Weight Correlation and Layer Effects in Pruned Feedforward and BERT Networks

RIBHAV BOSE, COLE FRANK, and ZACH HEMPSTEAD, University of Chicago, USA

It is possible to prune neural networks during or after training, resulting in accurate but smaller and sometimes more performant sparse models. However, it's not generally possible to train sparse networks from scratch without starting with a dense network. We are interested in properties of networks that have been trained and then pruned; it is possible that more analysis of such networks can inform more work on reducing the cost of producing trained sparse networks. This is especially important for transformer models, which are often large and, because they are newer, poorly-understood. Our initial approach is informed by Frankle et al. 2019, where we train small feedforward neural networks on MNIST, then prune them. We find very strong correlations between certain properties of input/output weights within nodes. We test out an initialization method that recreates these correlations and find that it may have a small positive effect. Informed by our analysis on simple feedforward networks, we prune pre-trained BERT and find similar correlations in the feedforward layers, although the strength of the correlation varies by layer. We also find correlations between W_Q and W_K within layers. Finally, we experiment with different one-shot pruning methods on BERT and testing performance on a masked language modeling task and 5 GLUE tasks. We find that global magnitude pruning performs the best at multiple sparsity levels across most tasks, and that magnitude pruning on only the lower half of layers in BERT remains extremely close in performance, with other methods exhibiting markedly worse performance.

Additional Key Words and Phrases: neural networks, large language models, sparsity, pruning, lottery ticket hypothesis

ACM Reference Format:
Ribhav Bose, Cole Frank, and Zach Hempstead. 2022. Weight Correlation and Layer Effects in Pruned Feedforward and BERT Networks. 1, 1 (December 2022), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

When new deep learning models are developed, insights into why the models are performant generally lag significantly behind the utility of the models. In particular, transformer models are large, novel, increasingly powerful, and poorly-understood. They are also expensive to train. While it is possible to prune them during training as with other models, [1] we know that the Lottery Ticket Hypothesis [2] holds for these models as well [3] meaning that there are sparse networks that will train faster and generalize well, but these subnetworks are hard to identify.

Work has been done on understanding the weight and gradient structures during initial training [4]. In particular, it has been observed that even after very little training has been done, weights are not i.i.d. - perturbing them causes significant performance degradation. It is our hope that if we can identify ways in which these weights are correlated, we can reproduce these correlations to make initial training faster, or eventually to start with sparse networks initialized in an intelligent way. To this end, we explore various types of weight-, node-, matrix- and attention-level correlations between weights of trained networks.

Authors' address: Ribhav Bose; Cole Frank; Zach Hempstead, University of Chicago, Chicago, IL, USA.

XXXX-XXXX/2022/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

We hypothesize that there will be correlations in the pruned network weights both in terms of the inputs and outputs of a given node and in terms of the self-attention query weights being correlated with the self-attention key weights. If training involves finding a subnetwork of salient paths carved out of the original dense network (as is suggested by the Lottery Ticket Hypothesis), certain nodes (or "features") that are more salient should have more non-zero inputs and outputs than other less salient nodes. Our hypothesis with regards to self-attention is that there will be strong correlations between query and key weight matrices, but weak correlations between query/value and key/value weight matrices. The role of the query/key matrices are to determine which individual embeddings should “attend” to other embeddings - if embedding E_x is highly activated by row j in W_K, and embedding E_y is highly activated by row j in W_Q, then the attention layer will transform the input embedding E_x into an output embedding with $W_V * E_y$ as a major component. In this way, the rows of W_K and W_Q are meaningfully linked, especially when an embedding “attends” to itself.

2 RELATED WORK

There is a large body of literature on sparsity and pruning of neural networks. Most relevant to our work are the original Lottery Ticket Hypothesis (LTH) paper and subsequent extensions of that research to larger and more complex networks. Frankle et al. (2020b) [5] are unable to scale the LTH as initially specified for larger networks (both in vision and NLP). Instead on these larger networks some relatively small amount of pre-training is needed before winning sub-networks emerge. They also try to test whether the weights in the pruned networks are i.i.d. by adding Gaussian noise and permuting the weights within layers and across layers before continuing to train. These perturbations result in worse performance, suggesting that the lottery ticket weights are not i.i.d. Our work builds on these findings by probing the nature of the dependence between the weights in the lottery tickets.

Chen et al. [1] specifically extend the original Lottery Ticket Hypothesis to pre-trained Bidirectional Encoder Representations from Transformers (BERT). They find that they are able to apply a basic iterative magnitude pruning (IMP) method to the pre-trained BERT model to find winning tickets for downstream tasks. In particular they show that on downstream GLUE tasks these matching tickets exist, and that the tickets emerge from the pre-trained initialization, a departure from the late-resetting method that had been explored in prior work. We adopt this approach of pruning pre-trained BERT as a jumping off point for our analysis of correlations and different pruning strategies.

Prasanna et al. [6] also explore the lottery ticket hypothesis for BERT with both a magnitude pruning method along with a structured pruning method. Their structured pruning method entails computing importance scores for the feed-forward networks and attention heads within each layer, and pruning the lowest scores. This work finds that subnetworks found from magnitude pruning tend to outperform those found from structured pruning, and magnitude pruning also tends to prune out more weights in matching tickets. We build on this literature using one-shot magnitude pruning methods, and test different heuristics related to magnitude pruning to see whether there exist easy to implement optimizations for pruning.

3 METHODS

3.1 MNIST

We train dense feedforward networks similar to Lenet-300-100 [7] on MNIST, except we use ReLU activation and no regularization, as the goals of regularization conflict with the goals of pruning. We perform magnitude pruning on each layer, although the output layer is pruned only half as much as other layers. Unless otherwise noted, we prune the output layer by 37.5% and other layers by 75%.

Pairwise Correlations. We train 100 models for 10 epochs and prune as described above. For each model, we assemble a table consisting of all pairs of weights in FC1 that share an output node, excluding pairs where either weight was pruned. We take the absolute value of all weights, and calculate the correlation between these weight

pairs. We repeat this analysis for pairs of FC2 weights that share input nodes, pairs of FC2 weights that share output nodes, and pairs of FC3 weights that share input nodes.

We perform a similar analysis with pairs of weights on opposite sides of nodes - i.e. all pairs where one weight is an input to a node in a hidden layer and the other weight is an output of the same node.

Reported metrics are the averages taken over all 100 models.

Node Analysis. We look at correlations between the following pairs of properties of nodes in hidden layer 1:

- The number of unpruned input/output weights
- The average value of nonzero input/output weights
- The average absolute value of nonzero input/output weights
- The fraction of nonzero weights that have a positive sign in inputs vs outputs

We report these values for a single (uncurated) model, and produce scatterplots for this model to visually demonstrate correlations. This analysis is repeated for nodes in the second hidden layer.

Initialization With Correlations. In the previous section, it is clear that in trained and prune networks, there is a strong positive correlation between the average absolute value of a node's inputs and its outputs. We investigate whether initializing weights with these correlations improves initial training. We accomplish this in the following manner, parameterized by the correlation factor f:

(1) Randomly initialize weights as normal
(2) Choose a value for hyperparameter $f \geq 1.0$
(3) Assign each hidden node i a value $s_i \sim \text{Uniform}(1.0, f)$
(4) With a 50% chance for each node, invert s_i (i.e. $s_i = \frac{1}{s_i}$)
(5) Multiply each weight connected to a hidden node by s_i (the values in the second weight matrix are multiplied by two different s_i)

The higher the value of f, the higher the average correlation of the absolute values of input/output weights within a node. If f is 1.0, the original initializations don't change.

We initialize 50 models, and for various values of f we update the initial weights as described above and train for a quarter of an epoch. Then, we evaluate the model's accuracy on the test set. We repeat this analysis using normal Kaiming initialization (as opposed to the default, uniform Kaiming), and also with `tanh` activation (as opposed to ReLU).

3.2 BERT

All of our experiments with the BERT architecture use the pre-trained HuggingFace implementation of 'bert-base-uncased'[1]. This model was pre-trained on two tasks–masked language modeling (MLM) and next sentence prediction (NSP)–across two datasets: BookCorpus and English Wikipedia. It has 110M parameters and is composed of 12 stacked encoder layers, each of the form shown in figure 1

3.2.1 BERT Correlation Analysis. For this part of the analysis we started by one-shot pruning the smallest 50% of each of the non-embedding parameter matrices of the pre-trained bert-base. We then separately analyzed the self-attention and feed-forward portions of each encoder layer for correlations between the remaining weights.

[1]https://huggingface.co/bert-base-uncased

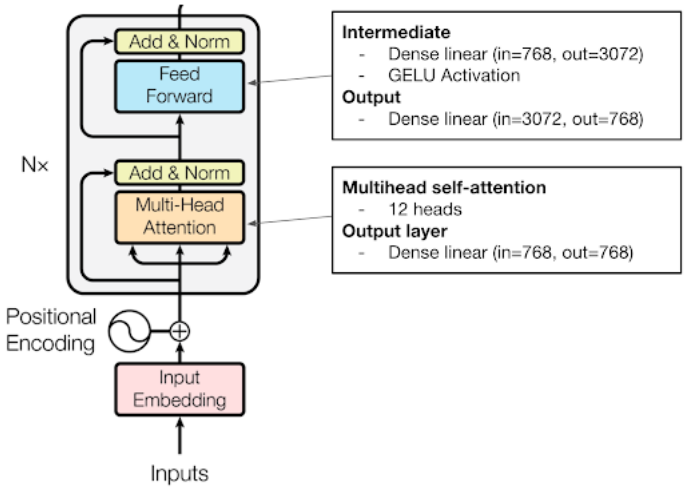

Fig. 1. Diagram of the encoder architecture in a bert-base-uncased

3.2.2 BERT Pruning Methods. From Prasanna et. al. [6], there is some indication that a structured pruning approach for BERT performs worse than an unstructured magnitude pruning process. However, since global magnitude pruning seems to be the main way of implementing a magnitude pruning process, we seek to answer whether this is the best way to magnitude prune large networks. It is possible that given the size of these networks, layers near the bottom of the network may be doing less work than those at the beginning of the network, and that pruning these bottom layers at a higher rate might result in improved pruned network performance. In order to compare network performance, we use five GLUE tasks and a masked language modeling task. In particular we use the MRPC, STSB, QNLI, MNLI, and SST2 tasks from GLUE, along with an MLM task using the wikitext dataset.

To test our hypothesis we test 4 different pruning methods, all at uniform sparsity levels.

(1) One-shot global magnitude pruning across all encoder layers in BERT
(2) One-shot magnitude pruning on ONLY the first 6 encoder layers
(3) One-shot magnitude pruning on ONLY the last 6 encoder layers
(4) One-shot global random pruning, serving as our random subnetwork benchmark

Note that if we prune 10% of weights in method 1, this would correspond to 20% of weights being pruned out in methods 2 and 3, to achieve uniform global sparsity across pruning methods.

At sparsity levels above 50%, since we cannot prune more than > 100% of either the first 6 or last 6 encoder layers, we instead move to a hybrid approach, which we will term 'mixed pruning'. In line with our hypothesis that lower layers do less work than upper layers, we prune the last 6 layers at a higher rate than the first 6 layers. For example, if we want to compare to a network that has 70% weights pruned by method 1, we create a subnetwork by pruning 60% of the weights in the first 6 layers, and 80% of the weights in the bottom 6 layers.

Table 1. MNIST Pairwise Correlations

Metric	Weight matrix			Node layer	
	FC1	FC2	FC3	Hidden 1	Hidden 2
Weights that share an output node	0.049	0.041	-	-	-
Weights that share an input node	-	0.142	0.155	-	-
Node input/output	-	-	-	0.060	0.056

Table 2. MNIST Node Correlations

Metric	Node layer	
	Hidden 1	Hidden 2
# of nonzero input vs output weights	0.930	0.804
# of nonzero input vs output weights (excl. nodes w/ all 0s)	0.556	0.804
Magnitude of nonzero input vs output weights	0.075	-0.132
Absolute magnitude of nonzero input vs output weights	0.822	0.847
Fraction of nonzero weights > 0 in input vs output	0.035	0.324

4 RESULTS

4.1 MNIST

4.1.1 Pairwise Correlations. Results reported in Table 1.

4.1.2 Node Analysis. Results reported in Table 2, and scatterplots are in Fig 2. Despite performing only absolute magnitude pruning within each layer, 113 out of 300 nodes in the first hidden layer had *all* their input and output weights set to 0.

4.1.3 Initialization With Correlations. See results at Table 3.

For our initial set ReLU/uniform models, running a t-test for $\pi_{f=1.1} - \pi_{f=1.0} > 0$ results in $p = 0.044$.

We realized that there was a source of random error that we weren't controlling for - each quarter epoch training run was looking at a random subset of the training data, in random order. This doesn't invalidate these analyses, but we were optimistic that if we reran the analyses controlling for this, we would achieve lower variance and thus more statistical power.

We reran the analysis using the default initialization and values of f ranging from 1.0 to 1.25, this time serving the same training data in the same order across all runs (with various f-values) for a single model. Despite the greater power the differences were not statistically significant: $\pi_{f=1.15} - \pi_{f=1.0} > 0$ with $p = 0.100$.

We also ran the same analysis on a version of the model with `tanh` activations instead of ReLU, and again did not get statistically significant results. $\pi_{f=1.05} - \pi_{f=1.0} > 0$ has $p = 0.316$.

4.2 BERT

4.2.1 BERT Correlation Analysis.

Feed-Forward Network. In the feedforward component of the BERT layer we found similar node-level input-output correlations to those in the simple feedforward network we trained on MNIST. Specifically we found strong correlations between the inputs and outputs of a node in terms of the count of non-zero inputs and

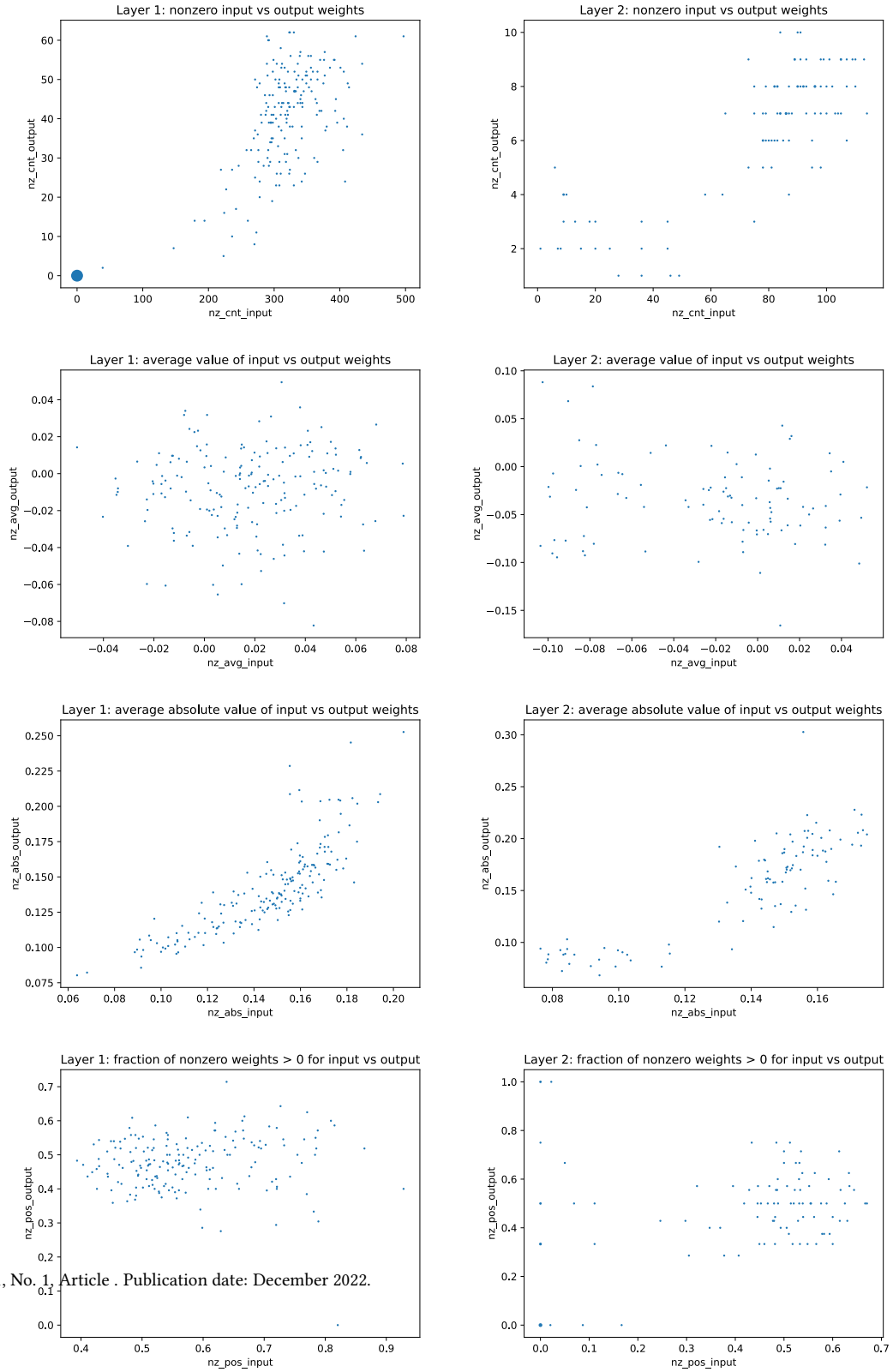

Fig. 2. MNIST Node property scatterplots

Table 3. MNIST Initialization with Correlations

f	Average test accuracy			
	No control for train samples		Control for train samples	
	ReLU, uniform	ReLU, normal	ReLU, uniform	`tanh`, uniform
1.0 (baseline)	0.828	**0.799**	0.832	0.897
1.05	-	-	0.830	**0.898**
1.1	**0.842**	0.781	0.836	0.897
1.15	-	-	**0.839**	0.897
1.2	-	-	0.830	0.897
1.25	0.831	-	0.828	0.896
1.5	0.835	-	-	-
2.0	0.806	-	-	-
2.5	0.802	-	-	-
3.0	0.766	-	-	-

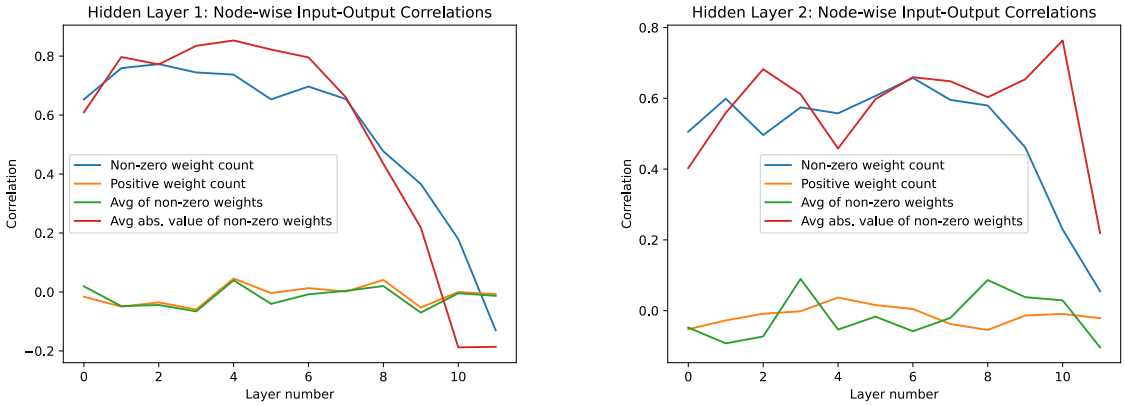

Fig. 3. Input-Output Correlations in the BERT Feed-Forward Networks

the average absolute value. The positive weight count and average values were uncorrelated. Interestingly the strong positive correlations for non-zero count and absolute value both decline from being very strong positively correlated in the early layers to being mildly negatively correlated by the final layer of the network. Fig 3. shows that this same general pattern holds for both of the hidden layers in the feed forward portion of the network.

Self-Attention. Next we turned our attention to the self-attention portion of each layer. We found that within each layer the query and value matrices tend to be correlated with each other, whereas the value matrices are correlated with neither the query matrix nor the key matrix (Fig 4).

These correlations are relatively modest compared to the input/ouput correlations we found in the feedforward networks, ranging from -0.05 to nearly 0.2, and generally becoming more positively correlated as we progress deeper into the network.

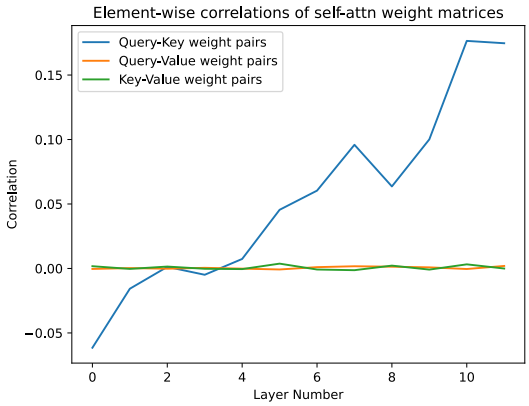

Fig. 4. Self-attention

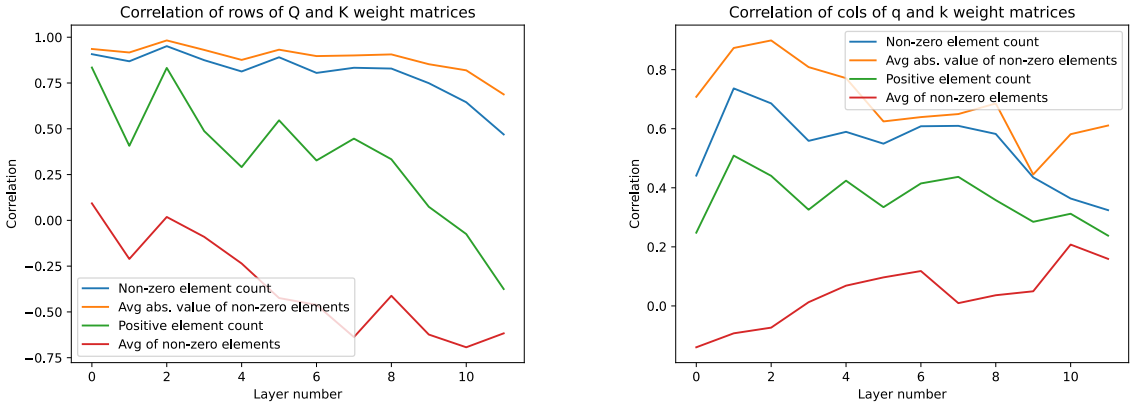

Fig. 5. Self-Attention Row and Column Analysis

The element-wise correlations above neglect any internal structure of the matrix. So next we analyzed correlations between the rows of the different self-attention weight matrices within a given layer. For each matrix we calculated 4 summary statistics for each row of the matrix: count of non-zero elements, count of positive elements, mean value, mean absolute value. For each of the three weight matrices this yielded 4 vectors of length 768 (the model dimension). For each summary statistic we checked the correlations of each of the 3 possible pairings of weight matrices (query-value, query-key, value-key). We found strong correlations between the summary statistic vectors of the query and key weight matrices. We did the same analysis for columns of the weight matrices and found weaker correlations: (see Fig 5)

4.2.2 *BERT Pruning Methods.*

Low Sparsity Networks. In Fig 6. we see that in general the global magnitude pruned network tends to perform the best across the baseline tasks within a given sparsity bucket. We also notice that at sparsities >=0.3, the

	MRPC	STSB	QNLI	MNLI	SST2	MLM Accuracy	MLM ppl.
full-base	0.857	0.876	**0.897**	0.821	**0.920**	0.512	22.003
full-0.1	0.853	0.877	0.895	0.820	**0.919**	**0.513**	**21.843**
random-0.1	0.814	0.871	0.885	0.808	0.913	0.462	29.731
upper-0.2	**0.863**	0.877	0.896	0.821	0.915	0.512	22.009
lower-0.2	0.853	**0.879**	**0.897**	**0.822**	0.916	0.512	21.899
full-0.3	**0.863**	0.878	**0.894**	**0.821**	**0.917**	**0.507**	**22.514**
random-0.3	0.754	0.759	0.813	0.766	0.893	0.330	89.841
upper-0.6	0.786	0.869	0.881	0.810	0.915	0.467	30.003
lower-0.6	0.841	**0.885**	0.890	0.816	0.912	0.500	23.197
full-0.45	**0.839**	**0.878**	**0.890**	**0.814**	**0.915**	**0.503**	**23.161**
random-0.45	0.761	0.296	0.630	0.700	0.837	0.252	184.569
upper-0.9	0.738	0.096	0.608	0.654	0.841	0.168	471.304
lower-0.9	0.834	0.876	0.851	0.780	0.905	0.402	43.325
full-0.495	**0.828**	**0.879**	**0.886**	**0.816**	**0.911**	**0.495**	**24.739**
random-0.495	0.744	0.277	0.623	0.690	0.838	0.238	226.522
upper-0.95	0.748	0.082	0.612	0.691	0.851	0.129	703.914
lower-0.95	0.822	0.875	0.850	0.774	0.905	0.393	47.630

Fig. 6. Comparing task performance for low sparsity networks across the 4 pruning methods

lower-pruned network outperforms the upper-pruned network across all tasks, and that in some cases the randomly pruned network outperforms the upper-pruned network. These results back up the idea that upper networks in general are more important for model performance than lower layers. We also notice that across all sparsities, the lower-pruned network has performance relatively close to the full-pruned network across all tasks, with the exception of MLM at sparsities >=0.45. The dropoff in the MLM task in particular could be due to BERT being trained on an MLM task, and thus pruning a large amount of weights even in the lower-layers can cause a larger dropoff in performance in this task compared to others.

High Sparsity Networks. Building off of the above results, in Fig. 7. there are cases of the mixed-prune networks exhibiting the highest performance in a given sparsity bracket, particularly for the MRPC, STSB, and QNLI tasks. Interestingly, the mixed-pruned networks within a given sparsity bucket tend to perform at similar levels, even with slight changes in the amount of each half the model that is pruned. It is however important to note that when we get to sparsities >=0.5, the dropoff from base model performance gets extremely large, which means that the practicality of implementing such a heuristic on an LLM likely yields little to no benefit.

5 DISCUSSION

Our results on whether initializing weights in a correlated manner may help in early training are inconclusive. Our analysis with less random error (and thus more statistical power) did not show a statistically-significant improvement, so it is possible that our initial significant result was a false positive. It may warrant more follow-up, however. Our analysis indicates that the feedforward layers of BERT shows similar correlation patterns to small feedforward networks trained in a different domain. Within attention layers, there are strong correlations in pruned vs unpruned and absolute value of weights between W_Q and W_K, as we predicted. Finally, we find that while global magnitude pruning tends to perform the best on baseline tasks, magnitude pruning focused on the lower layers of BERT shows promise as a simple heuristic to cut down model size. However, due to the

	MRPC	STSB	QNLI	MNLI	SST2	MLM Accuracy	MLM ppl.
full-base	**0.857**	**0.876**	**0.897**	**0.821**	**0.920**	**0.512**	**22.003**
full-0.7	**0.760**	**0.810**	**0.837**	**0.773**	**0.874**	**0.322**	**99.662**
random-0.7	0.735	0.154	0.615	0.605	0.815	0.150	574.143
upper-0.6-lower-0.8	0.741	0.160	0.617	0.640	0.836	0.129	668.620
upper-0.66-lower-0.74	0.729	0.176	0.614	0.648	0.827	0.132	663.858
full-0.9	0.725	0.138	0.594	0.354	**0.836**	**0.207**	**327.695**
random-0.9	0.748	0.064	0.608	0.553	0.814	0.152	626.551
upper-0.83-lower-0.97	0.748	0.150	0.609	**0.598**	0.813	0.150	603.192
upper-0.86-lower-0.94	**0.758**	**0.168**	**0.609**	0.594	0.826	0.148	596.855
full-0.95	0.739	0.084	**0.611**	**0.631**	**0.833**	**0.203**	**379.285**
random-0.95	0.748	0.073	0.606	0.548	0.811	0.150	651.032
upper-0.92-lower-0.98	0.748	0.108	0.607	0.587	0.823	0.150	608.311
upper-0.936-lower-0.964	**0.753**	**0.136**	**0.611**	0.566	0.820	0.150	599.360
full-0.99	0.748	0.034	0.505	0.566	0.817	0.187	483.925
random-0.99	0.748	0.070	0.610	0.537	0.821	0.133	768.780

Fig. 7. Comparing task performance between global magnitude pruned, randomly pruned, and mix pruned networks

performance dropoff at high sparsities with one-shot pruning in transformer models, there is likely little benefit in implementing these heuristics.

6 CONCLUSION

In this paper we analyzed weight-, matrix-, and node-level correlations of trained and pruned MLP and BERT models. We discovered some very strong correlations in both fully-connected and transformer layers. We tested a new initialization strategy based on these observations, with somewhat inconclusive results. Finally, we analyzed layer pruning strategies for BERT and found that we could improve over the baseline global pruning by focusing on certain layers. Our analyses improve understanding of Transformer models in the context of pruning and lottery tickets.

REFERENCES

[1] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained BERT networks. *CoRR*, abs/2007.12223, 2020.
[2] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018.
[3] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
[5] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. *CoRR*, abs/2002.10365, 2020.
[6] Sai Prasanna, Anna Rogers, and Anna Rumshisky. When BERT plays the lottery, all tickets are winning. *CoRR*, abs/2005.00561, 2020.
[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.