Abstract

In recent years language models have achieved state of the art performance on a wide variety of natural language processing tasks. As these models are continuously growing in size it becomes increasingly important to explore methods to make them more storage efficient. At the same time their increase cognitive abilities increase the danger that societal bias existing in datasets are implicitly encoded in the model weights. We propose an architecture which deals with these two challenges at the same time using two techniques: DiffPruning and adversarial Training. The result is a modular architecture which extends the original DiffPruning setup with an additional sparse sub-network applied as a mask to diminish the effects of a predefined protected attribute at inference time.

1 Introduction and Motivation

Since their introduction, Transformer models have led to remarkable improvements in a range of natural language processing tasks (Vaswani et al., 2017; Liu et al., 2018; Kitaev and Klein, 2018; Radford and Narasimhan, 2018). Particularly in transfer learning, pre-trained Transformer encoders have been successful due to their ability to produce rich vector representations of text elements. These representations are typically learned through task-agnostic language modelling procedures (pretraining) after which they can be finetuned for various downstream tasks (Devlin et al., 2019). Fine-tuning is typically achieved by adding a task specific model “head” on top of the trained encoder and then using a suitable loss function to further train the model parameters via conventional gradient based methods. This procedure requires storing a separate encoder instance with its own set of weights for each new task. This can pose a challenge as parameter counts of widely used pre-trained encoders like BERTbase or RoBERTabase can go beyond 100 Million and parameter counts of newer state-of-the-art models far exceed these numbers (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020). As a result these models can be too slow or large for those real world tasks in which compute power and disk storage are limited, for example in mobile or edge computing.

To mitigate these problems, various approaches have been explored for reducing the overall model size and number of parameters required for finetuning. One of the most common category of methods is weight pruning (Han et al., 2015; See et al., 2016; Zhu and Gupta, 2018). Generally, weight pruning refers to a procedure which is applied during or after finetuning to reduce the number of parameters which are non-zero. One simple approach for weight pruning is magnitude pruning (Han et al., 2015). Magnitude pruning removes weights based on absolute value, usually followed by further training the remaining weights. It has been shown that magnitude pruning for language models can maintain good performance with only 60% of the original weights (Gordon et al., 2020).

A different approach to weight pruning is the Adapter network (Houlsby et al., 2019) where task-specific modules are inserted at different locations in the pre-trained model. During finetuning, only the adapter parameters are trained while other weights are kept frozen. Thus this method does not reduce the number of parameters of the pre-trained model but aims to limit the information which needs to be stored additionally to adapt a pre-trained model to a task. Houlsby et al. (2019) show that Adapters can match the performance of a fully finetuned BERT encoder on the GLUE benchmark while adding only 3.5% parameters per task.

DiffPruning follows a similar goal as Adapters and has proved to be even more parameter-efficient (Guo et al., 2021). In DiffPruning a sparse mask is learned for each model parameter. Non-zero values in the mask represent differences between
pre-trained and finetuned weights. Experiments on GLUE show that DiffPruning with a target sparsity of 0.5% compared to the original model parameter count of BERT$_{base}$ can achieve a similar performance as full finetuning (Guo et al., 2021).

Our work expands the concept of DiffPruning by applying it to a bias mitigation setting. Past research has shown that different types of bias such as race and gender bias are prevalent in Machine Learning systems (Buolamwini and Gebru, 2018; Bolukbasi et al., 2016; De-Arteaga et al., 2019). Bias Mitigation aims to reduce or remove bias via various techniques in the pre- and post-processing stages of a Machine Learning workflow or during model training itself (Ntoutsi et al., 2020). One method emerged from the field of domain adaptation is adversarial training (Ganin and Lempitsky, 2015; Zhang et al., 2018). To reduce the impact of a pre-defined attribute on task predictions intermediate model outputs are fed into a task head as well as a second head which has the objective to predict this attribute. During the backward pass the sign of the gradients flowing back out of this head is reversed. This has the effect of "punishing" the rest of the model for enabling correct domain predictions.

Our contribution is a unified training procedure to produce sparse language encoders for bias mitigation. During training the encoder is either optimized for the task via normal finetuning or with DiffPruning. Additionally the model architecture features a DiffPruning mask which can be added to remove information about one or more pre-defined protected attributes while still being able to perform the original task. Weight-values for the two parameter groups are updated in a 2-step procedure for each batch. This technique has the effect of modularizing bias mitigation by separating out weight-values which remove biased information. It furthermore makes it possible to choose if the focus during inference should be on just performing a task as good as possible or if the task should be achieved without using information about certain pre-defined protected attributes.

2 Background and Related Work
2.1 Adversarial Training
To adapt a pre-trained text encoder for bias mitigation we rely on datasets which include two sets of labels. One set of labels can be defined as task labels while the second set can be interpreted as labels for a protected attribute for which no information should be stored. This setting enables us to use adversarial training for bias mitigation. The model architecture for adversarial training consists of four components:

1. A neural network which outputs a feature vector.
2. A neural network head which takes the feature vector and predicts class labels for a particular set of labeled training data.
3. A second neural network head which for the example of domain adaptation tries to predict which domain (i.e. dataset) a particular training sample originates from.
4. A gradient reversal layer which is inserted between the base network and the domain prediction head. During backpropagation this layer takes the gradients of the domain loss w.r.t. the feature vector and changes the sign before the gradients are propagated back further to update the weights of the base network. This has the effect of "penalizing" the base network if the feature vector contains information about the source domain of a data sample.
2.2 \textbf{L}_0 \textit{regularization}

In contrast to \textit{L}_1 or \textit{L}_2 regularization which both reduce the magnitude of parameters by penalizing large weights, \textit{L}_0 regularization does not shrink weight values (Figure 2a). It can be thought of as a binary penalization as it only differentiates between 1 (no penalty applied) and 0 (full penalty applied). It therefore is well suited to enforce sparsity. However, since it is not differentiable in its standard form \( \sum_{d=1}^{d} \delta_i \neq 0 \), optimizing it is difficult.

One solution is to approximate the \textit{L}_0 objective via stochastic gates to make it differentiable (Louizos et al., 2018). Stochastic gates in neural networks make use of the reparametrization trick (Kingma and Welling, 2014) to separate the parameters of a stochastic node \( p(z|x, \phi) \) from a noise distribution \( p(\epsilon) \). Categorical variables can be represented in this way by a concrete distribution which was discovered independently by Maddison et al. (2017) and Jang et al. (2017). Louizos et al. (2018) chose a binary concrete distribution with a hard sigmoid gate which they call hard concrete distribution (Figure 2b) to construct a differentiable \textit{L}_0 regularization objective as follows:

\begin{align*}
    z &= \min(1, \max(0, \sigma(\log \alpha - \beta \log(-\gamma \zeta) + \gamma))) \\
    \sigma &= \max(0, \min(1, \sigma(\log \alpha - \beta \log(-\gamma \zeta) + \gamma))) \\
    u &\sim \mathcal{U}(0, 1) \\
    s &= \sigma((\log u - \log(1 - u) + \log \alpha)/\beta) \\
    \bar{s} &= s(\zeta - \gamma) + \gamma \\
    z &= \min(1, \max(0, \bar{s}))
\end{align*}

At inference time \( z \) is made deterministic by setting it to \( \min(1, \max(0, \sigma(\log \alpha)(\zeta - \gamma) + \gamma)) \).

2.3 \textit{DiffPruning}

For \textit{DiffPruning} (Guo et al., 2021), each model weight \( \theta \) is reparametrized as an addition of pretrained weights \( \theta \) and a task specific mask \( \delta \). During finetuning only \( \delta \) is optimized while \( \theta \) is kept frozen. \( \delta \) is made sparse using mask \( z \) which is trained via \textit{L}_0 Regularization.

First, construct a mask \( z \) from a stretched and rectified binary concrete distribution

\begin{align*}
    u &\sim \mathcal{U}(0, 1) \\
    s &= \sigma((\log u - \log(1 - u) + \log \alpha)/\beta) \\
    \bar{s} &= s(\zeta - \gamma) + \gamma \\
    z &= \min(1, \max(0, \bar{s}))
\end{align*}

Finally, the \textit{L}_0 penalty term penalizes the probability \( p(z > 0) \) and can be formulated as

\begin{align*}
    \mathcal{L}_{L_0} &= \sum_{j=1}^{[\theta]} \text{Sigmoid}(\log \alpha_j - \beta_j \log(-\gamma \zeta)) \tag{2}
\end{align*}

At inference time \( z \) is made deterministic by setting it to \( \min(1, \max(0, \text{Sigmoid}(\log \alpha)(\zeta - \gamma) + \gamma)) \).

\begin{align*}
    \theta^* &= \theta \odot z \tag{1}
\end{align*}

For \textit{DiffPruning} (Guo et al., 2021), each model weight \( \theta^* \) is reparametrized as an addition of pretrained weights \( \theta \) and a task specific mask \( \delta^*_r \). During finetuning only \( \delta^*_r \) is optimized while \( \theta \) is kept frozen. \( \delta^*_r \) is made sparse using mask \( z \) which is trained via \textit{L}_0 Regularization.

\begin{align*}
    \theta^*_r &= \theta + (z \odot \delta^*_r) \tag{3}
\end{align*}
Figure 4: An overview of bias mitigation with DiffPruning. In a two-step iterative procedure, subnetwork $E_t$ together with the pre-trained weights is first used to predict the task. Here only the task-loss is used for backpropagation. In a second step a second subnetwork $E_d$ is added on top of $E_t$ and updated via adversarial training. The weights of task head $H_t$ are kept frozen during this step.

Figure 5: DiffPruning with adversarial Training

After including the $L_0$ loss term the optimization objective for DiffPruning thus becomes:

$$\min_{w, \alpha, \beta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(y_n, \text{model}(x_n; \theta + z \odot w)) + \lambda \sum_{i=1}^{d} \sigma \left( \log \alpha_{r,i} - \beta_{r,i} \log \left( -\gamma \right) \right)$$ (4)

Guo et al. (2021) chose to fix $L_0$ parameter $\beta = 1$ without explicitly stating a reason. It is however reasonable to assume this choice was made to simplify calculations and to reduce the number of parameters to optimize. While the $L_0$ objective leads to a large number of parameters being set to 0, one downside of the method is that it is difficult to achieve an exact sparsity rate. To solve this issue Guo et al. (2021) introduce a second training phase after $L_0$ regularization. At the beginning this phase magnitude pruning is applied to the diff vector $\delta_r$ to achieve a target sparsity by only keeping the largest $x\%$ of weights. The remaining weights are then further fine-tuned (Figure 3).

3 Diff Subnetwork Bias Mitigation

Putting the concepts mentioned in the previous section together, we propose a sparse architecture for bias mitigation (Figure 4). We choose a transformer encoder as the backbone and five parallel Classifiers for the adversarial head with loss averaging.

To allow for situations where a task should be learned with and without including information about a protected attribute, we propose a modular architecture. We extend DiffPruning to two separate Diff -masks stacked on top of each other. One mask is only trained on the task objective while the second mask is trained via adversarial training and can be added to achieve bias mitigation through removal of a protected attribute. As the combination of the pre-trained encoder and one or both masks produces two different hidden representations for a sample, optimizing this architecture requires a 2-step procedure (Figure 4). During each update of the adversarial mask we keep the weights of the task head frozen. This has the advantage of forcing
all embeddings to be in the same vector space and lead to good performance in our experiments. As second option would be to use two separate task heads, one for the embeddings $E_{pre} + E_t$ and one for unbiased embeddings $E_{pre} + E_t + E_d$.

4 Experiment Design

For evaluating our architecture, we chose two datasets which both satisfy the requirement of having two attribute labels while at the same time fitting contextually to the concept of bias mitigation:

- the BIOS dataset includes short biographies with two labels job title and gender. We define job title as the task label and gender as the protected attribute. (De-Arteaga et al., 2019)

- the PAN16 dataset contains Twitter tweets in English, Spanish and Dutch. We use only the English subset which has labels gender and age. We again define gender as the protected attribute. As the task objective we use mention detection (i.e. has another user been mentioned in a tweet) for which we preprocess the dataset to remove all "@"-symbols which identify a mention. (Rangel et al., 2016)

For evaluation we want to measure how well a model is able to perform a task as well as how much information about a protected attribute is contained in the embeddings. For this purpose we use two metrics:

- **Task accuracy** to ensure good performance on the task objective is maintained

- **Adversarial attack accuracy** describes the accuracy of a new classifier which is trained on predicting the protected attribute based on the hidden representation of the trained (with or without adversarial training) encoder. For a balanced dataset this metric should be close to 0.5 which would imply that no information about a protected attribute could be extracted from the embeddings during the attack.

While the dataset is relatively balanced between both sets of classes, we artificially create complete parity for the protected attribute within each job title. This has the purpose of keeping an adversarial attacker from picking up on the class imbalance and predicting the majority class for all samples.

As a pre-trained model we use a BERT architecture with four heads and an embedding space of 256\(^1\) (Devlin et al., 2019; Turc et al., 2019). We compare our proposed modular architecture to various baselines. As dense baseline we run regular finetuning. We furthermore evaluate a Diff-Pruning baseline with a sparsity rate of 90% and 95% meaning that 10% and 5% of parameters are non-zero respectively. 90% is approximately the sparsity Diff-Pruning reaches naturally without magnitude pruning and thus provides a good balanced between sparsity and performance. 95% is a natural increment to test higher sparsity rates. Finally we evaluate a modular baseline consisting of two dense networks which follows the same two step training procedure outlined for the sparse modular architecture in (Figure 4). For all architectures we evaluate task-only as well as adversarial training.

All dense models were trained for 20 epochs while all Diff-Pruning models were trained for 30 epochs. This is to account for the effect that Diff-Pruning has two phases and models need time to recover their performance after the magnitude pruning step. We fix the learning rate for the BERT weights to $2e^{-5}$ and the learning rate for the classifier heads to $1e^{-4}$ after a light parameter search. The adversarial heads consist of five identical classifiers with different initialization. The loss for the five classifiers is averaged and accuracy is measured via majority vote. We found it furthermore important to add one hidden layer to the adversarial head classifiers. This enabled the head to learn more complex features from the BERT embeddings thus de-biasing it more effectively. For the task classifier we did find and additional hidden layer to have a measurable impact on performance, therefore we only use input and output layer. For all classification heads we use Tanh as the activation function. For Diff-Pruning we keep all hyperparameters as chosen by Guo et al. (2021).

To evaluate how much information about the protected attribute is contained in the embeddings of a particular finetuned model we run an adversarial attack. For this purpose we reinitialize the adversarial head and train with the objective of predicting the protected attribute with the finetuned BERT embeddings as input. We run each adversarial attack for 40 epochs with early stopping.

\(^1\)https://huggingface.co/google/bert_uncased_L-4_H-256_A-4
with a defined sparsity rate and we do not enforce
which are equally well suited to perform the task
The performance of all model architectures is sum-
Table 1: Performance of different model architectures on the BIOS and PAN16 datasets. For both datasets the
Peters et al. (2018) found that lower layers of lan-
Diff-Pruning sparsity
Figure 6 in Appendix
training an additional bias mitigation mask with
modular counterparts. The reason for this is that a
architectures the capacity (i.e. number of parame-
performance (i.e. lowest adversarial attack accu-
Objective as others which are possible more easy to
finetuned Diff-Pruning model as the initial state for
furthermore see in our experiments that training
only the Embedding layer exhibits a significantly
higher sparsity rate, partly confirm this assumption.
We use the same $L_0$ penalty for all layers.
We found the LayerNorm parameters to be the
least dense however looking at Table 2 in Appendix
their overall share in the total model size is neg-
layers the attention value parameters as well as the attention output
parameters regularly achieved high sparsity rates
However they still exhibit a high variance across different initializations. A
high variance of sparsity rates across experiments is usually caused by structured Diff-Pruning (Guo et al., 2021) where an additional $L_0$ Objective is
learned to encourage entire parameter groups to be zeroed out. Our experiments indicate that this happens more frequently for modular diff-pruning.
6 Conclusion
In this paper we proposed modular Diff-Pruning, a parameter-efficient and effective architecture for bias mitigation. In a two step procedure modular
Diff-Pruning learns two separate sparse sets of parameters: one solely oriented towards high
task performance and the second containing delta-values needed for bias mitigation. Using the BIOS
data set we show that bias mitigation does not nec-

5 Results and Analysis
The performance of all model architectures is sum-
marized in Table 1. Our experiments show a clear
effect of adversarial training for bias mitigation
for all evaluated model architectures. Interestingly
task accuracy is largely unaffected by adversarial
training. This could mean that adversarial train-
merely encourages a model to rely on features
which are equally well suited to perform the task
objective as others which are possible more easy to
find but contain bias towards the protected attribute.
The performance for the objective of bias miti-
gation is most prominent for the modular architec-
tures with the modular baseline achieving the best
performance (i.e. lowest adversarial attack accu-
率). It should be noted that for sparse modular
architectures the capacity (i.e. number of param-
eters) can be in theory double the size of their non-
modular counterparts. The reason for this is that a
modular model contains two separate sparse mask
with a defined sparsity rate and we do not enforce
the masks to share their non-zero positions. We did
furthermore see in our experiments that training
both masks in a combined iterative procedure as
outlined in Figure 4 is important for achieving a
high degree of debiasing. Using an already fine-
tuned Diff-Pruning model as the initial state for
training an additional bias mitigation mask with
adversarial training did not yield the same results.

Diff-Pruning sparsity Figure 6 in Appendix
show how non-zero parameters are distributed for
Diff-Pruning masks with a sparsity rate of 10%. Peters et al. (2018) found that lower layers of lan-
guage encoders capture mainly syntactic information while higher layers capture semantic information. Previous works (Zhao et al., 2020; Zhang et al., 2021) have used this finding for masking approaches focusing on higher layers as one would expect that semantic information is more relevant for finetuning. Our experiments, which show that
only the Embedding layer exhibits a significantly
higher sparsity rate, partly confirm this assumption.

we observed a high number of zero values in the value and attention output parameter matrices. In Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Proceedings of the 1st Conference on Fairness, Accountability and Transparency, volume 81 of Proceedings of Machine Learning Research, pages 77–91. PMLR.

Maria De- Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. page 120–128, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 1, pages 4171–4186. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In International conference on machine learning, pages 1180–1189. Proceedings of Machine Learning Research.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. In Workshop on Representation Learning for NLP, volume 5, pages 143–155. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In Annual Meeting of the Association for Computational Linguistics / International Joint Conference on Natural Language Processing, pages 4884–4896. Association for Computational Linguistics.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, volume 28, pages 1135—1143. Curran Associates, Inc.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In International Conference on Machine Learning, volume 97, pages 2790–2799. Proceedings of Machine Learning Research.

Eric Jiang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In International Conference on Learning Representations. OpenReview.net.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In International Conference on Learning Representations.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In Annual Meeting of the Association for Computational Linguistics, pages 2676–2686. Association for Computational Linguistics.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam M. Shazeer. 2018. Generating wikipedia by summarizing long sequences. ArXiv, abs/1801.10198.
Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. ArXiv, abs/1907.11692.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l0 regularization. In International Conference on Learning Representations.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In International Conference on Learning Representations. OpenReview.net.

Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Isosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompaniatis, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, and Steffen Staab. 2020. Bias in data-driven artificial intelligence systems—an introductory survey. WIREs Data Mining and Knowledge Discovery, 10.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pretraining.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Pan16 author profiling. Zenodo.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In Computational Natural Language Learning, volume 20, pages 291–301. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. arXiv preprint arXiv:1908.08962v2.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30, pages 6000—6010. Curran Associates, Inc.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES ’18, page 335–340, New York, NY, USA. Association for Computing Machinery.

Xiongyi Zhang, Jan-Willem van de Meent, and Byron Wallace. 2021. Disentangling representations of text by masking transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 778–791, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In Empirical Methods in Natural Language Processing, pages 2226–2241. Association for Computational Linguistics.

Michael Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In International Conference on Learning Representations, volume 6.

A Appendix

| Parameter Name             | Size     |
|----------------------------|----------|
| word_embeddings            | 3’9068’160|
| intermediate.dense         | 1’310’720|
| output.dense               | 1’310’720|
| position_embeddings        | 655’360  |
| attention.self.query       | 327’680  |
| attention.self.key         | 327’680  |
| attention.self.value       | 327’680  |
| attention.output.dense     | 327’680  |
| intermediate.dense.bias    | 5’120    |
| token_type_embeddings      | 2’560    |
| others                     | 1’280    |

Table 2: Parameters size (i.e. number of scalars)
Figure 6: Modular Diff-Pruning 90% vs. separate task-only and adversarial Diff-Pruning models 90%: density per layer and module. Percentage values indicate the number of non-zero parameters in each layer averaged over 5 experiments. Top most and right most values show the totals for the respective layers and modules.