# When BERT Plays the Lottery, All Tickets Are Winning

**Sai Prasanna**[*]
Zoho Labs
Zoho Corporation
Chennai, India
saiprasanna.r@zohocorp.com

**Anna Rogers**[*]
Center for Social Data Science
Copenhagen University
Copenhagen, Denmark
arogers@sodas.ku.dk

**Anna Rumshisky**
Dept. of Computer Science
Univ. of Massachusetts Lowell
Lowell, USA
arum@cs.uml.edu

## Abstract

Large Transformer-based models were shown to be reducible to a smaller number of self-attention heads and layers. We consider this phenomenon from the perspective of the lottery ticket hypothesis, using both structured and magnitude pruning. For fine-tuned BERT, we show that (a) it is possible to find subnetworks achieving performance that is comparable with that of the full model, and (b) similarly-sized subnetworks sampled from the rest of the model perform worse. Strikingly, with structured pruning even the worst possible subnetworks remain highly trainable, indicating that most pre-trained BERT weights are potentially useful. We also study the "good" subnetworks to see if their success can be attributed to superior linguistic knowledge, but find them unstable, and not explained by meaningful self-attention patterns.

## 1 Introduction

Much of the recent progress in NLP is due to the transfer learning paradigm in which Transformer-based models first try to learn task-independent linguistic knowledge from large corpora, and then get fine-tuned on small datasets for specific tasks. However, these models are overparametrized: we now know that most Transformer heads and even layers can be pruned without significant loss in performance (Voita et al., 2019; Kovaleva et al., 2019; Michel et al., 2019).

One of the most famous Transformer-based models is BERT (Devlin et al., 2019). It became a must-have baseline and inspired dozens of studies probing it for various kinds of linguistic information (Rogers et al., 2020b).

We conduct a systematic case study of fine-tuning BERT on GLUE tasks (Wang et al., 2018) from the perspective of the lottery ticket hypothesis (Frankle and Carbin, 2019). We experiment with and compare magnitude-based weight pruning and importance-based pruning of BERT self-attention heads (Michel et al., 2019), which we extend to multi-layer perceptrons (MLPs) in BERT.

With both techniques, we find the "good" subnetworks that achieve 90% of full model performance, and perform considerably better than similarly-sized subnetworks sampled from other parts of the model. However, in many cases even the "bad" subnetworks can be re-initialized to the pre-trained BERT weights and fine-tuned separately to achieve strong performance. We also find that the "good" networks are unstable across random initializations at fine-tuning, and their self-attention heads do not necessarily encode meaningful linguistic patterns.

## 2 Related Work

Multiple studies of BERT concluded that it is considerably overparametrized. In particular, it is possible to ablate elements of its architecture without loss in performance or even with slight gains (Kovaleva et al., 2019; Michel et al., 2019; Voita et al., 2019). This explains the success of multiple BERT compression studies (Sanh et al., 2019; Jiao et al., 2019; McCarley, 2019; Lan et al., 2020).

While NLP focused on building larger Transformers, the computer vision community was exploring the Lottery Ticket Hypothesis (LTH: Frankle and Carbin, 2019; Lee et al., 2018; Zhou et al., 2019). It is formulated as follows: *"dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that – when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations"* (Frankle and Carbin, 2019). The "winning tickets" generalize across vision datasets (Morcos et al., 2019), and exist both in LSTM and Transformer models for NLP (Yu et al., 2020).

[*]Equal contribution



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 3208–3229,
November 16–20, 2020. 

| Task | Dataset | Train | Dev | Metric |
|---|---|---|---|---|
| CoLA | Corpus of Linguistic Acceptability Judgements (Warstadt et al., 2019) | 10K | 1K | Matthews |
| SST-2 | The Stanford Sentiment Treebank (Socher et al., 2013) | 67K | 872 | accuracy |
| MRPC | Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) | 4k | n/a | accuracy |
| STS-B | Semantic Textual Similarity Benchmark (Cer et al., 2017) | 7K | 1.5K | Pearson |
| QQP | Quora Question Pairs[1] (Wang et al., 2018) | 400K | n/a | accuracy |
| MNLI | The Multi-Genre NLI Corpus (matched) (Williams et al., 2017) | 393K | 20K | accuracy |
| QNLI | Question NLI (Rajpurkar et al., 2016; Wang et al., 2018) | 108K | 11K | accuracy |
| RTE | Recognizing Textual Entailment (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) | 2.7K | n/a | accuracy |
| WNLI | Winograd NLI (Levesque et al., 2012) | 706 | n/a | accuracy |

Table 1: GLUE tasks (Wang et al., 2018), dataset sizes and the metrics reported in this study

However, so far LTH work focused on the "winning" random initializations. In case of BERT, there is a large pre-trained language model, used in conjunction with a randomly initialized task-specific classifier; this paper and concurrent work by Chen et al. (2020) are the first to explore LTH in this context. The two papers provide complementary results for magnitude pruning, but we also study structured pruning, posing the question of whether "good" subnetworks can be used as an tool to understand how BERT works. Another contemporaneous study by Gordon et al. (2020) also explores magnitude pruning, showing that BERT pruned before fine-tuning still reaches performance similar to the full model.

Ideally, the pre-trained weights would provide transferable linguistic knowledge, fine-tuned only to learn a given task. But we do not know what knowledge actually gets used for inference, except that BERT is as prone as other models to rely on dataset biases (McCoy et al., 2019b; Rogers et al., 2020a; Jin et al., 2020; Niven and Kao, 2019; Zellers et al., 2019). At the same time, there is vast literature on probing BERT architecture blocks for different linguistic properties (Rogers et al., 2020b). If there are "good" subnetworks, then studying their properties might explain how BERT works.

## 3 Methodology

All experiments in this study are done on the "BERT-base lowercase" model from the Transformers library (Wolf et al., 2020). It is fine-tuned[2] on 9 GLUE tasks, and evaluated with the metrics shown in Table 1. All evaluation is done on the dev sets, as the test sets are not publicly distributed. For each experiment we test 5 random seeds.

[2]All experiments were performed with 8 RTX 2080 TI GPUs, 128 Gb of RAM, 2x CPU Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz. Code repository: https://github.com/sai-prasanna/bert-experiments.

### 3.1 BERT Architecture

BERT is fundamentally a stack of Transformer encoder layers (Vaswani et al., 2017). All layers have identical structure: a multi-head self-attention (MHAtt) block followed by an MLP, with residual connections around each.

MHAtt consists of $N_h$ independently parametrized heads. An attention head $h$ in layer $l$ is parametrized by $W_k^{(h,l)}, W_q^{(h,l)}, W_v^{(h,l)} \in \mathbb{R}^{d_h \times d}$, $W_o^{(h,l)} \in \mathbb{R}^{d \times d_h}$. $d_h$ is typically set to $d/N_h$. Given $n$ $d$-dimensional input vectors $\mathrm{x} = x_1, x_2, ..x_n \in \mathbb{R}^d$, MHAtt is the sum of the output of each individual head applied to input $x$:

$$\mathrm{MHAtt}^{(l)}(\mathrm{x}) = \sum_{h=1}^{N_h} \mathrm{Att}^{(l)}_{W_k^{(h,l)}, W_q^{(h,l)}, W_v^{(h,l)}, W_o^{(h,l)}}(\mathrm{x})$$

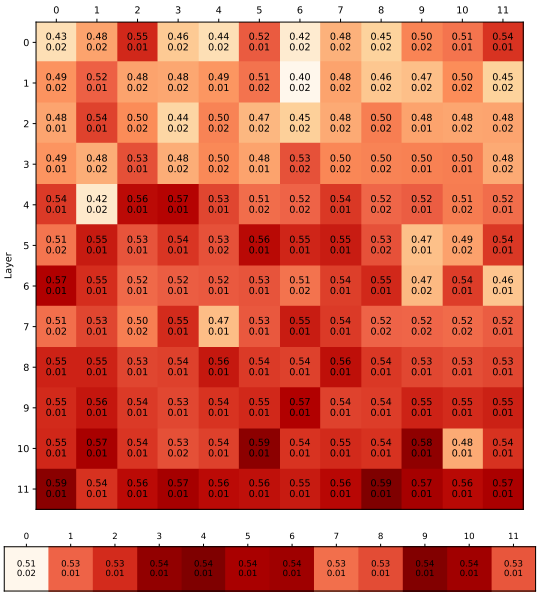
The MLP in layer $l$ consists of two feed-forward layers. It is applied separately to $n$ $d$-dimensional vectors $\mathrm{z} \in \mathbb{R}^d$ coming from the attention sub-layer. Dropout (Srivastava et al., 2014) is used for regularization. Then inputs of the MLP are added to its outputs through a residual connection.

### 3.2 Magnitude Pruning

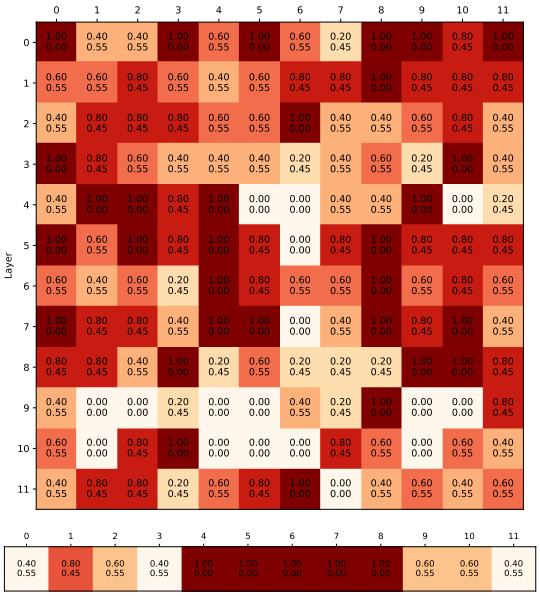
For magnitude pruning, we fine-tune BERT on each task and iteratively prune 10% of the lowest magnitude weights across the entire model (excluding the embeddings, since this work focuses on BERT's body weights). We check the *dev* set score in each iteration and keep pruning for as long as the performance remains above 90% of the full fine-tuned model's performance. Our methodology and results are complementary to those by Chen et al. (2020), who perform *iterative* magnitude pruning while fine-tuning the model to find the mask.

### 3.3 Structured Pruning

We study structured pruning of BERT architecture blocks, masking them under the constraint that at



(a) M-pruning: each cell gives the percentage of surviving weights, and std across 5 random seeds.



(b) S-pruning: each cell gives the average number of random seeds in which a given head/MLP survived and std.

Figure 1: The "good" subnetworks for QNLI: self-attention heads (top, 12 x 12 heatmaps) and MLPs (bottom, 1x12 heatmaps), pruned together. Earlier layers start at 0.

least 90% of full model performance is retained. Combinatorial search to find such masks is impractical, and Michel et al. (2019) estimate the importance of attention heads as the expected sensitivity to the mask variable $\xi^{(h,l)}$:

$$I_h^{(h,l)} = E_{x\sim X}\left|\frac{\partial \mathcal{L}(x)}{\partial \xi^{(h,l)}}\right|$$

where $x$ is a sample from the data distribution $X$ and $\mathcal{L}(x)$ is the loss of the network outputs on that sample. We extend this approach to MLPs, with the mask variable $\nu^{(l)}$:

$$I_{mlp}^{(l)} = E_{x\sim X}\left|\frac{\partial \mathcal{L}(x)}{\partial \nu^{(l)}}\right|$$

If $I_h^{(h,l)}$ and $I_{mlp}^{(l)}$ are high, they have a large effect on the model output. Absolute values are calculated to avoid highly positive contributions nullifying highly negative contributions.

In practice, calculating $I_h^{(h,l)}$ and $I_{mlp}^{(l)}$ would involve computing backward pass on the loss over samples of the evaluation data[3]. We follow Michel et al. in applying the recommendation of Molchanov et al. (2017) to normalize the importance scores of the attention heads layer-wise (with $\ell 2$ norm) before pruning. To mask the heads, we use a binary mask variable $\xi^{(h,l)}$. If $\xi^{(h,l)} = 0$, the head $h$ in layer $l$ is masked:

$$\text{MHAtt}^{(l)}(\text{x}) = \sum_{h=1}^{N_h} \xi^{(h,l)} \text{Att}^{(l)}_{W_k^{(h,l)}, W_q^{(h,l)}, W_v^{(h,l)}, W_o^{(h,l)}}(\text{x})$$

Masking MLPs in layer $l$ is performed similarly with a masking variable $\nu^{(l)}$:

$$\text{MLP}^{(l)}_{\text{out}}(\text{z}) = \nu^{(l)} MLP^{(l)}(\text{z}) + \text{z}$$

We compute head and MLP importance scores in a single backward pass, pruning 10% heads and one MLP with the smallest scores until the performance on the *dev* set is within 90%. Then we continue pruning heads alone, and then MLPs alone. The process continues iteratively for as long as the pruned model retains over 90% performance of the full fine-tuned model.

We refer to magnitude and structured pruning as *m-pruning* and *s-pruning*, respectively.

## 4 BERT Plays the Lottery

### 4.1 The "Good" Subnetworks

Figure 1 shows the heatmaps for the "good" subnetworks for QNLI, i.e. the ones that retain 90% of full model performance after pruning.

For s-pruning, we show the number of random initializations in which a given head/MLP survived

[3]The GLUE dev sets are used as oracles to obtain the best heads and MLPs for the particular model and task.

the pruning. For m-pruning, we compute the percentage of surviving weights in BERT heads and MLPs in all GLUE tasks (excluding embeddings). We run each experiment with 5 random initializations of the task-specific layer (the same ones), and report averages and standard deviations. See Appendix A for other GLUE tasks.

Figure 1a shows that in m-pruning, all architecture blocks lose about half the weights (42-57% weights), but the earlier layers get pruned more. With s-pruning (Figure 1b), the most important heads tend to be in the earlier and middle layers, while the important MLPs are more in the middle. Note that Liu et al. (2019) also find that the middle Transformer layers are the most transferable.

In Figure 1b, the heads and MLPs were pruned together. The overall pattern is similar when they are pruned separately. While fewer heads (or MLPs) remain when they are pruned separately (49% vs 22% for heads, 75% vs 50% for MLPs), pruning them together is more efficient overall (i.e., produces smaller subnetworks). Full data is available in Appendix B. This experiment hints at considerable interaction between BERT's self-attention heads and MLPs: with fewer MLPs available, the model is forced to rely more on the heads, raising their importance. This interaction was not explored in the previous studies (Michel et al., 2019; Voita et al., 2019; Kovaleva et al., 2019), and deserves more attention in future work.

### 4.2 Testing LTH for BERT Fine-tuning: The Good, the Bad and the Random

LTH predicts that the "good" subnetworks trained from scratch should match the full network performance. We experiment with the following settings:

- *"good" subnetworks*: the elements selected from the full model by either technique;
- *random subnetworks*: the same size as "good" subnetworks, but with elements randomly sampled from the full model;
- *"bad" subnetworks*: the elements sampled from those that did *not* survive the pruning, plus a random sample of the remaining elements so as to match the size of the "good" subnetworks.

For both pruning methods, we evaluate the subnetworks (a) after pruning, (b) after retraining the same subnetwork. The model is re-initialized to pre-trained weights (except embeddings), and the task-specific layer is initialized with the same random seeds that were used to find the given mask.

As mentioned earlier, the evaluation is performed on the GLUE[4] *dev* sets, which have also been used to identify the the "good" subnetworks originally. These subnetworks were chosen to work well on this specific data, and the corresponding "bad" subnetworks were defined only in relation to the "good" ones. We therefore do not expect these subnetworks to generalize to other data, and believe that they would best illustrate what exactly BERT "learns" in fine-tuning.

Performance of each subnetwork type is shown in Figure 2.The main LTH prediction is validated: the "good" subnetworks can be successfully retrained alone. Our m-pruning results are consistent with contemporaneous work by Gordon et al. (2020) and Chen et al. (2020).

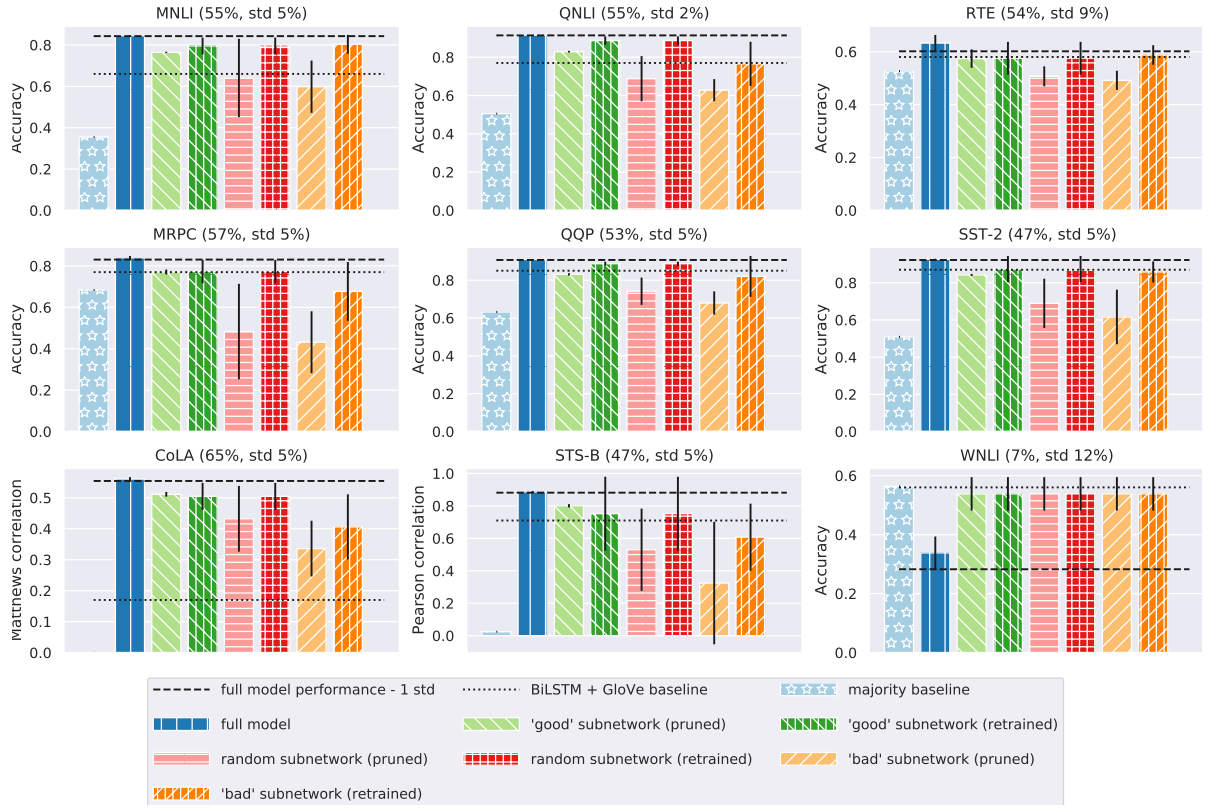
We observe the following differences between the two pruning techniques:

- For 7 out of 9 tasks m-pruning yields considerably higher compression (10-15% more weights pruned) than s-pruning.
- Although m-pruned subnetworks are smaller, they mostly reach[5] the full network performance. For s-pruning, the "good" subnetworks are mostly slightly behind the full network performance.
- Randomly sampled subnetworks could be expected to perform better than the "bad", but worse than the "good" ones. That is the case for m-pruning, but for s-pruning they mostly perform on par with the "good" subnetworks, suggesting the subset of "good" heads/MLPs in the random sample suffices to reach the full "good" subnetwork performance.

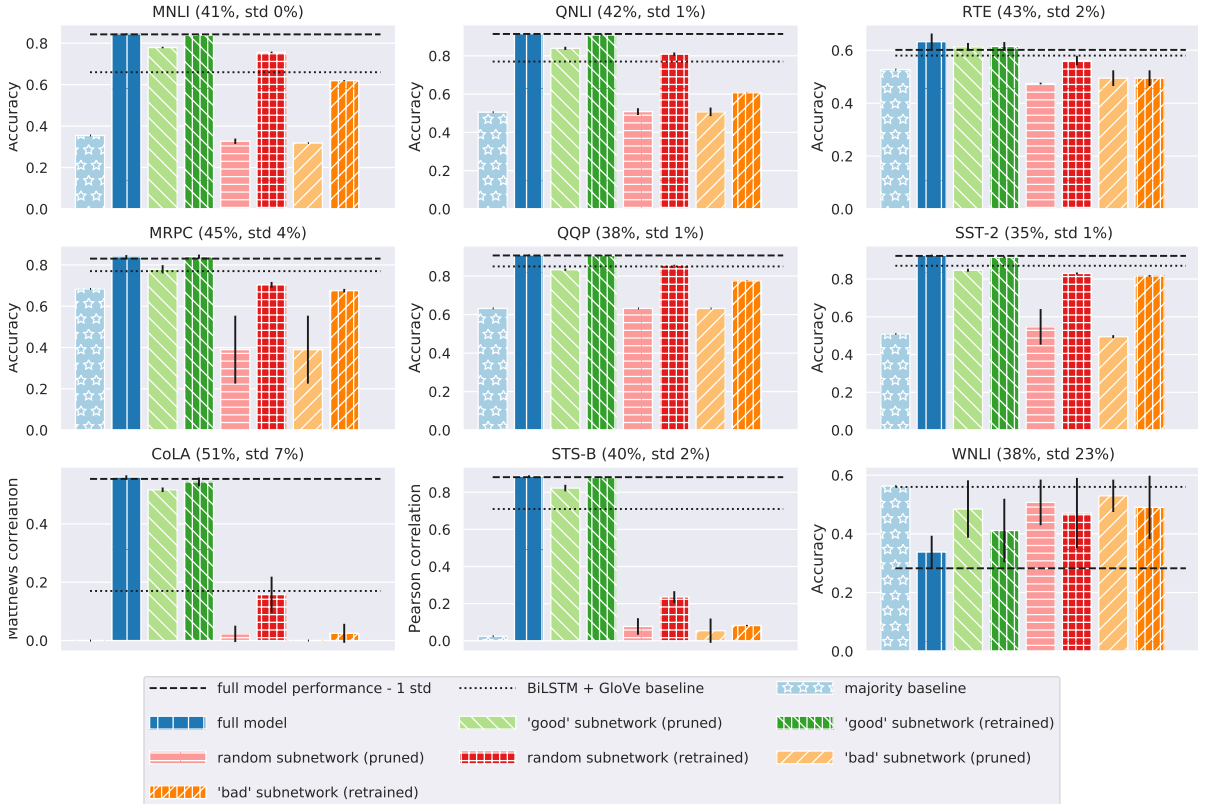
Note that our pruned subnetworks are relatively large with both pruning methods (mostly over 50% of the full model). For s-pruning, we also look at "super-survivors": much smaller subnetworks consisting only of the heads and MLPs that consistently survived across all seeds for a given task. For most tasks, these subnetworks contained only about 10-26% of the full model weights, but lost only about 10 performance points on average. See Appendix E for the details for this experiment.

[4]The results for WNLI are unreliable: this dataset has similar sentences with opposite labels in train and dev data, and in s-pruning the whole model gets pruned away. See Appendix A for discussion of that.

[5]For convenience, Figure 2 shows the performance of the full model minus one standard deviation – the success criterion for the subnetwork also used by Chen et al. (2020).




(a) S-pruning




(b) M-pruning

Figure 2: The "good" and "bad" subnetworks in BERT fine-tuning: performance on GLUE tasks. 'Pruned' subnetworks are only pruned, and 'retrained' subnetworks are restored to pretrained weights and fine-tuned. Subfigure titles indicate the task and percentage of surviving weights. STD values and error bars indicate standard deviation of surviving weights and performance respectively, across 5 fine-tuning runs. See Appendix C for numerical results, and subsection 4.3 for GLUE baseline discussion.

| Model | CoLA | SST-2 | MRPC | QQP | STS-B | MNLI | QNLI | RTE | WNLI | Average[6] |
|---|---|---|---|---|---|---|---|---|---|---|
| Majority class baseline | 0.00 | 0.51 | 0.68 | 0.63 | 0.02 | 0.35 | 0.51 | 0.53 | **0.56** | 0.42 |
| CBOW | **0.46** | 0.79 | 0.75 | 0.75 | 0.70 | 0.57 | 0.62 | 0.71 | **0.56** | 0.61 |
| BILSTM + GloVe | 0.17 | 0.87 | **0.77** | 0.85 | **0.71** | 0.66 | **0.77** | **0.58** | **0.56** | 0.66 |
| BILSTM + ELMO | 0.44 | **0.91** | 0.70 | **0.88** | 0.70 | **0.68** | 0.71 | 0.53 | 0.56 | **0.68** |
| 'Bad' subnetwork (s-pruning) | 0.40 | 0.85 | 0.67 | 0.81 | 0.60 | 0.80 | 0.76 | 0.58 | 0.53 | 0.67 |
| 'Bad' subnetwork (m-pruning) | 0.24 | 0.81 | 0.67 | 0.77 | 0.08 | 0.61 | 0.6 | 0.49 | 0.49 | 0.51 |
| Random init + random s-pruning | 0.00 | 0.78 | 0.67 | 0.78 | 0.14 | 0.63 | 0.59 | 0.53 | 0.50 | 0.52 |

Table 2: "Bad" BERT subnetworks (the best one is underlined) vs basic baselines (the best one is bolded). The randomly initialized BERT is randomly pruned by importance scores to match the size of s-pruned 'bad' subnetwork.

### 4.3 How Bad are the "Bad" Subnetworks?

Our study – as well as work by Chen et al. (2020) and Gordon et al. (2020) – provides conclusive evidence for the existence of "winning tickets", but it is intriguing that for most GLUE tasks random masks in s-pruning perform nearly as well as the "good" masks - i.e. they could also be said to be "winning". In this section we look specifically at the "bad" subnetworks: since in our setup, we use the *dev* set both to find the masks and to test the model, these parts of the model are the *least* useful for that specific data sample, and their trainability could yield important insights for model analysis.

Table 2 shows the results for the "bad" subnetworks pruned with both methods and re-fine-tuned, together with *dev* set results of three GLUE baselines by Wang et al. (2018). The m-pruned 'bad' subnetwork is at least 5 points behind the s-pruned one on 6/9 tasks, and is particularly bad on the correlation tasks (CoLA and STS-B). With respect to GLUE baselines, the s-pruned "bad" subnetwork is comparable to BiLSTM+ELMO and BiLSTM+GloVe. Note that there is a lot of variation between tasks: the 'bad' s-pruned subnetwork is competitive with BiLSTM+GloVe in 5/9 tasks, but it loses by a large margin in 2 more tasks, and wins in 2 more (see also Figure 2).

The last line of Table 2 presents a variation of experiment with fine-tuning randomly initialized BERT by Kovaleva et al. (2019): we randomly initialize BERT and also apply a randomly s-pruned mask so as to keep it the same size as the s-pruned "bad" subnetwork. Clearly, even this model is in principle trainable (and still beats the majority class baseline), but on average[6] it is over 15 points behind the "bad" mask over the pre-trained weights. This shows that even the worst possible selection of pre-trained BERT components for a given task still contains a lot of useful information. In other words, some lottery tickets are "winning" and yield the biggest gain, but all subnetworks have a non-trivial amount of useful information.

Note that even the random s-pruning of a randomly initialized BERT is slightly better than the m-pruned "bad" subnetwork. It is not clear what plays a bigger role: the initialization or the architecture. Chen et al. (2020) report that pre-trained weights do not perform as well if shuffled, but they do perform better than randomly initialized weights. To test whether the "bad" s-pruned subnetworks might match the "good" ones with more training, we trained them for 6 epochs, but on most tasks the performance went down (see Appendix D).

Finally, BERT is known to sometimes have degenerate runs (i.e. with final performance much lower than expected) on smaller datasets (Devlin et al., 2019). Given the masks found with 5 random initializations, we find that standard deviation of GLUE metrics for both "bad" and "random" s-pruned subnetworks is over 10 points not only for the smaller datasets (MRPC, CoLA, STS-B), but also for MNLI and SST-2 (although on the larger datasets the standard deviation goes down after re-fine-tuning). This illustrates the fundamental cause of degenerate runs: the poor match between the model and final layer initialization. Since our "good" subnetworks are specifically selected to be the best possible match to the specific random seed, the performance is the most reliable. As for m-pruning, standard deviation remains low even for the "bad" and "random" subnetworks in most tasks except MRPC. See Appendix C for full results.

## 5 Interpreting BERT's Subnetworks

In subsection 4.2 we showed that the subnetworks found by m- and s-pruning behave similarly in fine-tuning. However, s-pruning has an advantage in

[6]GLUE leaderboard uses macro average of metrics to rank the participating systems. We only consider the metrics in Table 1 to obtain this average.

that the functions of BERT architecture blocks have been extensively studied (see detailed overview by Rogers et al. (2020b)). If the better performance of the "good" subnetworks comes from linguistic knowledge, they could tell a lot about the reasoning BERT actually performs at inference time.

### 5.1 Stability of the "Good" Subnetworks

Random initializations in the task-specific classifier interact with the pre-trained weights, affecting the performance of fine-tuned BERT (McCoy et al., 2019a; Dodge et al., 2020). However, if better performance comes from linguistic knowledge, we would expect the "good" subnetworks to better encode this knowledge, and to be relatively stable across fine-tuning runs for the same task.

We found the opposite. For all tasks, Fleiss' kappa on head survival across 5 random seeds was in the range of 0.15-0.32, and Cochran Q test did not show that the binary mask of head survival obtained with five random seeds for each tasks were significantly similar at $\alpha = 0.05$ (although masks obtained with some *pairs* of seeds were). This means that the "good" subnetworks are unstable, and depend on the random initialization more than utility of a certain portion of pre-trained weights for a particular task.

The distribution of importance scores, shown in Figure 3, explains why that is the case. At any given pruning iteration, most heads and MLPs have a low importance score, and could all be pruned with about equal success.

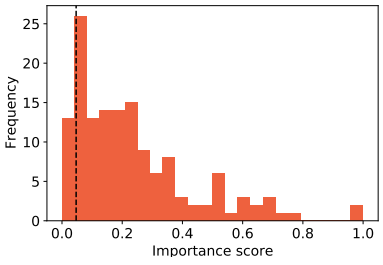



Figure 3: Head importance scores distribution (this example shows CoLA, pruning iteration 1)

### 5.2 How Linguistic are the "Good" Subnetworks?

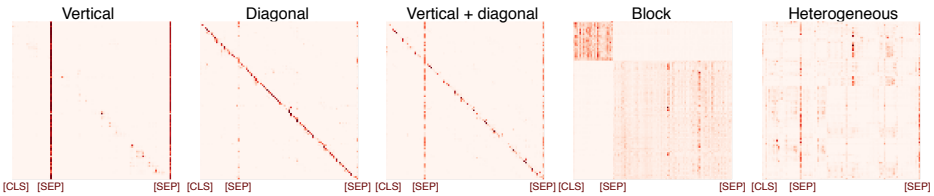
A popular method of studying functions of BERT architecture blocks is to use probing classifiers for specific linguistic functions. However, "the fact that a linguistic pattern is not observed by our probing classifier does not guarantee that it is not there, and the observation of a pattern does not tell us how it is used" (Tenney et al., 2019).

In this study we use a cruder, but more reliable alternative: the types of self-attention patterns, which Kovaleva et al. (2019) classified as diagonal (attention to previous/next word), block (uniform attention over a sentence), vertical (attention to punctuation and special tokens), vertical+diagonal, and heterogeneous (everything else) (see Figure 4a). The fraction of heterogeneous attention can be used as an *upper bound estimate* on non-trivial linguistic information. In other words, these patterns do not guarantee that a given head has some interpretable function – only that it could have it.

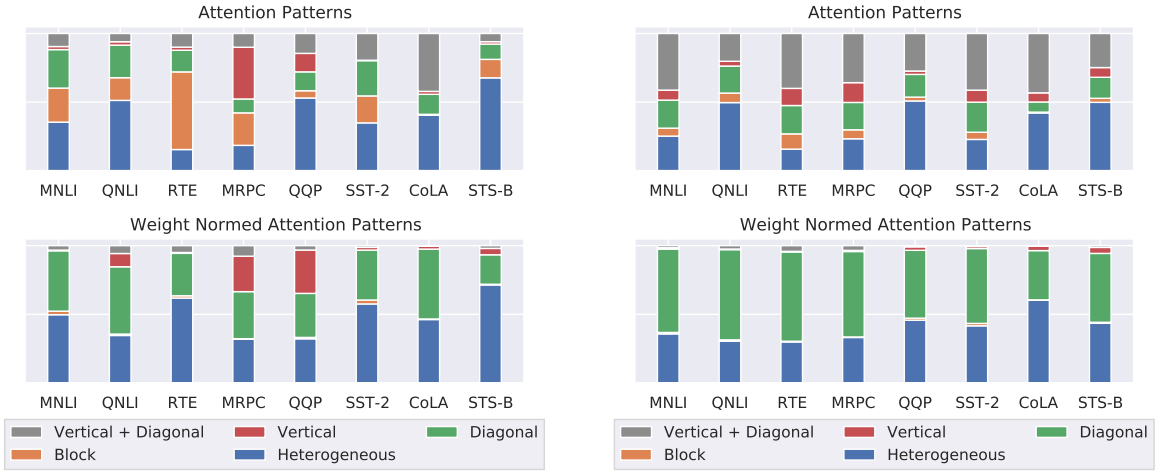
This analysis is performed by image classification on generated attention maps from individual heads (100 for each GLUE task), for which we use a small CNN classifier with six layers. The classifier was trained on the dataset of 400 annotated attention maps by Kovaleva et al. (2019).

Note that attention heads can be seen as a weighted sum of linearly transformed input vectors. Kobayashi et al. (2020) recently showed that the input vector norms vary considerably, and the inputs to the self-attention mechanism can have a disproportionate impact relative to their self-attention weight. So we consider both the raw attention maps, and, to assess the true impact of the input in the weighted sum, the L2-norm of the transformed input multiplied by the attention weight (for which we annotated 600 more attention maps with the same pattern types as Kovaleva et al. (2019)). The weighted average of $F_1$ scores of the classifier on annotated data was 0.81 for the raw attention maps, and 0.74 for the normed attention.

Our results suggest that the super-survivor heads do *not* preferentially encode non-trivial linguistic relations (heterogeneous pattern), in either raw or normed self-attention (Figure 4b). As compared to all 144 heads (Figure 4c) the "raw" attention patterns of super-survivors encode considerably more block and vertical attention types. Since norming reduces attention to special tokens, the proportion of diagonal patterns (i.e. attention to previous/next tokens) is increased at the cost of vertical+diagonal pattern. Interestingly, for 3 tasks, the super-survivor subnetworks still heavily rely on the vertical pattern even after norming. The vertical pattern indicates a crucial role of the special tokens, and it is unclear why it seems to be less important for MNLI rather than QNLI, MRPC or QQP.




(a) Reference: typical BERT self-attention patterns by Kovaleva et al. (2019).




(b) Super-survivor heads, fine-tuned

(c) All heads, fine-tuned

Figure 4: Attention pattern type distribution

The number of block pattern decreased, and we hypothesize that they are now classified as heterogeneous (as they would be unlikely to look diagonal). But even with the normed attention, the utility of super-survivor heads cannot be attributed only to their linguistic functions (especially given that the fraction of heterogeneous patterns is only a rough upper bound). The Pearson's correlation between heads being super-survivors and their having heterogeneous attention patterns is 0.015 for the raw, and 0.025 for the normed attention. Many "important" heads have diagonal attention patterns, which seems redundant.

We conducted the same analysis for the attention patterns in pre-trained vs. fine-tuned BERT for both super-survivors and all heads, and found them to not change considerably after fine-tuning, which is consistent with findings by Kovaleva et al. (2019). Full data is available in Appendix F.

Note that this result does not exclude the possibility that linguistic information is encoded in certain *combinations* of BERT elements. However, to date most BERT analysis studies focused on the functions of individual components (Voita et al., 2019; Htut et al., 2019; Clark et al., 2019; Lin et al., 2019; Vig and Belinkov, 2019; Hewitt and Manning, 2019; Tenney et al., 2019, see also the overview by Rogers et al. (2020b)), and this evidence points to the necessity of looking at their interactions. It also adds to the ongoing discussion of interpretability of self-attention (Jain and Wallace, 2019; Serrano and Smith, 2019; Wiegreffe and Pinter, 2019; Brunner et al., 2020).

Once again, heterogenerous pattern counts are only a crude upper bound estimate on potentially interpretable patterns. More sophisticated alternatives should be explored in future work. For instance, the recent information-theoretic probing by minimum description length (Voita and Titov, 2020) avoids the problem of false positives with traditional probing classifiers.

### 5.3 Information Shared Between Tasks

While the "good" subnetworks are not stable, the overlaps between the "good" subnetworks may still be used to characterize the tasks themselves. We leave detailed exploration to future work, but as a brief illustration, Figure 5 shows pairwise overlaps in the "good" subnetworks for the GLUE tasks.

The overlaps are not particularly large, but still

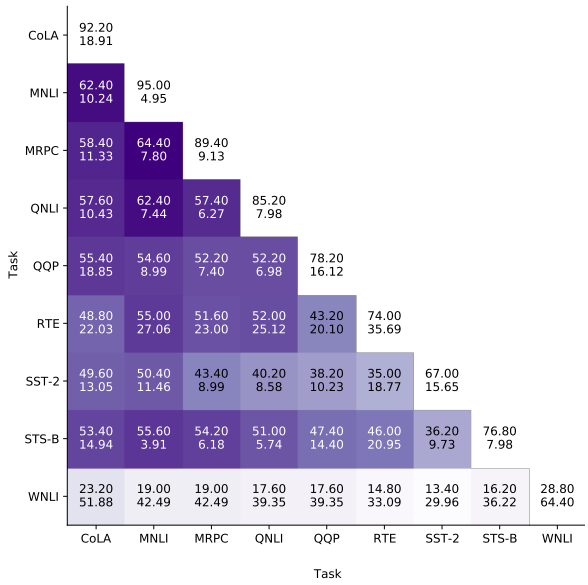



Figure 5: Overlaps in BERT's "good" subnetworks between GLUE tasks: self-attention heads.

more than what we would expect if the heads were completely independent (e.g. MRPC and QNLI share over a half of their "good" subnetworks). Both heads and MLPs show a similar pattern. Full data for full and super-survivor "good" subnetworks is available in Appendix G.

Given our results in subsection 5.2, the overlaps in the "good" subnetworks are not explainable by two tasks' relying on the same linguistic patterns in individual self-attention heads. They also do not seem to depend on the type of the task. For instance, consider the fact that two tasks targeting paraphrases (MRPC and QQP) have less in common than MRPC and MNLI. Alternatively, the overlaps may indicate shared heuristics, or patterns somehow encoded in combinations of BERT elements. This remains to be explored in future work.

## 6 Discussion

This study confirms the main prediction of LTH for pre-trained BERT weights for both m- and s-pruning. An unexpected finding is that with s-pruning, the "random" subnetworks are still almost as good as the "good" ones, and even the "worst" ones perform on par with a strong baseline. This suggests that the weights that do not survive pruning are not just "inactive" (Zhang et al., 2019).

An obvious, but very difficult question that arises from this finding is whether the "bad" subnetworks do well because even they contain some linguistic knowledge, or just because GLUE tasks are overall easy and could be learned even by *random* BERT (Kovaleva et al., 2019), or even any sufficiently large model. Given that we did not find even the "good" subnetworks to be stable, or preferentially containing the heads that could have interpretable linguistic functions, the latter seems more likely.

Furthermore, should we perhaps be asking the same question with respect to not only subnetworks, but also full models, such as BERT itself and all the follow-up Transformers? There is a trend to automatically credit any new state-of-the-art model with with better knowledge of language. However, what if that is not the case, and the success of pre-training is rather due to the flatter and wider optima in the optimization surface (Hao et al., 2019)? Can similar loss landscapes be obtained from other, non-linguistic pre-training tasks? There are initial results pointing in that direction: Papadimitriou and Jurafsky (2020) report that even training on MIDI music is helpful for transfer learning for LM task with LSTMs.

## 7 Conclusion

This study systematically tested the lottery ticket hypothesis in BERT fine-tuning with two pruning methods: magnitude-based weight pruning and importance-based pruning of BERT self-attention heads and MLPs. For both methods, we find that the pruned "good" subnetworks alone reach the performance comparable with the full model, while the "bad" ones do not. However, for structured pruning, even the "bad" subnetworks can be fine-tuned separately to reach fairly strong performance. The "good" subnetworks are *not* stable across fine-tuning runs, and their success is *not* attributable exclusively to non-trivial linguistic patterns in individual self-attention heads. This suggests that most of pre-trained BERT is potentially useful in fine-tuning, and its success could have more to do with optimization surfaces rather than specific bits of linguistic knowledge.

**Carbon Impact Statement.** This work contributed 115.644 kg of $CO_{2eq}$ to the atmosphere and used 249.068 kWh of electricity, having a NLD-specific social cost of carbon of \$-0.14 (\$-0.24, \$-0.04). The social cost of carbon uses models from (Ricke et al., 2018) and this statement and emissions information was generated with *experiment-impact-tracker* (Henderson et al., 2020).

## Acknowledgments

The authors would like to thank Michael Carbin, Aleksandr Drozd, Jonathan Frankle, Naomi Saphra and Sri Ananda Seelan for their helpful comments, the Zoho corporation for providing access to their clusters for running our experiments, and the anonymous reviewers for their insightful reviews and suggestions. This work is funded in part by the NSF award number IIS-1844740 to Anna Rumshisky.

## References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On Identifiability in Transformers. In *International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. *arXiv:2007.12223 [cs, stat]*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*.

W.B. Dolan and C. Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 1–9, Prague, Czech Republic. Association for Computational Linguistics.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. *ArXiv*, abs/2002.08307.

R. Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The Second Pascal Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and Understanding the Effectiveness of BERT. 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 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *arXiv:2002.05651 [cs]*.

John Hewitt and Christopher D. Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. 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 4129–4138.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in BERT track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-*

*guage Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556.

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*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *AAAI 2020*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention Module is Not Only a Weight: Analyzing Transformers with Vector Norms. *arXiv:2004.10102 [cs]*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. 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 4356–4365, Hong Kong, China. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. In *ICLR*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2018. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open Sesame: Getting inside BERT's Linguistic Knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

JS McCarley. 2019. Pruning a BERT-based Question Answering Model. *arXiv preprint arXiv:1910.06360*.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2019a. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. *ArXiv*, abs/1911.02969.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019b. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? *Advances in Neural Information Processing Systems 32 (NIPS 2019)*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. 2019. One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems 32*, pages 4932–4942. Curran Associates, Inc.

Timothy Niven and Hung-Yu Kao. 2019. Probing Neural Network Comprehension of Natural Language Arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2020. Learning Music Helps You Read: Using Transfer to Study Linguistic Structure in Language Models. *arXiv:2004.14601 [cs]*.

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. Association for Computational Linguistics.

Katharine Ricke, Laurent Drouet, Ken Caldeira, and Massimo Tavoni. 2018. Country-level social cost of carbon. *Nature Climate Change*, 8(10):895.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020a. Getting Closer to AI Complete Question Answering: A Set of Prerequisite Real Tasks. In *AAAI*, page 11.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020b. A Primer in BERTology: What we know about how BERT works. *arXiv:2002.12327 [cs]*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*.

Sofia Serrano and Noah A. Smith. 2019. Is Attention Interpretable? *arXiv:1906.03731 [cs]*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008, Long Beach, CA, USA.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. *arXiv:2003.12298 [cs]*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. 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, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. 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 11–20, Hong Kong, China. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: Lottery tickets in RL and NLP. In *ICLR*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL 2019*.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2019. Are All Layers Created Equal? In *ICML 2019 Workshop Deep Phenomena*.

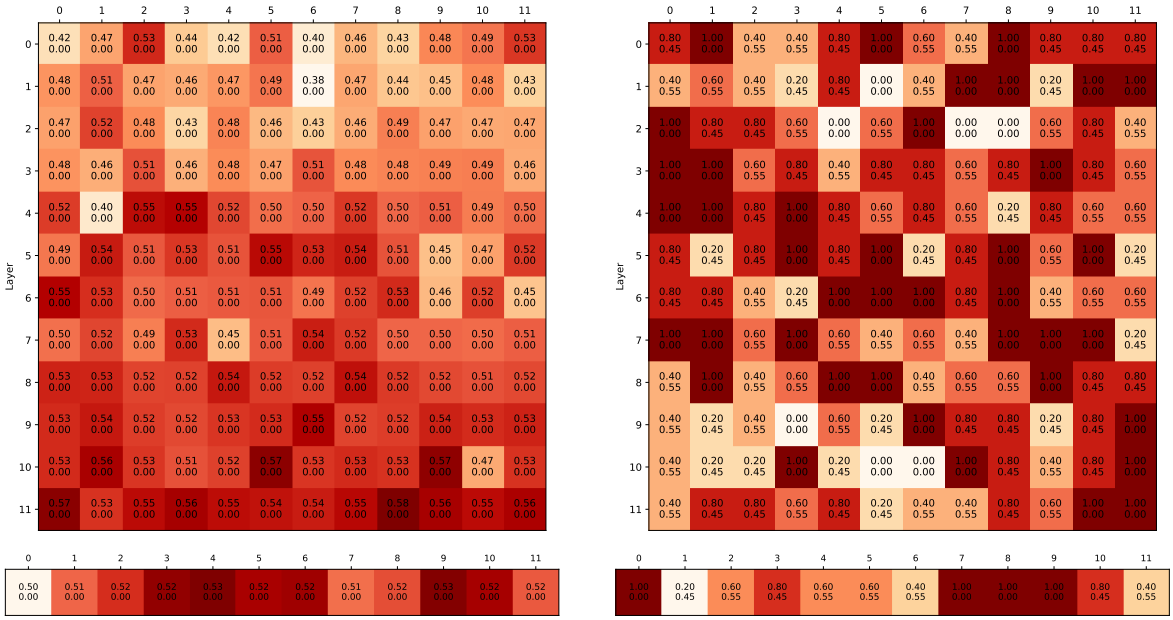
Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. In *Advances in Neural Information Processing Systems 32*, pages 3597–3607. Curran Associates, Inc.

## A "Good" Subnetworks in BERT Fine-tuned on GLUE Tasks

Each figure in this section shows the "good" subnetwork of heads and layers that survived the pruning process described in section 3. Each task was run with 5 different random seeds. The top number in each cell indicates how likely a given head or MLP was to survive pruning, with 1.0 indicating that it survived on every run. The bottom number indicates the standard deviation across runs.

The figures in this appendix show that each task has a varying number of heads and layers that survive pruning on all fine-tuning runs, while some heads and layers were only "picked up" by some random seeds. Note also that in addition to the architecture elements that survive across many runs, there are also some that are useful for over half of the tasks, as shown in Figure 15, and some *always* survive the pruning.

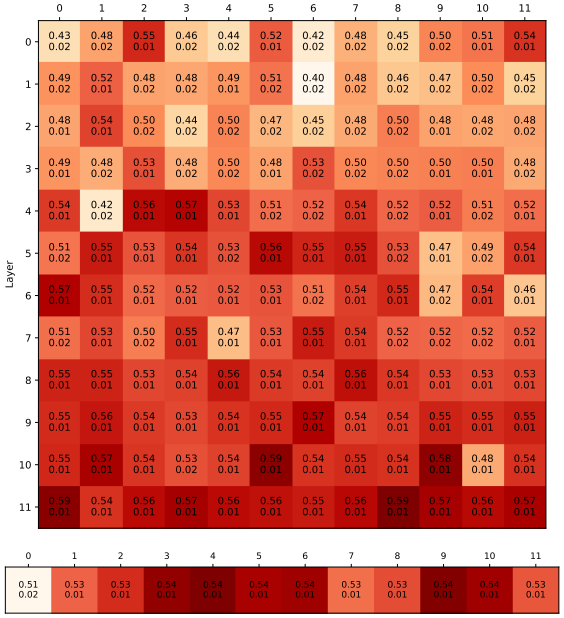
Visualizing the "good" subnetwork illustrates the core problem with WNLI, the most difficult task of GLUE. Figure 14 shows that each run is completely different, indicating that BERT fails to find any consistent pattern between the task and the information in the available pre-trained weights. WNLI is described as "somewhat adversarial" by Wang et al. (2018) because it has similar sentences in train and dev sets with opposite labels.



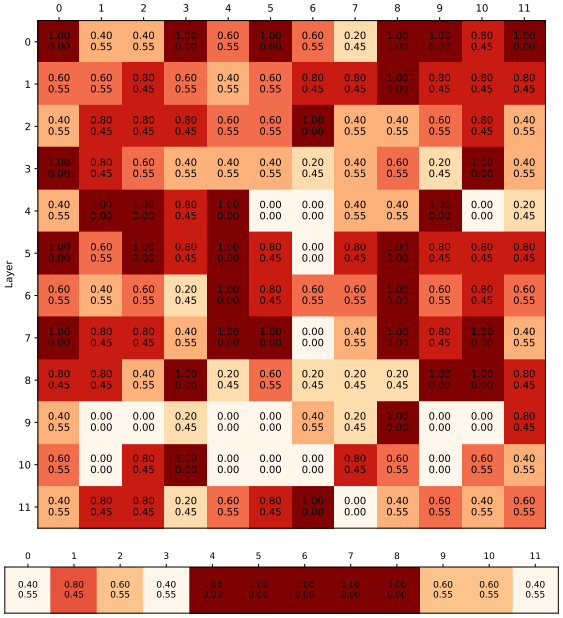

(a) M-pruning

(b) S-pruning

Figure 6: MNLI

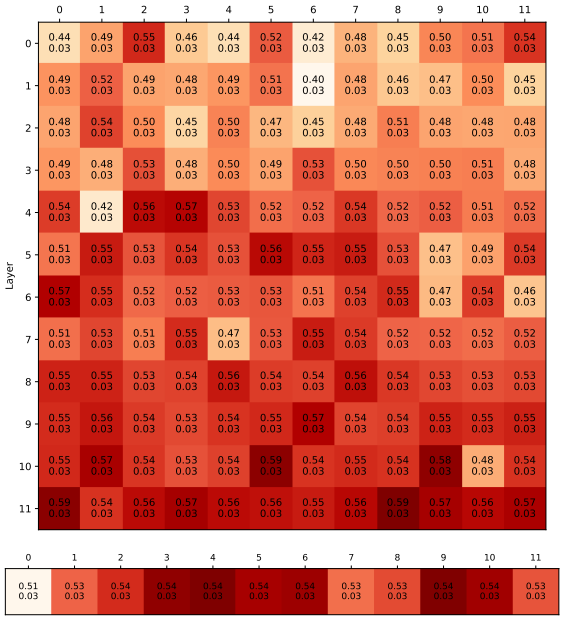



(a) M-pruning

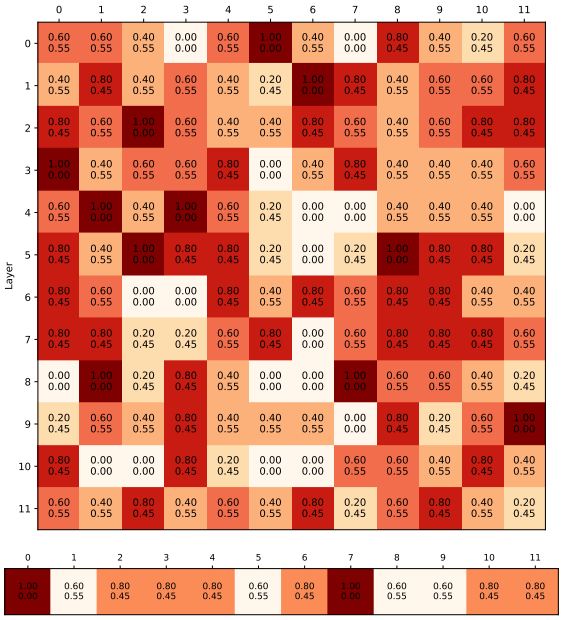



(b) S-pruning

Figure 7: QNLI

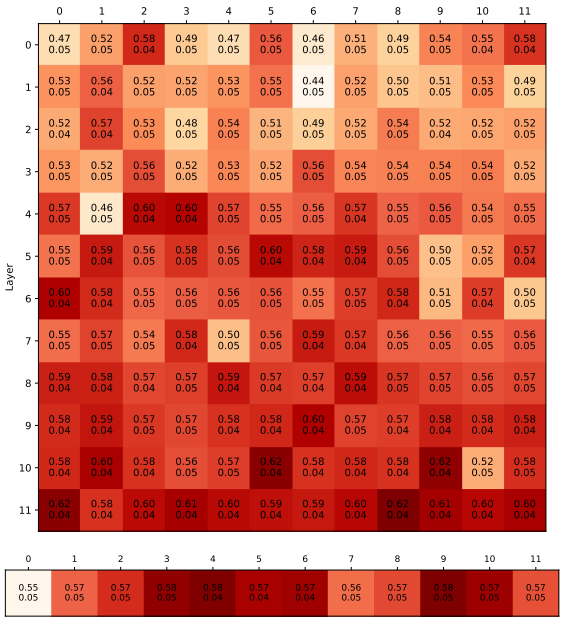



(a) M-pruning

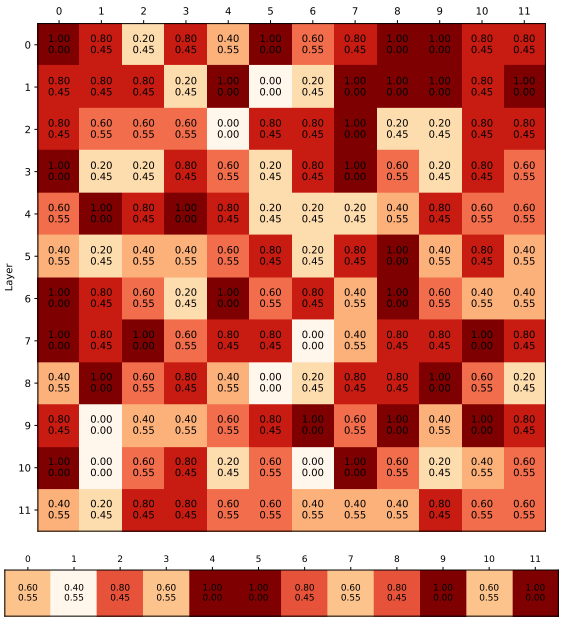



(b) S-pruning

Figure 8: RTE

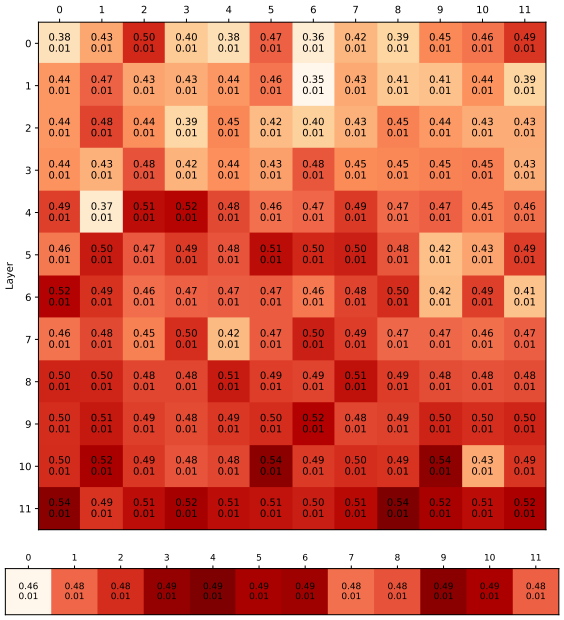



(a) M-pruning

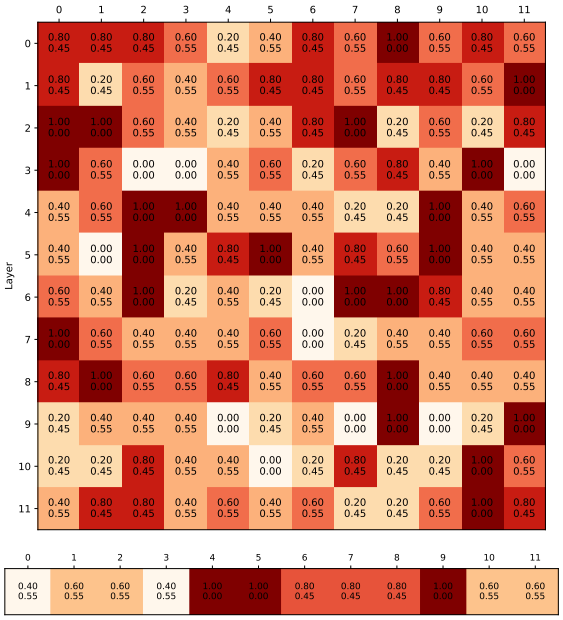



(b) S-pruning

Figure 9: MRPC

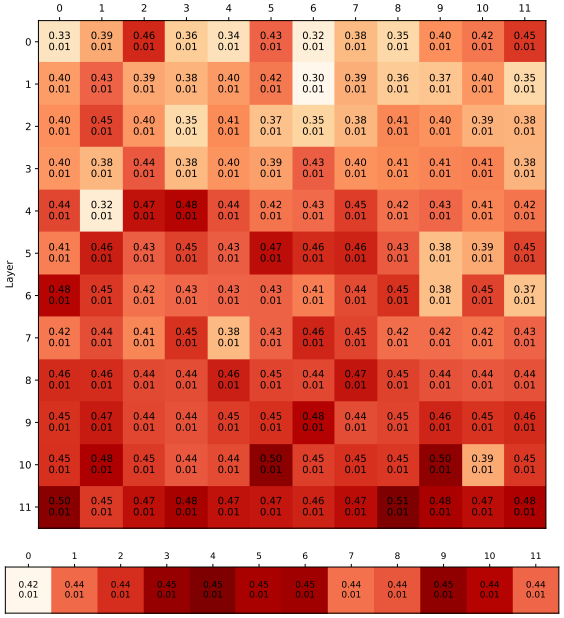



(a) M-pruning

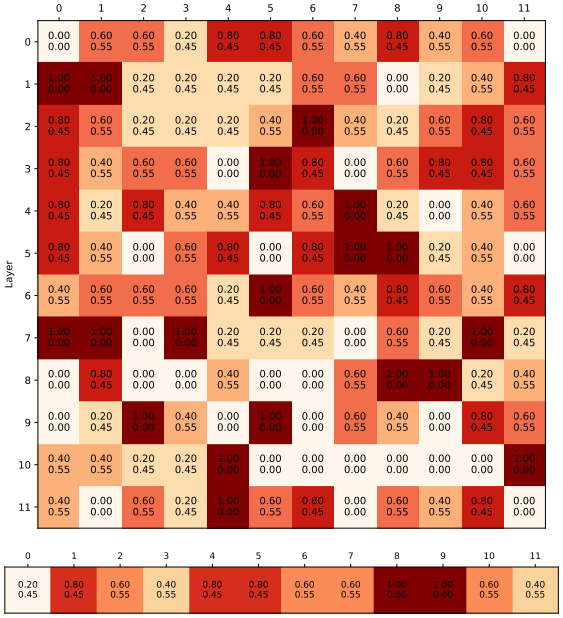



(b) S-pruning

Figure 10: QQP

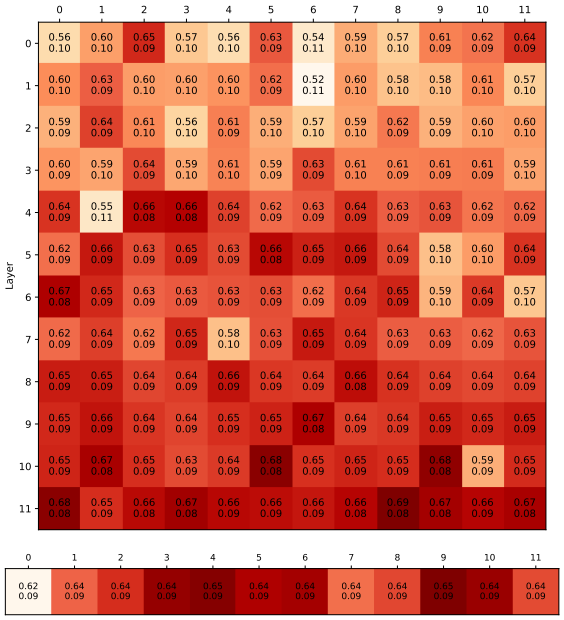



(a) M-pruning

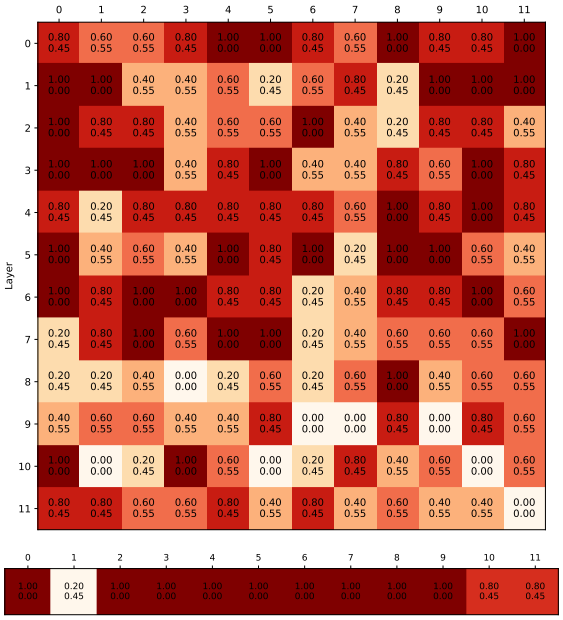



(b) S-pruning

Figure 11: SST-2

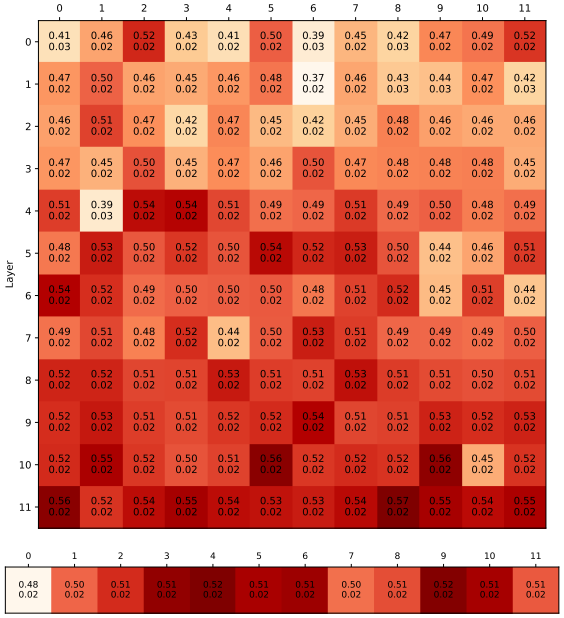



(a) M-pruning

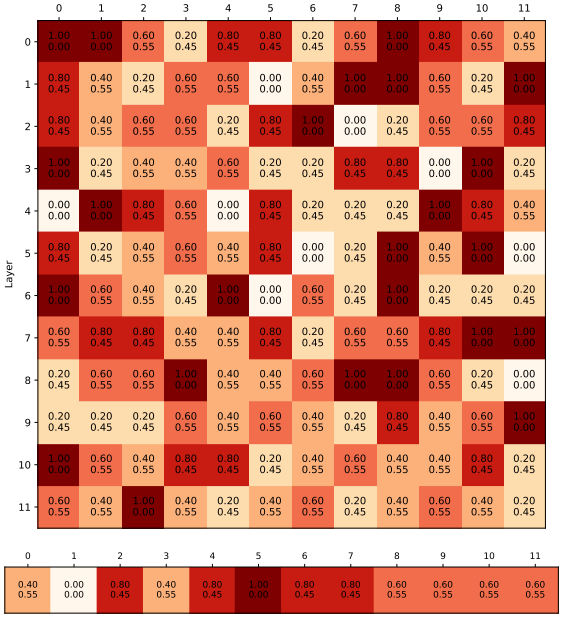



(b) S-pruning

Figure 12: CoLA

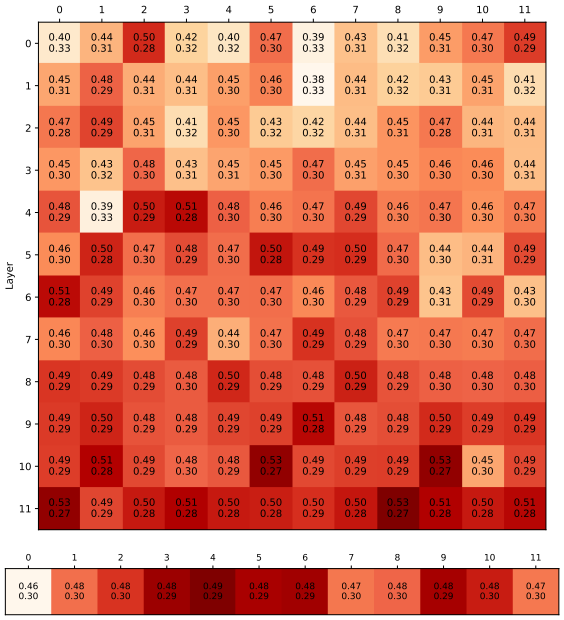



(a) M-pruning

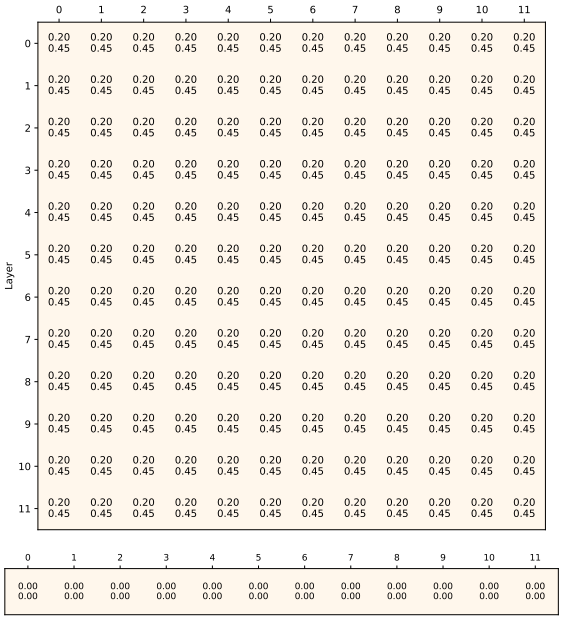



(b) S-pruning

Figure 13: STS-B




(a) M-pruning



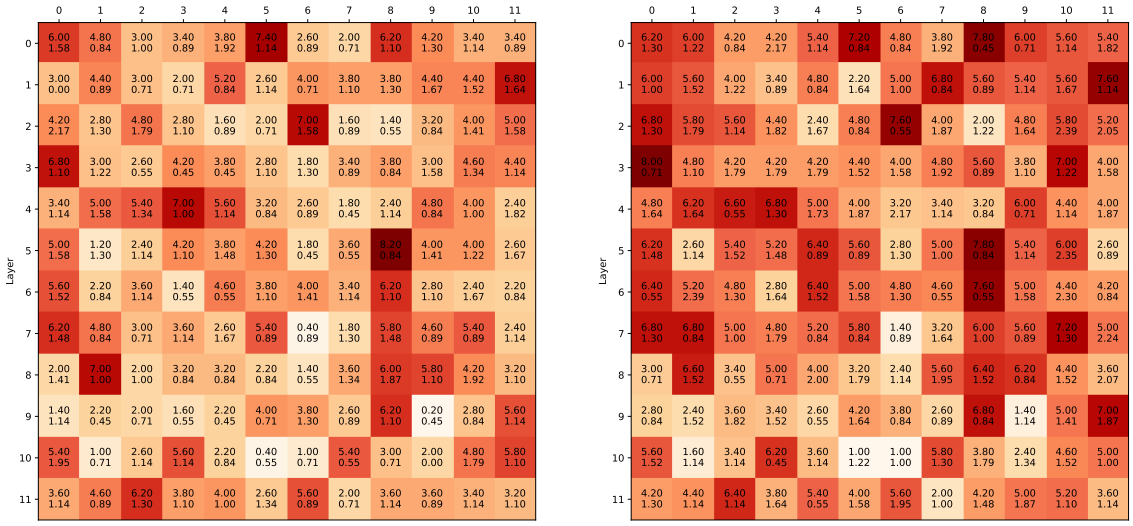

(b) S-pruning

Figure 14: WNLI

## B Iterative Pruning Modes

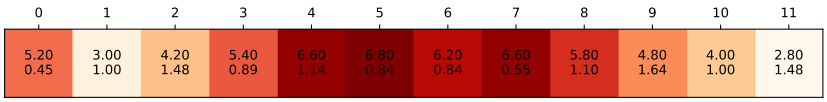
We conducted additional experiments with the following settings for iterative pruning based on importance scores:

- *Heads only*: in each iteration, we mask as many of the unmasked heads with the lowest importance scores as we can (144 heads in the full BERT-base model).
- *MLPs only*: we iteratively mask one of the remaining MLPs that has the smallest importance score (subsection 3.3).
- *Heads and MLPs*: we compute head (subsection 3.3) and MLP (subsection 3.3) importance scores in a single backward pass, pruning 10% heads and one MLP with the smallest scores until the performance on the *dev* set is within 90%. Then we continue pruning heads alone, and then MLPs alone. This strategy results in a larger number of total components pruned within our performance threshold.

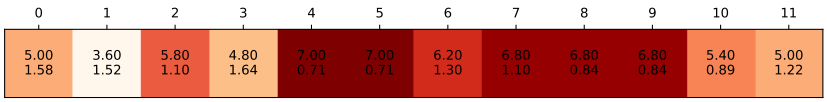



(a) Surviving heads (masking heads only)

(b) Surviving heads (masking heads and MLPs)




(c) Surviving MLPs (masking MLPs only)




(d) Surviving MLPs (masking heads and MLPs)

Figure 15: The "good" subnetworks: self-attention heads and MLPs that survive pruning. Each cell gives the average number of GLUE tasks in which a given head/MLP survived, and the standard deviation across 5 fine-tuning initializations.

## C Evaluation on GLUE Tasks

| Experiment | CoLA | MNLI | MRPC | QNLI | QQP | RTE | SST-2 | STS-B | WNLI |
|---|---|---|---|---|---|---|---|---|---|
| majority baseline | 0.00 ± 0.00 | 0.35 ± 0.00 | 0.68 ± 0.00 | 0.51 ± 0.00 | 0.63 ± 0.00 | 0.53 ± 0.00 | 0.51 ± 0.00 | 0.02 ± 0.00 | 0.56 ± 0.00 |
| full model | 0.56 ± 0.01 | 0.84 ± 0.00 | 0.84 ± 0.01 | 0.92 ± 0.00 | 0.91 ± 0.00 | 0.63 ± 0.03 | 0.93 ± 0.00 | 0.89 ± 0.00 | 0.34 ± 0.06 |
| S-pruning Subnetworks | | | | | | | | | |
| 'good' (pruned) | 0.51 ± 0.01 | 0.76 ± 0.00 | 0.77 ± 0.01 | 0.83 ± 0.00 | 0.83 ± 0.01 | 0.57 ± 0.03 | 0.84 ± 0.01 | 0.80 ± 0.01 | 0.54 ± 0.06 |
| 'good' (retrained) | 0.50 ± 0.04 | 0.80 ± 0.04 | 0.77 ± 0.06 | 0.89 ± 0.02 | 0.89 ± 0.01 | 0.58 ± 0.06 | 0.87 ± 0.07 | 0.75 ± 0.23 | 0.54 ± 0.06 |
| random (pruned) | 0.43 ± 0.11 | 0.64 ± 0.19 | 0.48 ± 0.23 | 0.69 ± 0.12 | 0.74 ± 0.07 | 0.51 ± 0.04 | 0.69 ± 0.13 | 0.53 ± 0.25 | 0.54 ± 0.06 |
| random (retrained) | 0.42 ± 0.23 | 0.79 ± 0.08 | 0.68 ± 0.21 | 0.84 ± 0.05 | 0.88 ± 0.03 | 0.56 ± 0.05 | 0.89 ± 0.01 | 0.79 ± 0.08 | 0.54 ± 0.06 |
| 'bad' (pruned) | 0.34 ± 0.09 | 0.60 ± 0.13 | 0.43 ± 0.15 | 0.63 ± 0.06 | 0.68 ± 0.06 | 0.49 ± 0.04 | 0.62 ± 0.15 | 0.32 ± 0.38 | 0.54 ± 0.06 |
| 'bad' (retrained) | 0.41 ± 0.11 | 0.80 ± 0.05 | 0.68 ± 0.14 | 0.77 ± 0.12 | 0.82 ± 0.11 | 0.59 ± 0.04 | 0.86 ± 0.06 | 0.61 ± 0.21 | 0.54 ± 0.06 |
| Importance Pruning - Super Subnetworks | | | | | | | | | |
| 'good' (pruned) | 0.13 ± 0.06 | 0.34 ± 0.01 | 0.39 ± 0.16 | 0.56 ± 0.03 | 0.63 ± 0.00 | 0.52 ± 0.02 | 0.54 ± 0.03 | 0.38 ± 0.07 | 0.54 ± 0.06 |
| 'good' (retrained) | 0.49 ± 0.02 | 0.76 ± 0.00 | 0.71 ± 0.00 | 0.83 ± 0.00 | 0.87 ± 0.00 | 0.50 ± 0.03 | 0.84 ± 0.00 | 0.80 ± 0.00 | 0.56 ± 0.00 |
| random (pruned) | 0.02 ± 0.05 | 0.32 ± 0.01 | 0.39 ± 0.16 | 0.50 ± 0.01 | 0.63 ± 0.00 | 0.47 ± 0.00 | 0.50 ± 0.01 | 0.06 ± 0.06 | 0.54 ± 0.06 |
| random (retrained) | 0.15 ± 0.15 | 0.75 ± 0.01 | 0.69 ± 0.01 | 0.76 ± 0.08 | 0.83 ± 0.04 | 0.50 ± 0.03 | 0.85 ± 0.00 | 0.12 ± 0.03 | 0.56 ± 0.00 |
| 'bad' (pruned) | 0.01 ± 0.01 | 0.32 ± 0.00 | 0.39 ± 0.16 | 0.49 ± 0.02 | 0.60 ± 0.07 | 0.52 ± 0.02 | 0.51 ± 0.02 | 0.06 ± 0.02 | 0.54 ± 0.06 |
| 'bad' (retrained) | 0.13 ± 0.03 | 0.77 ± 0.00 | 0.69 ± 0.01 | 0.57 ± 0.03 | 0.87 ± 0.00 | 0.49 ± 0.03 | 0.83 ± 0.01 | 0.13 ± 0.01 | 0.56 ± 0.00 |
| M-pruning Subnetworks | | | | | | | | | |
| 'good' (pruned) | 0.52 ± 0.01 | 0.78 ± 0.00 | 0.78 ± 0.02 | 0.84 ± 0.01 | 0.83 ± 0.01 | 0.61 ± 0.01 | 0.85 ± 0.01 | 0.82 ± 0.02 | 0.48 ± 0.10 |
| 'good' (retrained) | 0.54 ± 0.02 | 0.84 ± 0.00 | 0.84 ± 0.01 | 0.91 ± 0.00 | 0.91 ± 0.00 | 0.61 ± 0.02 | 0.92 ± 0.00 | 0.88 ± 0.00 | 0.41 ± 0.11 |
| random (pruned) | 0.02 ± 0.03 | 0.33 ± 0.01 | 0.39 ± 0.16 | 0.51 ± 0.02 | 0.63 ± 0.00 | 0.47 ± 0.00 | 0.55 ± 0.09 | 0.08 ± 0.04 | 0.51 ± 0.08 |
| random (retrained) | 0.16 ± 0.06 | 0.76 ± 0.00 | 0.70 ± 0.01 | 0.81 ± 0.01 | 0.86 ± 0.00 | 0.56 ± 0.02 | 0.83 ± 0.01 | 0.24 ± 0.03 | 0.47 ± 0.12 |
| 'bad' (pruned) | 0.00 ± 0.00 | 0.32 ± 0.00 | 0.39 ± 0.16 | 0.51 ± 0.02 | 0.63 ± 0.00 | 0.49 ± 0.03 | 0.49 ± 0.01 | 0.05 ± 0.07 | 0.53 ± 0.06 |
| 'bad' (retrained) | 0.02 ± 0.03 | 0.62 ± 0.00 | 0.68 ± 0.01 | 0.61 ± 0.00 | 0.78 ± 0.00 | 0.49 ± 0.03 | 0.82 ± 0.00 | 0.08 ± 0.00 | 0.49 ± 0.11 |

Table 3: Mean and standard deviation of GLUE tasks metrics evaluated on five seeds.

## D Longer Fine-tuning of "Bad" s-pruned Subnetworks

| Epoch | CoLA | SST-2 | MRPC | QQP | STS-B | MNLI | QNLI | RTE | WNLI | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0.422 | **0.873** | **0.71** | **0.832** | 0.651 | **0.805** | **0.764** | 0.579 | 0.498 | **0.6815** |
| 4 | 0.423 | 0.859 | 0.663 | 0.828 | 0.652 | 0.804 | 0.762 | 0.587 | **0.554** | 0.6813 |
| 5 | **0.432** | 0.862 | 0.665 | 0.831 | **0.668** | 0.801 | 0.752 | 0.590 | 0.523 | 0.6804 |
| 6 | 0.425 | 0.867 | 0.655 | 0.830 | 0.667 | 0.800 | 0.753 | **0.594** | 0.521 | 0.6791 |

Figure 16: The mean of GLUE tasks metrics evaluated on five seeds at different epochs (the best one is bolded). *Slight divergence in metrics from the previously reported ones due to this being an new fine-tuning run.

## E Performance of the "Super Survivor" Subnetworks

In this experiment, we explore three settings:

- *"good" subnetworks*: the subnetworks consisting only of "super-survivors": the self-attention heads and MLPs that survived in all random seeds, shown in Appendix A. These subnetworks are much smaller than the pruned subnetworks discussed in subsection 4.2 (10-30% vs 50-70% of the full model);
- *"bad" subnetworks*: the subnetworks the same size as the super-survivor subnetworks, but selected from heads and MLPs the *least* likely to survive importance pruning;
- *random subnetworks*: same size as super-survivor subnetworks, but selected from elements that were neither super-survivors, nor the ones in the "bad" subnetworks.

The striking conclusion is that on 6 out of 9 tasks the bad and random subnetworks behaved nearly as well as the "good" ones, suggesting that the "super-survivor" self-attention heads and MLPs did *not* survive importance pruning because of their encoding some unique linguistic information necessary for solving the GLUE tasks.

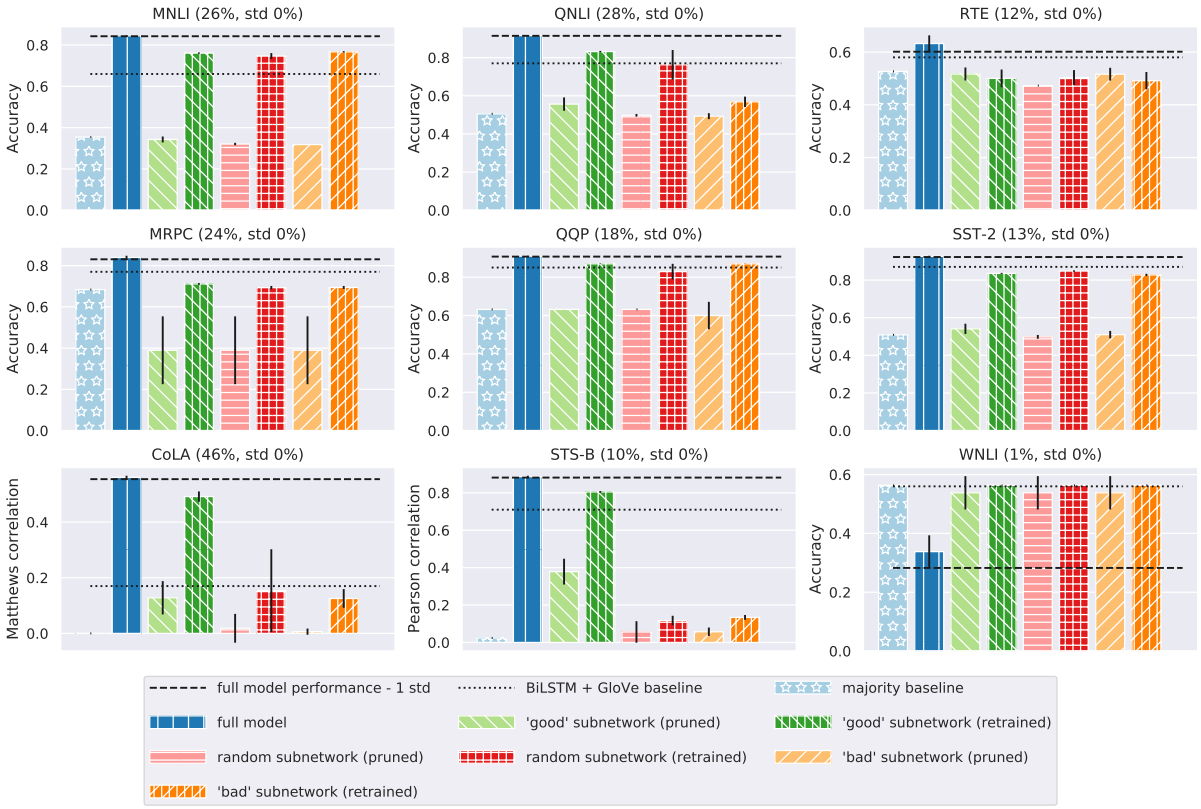


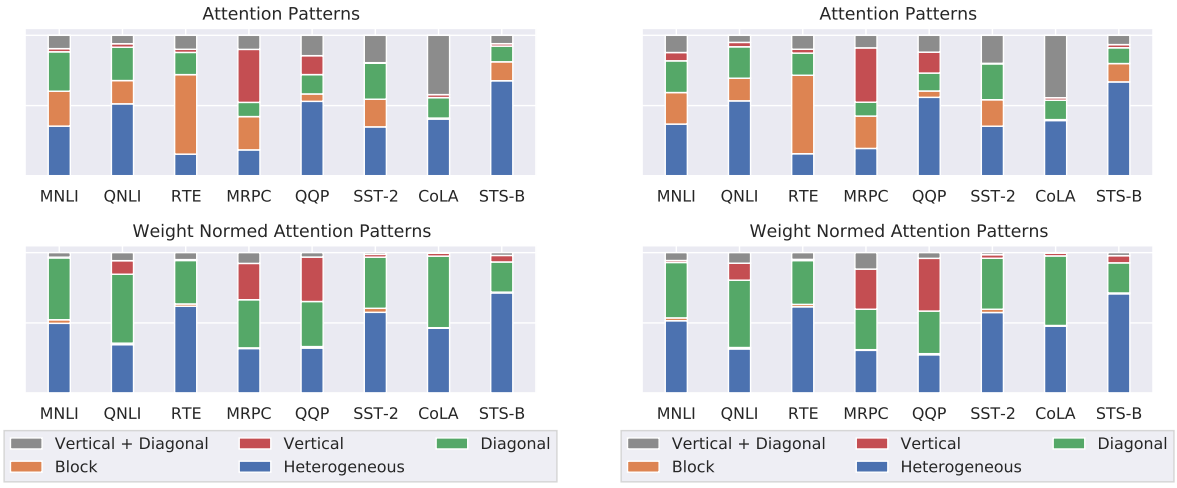

Figure 17: The performance of "super survivor" subnetworks in BERT fine-tuning: performance on GLUE tasks (error bars indicate standard deviation across 5 fine-tuning runs). The size of the super-survivor subnetwork as % of full model weights is shown next to the task names.

## F Attention Pattern Type Distribution

We use two separately trained CNN classifiers to analyze the BERT's self-attention maps, both "raw" head outputs and weight-normed attention, following Kobayashi et al. (2020). For the former, we use 400 annotated maps by Kovaleva et al. (2019), and for the latter we additionally annotate 600 more maps.

We run the classifiers on pre-trained and fine-tuned BERT, both the full model and the model pruned by the "super-survivor" mask (only the heads and MLPs that survived across GLUE tasks). For each experiment, we report the fraction of attention patterns estimated from a hundred dev-set samples for each task across five random seeds.

See Figure 4a for attention types illustration.




(a) Super-survivor heads, fine-tuned

(b) Super-survivor heads, pre-trained

(c) All heads, fine-tuned

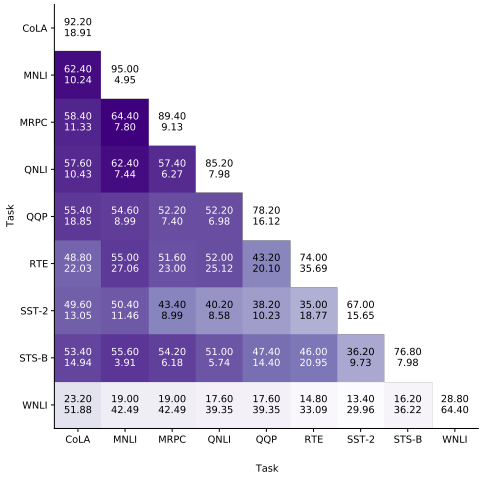
(d) All heads, pre-trained

Figure 18: Attention pattern distribution in all BERT self-attention heads and the "super-survivor" heads.

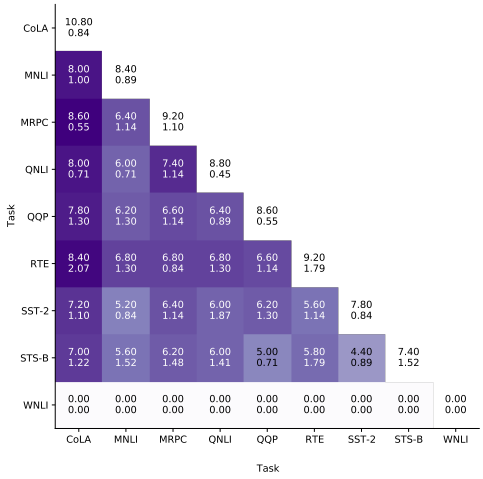
## G How Task-independent are the "Good" Subnetworks?

The parts of the "good" subnetworks that are only relevant for some specific tasks, but consistently survive across fine-tuning runs for that task, should encode the information useful for that task, even if it is not deeply linguistic. Therefore, the degree to which the "good" subnetworks overlap across tasks may be a useful way to characterize the tasks themselves.

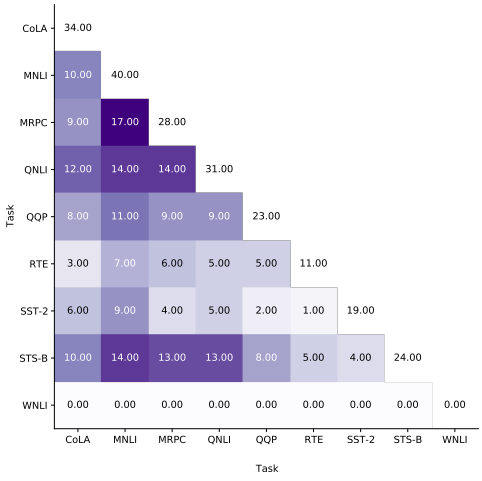
Figure 19 shows pairwise comparisons between all GLUE tasks with respect to the number of shared heads and MLPs in two conditions: the "good" subnetworks found by structured importance pruning that were described in subsection 4.1, and super-survivors (the heads/MLPs that survived in all random seeds).



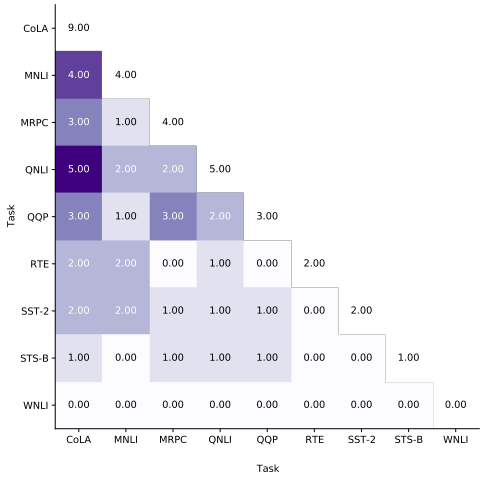

(a) Heads shared between tasks




(b) MLPs shared between tasks




(c) Super-survivor Heads shared between tasks




(d) Super-survivor MLPs shared between tasks

Figure 19: The "good" subnetwork: The diagonal represents the BERT architecture components that survive pruning for a given task and remaining elements represent the common surviving components across GLUE tasks. Each cell gives the average number of heads (out of 144) or layers (out of 12), together with standard deviation across 5 random initializations (for (a) and (b)).