unfreezing. Their main finding is that when the underlying LM pretraining is semantically similar to the downstream task there is less need to deactivate gradients, while the semantically different tasks benefit more from activating all gradients for fine-tuning.

Language Model Masking. While standard LM fine-tuning remains the defacto standard in NLP-based transfer learning, there has been other masking-related approaches. Zhao et al. (2020) have learned a mask over the weights instead of fine-tuning the weights, showing that this can lead to competitive performance for fine-tuning. In contrast to our work, we show that combining masking during fine-tuning is a preferred method for the same computational budget. Liu et al. (2021) use the change of the gradient magnitudes of a layer as a criterion to determine whether a layer is to be frozen. Hence, gradients that stagnate in a layer are most likely to be frozen during fine-tuning.

3 Proposed Methodology

In this section, we describe our main contribution, gradient dropout and variants thereof. We begin by first describing the self-attention blocks in transformers. Assume we have a sequence of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ where each vector $\boldsymbol{x}_i \in \mathbb{R}^d$ of d dimensions (e.g., $d = 512$). We define $\mathbf{Q} \in \mathbb{R}^{n \times d}$ to be a matrix representing the sequence where the i-th row of $\mathbf{Q}$ corresponds to $\boldsymbol{x}_i$. The key $\mathbf{K} \in \mathbb{R}^{d \times l}$, value $\mathbf{V} \in \mathbb{R}^{d \times l}$ and projection layer $\mathbf{U} \in \mathbb{R}^{d \times o}$ parameters are defined where $\mathbf{U}$ ensures the output dimensionality of the self-attention block is the same as the original input $\mathbf{Q}$. We can then define the self-attention as $\mathbf{Z} = \text{Softmax}(\frac{\mathbf{QK}}{\sqrt{dl}}\mathbf{V}^\top\mathbf{Q}^\top)\mathbf{QU}$ where $\mathbf{QU} \in \mathbb{R}^{n \times o}$ is matrix of new embeddings, $\mathbf{QKV}^\top\mathbf{Q}^\top \in \mathbb{R}^{n \times n}$ is a matrix representing the inner products in a new l-dimensional space and $\text{Softmax}\big(\mathbf{QKV}^\top\mathbf{Q}^\top\big)$ is a matrix where each row entry is positive and sums to 1. Note that *scaled* dot-product is used (normalization by $\sqrt{dl}$) to avoid vanishing gradients of the Softmax, which may occur when dl is large. The parameters for the j-th attention head $\mathbf{K}^j, \mathbf{V}^j \in \mathbb{R}^{d \times l}$, $\mathbf{U}^j \in \mathbb{R}^o$ for $j = 1, \ldots, n_a$ where n_a is the number of attention heads. Then we summarize the formulation of multi-headed self-attention as Equation 1,

$$\begin{aligned} \mathbf{Z}^j &= \text{Softmax}\big(\frac{\mathbf{QK}^j}{\sqrt{dl}}(\mathbf{V}^j)^\top\mathbf{Q}^T\big)\mathbf{QU}^j \\ \tilde{\mathbf{Z}} &= \text{Concat}(\mathbf{Z}^1, \ldots \mathbf{Z}^n_a) \\ \mathbf{Z} &= \text{Feedforward}(\text{LayerNorm}(\tilde{\mathbf{Z}} + \mathbf{Q})) \end{aligned} \tag{1}$$

where $\mathbf{Z}^j \in \mathbb{R}^{n \times d_a}$ and $\tilde{\mathbf{Z}} \in \mathbb{R}^{n \times d_a n_a}$, with d_a being the dimensionality of the self-attention output.

Gradient Dropout After backpropogation, we apply a random binary mask on the gradients of $\mathbf{K}, \mathbf{V}$ and $\mathbf{U}$. For simplicity, let us assume $\theta := \{\mathbf{K}, \mathbf{V}, \mathbf{U}\}$ and the gradients of θ are represented as $\mathbf{g} := \nabla_\theta \mathcal{L}^s(f_\theta(\mathbf{Q}), \mathbf{Y})$, where $\mathbf{Y} \in \mathbb{N}^{n \times d}$ represents one-hot targets of dimension d. A binary mask $\mathbf{m}$ is then generated from a predefined distribution (e.g., Bernoulli or Gaussian) and applied over the gradients. The gradient update rule with gradient dropout can then be expressed as,

$$\theta'_l = \theta_l - \alpha * \mathbf{g}_l \odot \mathbf{m}_l \tag{2}$$

where α is the learning rate, $\odot$ performs the Hadamard product (i.e., the element-wise product of tensors) and $l \in L$ is the layer index. Given that the stochastic noise induced by SGD through random mini-batch training regularizes DNNs, we too expect that the random dropping of gradients will have a similar regularization effect. When $\mathbf{m}$ is generated from a Bernoulli distribution, we randomly zero the gradient with probability p, in which the process of sampling m is formulated as:

$$\mathbf{m} \sim \text{Bernoulli}(1-p)/(1-p) \tag{3}$$

where the denominator $1 - p$ is the normalization factor. Note that, different from Dropout (Srivastava et al., 2014) which randomly drops the intermediate activations in a supervised learning network under a single task setting, we perform the *dropout on the gradient level*. We focus on binary masks for $\mathbf{m}$ as it is computationally efficient to generate and store low precision boolean tensors, in comparison to continuous noise such as the Gaussian distribution. Lastly, when applying gradient dropout layer-wise (**Layer-GradDrop**), $\boldsymbol{m} \in \{0, 1\}^L$ where l-th

Model	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.
Original XLM-R																
XLM-R$_{\text{Base}}$ Conneau et al.	84.6	78.4	78.9	76.8	75.9	77.3	75.4	73.2	71.5	75.4	72.5	74.9	71.1	65.2	66.5	74.5
XLM-R$_{\text{Large}}$ Conneau et al.	88.8	83.6	84.2	82.7	82.3	83.1	80.1	79.0	78.8	79.7	78.6	80.2	75.8	72.0	71.7	80.1
FILTER																
XLM-R$_{\text{Large}}$ Fang et al.	88.7	77.2	83.0	82.5	80.8	83.7	82.2	75.6	79.1	71.2	77.4	78.0	71.7	79.3	78.2	79.2
XLM-R$_{\text{Large}}$ (*translate-train*)	88.6	82.2	85.2	84.5	84.5	85.7	84.2	80.8	81.8	77.0	80.2	82.1	77.7	82.6	82.7	82.6
FILTER	89.7	83.2	86.2	85.5	85.1	86.6	85.6	80.9	83.4	78.2	82.2	83.1	77.4	83.7	83.7	83.6
FILTER + Self-Teaching	89.5	83.6	86.4	85.6	85.4	86.6	85.7	81.1	83.7	78.7	81.7	83.2	79.1	83.9	83.8	83.9
Ours																
XLM-R$_{\text{Large}}$	88.35	76.51	82.01	83.13	80.12	84.54	82.61	75.22	78.07	71.00	77.35	78.63	71.85	79.72	79.64	79.25
+GradFreeze-TopBottom	88.83	77.83	80.76	83.25	80.73	84.46	83.22	74.18	79.24	72.05	76.10	77.59	70.52	80.03	79.44	79.23
+GradFreeze-BottomUp	84.95	75.15	78.49	82.10	80.03	83.88	81.02	74.58	78.36	72.05	75.83	77.08	69.17	79.43	79.01	78.07
+GradDrop	90.01	78.19	**82.37**	83.53	80.68	84.82	**83.69**	76.18	78.72	**72.73**	77.03	79.04	**72.69**	**80.68**	79.23	**79.97**
+Anneal-GradDrop	88.49	76.88	82.81	83.54	80.13	85.07	82.90	**78.11**	78.14	71.04	76.72	78.39	72.47	80.17	79.28	79.58
+Anneal-Layer-GradDrop	**90.68**	78.19	82.93	**83.57**	80.96	85.26	83.53	76.27	79.12	71.93	77.43	77.59	72.49	79.44	79.72	79.94†
+Layer-GradDrop	88.65	76.97	81.99	81.43	81.38	83.11	82.99	76.45	**80.30**	68.53	**78.23**	78.05	71.47	79.38	79.42	79.22
+GradDrop-Epoch	88.27	**82.77**	83.13	81.25	**88.71**	**85.30**	83.25	77.07	78.67	71.45	77.31	**79.40**	72.53	80.20	**79.72**	79.94†

The best performance obtained are marked in **bold**, while the second best results are indicated with †.

Table 1: XNLI zero-shot accuracy (apart from 'en') for each language. Results of fine-tuned XLM-R from prior work (Conneau et al., 2020a; Fang et al., 2020) are from the XTREME benchmark (Hu et al., 2020).

element in $\mathbf{m}$ corresponds to whether that layers gradients are activated or not. When $\boldsymbol{m}_l = 1$, a one matrix $\mathbf{1}$ of the same dimensionality as $\mathbf{g}_l$ is applied, and and zeros when $\boldsymbol{m}_l = 0$[1]. We also apply each mask per minibatch or per epoch using a scheduled dropout rate, namely a linear schedule that begins at $p = 0.9$, reduces by $p_\epsilon := p_{\epsilon-1} - 1/T$ at each epoch ϵ until the last epoch T is reached where $p = 0$. In subsequent tables, models that have term "Anneal-" use this annealed GradDrop schedule. We posit that the main generalization benefits given by sparsely freezing gradients can be explained by how it slows down the total amount of gradient flow for each consecutive mini-batch during fine-tuning. This is particularly important for tasks that are more distant from the original self-supervised pretraining objective used prior to fine-tuning, i.e., converging too fast on a distant task may lose the generalization benefits given by the pretrained state.

Epoch-wise Gradient Dropout We also propose a variant of GradDrop whereby the same dropout mask is applied to all mini-batches for a single epoch. The mask can be reset for successive epochs by uniformly sampling from the aforementioned Bernoulli distribution at the same dropout rate as before. However, we also consider an accumulative mask whereby we sample from the Bernoulli distribution *without* replacement for each epoch and this is what we use for our experiments. Figure 1 shows the difference between the proposed *GradDrop-Epoch* when previous epoch masks are frozen once a new mask is applied (**GradDrop-Epoch-Toggle**) and when the previous epoch masks are left unfrozen (**GradDrop-Epoch**). In both cases, sampling without replacement is used, unlike standard GradDrop and like gradual unfreezing. This similarity to gradual unfreezing w.r.t. sampling without replacement aims to improve the stability during fine-tuning as only a subset of parameters are being updated for a whole epoch.

[1]Please see the supplementary material for a pseudocode example of GradDrop used with XLM-R.

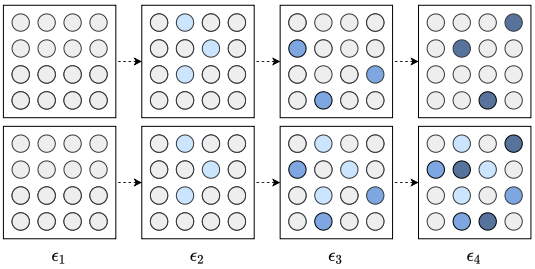

Figure 1: **GradDrop-Epoch-Toggle (Top) and GradDrop-Epoch (Bottom)**. Grey represents frozen gradients, blue represents active gradients where darker blue indicates the recency of gradients turned on.

4 Results

Table 1 shows the previous SoTA results on XNLI, our fine-tuned XLM-R$_{\text{Large}}$, *GradFreeze* (i.e., gradual unfreezing), GradDrop and its variants. Standard GradDrop outperforms its other variants and all prior SoTA fine-tuning methods, including gradient freezing. Our proposed methodology reports a 0.72% increase in zero-shot accuracy for GradDrop compared to standard fine-tuning.

Understanding Score. We show the average task *understanding* score for our GradDrop vari-

Models	Translation	#Params	XNLI	NC	NER	PAWSX	POS	QAM	QADSM	WPR	MLQA	Avg.
M-BERT (Liang et al., 2020)	Yes	550M	66.3	82.7	78.2	87.2	74.7	66.1	64.2	73.5	60.7	72.6
FILTER+Self-Teaching Fang et al.	Yes	550M	83.9	83.5	82.6	93.8	81.6	73.4	71.4	74.7	76.2	80.1
XLM-R_{Large}-T (Fang et al., 2020)	Yes	550M	82.6	-	-	-	-	-	-	-	-	-
Unicoder (Huang et al., 2019)	No	255M	75.3	83.5	79.70	90.1	79.6	68.9	68.4	73.9	66.0	76.1
XLM-R_{Large} (Conneau et al., 2020b)	No	550M	80.1	-	-	-	-	-	-	-	-	
XLM-R_{Large} (Fang et al., 2020)	No	550M	79.2	83.2	-	-	-	-	-	-	-	-
XLM-R_{Large} (Ours)	No	550M	79.25	83.21	80.61	89.23	80.38	69.82	71.05	73.27	70.21	77.45
+GradFreeze-TopDown	No	550M	79.23	83.65	78.64	88.33	78.11	71.01	71.01	72.61	70.55	77.02
+GradFreeze-BottomUp	No	550M	78.07	80.41	76.35	87.31	73.32	64.81	68.56	71.17	69.64	74.40
+GradDrop	No	550M	79.97	83.41	80.21	91.46	81.00	71.72	71.02	73.44	71.29	78.17
+Anneal-GradDrop	No	550M	79.58	81.87	81.87	91.18	80.72	71.46	70.71	73.27	71.20	77.98
+Anneal-Layer-GradDrop	No	550M	**79.94**	**84.24**	81.48	91.06	80.88	72.31	71.16	74.25	71.40	78.52
+Layer-GradDrop	No	550M	79.22	83.73	81.85	**92.23**	**81.42**	**72.33**	**71.39**	74.51	**72.55**	**78.77**
+GradDrop-Epoch	No	550M	79.94	83.73	**82.41**	91.15	80.45	72.98	70.52	**74.61**	72.01	78.64†

The best performance obtained are marked in **bold**, while the second best results are indicated with †.

Table 2: **Zero-Shot Cross-Lingual Performance Per Task and Overall Average Score (Avg.).**

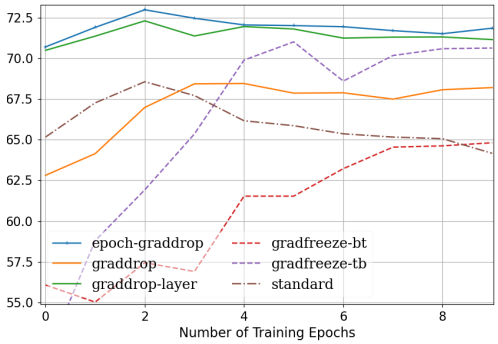

Figure 2: **QAM Test Performance Over Train Epochs.**

ants and previous baselines in Table 2. We find that GradDrop-Epoch and Layer-GradDrop are two methods which consistently outperform the remaining GradDrop variants, standard fine-tuning and in some cases, FILTER which uses translation data. To our knowledge, Layer-GradDrop sets a SoTA results on XGLUE for methods which *do not* use *translate-train* or translation language model *cross-lingual* alignment pretraining. Additionally, Layer-GradDrop is only 1.4 understanding score points from FILTER with their self-teaching loss.

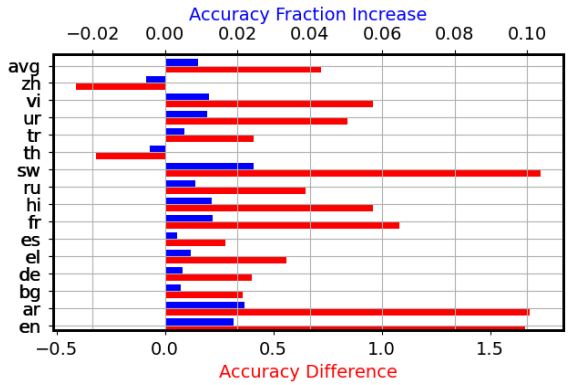

Figure 3: **Test Performance Increase By Language in XNLI.** Red bars indicate accuracy increases or decreases, while blue indicates the fractional increase of GradDrop over standard fine-tuning.

Training Stability and Convergence. We also analyse the stability of different GDVs, compared to standard fine-tuning and gradual unfreezing in Figure 2 which shows the development set accuracy for Question Answer Matching over consecutive training epochs. We find that both Epoch-GradDrop and Layer-GradDrop maintains performance for further training epochs while standard fine-tuning decreases as the model begins to overfit. This can be attributed to a reduction in the number of parameters being trained at any given epoch.

Per Language Analysis. We also inspect what languages do GDVs improve performance the most when compared to standard fine-tuning. We analyse XNLI which includes well-resourced and under-resourced languages in the evaluation set. Figure 3 shows how our best performing GDVs increase over standard fine-tuning and which languages we mostly attribute to the increase in average score. We find that biggest gains are made on Swahili and Arabic. We conclude that GradDrop improves performance on under-resourced languages in particular. We posit that this may be because GradDrop forces the model to be robust to static gradients during training on English only, reducing the effects of overfitting to the English language.

5 Conclusion

We find that our proposed GradDrop variants outperform standard fine-tuning of cross-lingual pretrained transformers. Specifically, epochwise- and layerwise- gradient dropout consistently outperform standard fine-tuning, gradual unfreezing and other baselines. Additionally, it is competitive against SoTA methods that use translation data and language alignment pretraining. We also find that gradient dropout particularly improves fine-tuning performance for under-resourced languages.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*, pages 8440–8451.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2020. Filter: An enhanced fusion method for cross-lingual language understanding. *arXiv preprint arXiv:2009.05166*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 12-18 July 2020, Virtual Conference*.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Empirical Methods in Natural Language Processing*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, et al. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. *arXiv preprint arXiv:2004.01401*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. 2021. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Virtual Conference*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Mengjie Zhao, Tao Lin, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. *arXiv preprint arXiv:2004.12406*.

A Example Appendix

This is an appendix.