Gated ViT structured pruning, with a lottery ticket hypothesis perspective

Marc Alwan, Krishna Deshpande, Marcin Anforowicz
University of Washington
{malwan, kdeshpan, manfor}@cs.washington.edu

July 14, 2026

ABSTRACT

Unstructured, global pruning is highly effective at reducing parameter count while maintaining effective test-time accuracy. Yet, it is not used in practice, as improvements in inference times are limited despite the lower parameter count due to the sparsity of the final model. In contrast, structured pruning yields a denser, more efficient model but often cannot capture the same level of performance at similar pruning ratios. We look at this problem from the perspective of the lottery ticket hypothesis and test various structured pruning procedures across different modules (attention heads, entire layers, and groups of neurons) in a ViT to discover which granularities can maintain test time performance while still producing a dense model. Our approach multiplies each module's output by a learnable L1-regularized gate to encourage sparsity during training. We find that pruning 95% of attention heads results in minimal accuracy loss, in line with prior work.

1 INTRODUCTION

The lottery ticket hypothesis paper shows that one can strategically remove most parameters in a randomly initialized neural network, train only the remaining sub-network, and get performance comparable to the full network (Frankle & Carbin, 2019). This lucky remaining combination of parameters is called the "winning ticket". A "winning ticket" corresponds to a set of dimensions that SGD can descend on to get good performance. The lottery ticket hypothesis appears to apply to most deep networks, including models such as BERT transformers (Chen et al., 2020).

The lottery ticket hypothesis implies that most parameters in a neural network are not strictly necessary. So why not just train a smaller model? Chang et al. (2020) shows that starting with a large model and pruning it results in significantly higher accuracies than training a small model directly.

A possible explanation for this is that small models have less sub-networks, reducing the likelihood that one of them is a winning ticket. The number of possible sub-networks scales roughly $O(2^n)$ with parameter count n. So a large model is more likely to have winning tickets, and pruning isolates them, while discarding the rest of the neural network. Unstructured pruning removes individual weights and biases from a neural network. It has been shown to reduce the model parameter count by over 95% (LeCun et al., 1989).

Unfortunately, unstructured pruning is not without its faults. Unstructured pruning sets elements of weight matrices and bias vectors to zero. This results in sparse matrices which generally aren't any faster at inference on GPUs than non-sparse matrices.

A more GPU-friendly alternative is structured pruning, which removes chunks of a model at a time, allowing entire matrix multiplications to be eliminated. A concern with structured pruning is that it causes a larger decline in accuracy for the same number of parameters removed. From the perspective of the lottery ticket hypothesis, if the number of prunable modules is m, then there are

$O(2^m)$ possible resulting sub-networks. As m decreases, the probability that one of the possible sub-networks contains a winning ticket decreases. Nevertheless, structured pruning can still achieve good results. We hypothesize that with the correct approach, structured pruning can approach the accuracy of unstructured pruning.

To test this hypothesis and better understand how pruning granularity (the size of modules being pruned) impacts accuracy, we present experimental results of structured pruning at different granularities in a ViT model trained on the CIFAR-100 and tiny-imagenet. We test pruning attention heads, entire layers, and 16 and 32 neuron sized chunks of projection layers.

2 Related Works

Network Slimming. (Liu et al., 2017) This is a structured pruning procedure that removes entire convolution channels from a CNN. It assumes that the inputs to each convolution are passed through a BatchNorm layer with a learnable scale parameter γ. They L1 regularize this scale parameter and remove convolutions with the lowest scale parameter magnitudes, setting the outputs to zeros.

Structured pruning of LLM attention heads. Michel et al. (2019) show that most attention heads in transformer models can be removed after training with minimal drop in performance. Through head masking experiments on machine translation and natural language inference tasks, they find that many heads contribute little to the final prediction. Similarly, Voita et al. (2019) used linguistic analysis to show that attention heads exhibit varying levels of importance. While a few specialized heads capture syntactic and positional information, the majority can be pruned with negligible degradation.

Structured pruning of ViTs. Yu & Wu (2021) prune attention heads, MLP dimensions, embeddings, and other components. They calculate an importance score for each channel by temporarily masking it out and computing the KL-divergence between the output logits of the original, unpruned model and the masked model. Channels that cause a high KL-divergence when removed are retained, whereas others are pruned.

3 Granular Pruning Procedure

3.1 Overview

Our pruning approach is closely inspired by the *Network Slimming* procedure. Our approach does not require BatchNorm, works on architectures beyond CNNs such as Transformers, and is designed to leverage skip connections. We take a module with a skip connection, and add a gate to its output. The gate has a single parameter that multiplies the outputs before they are added to the skip connection. If the gate parameter goes to zero, the module is entirely bypassed by the skip connection. In this case, the module can be safely removed without affecting the neural network.

To encourage gate parameters to go to zero, we apply L1 regularization to them. This by itself leads internal module weights to grow, countering this effect. To discourage internal parameters from growing too far, we apply regularization on them too. As we aren't trying to achieve sparsity within the modules, we use L2 regularization for this purpose.

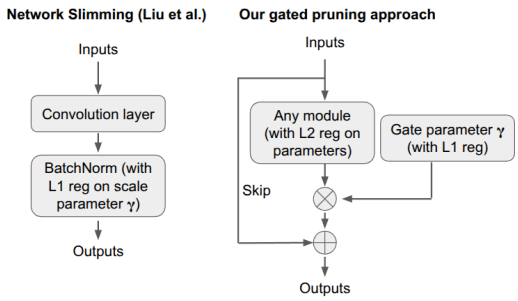


Figure 1: Accuracy of different-sized models.

3.2 PRUNING PROCEDURES

We employ three major variations for our pruning procedure:

1. Train the model to convergence while freezing all gate parameters to 1. During this pre-pruning training phase, no L1 loss or gradients are propagated to the gate parameters (but L2 regularization is used on the rest of the parameters). After the model has converged, unfreeze all the gate parameters, and train the model while applying L1 loss to the gate parameters and backpropagating the loss. Every n epochs, "prune" some percentage of the remaining groups, iterating over this process until a desired pruning ratio is achieved, where n is a hyperparameter.
2. Train the model to convergence while applying L1 loss to all gate parameters (and L2 regularization for the rest of the parameters). After the model has converged, train the model while applying L1 loss to the gate parameters and backpropagating the loss. Every n epochs, "prune" some percentage of the remaining groups, iterating over this process until a desired pruning ratio is achieved.
3. From the very beginning of the training procedure, apply L1 loss to all gate parameters (and L2 regularization for the rest of the parameters) and every n epochs, "prune" some percentage of the remaining groups, iterating over this process until a desired pruning ratio is achieved.

While these three pruning procedures are similar, there are significant differences between the three of them.

Procedure one is most effective for large models that have already been trained, because the pruning procedure is entirely post-hoc. Even if the original model architecture was not designed for this pruning procedure, the required gate parameters can be inserted into the model afterward, and the iterative pruning and fine-tuning procedure can still be run.

For models that are trained with the pruning procedure in mind from the very beginning, procedures two and three may be a better fit. During the main training procedure, L1 loss is applied from the very beginning, helping to earlier isolate sub-networks that do not contribute as much to the accuracy of the model. However, procedure two only prunes groups after the original model has fully converged, while procedure three iteratively prunes groups during the main training procedure. In this manner, procedure three more closely mimics the original lottery ticket hypothesis pruning approach Frankle & Carbin (2019), though with some differences (such as not re-initializing weights from scratch). However, procedure two may better take advantage of the sparsity produced by the L1 regularization, because it is allowed to train to convergence before any pruning takes place.

4 EXPERIMENTAL RESULTS

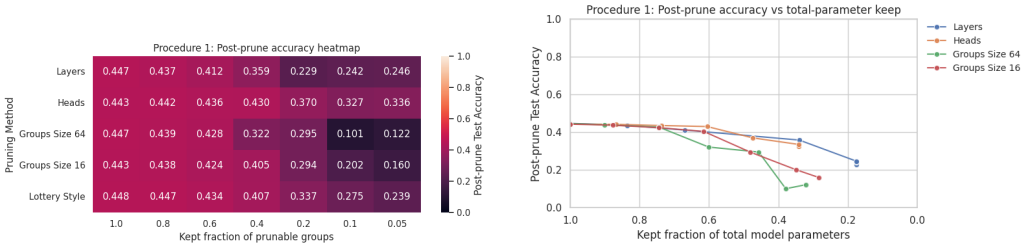
4.1 EXPERIMENT SETUP

To evaluate our pruning procedures, we construct instances of the same ViT model that vary according to the pruning granularity we wish to prune at. We list these below:

- **Layer-wise pruning:** We attach the gate parameter directly to the output of each transformer block, which contains the attention mapping and projection mechanism, along with a feed-forward network that follows it.
- **Head-wise pruning:** We attach the gate parameter to the output of each head in the transformer blocks. Because our ViT model applies multi-headed attention, there are multiple gate parameters per layer. Note that because the feed-forward network does not contain any heads, the pruning ratio achievable by this type of pruning is lower than that of the other pruning approaches (which is reflected by our results).
- **Grouped-neuron pruning:** We design a custom linear module that groups neuron outputs into specific sizes, depending on how it is parameterized. For each group in the layer, we attach a gate parameter. We note that this implementation does not contain a skip connection, which may lower performance. We use group sizes of 64 and 16.

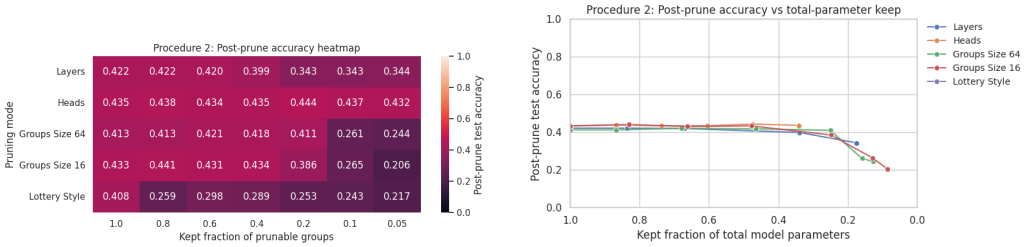
- **Unstructured lottery ticket pruning:** We apply the iterative, unstructured pruning approach described by the original lottery ticket paper. For this, all gate parameters are frozen.

To test the different granularities, we apply our three different procedures to the openly available CIFAR-100 dataset and plot the results below. The graphs on the left-hand side show test accuracies for the various granularities after the procedure has been completed, demonstrating the effects of removing fractions of the prunable groups. The graphs on the right-hand side show similar values, but the x-axis instead shows the true fraction of parameters that have not been pruned (since the various granularities may not all achieve the same exact scarcities due to the implementation details).



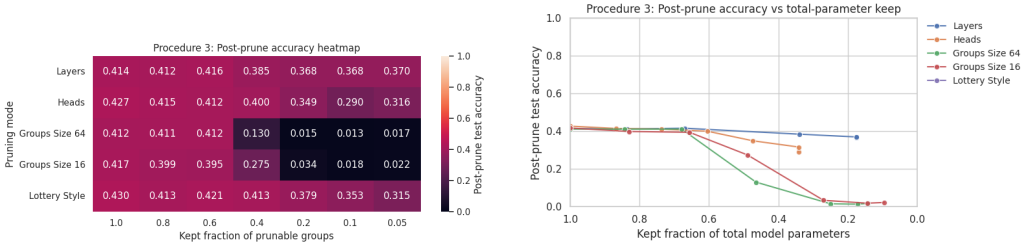
(a) Fraction of total groups that can be pruned (b) True fraction of model parameters

Figure 2: Train to convergence (with no gate L1 loss), then iteratively prune/train (with gate L1 loss)



(a) Fraction of total groups that can be pruned (b) True fraction of model parameters

Figure 3: Train to convergence (with gate L1 loss), then iteratively prune/train (with gate L1 loss)



(a) Fraction of total groups that can be pruned (b) True fraction of model parameters

Figure 4: Iteratively prune and finetune with gate L1 loss

5 Discussion

5.1 Analysis

Remarkably, we find that the structured pruning approaches often achieve better test accuracies at finer granularities. This is more pronounced in procedures one and three, where there is a steep

drop-off in accuracy for the finer granularities with group sizes 64 and 16. Surprisingly, even the iterative globally unstructured pruning performs slightly worse than the more structured pruning methods. This is rather unexpected, as unstructured pruning often performs better than structured pruning, due to its ability to more closely isolate sub-networks in the larger network.

However, this particular behavior may be due to the nature of transformers. Previous works, such as Voita et al. (2019) and Michel et al. (2019), find that more attention heads do not necessarily improve performance. In fact, removing heads and leaving behind more specialized heads can even perform better. Some researchers even find that layers can be reduced to just one head, using greedy and magnitude-based approaches. This also suggests that entire layers can be removed if all heads in that layer do not perform well. This supports our results. Due to the inherent structure of transformers, large swaths of heads without degrading performance too much. In procedure two, nearly 95 percent of heads are removed without seriously degrading performance.

In contrast, the grouped-neuron pruning performed very poorly, especially in procedure three. We suspect this is due to a few factors. Since there are many more groups with gate parameters, it is possible that the L1 normalization loss fails to isolate the most and least important groups, leading to poorer pruning selections. It may also be more sensitive to the L1 hyperparameter constant. Furthermore, the implementation of the custom grouped-neuron linear layer simply sets the output of pruned groups to 0, and with no residual connection, bottlenecks may form in how the data flows through the network.

5.2 Limitations

Due to limited compute, and the large number of model configurations needed to be trained, it was intractable to perform an extensive hyperparameter search for our plots. It is possible that certain comparisons with the different procedures could have yielded different results with more effective hyperparameters. Likewise, larger and higher-dimensional datasets are similarly intractable, due to the significantly longer training times. However, we also performed the same experiments on the tiny-imagenet dataset with similar results (plots omitted for brevity).

6 Conclusion

In this paper, we tested the accuracy decline from pruning at a granularity of attention heads, transformer layers, and neuron groups of size 32 and 64. We compared this to unstructured lottery-ticket-hypothesis-style pruning. We observed that pruning high ratios of ViT heads and layers is possible with very little accuracy impact. This supports training large models and structurally removing the majority of their parameters as a way of balancing high accuracy and generalization with small model size and GPU-friendly performance.

Furthermore, while decreasing the number of modules may decrease the number of potential pruned configurations available, with coarser granularities, there exists a number of parameter configurations per each module. This significantly increases the number of subnetworks selectable at higher granularities. This is backed by the results, which suggest, in contrast to our initial assumptions, pruning at coarser granularities does not substantially reduce the number of winning tickets that may be selected.

For future work, we would like to run sweeps across hyperparameters such as the L1 gate regularization strength, number of finetuning epochs, and more. Additionally, quantifying the performance-accuracy tradeoff between pruning and the closely related mixture-of-expert approach could be insightful. Finally, it might be useful to compare the size compression achievable via pruning to that achieved via distillation, along with comparing models directly trained with the same parameter as the pruned models.

7 Our earlier experiments

We arrived at our structured pruning direction near the end of the quarter. Prior to that, we ran a collection of related experiments studying why large models generalize better to unseen data. A real

research paper would focus just on the final direction, but as this is a capstone, we're including notes from our other experiments here, in a blog post style.

7.1 LARGE MODEL PERFORMANCE

"Deep double descent" is the observation that large overparameterized models tend to overfit less and generalize better to new data (Nakkiran et al., 2019) than small models. This can be counterintuitive, as large models are theoretically capable of overfitting training data in much more complex ways that don't generalize.

To observe double descent, we trained 3 CNNs and 3 ViTs of varying sizes. To make overfitting clearer, we trained on a subset of CIFAR-10, with just 6000 training images. For fairness, we trained each model with similar hyperparameters (see our code) until maximum validation accuracy did not increase for 3 epochs.

We used a basic CNN architecture composed of a series of convolution layers followed by a linear layer. Each convolution layer is a 3x3 convolution, followed by a 2D batchnorm, ReLU, and 2x2 max pool. The three sizes we used were:

Table 1

Model	Layer channel counts
Small CNN	3, 8
Medium CNN	3, 16, 32
Large CNN	3, 128, 256, 512

Our vision transformer is also straightforward. It splits the image into 4x4 patches, which are projected to an embedding dimension, with added learned positional embedding. The backbone is a stack of transformer layers, each with multi-head self-attention and a 2-layer perceptron. Classification is done by average-pooling across patch tokens, flattening, and then a linear output layer for 10 classes. The three sizes we used were:

Table 2

Model	Embedding dimension	Num layers	Num heads	MLP hidden layer size ratio
Small ViT	32	4	2	2.0
Medium ViT	64	6	4	4.0
Large ViT	128	8	8	4.0

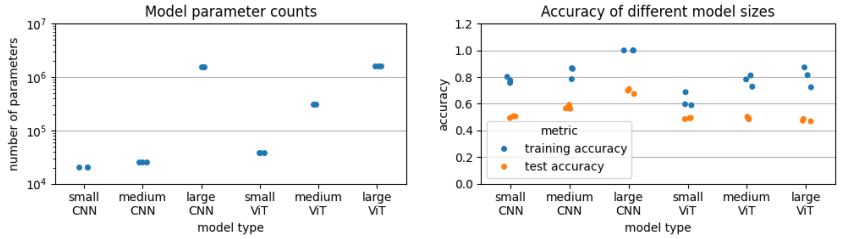


Figure 5: Accuracy of different-sized models.

Our results show that adding more parameters to our CNN improved its training and test accuracy. However, adding more parameters to our ViT only improved training accuracy and didn't affect test accuracy. While behavior differed across architectures, adding more parameters didn't reduce test accuracy in any case.

7.2 GENERALIZATION INVESTIGATION

We attempted to investigate why increasing model size doesn't hurt generalization. Our hypothesis was that large models can descend more directly to a solution, whereas small models meander more, memorizing and overfitting along the way.

Valle-Pérez et al. (2019) argue that random initialization (He et al., 2015) is biased to sample simple functions, as they occupy larger volumes of the parameter space than complex functions. Also, infinitely wide neural networks are like Gaussian processes (Lee et al., 2018) which output similar values for similar inputs, forming a smooth, simple function. Thus, we view initialization as a simple starting point from which training progresses.

Nagarajan & Kolter (2019) says that SGD on deep networks tends to settle in solutions close to initialization, rather than searching the entire hypothesis space.

Choromanska et al. (2015) says that overparametrization increases the amount of good local minima. Suboptimal critical points are likely to be saddle points, simply because there are so many dimensions in the loss landscape. Similarly, Garipov et al. (2018) says that overparametrized models tend to have more connected minima, with fewer suboptimal minima.

We interpret this through the lens of the lottery ticket hypothesis. We hypothesize that overparameterization helps because it provides more trainable subnetworks (winning lottery tickets) near initialization, reducing the amount of movement required to reach a good solution. On the other hand, a small model without a winning ticket has to meander longer through the loss landscape to find a solution, which causes overfitting, and takes it further from its simple initialization, more likely to a sharp minima.

To test if this is the case, we tracked measures of descent directness during training for different networks. Our "cosine directness" computes the cosine similarities between adjacent steps and takes the average of this for each epoch. We also calculate the RMS of the difference between parameters at initialization and after training.

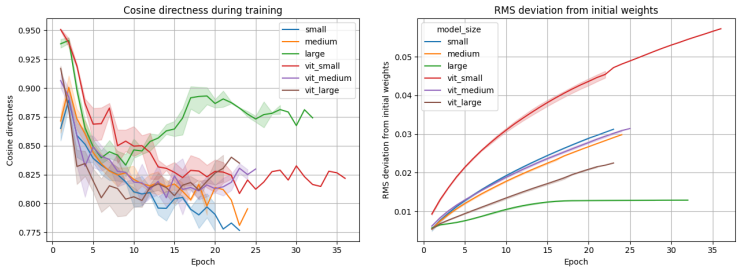


Figure 6: Accuracy of different-sized models.

We get mixed results for cosine directness. Cosine directness during training was higher for larger CNNs. However, this trend appeared to be reversed and less significant with ViTs. RMS deviation of larger models tended to be lower, suggesting they do settle closer to their initialization.

However, these results should be taken with a grain of salt, as comparing similarity and deviation across different dimensional space may not be the most meaningful, even for seemingly dimension-invariant measures like cosine directness.

7.3 REGULARIZING TOWARDS INITIALIZATION

We decided it may be insightful to test whether incentivizing a solution closer to initialization improves generalization. We did this by creating a custom L2 regularizer that penalizes distance from initialization, rather than distance from the origin. We tested it on our medium CNN, on the subset of CIFAR-10.

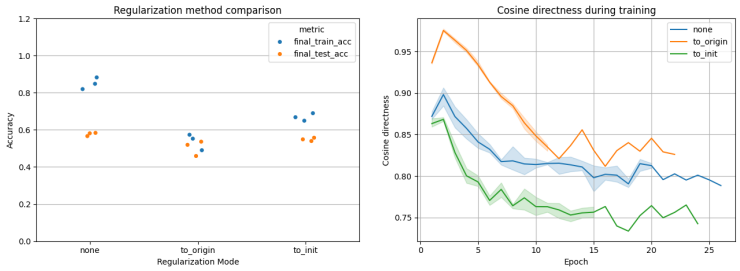


Figure 7: Accuracy of different-sized models.

We did not observe a significant test accuracy difference between regularizing towards the origin and towards the initialization point.

We observed that regularizing toward initialization decreased cosine directness during training, whereas regularizing toward the origin increased it. This seems to make sense, as a model pulled towards initialization will probably slide around in that region more. Whereas a model pulled towards the origin, will slide more consistently in that direction.

7.4 PRUNING-AWARE REGULARIZATION

In our first experiment, we observed that increasing model size can improve generalization. However, models can be pruned aggressively, showing that most parameters are not needed at inference. A common method of unstructured pruning is magnitude pruning, where you remove parameters with the lowest magnitudes. With this in mind, we proposed a custom regularizer in order to test we could improve the performance of magnitude pruning.

"iterative_train_prune" is the iterative magnitude pruning algorithm that the lottery ticket hypothesis paper uses. This algorithm trains the model from initialization many times, each time removing the lowest-magnitude weights, without using any regularization in particular.

"bottom l1" and "bottom l2" only apply the decay of the regularizer to the k% of weights being pruned with the lowest magnitude. The intent is to prevent them from messing with the important high-magnitude weights that might be part of winning tickets.

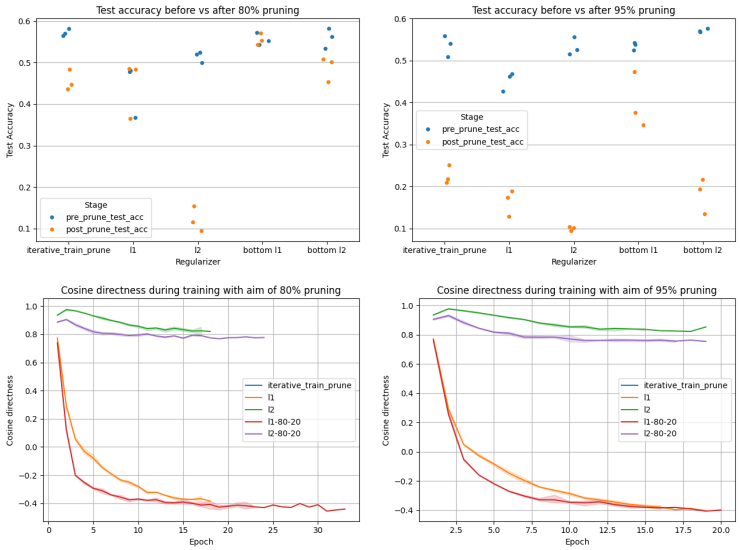


Figure 8: Accuracy of different-sized models.

The “bottom 11” algorithm achieved the highest post-pruning test accuracy for both 80% and 95% pruning.

Interestingly, the L1 regularizers significantly decreased cosine directness, even making it negative. This could be because L1 regularization more significantly impacts the large portion of weights that are low-magnitude, whereas L2 focuses more on the few high magnitude weights. In this case, low cosine directness did not correlate with overfitting or poor generalization, suggesting it is may not be the best signal for prediction generalization.

7.5 DISTILLATION

Seeing that unstructured pruning is effective, yet doesn’t yield GPU-friendly shapes, we decided to experiment with alternative methods of model size reduction. One such method commonly used on language models is distillation.

We trained our large CNN on CIFAR-100, and then distilled it into our small CNN. We used a temperature of 4.0 and an alpha of 0.8.

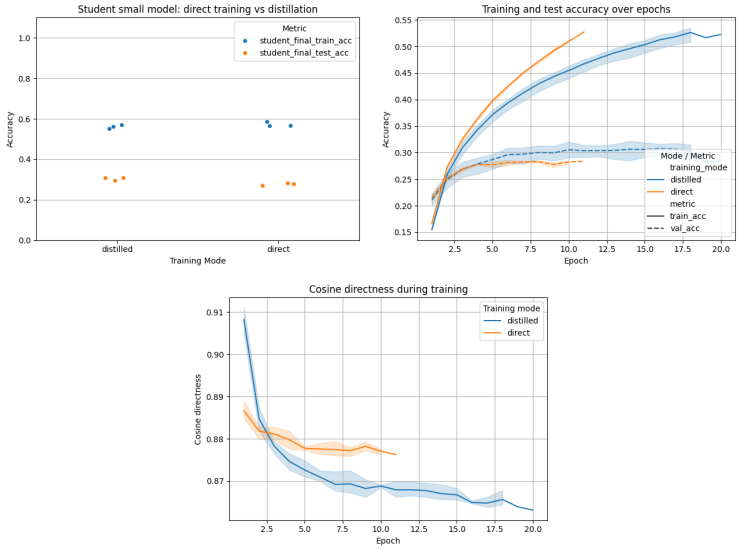


Figure 9: Accuracy of different-sized models.

We were not able to get distillation to reach a higher accuracy than just training the small model directly. It is possible that we did not do enough hyperparameter tuning. Another factor is that distillation works better on large language models because their output probability distribution has thousands of classes, allowing student models to better pick up “dark knowledge”, as opposed to CIFAR-100’s only 100 classes.

Interestingtly, we noticed the model trained through distillation started descending with a higher cosine directness and finished with a lower cosine directness than the small model trained directly. This could be because learning from a teacher’s probability distribution smoothens and simplifies the loss landscape, allowing the student to initially descend directly, before meandering a bit to fit more closely to the data.

Seeing that distillation seems to have a limit on how effectively it can reduce model size, we decided to run experiments of structured pruning, which may offer a middle ground between unstructured pruning and distillation (which is covered by our main dicussion).

REFERENCES

Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks, 2020. URL https://arxiv.org/abs/2012.08749.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks, 2020. URL https://arxiv.org/abs/2007.12223.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks, 2015. URL https://arxiv.org/abs/1412.0233.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018. URL https://arxiv.org/abs/1802.10026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL https://arxiv.org/abs/1502.01852.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes, 2018. URL https://arxiv.org/abs/1711.00165.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming, 2017. URL https://arxiv.org/abs/1708.06519.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one?, 2019. URL https://arxiv.org/abs/1905.10650.

Vaishnavh Nagarajan and J. Zico Kolter. Generalization in deep networks: The role of distance from initialization, 2019. URL https://arxiv.org/abs/1901.01672.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt, 2019. URL https://arxiv.org/abs/1912.02292.

Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, 2019. URL https://arxiv.org/abs/1805.08522.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned, 2019. URL https://arxiv.org/abs/1905.09418.

Hao Yu and Jianxin Wu. A unified pruning framework for vision transformers, 2021. URL https://arxiv.org/abs/2111.15127.

A ACKNOWLEDGMENTS

We'd like to thank our CSE 481M professor Pang Wei Koh and our teaching assistants, Rulin Shao and Zhiyuan Zeng. This project would not have been possible without their suggestions, feedback, and support.

B CODE REPOSITORY

All our experiment code is available at `https://gitlab.cs.washington.edu/manfor/cse481m_capstone` for reproducibility.

C AUTHOR CONTRIBUTIONS

C.1 MARC ALWAN

Wrote much of the code for the granular-wise pruning experiments for the ViT and ResNet. Wrote code for cosine similarity and other metrics. Wrote much of the results, discussion, and methods sections. Created much of the visualizations used in this report and previous reports. Did basic writing work for the previous reports.

C.2 KRISHNA DESHPANDE

Wrote code for our initial version of the ViT model and its different sizes. Wrote code for running and visualizing how different sized CNN and ViT models performed on our initial experiments regarding pruning, regularization, and distillation, on the CIFAR-10 and CIFAR-100 datasets. Added the tiny-imagenet dataset, and wrote code to run and evaluate how our updated ViT model, a medium-sized CNN model, and our ResNet-18 and ResNet-50 models, performed on our pruning experiments. Did writing work for the project reports, project presentations, and took notes. Added comments to document our code.

C.3 MARCIN ANFOROWICZ

Created the testing framework skeleton which runs trials of experiments on GPUs with different independent variables. Wrote the initial experiments comparing model size, descent metrics, regularization toward initialization, pruning aware regularization, and distillation. Drafted many of the milestone reports. Wrote notes and a starting a rough draft for this final report.