Maastricht University

Do Transformer Image Classification Models Contain Winning Tickets?

MASTER'S THESIS ARTIFICIAL INTELLIGENCE

Author
Christopher DU TOIT

Supervisor & Examiner 1
Dr. Mirela POPA

Examiner 2
Dr. Enrique HORTAL

With the ever increasing sizes of models, the need for efficient compression techniques is required such that the models can be compressed while maintaining an equivalent accuracy. In this Master's Thesis, a compression idea called the Lottery Ticket Hypothesis (LTH), which utilizes pruning, is applied to the Vision Transformer (ViT) for the first time. The LTH is a method which prunes models before they are fully trained, which is ideal as sometimes training a non-pruned model takes a long time. This paper also explores the effect of the LTH on the ViT and also introduces two deviant methods which give insight into how important the characteristics of the LTH are for the ViT. The common problem of deciding how much of a model should be pruned when it is not fully trained is tackled by introducing a novel pruning factor criterion.

June 2022

Contents

1 Introduction

In machine learning, the common goal is to have high performing models and often 'bigger is better' is the case, resulting in models that exceed multi-millions of parameters. This is particularly the case in Computer Vision, and so far most of the images or videos used for training are not even in high definition so there is still a lot of potential for increased parameter counts. Consequently, having such large parameter counts creates some core issues, such as the need for high computing power to train or even simply to run the models which results in a higher energy consumption, making the industry not very green. Another issue is how to deploy such large models. For instance, mobile phones, small robots, cars, surveillance cameras (and more...) have relatively tiny computing power compared to the large super computers that researchers have access to.

As a result, the need for effective compression techniques has increased dramatically. A particular compression method is called 'pruning', which is to remove or *sparsify* (convert to 0) some of the model parameters. The idea with pruning is that it reduces the computational complexity of the model. Usually, parameters are 32-bit numbers meaning that when you apply multiplication between 2 parameters of that size, although very fast, it is still quite complex. Now imagine computing this multiplication thousands or millions of times, one can see that it can become quite slow. Therefore, if one of the parameters in the multiplication is sparsified, the multiplication becomes significantly quicker, but what is even more efficient is to remove the parameter altogether such that there is no multiplication in the first place, but this is often difficult due to dimension problems within the model.

Most pruning methods are performed on models that have already been fully trained by simply pruning until the accuracy drops below a threshold. However, one may argue that if there exists a smaller subnetwork within a larger network, which has similar accuracy when fully trained, then why not just start from this subnetwork in the first place? This question has led to an emerging pruning type which is to prune before training which has the significant advantage that once the model has been pruned, the training of the model to full capacity is significantly faster as the complexity per iteration is reduced and it often can require fewer iterations due to the smaller size of the model. Currently, there are some key issues with this idea of *pruning before training* such as, "how do we know when to stop pruning?" since the accuracy of an untrained model is poor, or "how do we find the correct parameters to prune?". An example of one of these methods is the Lottery Ticket Hypothesis (LTH) [Frankle and Carbin, 2018], which is a pruning method designed for Convolutional Neural Networks (CNNs) [O'Shea and Nash, 2015] and the idea is to train a model for a small amount of iterations (about 1/12th of the total number of iterations needed to train a full model), prune the model, then reset the model and train the model fully.

On a different topic, transformers are a Deep Learning architecture which utilizes a Multi-Head Self-Attention mechanism to find relations between a sequence of input tokens. Transformers have been predominantly used in Natural Language Processing, with some models being quite famous such as BERT [Devlin et al., 2019]. In the last few years, the transformer architecture has seen recent success in Computer Vision with famous models such as the Vision Transformer (ViT) [Dosovitskiy et al., 2021]. However, the ViT has a vast amount of parameters and requires a lot of computational power to be trained. Hence, it has recently been the focus of many new pruning techniques, of which will be discussed.

There are two ways to prune a transformer model, pruning weights within the attention heads (unstructured pruning) or removing entire attention heads (structured pruning). The focus of this project is to prune, the attention heads only as this allows for complete removal of the parameters, rather than just sparsification, resulting in a more efficient model. However, the problem remains of deciding how to accurately find which attention heads to prune. In addition, the main focus of this Master's Thesis will be to apply the LTH on the ViT, for the first time, to see if the Vision Transformer contains any *Winning Tickets* (subnetworks). The issue regarding "how to know when to stop pruning" will be addressed and some in-depth analysis of the LTH method will be done, mainly to see if *weight rollback and retraining* is crucial to the success of pruning the ViT.

Hence, the main problem statement is:

Applying the Lottery Ticket Hypothesis to Transformer Image Classification

To guide this research, the following research questions will be answered throughout this report.

1. How can the Lottery Ticket Hypothesis be applied to the Vision Transformer?
2. How can important heads be detected?
3. How to decide when to stop pruning? i.e. How to know what percentage of heads to prune?
4. Does the Vision Transformer contain lottery tickets?
5. How important is *weight rollback and retraining* in the Lottery Ticket Hypothesis?

2 Background and related works

In this section, some background and related works about pruning methods and the transformer model will be discussed. There are two important things to note here: 1. the research is limited to compression techniques involving pruning (as opposed to including other methods such as knowledge distillation, quantization etc.) because otherwise, the scope of the project will be too broad. 2. Vision Transformers are still relatively new in the CV domain and hence most of the current work done for compressing CV transformer models do not have their codes released yet.

2.1 Pruning methods

In theory, there are 2 main types of pruning methods; 1. pruning after the model has been fully trained [Parnami et al., 2021], [Tang et al., 2021], [Zhu et al., 2021], [Michel et al., 2019] and 2. pruning before the model is fully trained [Frankle and Carbin, 2018], [Frankle et al., 2019]. One could even argue that there is a third type which is to prune and train together [Chen et al., 2021] but this is more towards the second pruning methods, i.e. pruning before the model has been fully trained. Furthermore, there are also 2 different types of pruning; 1. Unstructured pruning where individual weights or patches are pruned or set to 0 (sparsified) and 2. Structured pruning where whole attention heads are pruned/sparsified.

In this section, some significant methods from each of the pruning types are discussed to build some background on the current progress of model compression for Transformer models as well as to give some motivation to some of the decisions made throughout the thesis. Credits must be given to the YouTube channel Matchue, who provides the clear diagrams of figures 1, 3 and 4.

2.1.1 Pruning after training

Pruning after training is the most common method of pruning as it is easier to identify important parameters but the main downside is that you still need to fully train the model. Figure 1 demonstrates the standard pruning after training procedure for a standard Neural Network (i.e. a CNN).

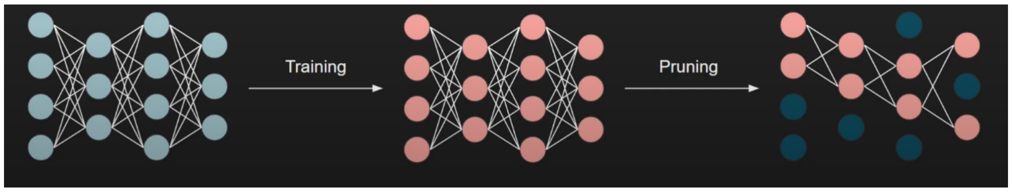

Figure 1: Pruning after training, taken from Matchue on YouTube.

Pruning Attention Heads of Transformer Models Using A* Search This method uses the A* search method to prune attention heads (structured pruning) of the BERT model for NLP [Parnami et al., 2021]. The idea is to first fine-tune the pre-trained model on the training dataset, then efficiently prune attention heads without having to visit all the possible combinations of pruned attention heads. The algorithm stops pruning once the model accuracy falls below a certain pre-determined accuracy value. The standard A* method tries to minimize the following:

$$f(n) = g(n) + h(n)$$

Where $g(n)$ is the cost of the path and $h(n)$ is the heuristic function that estimates the cost of the cheapest path from n to the goal. In this paper, $g(n)$ is the drop in performance of the model and hence the A* method will stop once the budget has been expended. $h(n)$ is the estimated cost of pruning the same attention head in the current iteration. This method is an expensive pruning method as it has to test it's score on the validation dataset every time it prunes a single head.

Are sixteen heads really better than one? [Michel et al., 2019] provides a structured method for pruning attention heads in the NLP domain. First they fine-tune the model on the dataset and then they iteratively prune a percentage of the attention heads until the score drops below a certain threshold. They then fine-tune the model to regain any loss in accuracy. Their formula for measuring head sensitivity is important for this project and is discussed further below in section 3.

Vision Transformer Pruning This paper provides a simple unstructured sparsity method where it simply aims to reduce the computation sizes of the Multi-Head Self-Attention (MHSA) and MLP in the Vision Transformer by pruning all weights/patches that are below a pre-determined value [Zhu et al., 2021]. This is done on a trained model and then after pruning, it is then fine-tuned to recover any accuracy loss (as opposed to only fine-tuning a layer as done in [Tang et al., 2021]). Figure 2 represents the unstructured pruning from [Zhu et al., 2021], where the patches in an attention head with the lowest magnitude are sparsified.

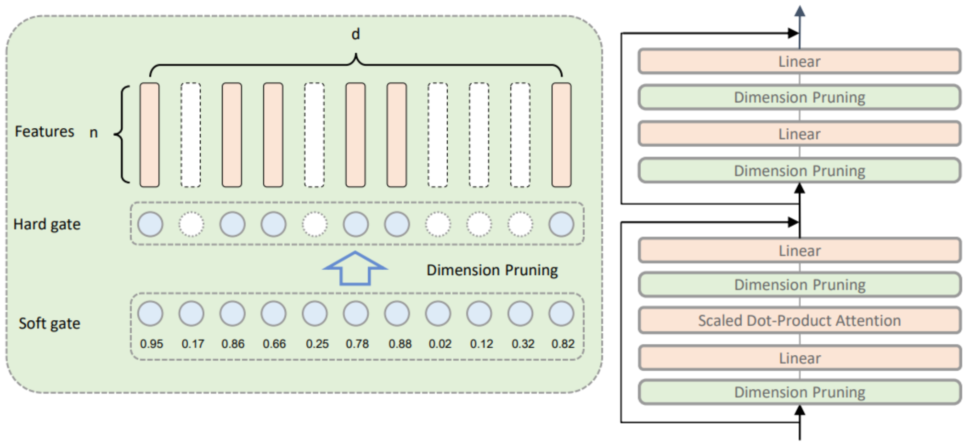

Figure 2: Unstructured Vision Transformer pruning, from [Zhu et al., 2021].

Patch Slimming for Efficient Vision Transformers This paper introduces an unstructured pruning method to remove patches from the Vision Transformer by first identifying the effective patches in the last layer and then using these to find the effective patches in the previous layers [Tang et al., 2021]. Moreover, their pruning procedure is as follows:

1. Start with the pre-trained model.
2. Iterate layer by layer.
3. For each layer, they sample a subset of the training dataset to calculate the significance scores s of the patches.
4. Iteratively, they prune the top r of the patches and fine-tune the layer for a few epochs and repeat until the performance of the layer drops below a predetermined value.
5. The patches that survive a layer will also survive the previous layers.

Multi-Dimensional Model Compression of the Vision Transformer [Hou and Kung, 2022] provide a method which combines both structured and unstructured pruning by pruning the number of neurons in MLP, heads in the Multi-Head Self-Attention (MHSA) and the sequence length simultaneously. To achieve this, they use a criterion which measures the statistical dependency between the features of a dimension and the output predictions, which is provably a complicated method. In order for this method to work, it has to start with a trained model.

2.1.2 Pruning before training

So far, all the compression methods discussed are traditional *prune after training* methods. However, one may think that if there exists a smaller subnetwork within a larger network that contains the same performance capabilities as the larger network, then why not just start from that subnetwork in the first place and only train that smaller network? Well, the issue is that we don't know what exactly the architecture or size of the smaller subnetwork should be. In this section, the novel idea of finding these subnetworks, introduced by the paper from [Frankle and Carbin, 2018], i.e. The Lottery Ticket Hypothesis (LTH) will be discussed and related works, in the transformer domain, that have stemmed from it. This project is also based off of this idea of pruning before training. Figure 3 depicts the pruning before training for a standard Neural Network (i.e. a CNN).

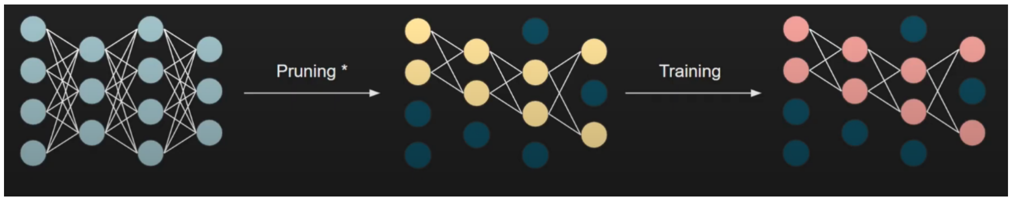

Figure 3: Pruning before training, taken from Matchue on YouTube.

The Lottery Ticket Hypothesis Although LTH method is designed for Convolution Neural Networks, the idea can be applied to the Transformer architecture as will be seen below. The hypothesis is given as,

"A randomly-initialized, dense neural network contains a sub-network that is initialized such that - when trained in isolation - it can match the test accuracy of the original network after training for at most the same number of iterations."

Basically the idea is that instead of training this large network, we look for sub-networks of this large network, take the best suited one and train that network instead. And by the LTH, this sub-network should get equivalent results to the fully trained large network.

Their method for finding these subnetworks, which they call *winning tickets* is as follows:

1. Randomly initialize a neural network.
2. Train the network for some small amount of iterations.
3. Prune a per-determined percentage number of these parameters, creating a mask. In this case, they simply prune the lowest magnitude weights.
4. Reset the remaining parameters to their initialized values from step 1, creating a winning ticket.

This method is a *one-shot* method where the network is trained once. An alternative and better method is the *iterative pruning* method which repeats the above but prunes $p^{\frac{1}{n}}\%$ of the weights for each iteration n, but this method is clearly more computationally expensive. In further work by the same team, they find that going back to the initialized weights isn't always the best, and that going back to an earlier point in training is better [Frankle et al., 2019]. It is also noted that the if the model is then randomly reinitialized to completely new weights after pruning, it will under-perform which implies that the method is sensitive to the initial weights.

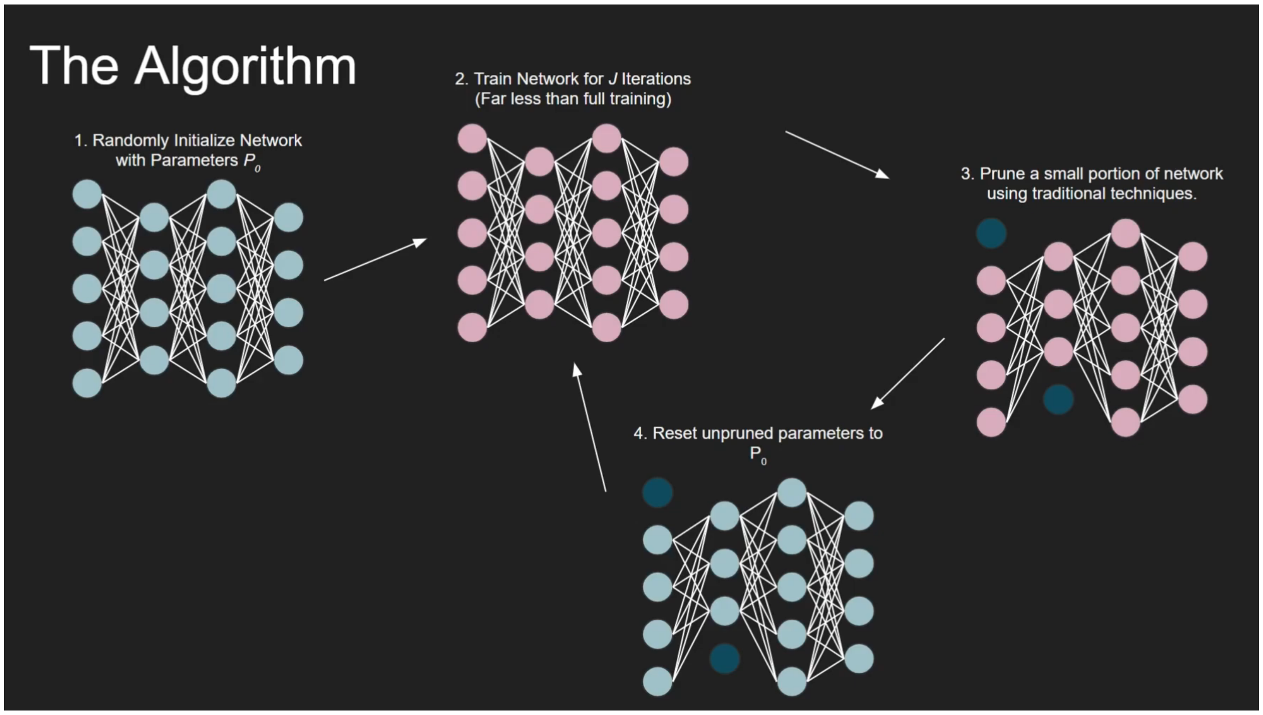
The Algorithm
1. Randomly Initialize Network with Parameters P_0
2. Train Network for J Iterations (Far less than full training)
3. Prune a small portion of network using traditional techniques.
4. Reset unpruned parameters to P_0

Figure 4: The full LTH method, taken from Matchue on YouTube.

Figure 4 depicts the LTH pruning method for general Neural Networks (i.e. CNNs), where the model is initialized in step 1, then trained in step 2, pruned in step 3, reinitialized in step 4 and then step 2 to 4 are repeated until the model is sufficiently pruned. The model is then fully trained.

The Lottery Ticket Hypothesis for pre-Trained BERT networks This is a direct application of the LTH to the NLP transformer model, BERT [Chen et al., 2020], but in an unstructured manner. Moreover, they train the network for some small amount of steps, then prune some percentage of the of the lowest magnitude weights and then roll back the model to an earlier point in training and repeat. [Prasanna et al., 2020] is a very similar paper and they state that they are using the LTH method but this is in-fact not true in the sense that they do not prune before training. Their method is to take a pre-trained BERT model, fine-tune it on the task at hand and then iteratively prune some percentage of parameters until the score is below some threshold of the original score. They apply this in two separate methods, one is unstructured where they perform magnitude pruning, which means pruning the weights which have the lowest magnitude. What is interesting is the is the structured method where they prune attention heads and **also** MLPs for the first time, by using the head sensitivity metric from [Michel et al., 2019] and adapting this slightly for the MLP sensitivity.

Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture [Brix et al., 2020] is very similar to [Chen et al., 2020] in terms of how it utitizes the LTH to prune the transformer architecture in the NLP domain. What is most significant in this paper is that they show that resetting to an earlier point in training, rather than to the initial weights, works well. In addition, they show that it's more the sign of the parameters that matter when resetting, rather than the values themselves.

Chasing Sparsity in Vision Transformers: An End-to-End Exploration [Chen et al., 2021] follows a style similar to the LTH, which is to prune before training for Vision Transformers. However, the rest of the work is quite different. They provide 3 separate methods but the one of interest is the structured pruning method. Their method is summarized as follows:

1. Randomly initialize a sparsity distribution for pruning heads.
2. Train the model for some iterations.
3. Prune the model according to the head sensitivity from [Michel et al., 2019].
4. 'Regrow' heads that score high magnitude gradients.
5. Repeat steps 2-4 until the model is trained and pruned sufficiently.

Note that this is a sparsity method, meaning that they don't completely remove attention heads but rather just set them to 0. This is so that they are able to 'regrow' attention heads but this also means that it is not as hardware friendly or as efficient as simply removing the heads altogether.

Other mentions There are some other papers which utilize the LTH to some degree but their methods are not as significant to this project. For instance, [Yu et al., 2019] applies LTH to NLP and Reinforcement Learning (RL) and show that for transformer models, the iterative pruning with LTH performs the best.

[Renda et al., 2020] introduce *learning rate rewinding*, which rewinds the learning rate and not the weights themselves, and compares it with standard pruning and *weight rewinding* from LTH. They show that the rewinding methods are superior to the standard pruning method and that the learning rate rewinding is equivalent and sometimes superior to the weight rewinding method.

2.1.3 Conclusion of pruning methods

Most of the methods described in the previous subsections either use magnitude pruning when pruning in an unstructured manner or the head sensitivity metric from [Michel et al., 2019] when pruning in a structured manner. In all methods, the pruning is done by sparsification, meaning that they don't actually remove the parameters but rather just set them to 0, which isn't as hardware friendly. Although [Chen et al., 2021] states that they apply the LTH to ViT, they do not actually do this in the vanilla form and in-fact, there is no known work that has been done on applying the vanilla LTH method to ViT. Also, what should be noticed is that most methods just prune some percentage of the parameters but do not have a smart or automated way of finding the absolute maximum number of parameters to prune. Lastly, none of the papers examine the effect of weight resetting and retraining in the LTH method.

Therefore, the work in this thesis aims to address some of these observations and will be discussed in section 3.

2.2 The Vision Transformer

The idea behind the Vision Transformer (ViT) [Dosovitskiy et al., 2021] was to use the vanilla Transformer, which is predominantly used in Natural Language Processing (NLP) [Vaswani et al., 2017], [Devlin et al., 2019], and apply it to the image domain with as little change as possible. The way the ViT works is, instead of inputting a sequence of words into the Transformer, they cut an image up into equal patches and feed these patches as embedded tokens into the model. Then the model computes the query $\mathbf{q}$, key $\mathbf{k}$ and value $\mathbf{v}$ matrices which are used to compute the attention between each patch and every other patch. Therefore, for an image with N by N patches, for each patch there are $N+1$ computations of attention. The $+1$ counts for the learnable classification token which is used to give the output classification of the image. This is repeated for L encoding blocks and H attention heads.

$$\mathbf{A}_p^{(l,a)} = SM\left(\frac{\mathbf{q}_p^{(l,a)\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}_0^{(l,a)} \left\{\mathbf{k}_{p'}^{(l,a)}\right\}_{p'=1,\dots,N}\right]\right)$$

Here, A is the attention, $l \in L$ is the layer, $a \in H$ is the attention head and p is the patch. SM represents a SoftMax computation. Then, a weighted sum of the value matrices and the self-attention coefficients $\mathbf{A}_{(p)}^{(l,a)}$ is computed.

$$\mathbf{s}_p^{(l,a)} = \mathbf{A}_{p,0}^{(l,a)}\mathbf{v}_0^{(l,a)} + \sum_{p'=1}^{N} \mathbf{A}_{p,p'}^{(l,a)}\mathbf{v}_{p'}^{(l,a)}$$

In a general Transformer architecture, one layer consists of a Multi-Head Self-Attention (MHSA) block followed by a MLP, where the MHSA is simply just multiple attention heads stacked upon each other. The base size for a transformer is usually 12 layers with 12 attention heads. Figure 5 shows the Vision Transformer architecture. Moreover, the image depicts how the image is split into fixed-size patches, then the patches are linearly embedded with added position embeddings and fed as a sequence of vectors into the transformer encoder.

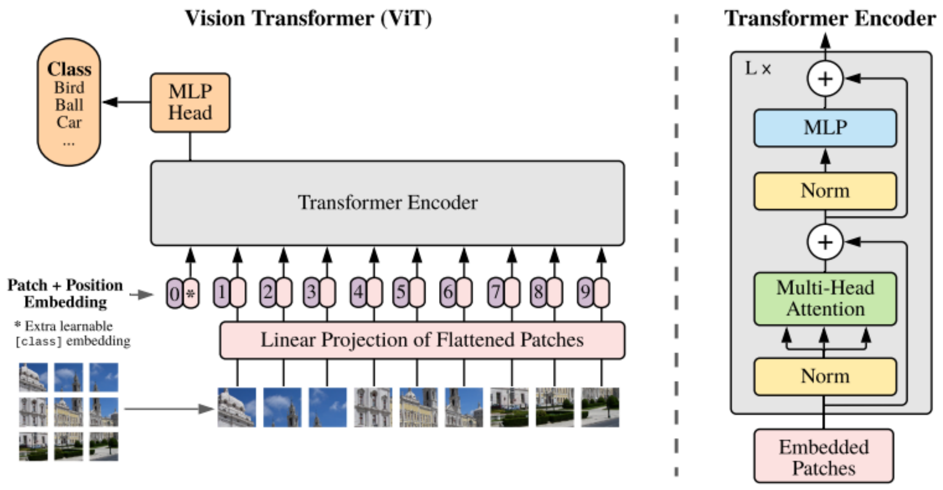

Figure 5: Transformer architecture, taken from [Dosovitskiy et al., 2021]

3 My contributions

In this project, the focus is to prune the model before training, in a structured manner, because of it's computational saving advantages (i.e. does not require fully training the model before pruning) and hardware efficiency. Pruning attention heads simply results in a reduced Multi-Head Attention block and corresponding MLP layer, shown in figure 5. In addition, most methods simply prune by *sparsifying* parameters but in this case, the goal is not to sparsify but to simply remove attention heads completely from the model, resulting in a faster and more hardware friendly model. However, the issue of trying to find out how to know when to stop pruning remains. The general method is based on the Lottery Ticket Hypothesis (LTH) and this project is the first time that the LTH has been applied to the Vision Transformer, in it's vanilla form. In addition to this method, 2 other deviations of this method have been implemented and tested on, with the intention that these method deviations will give us some more insight into how the ViT performs with the LTH.

3.1 LTH for VIT

The main method is as such; first train the model on a small subset of the dataset, evaluate the performance of the model on the evaluation dataset and then detect which heads are the least important by using a head sensitivity metric described below. To get a good relative comparison between the heads, the head sensitivity values are normalized globally and then locally since this has a provably improved performance (see section 5.1. The heads are then pruned and if all the heads of a layer are pruned then the corresponding MLP will also be removed, resulting in a completely pruned layer.

Then, the model is reset to an earlier checkpoint in training, as opposed to new randomly initialized weights as further work by [Frankle et al., 2019] proves that this provides a significant improvement. This is one iteration of the pruning algorithm. The step size for the next step of the pruning algorithm depends on the performance of the previous iteration. More about this below.

Algorithm 1 Finding ViT head mask with LTH

```
1: **Initialize:**
       Head mask $\mathbf{M} \leftarrow \mathbf{1}$ (Num layers $\times$ Num heads)
       Initialize model $f(x;\theta_0, y_0)$
       Initialize pruning factor $p$
2: Train $f(x;\theta_0, y_0)$ for some epochs $t << j \leftarrow f(x;\theta_i, y_i)$
3: **while** $i <$ num epochs **do**
4:     Apply mask $f(x;\mathbf{M} \odot \theta_i, y_i)$
5:     Train for $j$ epochs $f(x;\mathbf{M} \odot \theta_i, y_i) \leftarrow f(x;\mathbf{M} \odot \theta_j, y_j)$
6:     Evaluate and find head sensitivity $I$
7:     Normalize head sensitivity
8:     Prune $p$ of lowest scoring heads
9:     Update $\mathbf{M}$
10:    Rollback model $f(x;\mathbf{M} \odot \theta_j, y_j) \leftarrow f(x;\theta_i, y_i)$
11:    Update $p$
12:    **if** $p < 0$ **then**
13:        Rollback head mask
14:        Update $p$
15:    **end if**
16: **end while**
17: **return** $\mathbf{M}$, $f(x;\theta_i, y_i)$
```

3.2 Head Sensitivity

However, a problem already arises as to how does one decide which heads to prune? Well, the idea is to use a *head sensitivity* metric, which is inspired by [Michel et al., 2019] and [Prasanna et al., 2020] but follows more accurately along the lines of [Molchanov et al., 2016]. [Molchanov et al., 2016] use the Taylor Series Expansion of the *difference gradient* which is the difference between the cost of pruning a feature map and not pruning a feature map. This resulted in the following:

$$\Theta(z_l) = \left| \frac{1}{M} \sum_m \frac{\delta C}{\delta z_l^m} z_l^m \right| \cdot \frac{1}{B}$$

Where z is the feature map, C is the cost or loss of the outputs, M is the length of the feature map z and l is the layer in question and B is the batch size. Intuitively, this function gives higher values to more important features/weights and lower values to less important features/weights.

To adapt it to the Vision Transformer, the feature map z becomes the attention map, the gradient of the cost function becomes the gradient of the loss function. The adapted formula is given below;

$$I_l = \left| \frac{1}{M} \sum_m \frac{\delta \mathcal{L}}{\delta A_l^m} A_l^m \right| \cdot \frac{1}{B}$$

In practice, to compute the head importance with this criterion, the model has to iterate through the validation dataset and store the attention maps and the gradients of those attention maps, which is more computationally expensive that simply doing magnitude pruning for example. However, this method is more rewarding as it can accurately find the least important heads.

3.3 Pruning factor criterion

Now to try and determine how many heads to prune, I designed a novel pruning factor criterion which computes the next pruning factor p_{i+1} based on the initial pruning factor p_0, the pruning threshold T, the initial score s_0 and the current score s_i. Some important characteristics of this criterion are,

1. The whole model should be able to be pruned in $\frac{100}{p_0}$ iterations if the problem at hand is trivial.
2. The model should not be pruned to a score below the $T * s_0$.
3. It knows when it has pruned too much (and should result in some sort of backtracking).

With these characteristics in mind, the following is produced.

$$p_{i+1} = p_0 \left(\frac{(1-T) - \frac{s_0 - s_i}{s_0}}{1-T} \right)$$

This pruning factor criterion is able to backtrack if the value is negative, and if the drop in scores is very significant, it will keep halving the initial pruning factor p_0 until the pruning factor is positive again or it ends the search (as the end point is found). Note that, one could change the p_0 to p_i but p_0 is more robust to sudden large drops in accuracy between to steps.

3.4 Deviant methods

To further investigate the LTH properties on the ViT, two more methods are introduced. The first one is called the *One Train* method, which means that instead of training for a small number of epochs and resetting the weights every iteration, the model is trained once and never reset. This is clearly more efficient but the expectation is that correct heads may not be accurately pruned and the stopping criteria will not work as well because without retraining, the validation accuracy can drop quite significantly. This method will give insight into seeing how important weight resetting and retraining is.

The other deviant method is to train and prune iteratively, inspired by [Chen et al., 2021]. The model is trained and pruned and instead of resetting the weights after each iteration, the model simply continues training from that point, but with the learning rate reset instead. This is repeated until all the possible heads are pruned and then it is fine tuned for some further epochs. This method could be more efficient but in the case of CIFAR100, it is actually not. This method will see how weight resetting and retraining will compare to iterative pruning and training (without resetting).

4 Experiments

In this section, the experiments that were performed are discussed and motivated. The results can be found in the following section, section 5. To encourage consistent results, weight sharing is often utilized for the initialized models.

Model and datasets For all the experiments, the DeiT-B model [Touvron et al., 2020] was used as this seems to be standard procedure in looking at pruning methods [Zhu et al., 2021], [Tang et al., 2021], [Hou and Kung, 2022], [Chen et al., 2021]. The DeiT-B model architecture is identical to the ViT-B except the Deit-B model has an additional distillation token. This distillation token is ignored as it's intended use is for knowledge distillation which is another pruning type not considered in this project. Another difference is that DeiT-B is pretrained on ImageNet-1k and ViT-B is pretrained on the full ImageNet dataset. The architecture consists of 12 attention heads per layer, followed by an MLP, and there are a total of 12 layers in the model, resulting in 144 heads total.

The datasets used for all the experiments, except the normalization, are CIFAR10, CIFAR100 and MNIST, where the sizes of these datasets are in table 4. CIFAR100 consists of 100 different classes with examples such as beaver, crocodile, bed, bicycle, house etc. and the top score for this dataset, according to PapersWithCode.com, is EffNet-L2 (SAM) [Foret et al., 2020], which is a CNN and it scores considerably higher than all other methods, scoring 96.08%. On the other hand, DeiT-B scores 90.8% on CIFAR100.

CIFAR10 is similar to CIFAR100 but with 10 classes only and 10 times the amount of training/test data per class. Most top performing models score 99% and above for CIFAR10, with DeiT-B scoring 99.1%.

Lastly, MNIST is a very well known dataset in the CV domain and consists of low resolution images of the numbers 0 to 9. This is a trivial dataset which get scores about 99.5% on average, but with that being said, there is no official score for MNIST with the DeiT-B model.

Datasets				
Name	Train size	Test size	Validation size	# of classes
CIFAR100	45 000	10 000	5 000	100
CIFAR10	45 000	10 000	5 000	10
MNIST	63 000	10 000	7 000	10

For the experimental configurations, please check the appendix section A.1.

4.1 Normalization

For this experiment, different types of normalization techniques for normalizing the head sensitivity are tested to see which gives the best performance. The normalization types are split into two: layer-wise and global. In layer-wise normalization, the head importance of each layer is normalized by L_2 normalization. In global normalization, the importance of all the heads of all layers are normalized using the min-max normalization. The choices of these normalizations were taken from HuggingFace Bertology. This experiment will only be performed on the CIFAR100 dataset.

4.2 Pruning Threshold

For this experiment, different pruning thresholds, for the pruning factor criterion, are tested on each of the datasets CIFAR100, CIFAR10 and MNIST, for each of the 3 methods. Then, for each of the threshold values, the final produced head mask is then applied to the model and fully trained to get a fully trained score. The idea behind this experiment is to analyze how the validation results compare with the fully trained results for different threshold values, different dataset difficulties and different mask finding methods. Also, this experiment will show how well the novel pruning factor criterion is performing.

Note that, although only significant in the case of MNIST and CIFAR10, whole layers (and hence the corresponding MLP layer) can be pruned if the pruning algorithm decides it.

4.2.1 LTH head masks

The LTH head masks is the masking method of which closely follows the Lottery Ticket Hypothesis (LTH), as described in section 3.1. For this, the threshold values to be tested for the smart iteration algorithm will be: $\{0.90, 0.92, 0.94, 0.96, 0.98, 0.99, 0.999\}$. These values are chosen such that there is potential for the pruning method to go from over pruning to under pruning the model. Also, an additional experiment will be performed here, which entails *limited pruning* such that whole layers cannot be pruned, i.e. there must be at least one attention head in each layer, but it is clear that for MNIST and CIFAR10 that this limits the full pruning capabilities, although in the case of CIFAR100 there is not difference because CIFAR100 does not get compressed enough. For the limited pruning results, please check the section in the appendix A.2.1.

4.2.2 One train head masks

This method is similar to the previous method but instead of rolling back the weights and retraining, this method instead trains the model once for a small number of iterations and simply reuses this model for all iterations of the mask finding method, described in section 3.4. The threshold values to be tested for the smart iteration algorithm will differ for each dataset as each of the datasets have varying levels of difficulties. The way the pruning factors will be done is to try and get similar pruning percentages to the previous section, 4.2.1. The interest in this particular experiment is to test how important rolling back and retraining is as stated in the LTH hypothesis.

4.2.3 Train and prune

For the train and prune method, the pruning thresholds will also differ per dataset but it should be noted that because of the constant continuation in training, the pruning thresholds can remain higher than in the main LTH head masking method. The idea for this method is to try and see if this method can be more efficient than the LTH method and also to investigate what happens when the weights are not reset. In addition, to try encourage the model to adapt to the new architecture, the learning rate is reset each iteration, despite the weights not being reset.

5 Results

5.1 Normalization

In figure 6, it is clear that at least layer-wise normalization is needed because no normalization and global normalization perform quite poorly. Layer-wise or both normalizations perform quite similarly and in this project, the normalization with global first is taken to be the standard due to it's slight improvement over the other two types. The need for normalization makes sense as it gives the method a better sense of which head is better performing in relation to the other heads.

5.2 Pruning Threshold

5.2.1 LTH head masks

In figure 7, there is a direct correlation between the final validation result from the mask finding method and the fully train model. The model accuracy is able to stay within 2% accuracy and prune up to 65% of the attention heads. One can also see that 65% is a tipping point for the model as the accuracy drops significantly when pruning beyond this value. An anomaly is the threshold value for 99.9% where it actually

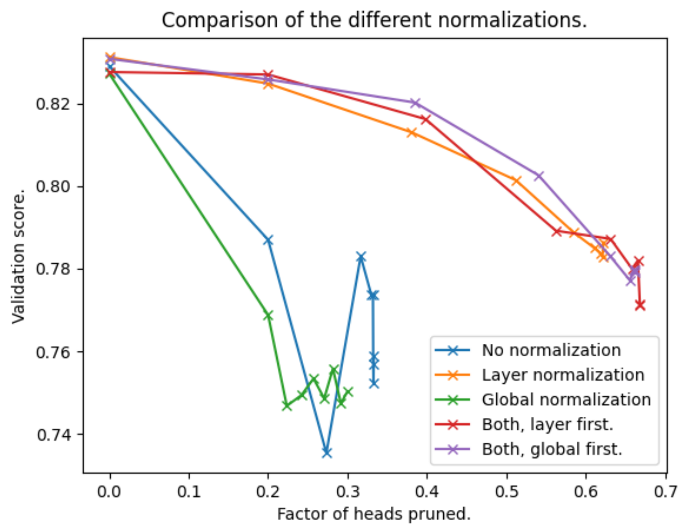

Figure 6: Comparison of different normalizations with CIFAR100.

performs better than the threshold value 99% but this is probably due to chance or a slightly lower starting validation score and is not a significant result.

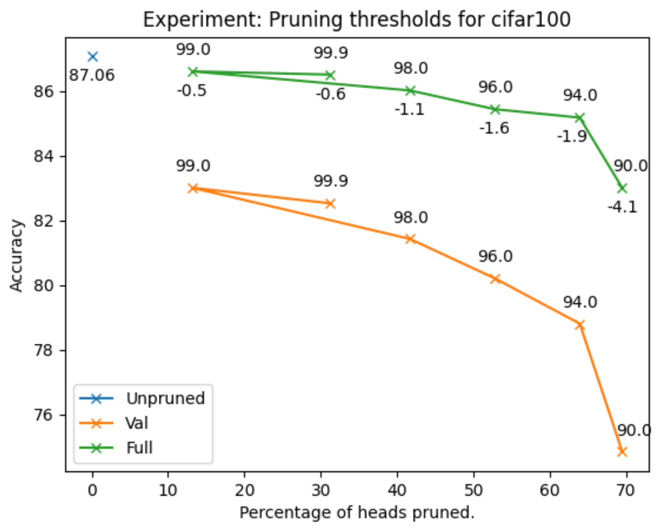

Figure 7: Pruning thresholds CIFAR100, LTH head masks. The labels for each of the data points represents the pruning threshold (above) and the difference in accuracy to the unpruned model (below).

In figure 8, again there is a direct correlation between the validation accuracy and the fully trained model accuracy and the method is able to prune the model up to about 80% of the model and stay within 2% accuracy. About 80% is also the tipping point for the model with the higher pruning percentages performing much worse.

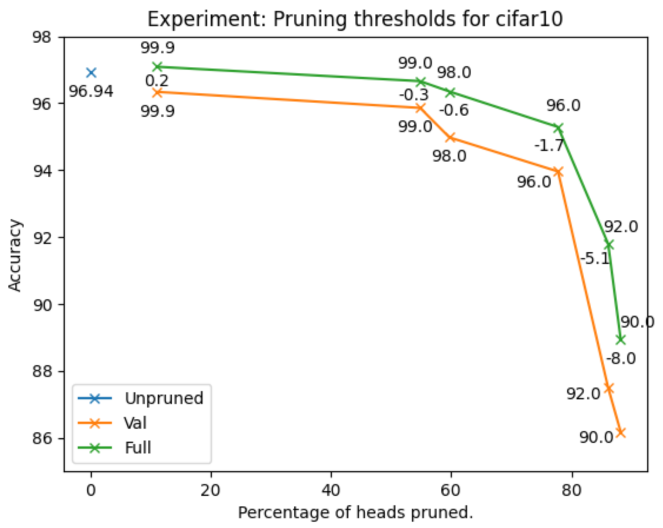

Figure 8: Pruning thresholds CIFAR10, LTH head masks. The labels for each of the data points represents the pruning threshold (above) and the difference in accuracy to the unpruned model (below)

In figure 9, the situation is quite different compared to the other 2 datasets due to the triviality of the problem. Hence, please notice the difference in layout of the plot compared to the previous 2 figures as this figure now has the threshold values on the x-axis and the pruning percentages above the green data points and below the orange data points. The LTH method is able to successfully prune all heads and layers except 1, with minimal loss in accuracy, in all cases except in some unique cases where the threshold value was 96% and 99%, where in the latter the accuracy is (slightly) even better than the original unpruned model score.

5.2.2 One train head masks

For the results *One Train*, there are some common trends. For instance, the method under-performs for all 3 datasets when compared to the LTH method and the model struggles to decide how much of the heads to prune because of the steep loss in accuracy between pruning iterations, which is due to the lack of retraining.

In figure 10 correlation between the validation accuracy and the full model accuracy is not as strong as in the corresponding LTH method, figure 7. Moreover, the validation accuracy drops much more significantly, which is expected due to the lack of retraining. The method also does not perform as well because the full model accuracy already drops below 2% of the unpruned accuracy before 65% of the model is pruned as in the LTH method. What is also interesting to note is that the method is sensitive to the initialized weights. Moreover, figure 18 in section A.2.2 shows the fully trained scores of the model with randomly initialized weights after mask finding and it is clearly inferior to keeping the initialized weights that the method pruned for.

In figure 11, the situation is similar to CIFAR100, where the validation score drops much more significantly as more heads are pruned. The *One Train* method also does not perform as well on the CIFAR10 dataset as the accuracy differs by more than 2% before 80% of the model has been pruned.

In the case of MNIST, it is known from the results with the LTH method that it can perform just as well when there is only 1 attention left. However, the main issue of this method is that no matter how low the pruning threshold was, the model could never get to prune all the 143 heads (of the total 144 heads), as shown in figure 12.

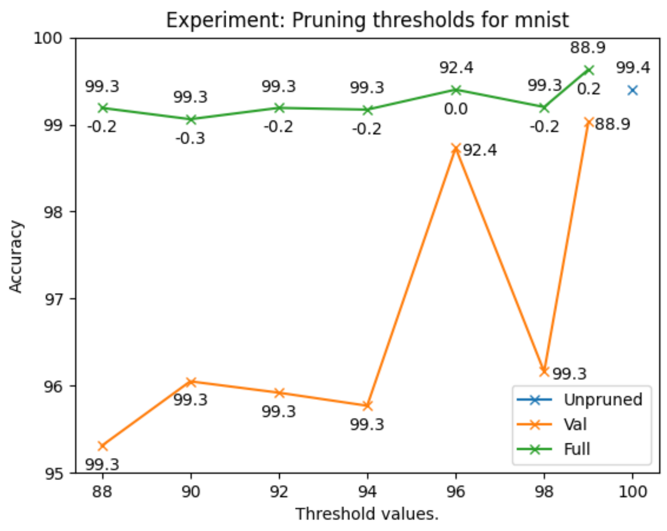

Figure 9: Pruning thresholds MNIST, LTH head masks. The labels for each of the data points represents the percentage of heads pruned and the difference in accuracy to the unpruned model.

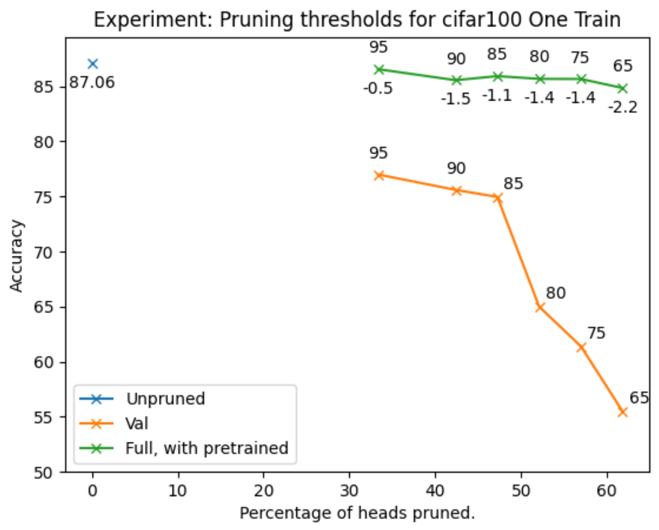

Figure 10: Pruning thresholds for CIFAR100 for the One Train method. The labels for each of the data points represents the pruning threshold (above) and the difference in accuracy to the unpruned model (below).

5.2.3 Train and prune

In general, the train and prune method was inferior to the LTH method and although there was potential for this method to be more efficient, it was found that in the case of CIFAR100 more epochs were needed to get a decent accuracy meaning that it had about the same amount of computational time as the standard LTH method.

With that being said, figure 13 represents the *Train and Prune* method for the CIFAR100 dataset and it is clear that the method is quite unstable, meaning that it fluctuates in scores, and also under performs when compared to the LTH method. The instability of the model could be due to the fact that the learning rates are rolled back.

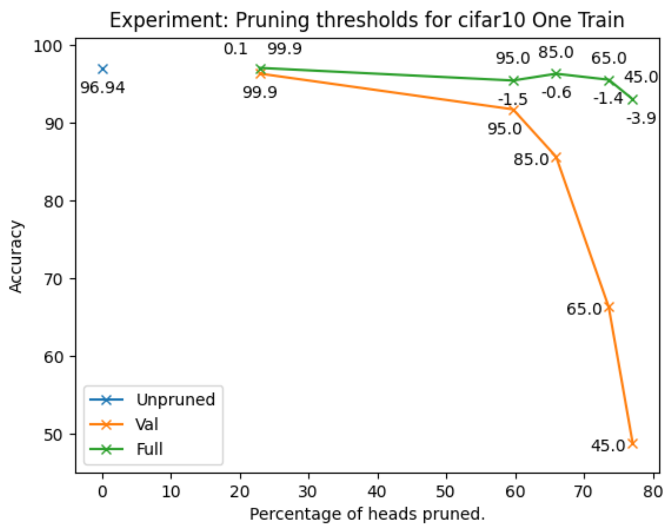

Figure 11: Pruning thresholds CIFAR10 for the One Train method. The labels for each of the data points represents the pruning threshold, and the values below the full train models represent the difference between the full model accuracy and the pruned model.

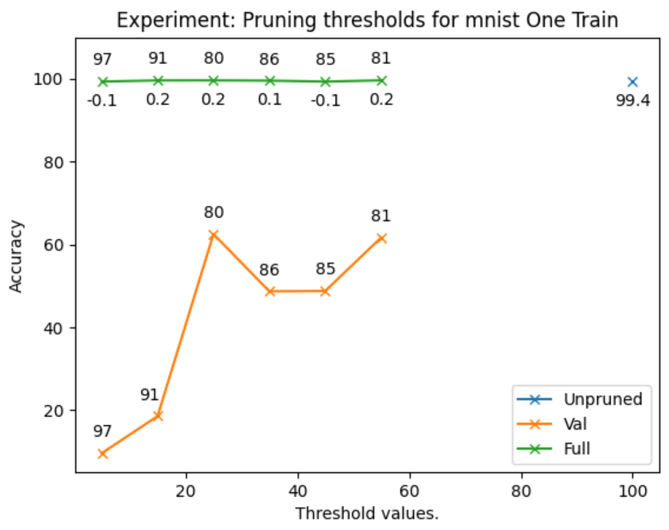

Figure 12: Pruning thresholds for MNIST for the One Train method. The labels for each of the data points represents the pruning threshold, and the values below the full train models represent the difference between the full model accuracy and the pruned model.

In figure 14, the method shows to have comparable performance to the LTH method but under performs as the accuracy difference is more than 2% before 80% of the model has been pruned. Despite that, this method was more efficient as it found the headmask and trained in about 4 epochs total whereas the LTH method used about 6 epochs.

In 15, the method is able to prune all the heads except 1 as in the LTH method when using thresholds 94% and 98% while maintaining a great accuracy at 99.3%, the model didn't prune as much but still had high accuracy. It is clear that the method could solve the MNIST problem.

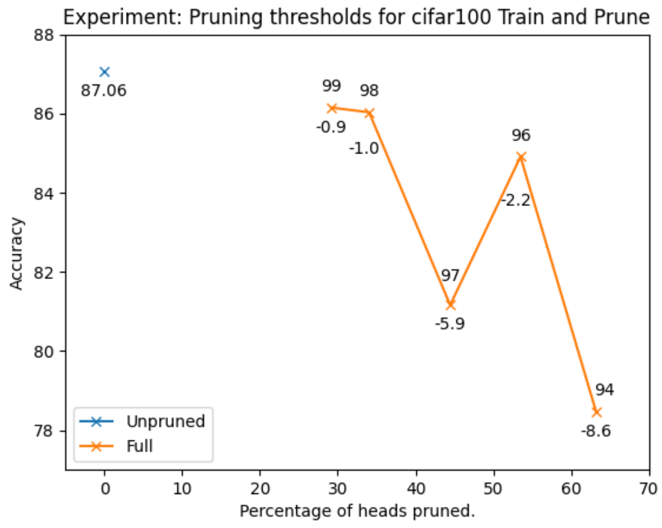

Figure 13: Pruning thresholds CIFAR100, Train and Prune method. The labels for each of the data points represents the pruning threshold, and the values below the full train models represent the difference between the full model accuracy and the pruned model.

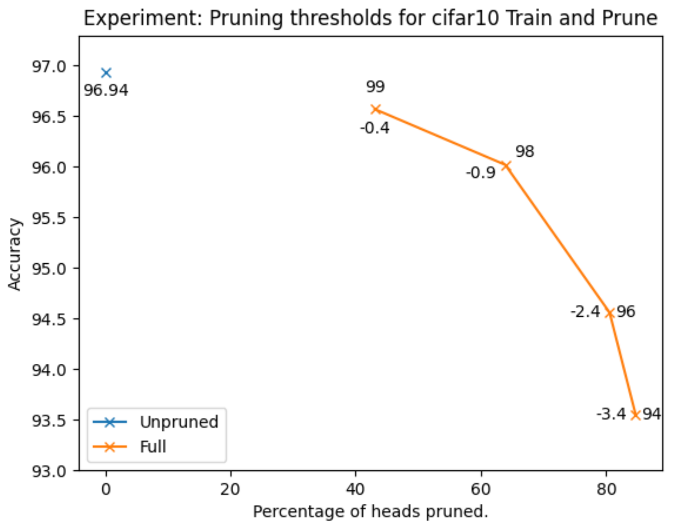

Figure 14: Pruning thresholds CIFAR10, Train and Prune method. The labels for each of the data points represents the pruning threshold, and the values below the full train models represent the difference between the full model accuracy and the pruned model.

6 Discussion and Further Research

In this section, the research questions are explicitly answered and discussed with support from the experiments.

1. How can the Lottery Ticket Hypothesis be applied to the Vision Transformer? The LTH method, designed initially for CNNs, simply trained for a small amount of iterations, pruned some percentage of the weights with the lowest magnitude, reset the weights and repeated. The LTH method for ViT is

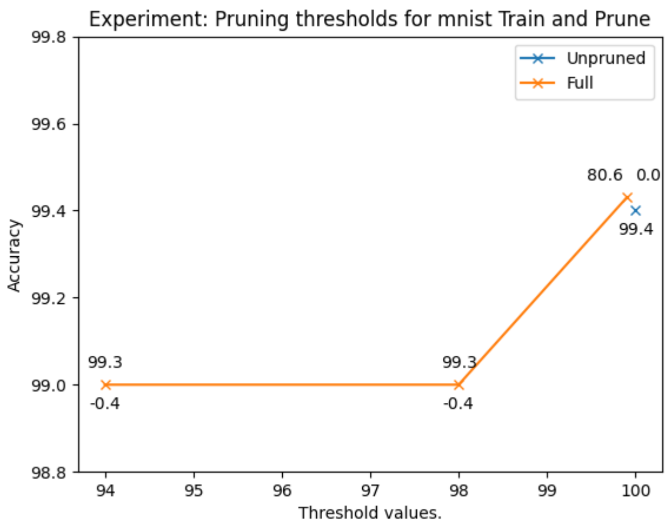

Figure 15: Pruning thresholds MNIST, Train and Prune method. The labels for each of the data points represents the percentage of heads pruned, and the values below the full train models represent the difference between the full model accuracy and the pruned model.

different, instead of pruning individual weights, whole attention heads were pruned in a structured manner as this has some advantages such as complete removal. Simple magnitude pruning will not work for an attention head which contains a multitude of weights and hence a head sensitivity metric was used. In addition, rather than just pruning some $p\%$ of the model regardless of the performance, a smart pruning factor iterator was used instead.

2. How can important heads be detected? This research question is implicitly answered in research question 1. Moreover, the important heads are discovered by their scores on a head sensitivity metric which was adapted from [Molchanov et al., 2016], meaning that the greater the score, the more the head has a particular importance on the outcome and hence the more important the head is. This is because the head sensitivity metric is the Taylor Series Expansion of the 'difference gradient' which is the difference between the cost of pruning an attention head and not pruning an attention head.

3. How to decide when to stop pruning? i.e. How to know what percentage of heads to prune? To tackle this problem, a novel pruning factor criterion was introduced which takes into consideration the initial pruning factor, the pruning threshold and the gradient between the current score and the initial score.

The results of the experiments show that the pruning factor criterion also works well as it is able to prune whole models if possible and also prune the maximum amount of heads while staying within a threshold, hence the pruning factor criterion has utilizes the desired characteristics stated in section 3.3.

4. Does the Vision Transformer contain lottery tickets? From the results of the experiments, there are some key things to note. The LTH method seems to be quite successful on the 3 datasets with 65% of the CIFAR100 model being pruned and staying within 2% accuracy and 80% of the CIFAR10 model while also staying within 2% accuracy. In the case of the trivial dataset MNIST, the method was able to prune all layers and heads except 1, resulting in a 99% pruned model. What was also interesting to see was the strong correlation between the validation accuracy and the fully trained model accuracy. Moreover, one could potentially figure out which threshold value is best for their model by looking at the validation scores before fully training a model. This implies that the Vision Transformer does contain lottery tickets.

Also, it was noticed that the method of pruning before training was sensitive to it's initial weights, meaning that if you prune some heads and then reset the weights to completely new weights before training, the performance will be significantly worse than keeping the same initial weights when finding the head masks.

5. How important is weight rollback and retraining in the Lottery Ticket Hypothesis? To understand the effect of weight rollback and retraining, 2 deviant methods were introduced, *One Train* and *Train and Prune.* For *One Train*, the weights was never rolled back and the model was only trained once and for the *Train and Prune* method, the model was trained after each pruning iteration without weight rollback.

In the *One Train* method, the results show that the main issue was the steep loss in validation accuracy during pruning, making it difficult for the method to decide when to stop pruning. With that being said, the final results were also inferior to the LTH method for the same percentage of heads pruned. The intuition behind this is because when the model has access to a lot of parameters it is not forced to compress the information to only a few parameters and hence there is a lot more dependencies between the parameters. Hence, the model accuracy drops significantly when some of these parameters are pruned. Moreover, the model still under performs when fully trained because it is not able to accurately prune the right heads, whereas in the LTH method, the method is able to prune the correct heads because each iteration, more of the model is pruned and the model is forced to focus it's information in just a few heads and hence there is a larger difference between important and less important heads, making it easier to identify which heads to prune.

The *Train and Prune* method performed surprisingly bad, especially in the case of a more complex dataset such as CIFAR100, where it was also unstable. Even though this method performed relatively better on CIFAR10 and MNIST, the results were still not better than the LTH method for CIFAR10, although in the case of MNIST, it did in-fact solve the problem.

These results of the deviant methods has made it clear that the LTH method is superior to the other two deviant methods, implying that the *weight rollback and retraining* is important.

Comparison to other methods Unfortunately, it is difficult to compare with other recent works as a lot of researchers only test on the ImageNet dataset, which is a very big dataset and requires a lot of computational resources (more about this below). In addition, the LTH has only been applied to CNN models so it is also difficult to grasps how good this method actually is relatively. With that being said, the LTH authors, [Frankle and Carbin, 2018], score 90.5% when the model, Resnet-18 [He et al., 2015], is pruned to 27.1% of the original size for the CIFAR10 dataset. Resnet-18 has 11.4M parameters, so the resulting size is 3.09M parameters. In our case, CIFAR10 scores 95.25% accuracy, with 80% of the parameters pruned, but the size of DeiT-B is 86M parameters, resulting in a model of 17.2M parameters. This is much larger, but in ViT models are bigger than CNNs in general.

Further research In a research project, there is always more work that can be done. Firstly, to get an accurate understanding of how well this method performs compared to other pruning methods, the method should be trained on the ImageNet-1k [Russakovsky et al., 2014]. This was actually a goal for the project but unfortunately, there were massive delays in getting a license for the dataset and then the computational resource I was using did not grant access to their ImageNet-1k dataset to use for training until only after the University lost their partnership with them. With that being said, to have equivalent ImageNet-1k results, the method has to be trained for about 300 epochs and this takes 5 days on 8 GPUs, making the task very computationally heavy.

Other ideas include trying the *Train and Prune* method without learning rate resetting as this could be a reason for the poor performance of this method. Another idea is to use multi-dimensional pruning, which is to prune parameters within the attention heads or apply patch slimming to the input parameters. One could also maybe try and find a direction explanation or derive a formula for the relation between the validation accuracy and the fully trained model accuracy in the LTH method.

7 Conclusion

In this project, the Lottery Ticket Hypothesis is successfully applied to the Vision Transformer, in a structured manner by utilizing a commonly used head sensitivity metric and a novel pruning factor criterion. Moreover, this method was able to prune a model trained on CIFAR100 by up to 65% and CIFAR10 by up to 80% while maintaining 2% accuracy. The LTH method was also able to completely prune the Vision Transformer from 144 heads and 12 MLPs to just 1 head and 1 layer while maintaining almost identical results, for the MNIST dataset. This proves that indeed, the ViT contains *Winning Tickets*.

Some more in-depth study was put into understanding the LTH method on the ViT by introducing two more methods, *One Train* and *Train and Prune*, where the former simply never resets it's weights but also never trains after the first iteration, and the latter trains and prunes iteratively without resetting the weights. Both methods fall short of the LTH method, showing that indeed the method of weight resetting and iterative pruning is the best performing.

References

[Brix et al., 2020] Brix, C., Bahar, P., and Ney, H. (2020). Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture.

[Chen et al., 2021] Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., and Wang, Z. (2021). Chasing sparsity in vision transformers: An end-to-end exploration.

[Chen et al., 2020] Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. (2020). The lottery ticket hypothesis for pre-trained bert networks.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

[Foret et al., 2020] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization.

[Frankle and Carbin, 2018] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

[Frankle et al., 2019] Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2019). Stabilizing the lottery ticket hypothesis.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Hou and Kung, 2022] Hou, Z. and Kung, S.-Y. (2022). Multi-dimensional model compression of vision transformer.

[Michel et al., 2019] Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one?

[Molchanov et al., 2016] Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference.

[O'Shea and Nash, 2015] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks.

[Parnami et al., 2021] Parnami, A., Singh, R., and Joshi, T. (2021). Pruning attention heads of transformer models using a* search: A novel approach to compress big nlp architectures.

[Prasanna et al., 2020] Prasanna, S., Rogers, A., and Rumshisky, A. (2020). When bert plays the lottery, all tickets are winning.

[Renda et al., 2020] Renda, A., Frankle, J., and Carbin, M. (2020). Comparing rewinding and fine-tuning in neural network pruning.

[Russakovsky et al., 2014] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). Imagenet large scale visual recognition challenge.

[Tang et al., 2021] Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., and Tao, D. (2021). Patch slimming for efficient vision transformers.

[Touvron et al., 2020] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2020). Training data-efficient image transformers &; distillation through attention.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Yu et al., 2019] Yu, H., Edunov, S., Tian, Y., and Morcos, A. S. (2019). Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp.

[Zhu et al., 2021] Zhu, M., Tang, Y., and Han, K. (2021). Vision transformer pruning.

A Experiments and results

A.1 Experiment configurations

Some parameters remained the same for all of the experiments. These are:

- Batch size: 8
- Number of pruning epochs: 8
- Initial pruning factor: 20%
- Initial learning rate: 0.0002

In addition, the same data augmentations used to pretrain the DeiT-B model are used on each of the datasets.

LTH method Since each of the experiments were using datasets of similar sizes, the experimental configurations were identical. For each pruning epoch, the model is trained for 1/12th of the full amount of training epochs which is 1000 training steps, where each training steps processes 8 images, resulting in 8000 training images in each pruning iteration. The model is then rolled back to the 100th training step and retrained from there. When the mask is found, the model is reset once again and trained for 4 epochs.

One Train The parameters are identical to the LTH method but there is no resetting of the weights and retraining.

Train and Prune For each pruning epoch, the model is trained for 1000 training steps, like in the LTH method, but the weights are not reset. Instead, the learning rate is reset and the model is continuously trained for anoher 1000 training steps each pruning iteration. After finding the masked model, the models are fine-tuned for a further 2 epochs for CIFAR10 and MNIST and 4 epochs for CIFAR100 since 2 epochs for CIFAR100 significantly under-performed.

A.2 Pruning Threshold

A.2.1 Limited Pruning

In figure 16, the results show that the 'limited' pruning is similar to the 'non-limited' pruning in figure 8. Moreover, the limited pruning performs worse here until the 96% threshold and then it actually improves over the non-limited pruning. However, at that point, the results are not so significant because the accuracy drop in both figures is too low.

It is clear that in figure 17, the 'limited' pruning is indeed really limiting the pruning performance of the LTH method as we know that the LTH method is able to prune almost all the heads and layers, as shown in figure 9.

A.2.2 One train with reinitialized weights

Figure 18 shows the result of the *One Train* method when reintializing the weights to new random values after finding the head mask. It is clear that one should not reinitialize the model as the results are consistently about two times worse than the results shown in figure 10.

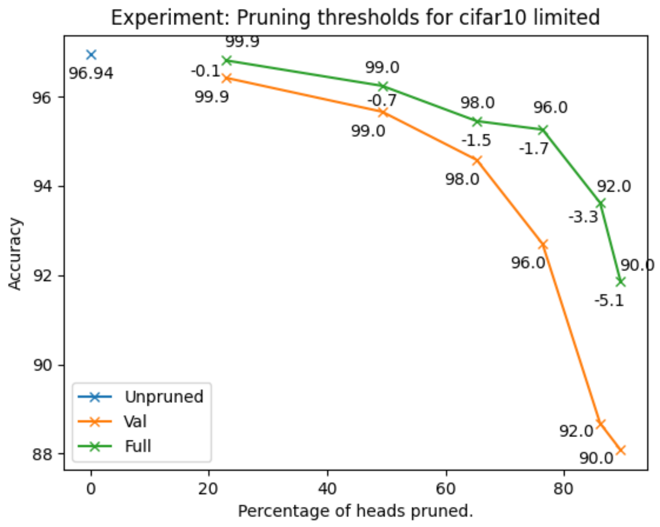

Figure 16: Pruning thresholds CIFAR10, LTH head masks with limited pruning (i.e. whole layers cannot be pruned). The labels for each of the data points represents the pruning threshold (above) and the difference in accuracy to the unpruned model (below).

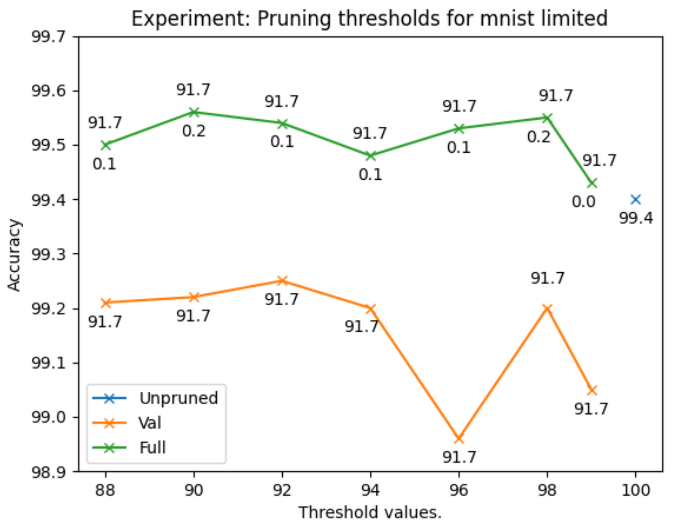

Figure 17: Pruning thresholds MNIST, LTH head masks with limited pruning (i.e. whole layers cannot be pruned).The labels for each of the data points represents the pruning threshold (above) and the difference in accuracy to the unpruned model (below).

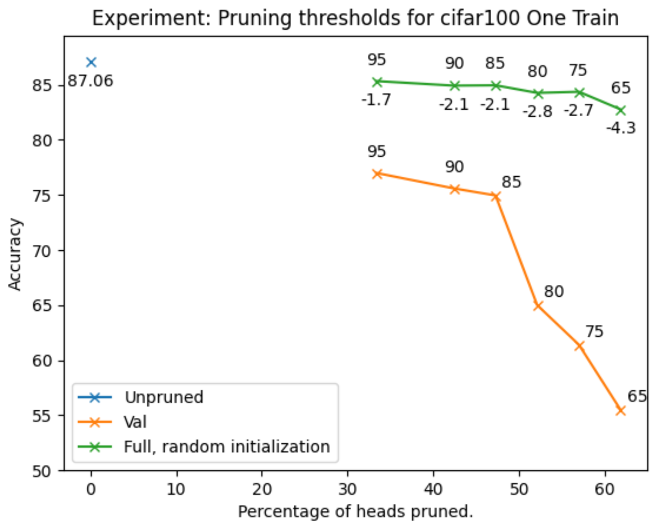

Figure 18: Pruning thresholds CIFAR100, One Train method with randomly intialized weights. The labels for each of the data points represents the pruning threshold, and the values below the full train models represent the difference between the full model accuracy and the pruned model.