Universiteit Leiden

Master Computer Science

Combining pruning and mixture of expert for finetuning vision transformers

Name:	Daniel van Heuven van Staereling
Student ID:	s4056779
Date:	August 2025
Specialisation:	Artificial Intelligence
1st supervisor:	Mitra Baratchi
2nd supervisor:	Valerio Marsocci
3rd supervisor:	Julia Wasala

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Foundation models can help make automatic classification of images more accessible, as the model does not require excessive training compared to setting up an entirely new neural network from scratch. Due to the size of the model and the difficulty of generalizing because of the difference between the pretraining tasks, which do not translate fully to downstream tasks, performance still lacks sometimes. Better memory use would help in making finetuning the models more accesible and better generalization might make the methods show better performance helping in better decision making. Hence, we propose using a combination of Taylor pruning, which uses Taylor expansions for pruning, and mixture of experts, which uses multiple instances of a model to generalize better. By implementing mixture of experts during finetuning, we aim to improve the predictive accuracy on downstream tasks while keeping memory low by using Taylor pruning. We used the Taylor pruning method, which uses data to prune to differentiate the experts, to try to push them in different directions. Furthermore, we did a visual analysis to look at how experts differ after training the router to see what parts of images experts look at based on masks. From the analysis, we learned that experts show differences in the images in some layers but not others. We evaluated the models on two different pretrained vision transformers for three total datasets of the geospatial and naturaling imaging domains. We compared the use of mixture of experts and pruning combined to the model with only pruning or only mixture of expert and to the default model without additions. For the comparisons we looked at the memory use on GPU during training and the predictive accuracy on the validation splits. The results show the difficulty of using mixture of expert during finetuning instead of pretraining.

Contents

1 Introduction

In the last few years, machine learning has seen significant growth. Most notably in natural language processing, with Large language models making an impact. Because of the improvements in NLP, many people have been trying to improve in areas like computer vision by using models like large language models, but for images. Bigger models like this, which are called foundation models, are challenging to train, but more and more data is available, making it possible to have these big models that apply to more tasks. Using a lot of data on these big models helps in generalizing since there is a lot of information in the model.

Nonetheless, computer vision is still a difficult task. Especially since not all tasks have a large number of annotated images readily available. Furthermore, there is a big contrast between vision tasks and NLP tasks. NLP tasks have semantic information hidden in the meaning of the words that relate to the context. Images consist of pixels that are just numbers. These are more difficult to put into a foundation model. As a consequence, image models often need a lot of annotated images, which takes a lot of time to get. Alternatively, pretrained models use tasks for pretraining the model, which are different from the downstream task for which we want to use it. Both cases often require that the model still needs finetuning on downstream tasks. The representation that foundation models learn for images helps in faster finetuning. However, it does not help with generalization for specific tasks as much since most of the decoded information is not specific to the task, and in some areas, images can have difficult to recognize boundaries.

Mixture of experts can help generalize (Jacobs, Jordan, Nowlan, & Hinton, 1991). Mixture of experts is a way of routing tokens in tasks to different sub models called experts that process the data and then combine their output, so that experts specialise on different aspects of the data, so that the predictions are better. These mixture of experts models have shown great performance on multi-task learning and in architectures of foundation models (Fan et al., 2022) (Lepikhin et al., 2020). Much focus for mixture of experts is on using them for pretraining. However, there is less research on using mixture of experts for finetuning for downstream tasks. The problem is that pretraining big models takes a lot of time and many resources. Hence, we want to use a pretrained model and finetune it for a task instead of pretraining a whole model from scratch every time. So we want to see the effectiveness of mixture of experts in finetuning. The benefit of mixture of experts is that it can learn to focus on different tokens in the data. It can do this for pretraining, but it should also be able to do this for finetuning. Images in a single dataset still have different images from which the experts can learn. For classification and segmentation, images have different labels that convey different information that the experts can learn. The implementation can be similar to pretraining models that replace the feedforward mlp layer in a transformer block. The difference is that we now copy over the pretrained MLP for each expert instead of randomly initialising. The problem is that foundation models are very large, and mixture of experts only increases the size. Furthermore, for pretraining, models initialize experts differently, while now we copy them over so the experts start the same. If copying were the only step, that would not be good since they must be different to focus on different aspects. Hence, we also use pruning. Pruning removes weights based on an estimation of the most and least important weights for the model.

In this research, we propose a combined model of mixture of expert with pruning for finetuning vision transformer foundation models. The mixture of experts approach improves generalization for finetuning downstream tasks while the pruning ensures the experts are different and the memory usage stays low. The first challenge is finding a way for the experts to differ using pruning. Pruning alone is deterministic, which means the experts will stay the same. This means we need to make sure the experts are different somehow. The second challenge is to balance the experts the right way so that experts are not overused or randomly chosen. Hence, we analyse different approaches to tackle these problems. Before continuing our contributions, it is important to mention that this implementation did not find an improvement. Different approaches showed no change or a decrease in performance. Hence, part of the research involves an analysis of the models used on the data and how the router learns to differentiate experts based on visual analysis.

Our contributions are the following:

- We propose a new way of using mixture in finetuning by combining it with a pruning approach to combine the strengths of generalization and pruning.
- We evaluate our methods by looking at accuracy and memory usage on two foundation models trained on three different datasets from the natural image domain and the geospatial domain.
- We compare our methods to the base foundation models, which do not have any of the additions of pruning and mixture of experts, we compare to the model with only finetuning, and we compare to the model with only mixture of expert
- We compare with alternatives of the combined model that use different approaches we implemented.
- We do a visual analysis of how experts focus on different parts of the images.

In the following chapters, we will go through the research of this thesis. First, we cover existing work and methods related to our method in Section 2. Section 3 mentions some extra information about the background that is important to understand the methods and results. We explain the method in section 4. Then, in section 5, we explain how we perform the experiments to answer the research questions for our contributions, which we analyse in the results in section 6. Finally, we end with the conclusion in section 7.

2 Related work

There is much research regarding foundation models, mixture of experts and memory reduction methods. In this section, we discuss some of the existing methods.

2.1 Foundation models

Up and coming for machine learning tasks are foundation models. Foundation models are pretrained on a large amount of unlabeled data with some self-supervised task to learn a representation of the data for later use, for other tasks (Bommasani et al., 2021). Foundation models for NLP are easier. Text inherently has the semantic information and can be learned from context because of the distributional hypothesis (Harris, 1954). With images, this is harder. An image consists of just a large number of pixels. These pixels are just numerical values without information about a particular pixel value. The pixels are only important in groups, making it more complicated because a certain pixel value can vary in different scenarios. Due to the difficulty in vision tasks, there are multiple different methods for pretraining a vision foundation model (K. He et al., 2022) (Chen, Kornblith, Norouzi, & Hinton, 2020)(C. Zhou et al., 2024). Because there are different pretraining methods, using or comparing vision foundation models can be difficult. We look at two specific foundation models.

Image-net transformer Most foundation models nowadays use transformers (Lepikhin et al., 2020) (Touvron et al., 2023) (Zhai et al., 2022). Basic vision models use natural images. Natural image datasets can contain images like dogs, cats, cars, and other things. This model is pretrained on ImageNet-21k data (Deng et al., 2009) with a supervised classification task (B. Wu et al., 2020). The model was then finetuned with ImageNet data (Russakovsky et al., 2015). The ImageNet dataset is another version of ImageNet-21k with 21,000 labels, while ImageNet has 1,000 labels. The model of the architecture uses a simple vit-base backbone (Dosovitskiy et al., 2020). Hence, there are 12 different transformer blocks in the model. Each transformer block has the standard layout for a vision transformer. Each image has a division of 14 by 14 patches, which, when flattened, is 196 patches. These 196 patches get one extra token representing the class label. Each patch has a feature vector of 768 features for a vit-base model. For a vit-large architecture, this would be more. There is no special decoder. There is only a linear classification head. The head is a linear layer with the input dimension of the number of features, and as the output dimension, the number of classes. It is difficult to generalize to other tasks because it is trained for specific classification labels, even though it has a lot of data. Hence, finetuning is still important.

SatMAE There are also foundation models that do not work on natural images. A subgroup of foundation models made for geospatial data is called geospatial foundation models. These work on satellite images for remote sensing tasks. Satmae is one of the foundation models (Cong et al., 2022). The letters stand for satellite masked autoencoder. The masked autoencoder is the method used for pretraining (K. He et al., 2022). For natural image pretraining, some datasets have many different labels. So when trying to learn an internal representation of images, it is easier to find data for it, and it is possible to use supervised learning as a pretraining task.

Even though masked autoencoders are a possible pretraining method for natural images, they are especially useful for satellite images. Satellite images are often very similar, and datasets are often much smaller. As a consequence of the small datasets, it is impossible to use supervised learning as a pretraining task. Different satellite images can also have different things happening since they often cover big land areas.

Masked autoencoders are a pretraining method that works with only the image itself. The goal is to learn a representation of these images by only using the image itself. A masked autoencoder takes the original image as the output for what pixel values to predict. The input is also the original images, but the model masks some parts of the images. The model divides the images into patches across the height and width. Some patches get a mask. The model predicts what is behind these masks based on the other visible patches. The model has two parts, an encoder and a decoder. The model puts the images through an encoder, which encodes the information from the image into a low dimensional vector, and this is put again through a decoder to predict the full image. After pretraining, we can remove the decoder and use it with a specific task head to finetune on a downstream task. Since the pretraining task only learns representations of the images, it is still necessary to learn specific information for downstream tasks and finetune the model for masked autoencoders. Generalization is also important since satellite images can look very similar even though two images are from different classes.

2.2 Memory reduction methods

There are different methods for reducing memory. Most of them are variants from pruning or low-rank adaptation. Pruning removes weights from the neural network, while low-rank adaptation replaces layers of the network with a new lower rank matrix (Hu et al., 2022). These methods are always used on pre-trained networks since there is existing information that we want to maintain while reducing memory, since pretraining happens primarily for big models.

2.2.1 Pruning

Pruning consists of two main approaches: structured or unstructured pruning. Unstructured pruning is any method that removes weights from any point in the network (Han, Pool, Tran, & Dally, 2015). Simple variants are greedy approaches like random pruning or max weight pruning. Other approaches use extra information like data, which is the case for Taylor pruning (Molchanov, Mallya, Tyree, Frosio, & Kautz, 2019). Taylor pruning looks at a subset of the data and then approximates the importance of the weight by looking at the gradient change and using Taylor approximations. Structured pruning removes an entire group of weights (Y. He & Xiao, 2023) (Li, Kadav, Durdanovic, Samet, & Graf, 2016). This method usually contains more steps but often results in higher memory reduction.

Different methods change the pruning process. Iterative pruning prunes multiple times for different iterations (Liu et al., 2024). Other methods include recap, which does iterative pruning but with the addition that they do not reinitalize the optimizer each iteration, and they do the pruning on CPU (Ilhan et al., 2024). Alternatively, prebackrazor, which does the pruning as part of the finetuning by implementing it in the backpropagation (Yu et al., 2024). These are for regular models, but there are also pruning methods that work on a mixture of expert models. However, these pruning methods work on pretrained mixture of expert models (Xie et al., 2024) (Lu et al., 2024). They use the information from the pretrained router to prune or remove entire experts for specific downstream tasks. There is no alternative for finetuning, where the router or experts do not provide information.

2.2.2 Low rank addaptation

Lora is different. Lora uses a different way of reducing the memory cost of fine-tuning by freezing the original pretrained weights and learning only a pair of small task-specific matrices next to the original weights in the model (Hu et al., 2022). Concretely, instead of updating the complete model during fine-tuning, Lora updates only the weight changes in the form of two low-rank matrices. During training, the forward pass adds the product of these low-rank matrices to the frozen weights. Hence, the model is the same, but the training cost is substantially less because there are fewer optimizer parameters (since the low-rank matrices are smaller).

Because LoRA does not change the original weights, the pretrained knowledge is kept intact, and the model can store multiple tasks by keeping several LoRA adapters without duplicating the base model. The low-rank matrices can be combined into the complete model with a single matrix addition at inference time. Hence, memory saving is only during the storage of multiple models and finetuning, not during inference. During inference itself, there is no reduction in memory storage. Pruning does not have that drawback, and mixture of experts can make it possible to use it on multiple tasks with a single model.

2.3 Mixture of expert models

Mixture of experts is a technique that is becoming more popular for neural network models. Mostly in foundation models, this is very useful since it saves on memory usage during inference and computational cost while keeping the number of total parameters high. It also makes extending to multi-task learning easier, like M3Vit (Fan et al., 2022). Many models are created with mixture of expert in mind and show good performance (Lepikhin et al., 2020) (Fedus, Zoph, & Shazeer, 2022).

A good example of a model that uses mixture of expert during finetuning is MoeLora (X. Wu, Huang, & Wei, 2024). MoeLora uses Lora for mixture of expert by using different trained Lora models and combining them as experts. Multi-task learning uses multiple instances of the model for different tasks as well. However, in this case, MoeLora uses a single vocabulary for the different tasks, which is possible for NLP tasks. The problem is that many of these models use mixture of experts during pre-training and not during finetuning. Either that or they use multi-task learning like M3Vit, where it has to add the task as a feature with a different decoder for each task, which does not add to generalization for each individual task.

Mixture of experts works with a router that chooses which expert to use for a data instance. The router depends on the use case and how you implement it. Multitask learning, like M3Vit, uses a different router than Gshard. For standard tasks like our case, we can use a standard router that decides which expert to choose based on the same input as the experts.

A problem is the routing balance. If the router uses one expert, it will be biased to use it again since that expert has had more training. Hence, the routing network keeps using the same expert. We want all experts to train an equal amount on the data. Therefore, it is a problem if the router picks only one expert. Hence, there are different balancing techniques to make sure this happens. Like using an additive auxiliary loss (Lepikhin et al., 2020). We will explain this one in more depth later. In short, based on the batch balance, it adds a loss for each layer for backpropagation. Another balancing technique is to use bias (Wang, Gao, Zhao, Sun, & Dai, 2024)s. After computing the router values, we add a bias value to alter which experts the router will choose. The bias changes after each forward pass based on balancing. If an expert receives many tokens, the bias decreases, and if an expert gets very few tokens, the bias increases. Another routing method is expert choice (Y. Zhou et al., n.d.). Expect choice works by selecting tokens for each expert instead of picking an expert for each token. Selecting based on tokens ensures that experts train an even amount. However, in some cases, images would benefit from an uneven split in tokens. Hence, this could be balancing too much.

2.4 Relevance

To the best of our knowledge, this study is the first to look at using a mixture of an expert only for finetuning and combining it with a pruning approach. Our model tries to improve on standard foundation models. Earlier papers tried improving a model by implementing a mixture of experts on pretraining, like Gshard, or using multi-task learning by using experts separately for different tasks, like M3Vit. Instead, we look at the improvements on individual tasks and let the router learn the differences between classes. However, since we approach this task differently, there is a big memory increase, which we try to tackle by using mixture of experts in combination with pruning. Furthermore, by pruning each expert on different parts of the data, we expect this to help in the differentiation process of learning to specialise experts.

3 Theory

3.1 Pruning

Multiple methods exist to decrease memory size. One of these is pruning. It is a simple approach. By masking a percentage of the weights in a neural network model, we remove weights that require memory, so we save memory.

The difference in which weights to remove can have a big impact on the eventual performance of the pruned model. Many methods exist for estimating the weight importance, which determines which weights are removed and which stay.

There are two variants of how to prune. Pre-finetuning pruning and post-finetuning pruning, Post-finetuning pruning does finetuning first and then pruning. The benefit here is that the weights can be accurately estimated and we can get better performance on the downstream tasks. Pre-finetuning does pruning first and then finetuning. Since the pretraining task does not always connect accurately enough to the downstream task it can be difficult to estimate the important weights. Since after finetuning the weights might change. The benefit of pre-finetuning pruning is that the pruning happens on cpu so it requires less memory space on GPU during the finetuning process so we can save on GPU size.

3.2 Mixture of experts

Mixture of experts uses different instances of the same model and trains the combined pool of experts on a task. During training, an extra router network looks at the input images. The gating network routes the image or token input to different experts, so not all experts look at each image or token. Therefore, each expert can train on different aspects of the image and improve generalization. An example is shown in Fig. 1. The routing network receives the same image tokens as the experts. The router output is N values representing a score for each expert. A softmax function turns the scores into probabilities to determine how to combine the output of the expert. The highest expert receives the tokens, or the expert's output is scaled by the probability and combined.

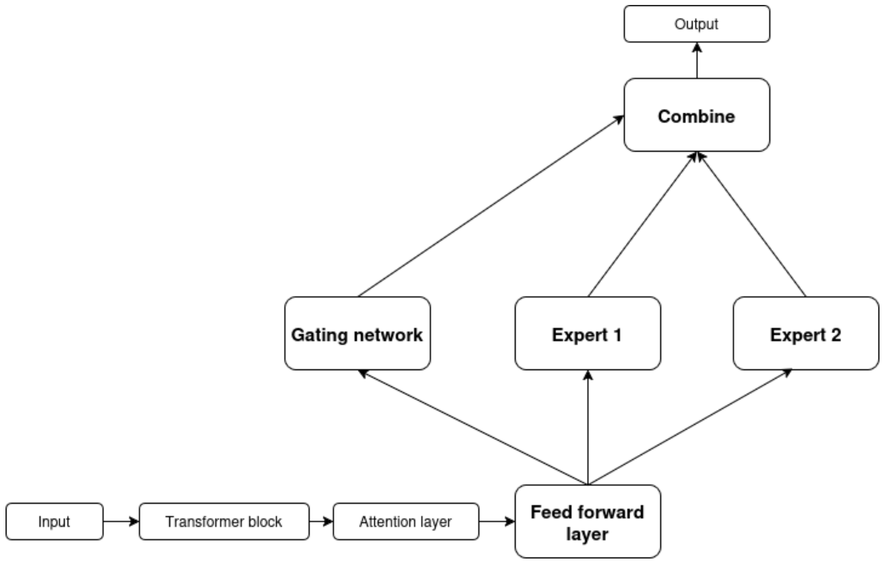


Figure 1: This shows the structure of a mixture of expert model

Auxilary loss To balance expert use, it is possible to add an auxilary loss to the normal loss. The loss depends on how well the experts are balanced. For each transformer block, we add the loss as in the equation below.

$$loss = \sum^{B} \sum^{E} \lambda N P \times f \tag{1}$$

Where λ is a tunable parameter, N is the number of experts, P is the fraction of the router probability given to each expert, and f is the fraction of tokens dispatched to each expert. The loss is a sum over each expert E and block B in the transformer. It is not possible to see which expert or which block contributes more to the loss, the goal is just to lower the loss.

3.3 Entropy

Entropy describes randomness. Entropy gives us an idea of the level of randomness. A high entropy value means there is a lot of randomness and disorder. A very low value means the process is ordered or expected, not random. In the case of mixture of experts, a high entropy value shows that experts are chosen at random, while a low entropy value means it always picks the same expert, so we know the expert the router will choose at all times. A simple equation for calculating entropy is shown below.

$$entropy = \sum^{E} -p \log p \tag{2}$$

E is the individual expert, and p is the expert probability for a token. The number of items or experts determines the highest possible entropy. The boundaries are $\log E$ for the upper limit, and 0 is the lowest value.

3.4 Taylor Series

A Taylor series or Taylor expansion is a mathematical notion of a series of a function that is infinitely differentiable. The series is described like this:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ... = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{3}$$

Using the Taylor series, it is possible to approximate a function by taking a certain order of the series. Calculating certain order terms is easier in some cases than the whole function, which is why it is useful. Many cases use first order and second order Taylor series or approximations. The series starts at the zeroth order term and then continues to the nth order. Hence, first order Taylor approximations use the first two terms, while second order Taylor approximations use three terms which makes approximations more accurate.

4 Methods

This section describes the proposed method of implementing mixture of experts on foundation models with pruning. We first clearly state the problem, then we explain the concepts we use and how they interact to solve the problem and then we go in depth on the methods that we implemented.

Problem statement It can be challenging to generalize information across classification tasks. Foundation models are suitable for this but are difficult to use, especially for image tasks. Hence, we want to improve their generalization further, but we do not just want to increase the number of parameters since foundation models are already so big.

To tackle this, we implement mixture of experts and pruning. Mixture of experts and pruning are already existing methods. Our method combines them with the way we prune the different experts with different data instances. Hence, we use Taylor pruning, which uses data to estimate weight importance.

We perform multiple tasks, as we explained. The foundation models were pretrained on 224 by 224 images. Hence, we preprocess by cropping and reshaping the images to fit this size. Formally, we can express the problem after preprocessing like the following:

We have input $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^{224\times224\times3}$ and $y_i \in \{0, 1...C-1\}$. x is the image with size 224 by 224 with RGB channels, and y is the class integer.

The model acts as a function that transforms the image output into the label prediction. $f(x) \rightarrow y$. The goal is to optimize the function to get the correct label. To improve this process, we want to replace part of the model with the mixture of experts approach. However, we want the model's memory use to stay low. How we do this is explained in the following sections.

Overview The goal is to improve the finetuning process of pretrained foundation models. Specifically, we want to improve pretrained vision transformers by implementing mixture of experts and pruning. Vision transformers have a repeating structure because they have multiple blocks, each with an attention layer and an mlp layer. We implement mixture of experts by replacing the mlp layer with the mixture of expert module. Then we prune the whole model to lower the memory requirements.

4.1 Applying Mixture of expert

Experts For the method, we combine mixture of experts and pruning. Like we showed in Section 3, mixture of experts consists of multiple modules called experts. Transformers have an attention layer and a feedforward mlp layer. The mlp layer has two linear layers, making it a prime candidate to replace. It is possible to replace the attention layer. However, we chose to replace the mlp layer for ease of use since that is also more common (Fedus et al., 2022) (Lepikhin et al., 2020).

The mlp layer has an ingoing and outgoing feedforward layer that goes from 768 features to 3072 and back. Since we perform pruning on the expert layer, replacing both linear layers with a single expert instead of either one is easier. Pruning can remove some of the input/output features that are not important. Hence, the output from layer 1 must have the same number of features as the input for layer 2. However, if one expert removes a different feature from the other, this can affect the output and input. If we treat the whole mlp layer as one expert then this is not a problem. If we remove all weights connected to an input or output node, we keep the node to avoid issues. However, we can remove intermediate nodes because we treat the whole mlp as one expert. Then we replace the layer with MoE. For pretraining methods, they then initialize the MoE. However, we use it for finetuning, so the mlp has pretrained weights. Therefore, we copy the weights over. An important part of mixture of experts is that the experts need to be different. Hence, they start out differently. Since we copy the same weights over, we need to differentiate the experts in another way. The main way is by doing the pruning, which we explain in the next section.

Router network The other part is the router. The router is the most important part of mixture of experts since the router decides which expert gets which tokens. Transformers divide images into 14 by 14 patches, and each patch is embedded as a feature vector. The patches make it easier to work with the router since we can use a linear router network. From the entire batch of images, the router takes each patch from each image and transforms it into a continuous feature vector for the number of experts. We take a softmax of this function for the next step to get the probabilities. Either we use 100% of one expert's output. Alternatively, we combine multiple experts by multiplying by the probability of the patch.

4.2 Pruning

Taylor pruning Taylor pruning, which we do after setting up the model but before training is an existing method. Taylor pruning works with the Taylor series which we explained in the Theory.

The method wants to estimate which weights have the most impact when removing them. The equation formalizes the importance estimation with I_m representing the importance of weight m. W is the weight matrix, E is the loss factor, and D is the dataset.

$$I_m = \Big(E(d, \mathbf{W}) - E(D, \mathbf{W}|w_m = 0)\Big)^2 \tag{4}$$

By removing a weight m and seeing the change in the model's performance, it is possible to estimate the impact. However, it takes too much time to compute the full equation for each weight. Hence, we use second-order Taylor expansions. Using the first-order approximation, this gets the following equation:

$$I_m^{(1)} = \left(\frac{\partial E}{\partial w_m} w_m\right)^2 \tag{5}$$

$I_m^{(1)}$ is the importance score for the first-order Taylor expansion and w_m is the weight m. The formula only uses the gradient and the weight score. Hence, it is a simple equation that only needs a single forward and backward pass to estimate the importance of all the weights without the need to remove weights one by one. The weight pruning happens based on the lowest importance to the highest importance until we have pruned the required amount of weights.

Pruning approach The experts are copies of the pretrained mlp layer. Experts require diversity to diverge during training so they cannot be too similar. To differentiate the experts, we prune them based on different data. This is why Taylor pruning is a good method to apply to our method. Instead of pruning the whole model at once with the same data, we do it in steps. First, we take multiple subsets of the data. We take from each class close to the same number of instances. Then, we create the subsets by combining the instances of classes so that each subset contains instances from the same number of classes, with the same number of subsets as experts. If there are 12 classes and 4 experts, expert 1 gets all instances from classes 1,2, and 3. Expert 2 gets all instances from classes 4,5, and 6, and so forth.

The second step is to prune each expert with the subsets. Normally, for mixture of experts, the router decides which expert gets which patches of tokens. In this pruning step, we change this by only choosing one expert for each image in a subset. Hence, we send all patches in an image for a subset to one expert.

The last step is to prune the attention layers with all labels. We do this again with a subset by sampling the same number of instances from each class so that the total number of instances is the same (10 times the batch number) as for each individual expert. This step would be the same as just regularly pruning everything. In this step, we let the router choose the experts again during the forward pass for weight importance estimation.

5 Experiments

In this section, there is a description of all the experiments that were necessary for answering the research questions.

- Can pruning of the model with MoE implemented for a foundation model reduce memory while achieving improved or similar performance compared to the finetuned base model and other baselines?
- Can the Pruning decrease the required memory size for a foundation model while maintaining performance compared to the finetuned base model and other baseline models?
- Can MoE improve the performance of a foundation model compared to the finetuned base model and other baseline models?
- For mixture of experts how does the router differentiate what tokens to send to which experts after training for a vision transformer?

The main goal of the research is to see whether the model of combining pruning with mixture of experts is worth using. Hence, the goal is to see whether the model has better predictive accuracy than baselines or has better memory usage. Therefore, the first research question tries to answer whether there is better or similar performance compared to baselines with less memory usage. The second research question checks whether pruning behaves as expected and decreases memory size while keeping close in accuracy for the specific foundation models we used. The third research question tries to answer whether mixture of experts even works without any pruning when applied during the finetuning process to improve the model performance. Since it will not improve memory, it can only improve predictive accuracy. Performance increase has to be a consequence of the routing method and not from increasing the number of parameters. Hence, we examine the experts in research question four. The results for mixture of expert show no improvement. Hence, the last question analyses how the router differentiates which tokens to send to which experts by looking at image masks. We explain the steps for answering the questions in the following sections. First, we start with an explanation of the data. Followed by the baselines, the chosen hyperparameters, and the setup for doing the experiments.

5.1 Data

To simplify the analysis of the experts, we used several classification datasets and no segmentation datasets. Only using classification tasks helps explain the differences between experts since it is easier to isolate different classes. We used three datasets: two from the geospatial image domain and one from the natural image domain. The geospatial datasets came from GEO-Bench (Lacoste et al., 2023). These are popular datasets adjusted for benchmarking. The natural image processing dataset is MIT Indoor scene recognition(Quattoni & Torralba, 2009).

m-forestnet Forestnet is a dataset consisting of satellite images taken of areas where all or part of the forest is removed. m-forestnet means that it is an adjusted variant created by geo-bench (Lacoste et al., 2023). The dataset has 12 classes, each being a different driving factor for why the forest area is gone. We picked the dataset based on the moderate amount of labels and because state of the art performance was low enough to have possible improvement.

m-pv4 Pv4 is another dataset from geo-bench (Lacoste et al., 2023). The dataset shows images that may or may not contain a solar panel. The only two labels are binary for images containing a solar panel or not. Since it only has two labels, it gives a different perspective when analysing the experts.

MIT Indoor scene recognition This dataset is for the natural image domain (Quattoni & Torralba, 2009). The task is scene classification. Different locations can be the living room, the airport, or others. There are a total of 67 classes. The performance for the models on this dataset is around 70%, making it easier to see whether there is improvement.

5.2 Baselines

The mixture of expert and pruning method works as an application to a pretrained model. Therefore, the baseline is the foundation model SatMAE and the tinyimagenet pretrained model. We also compare the pruned mixture of expert model to the pruned model. Hence, the baselines are the default model and the pruned model.

Satmae As discussed in the related work, Satmae is a geospatial foundation model (Cong et al., 2022). The structure is a standard vit-base model (Dosovitskiy et al., 2020). The structure has 12 transformer blocks with an attention layer and an mlp layer. The images are split into 14 by 14 patches, each of which has 768 feature tokens. The attention layer has 16 attention heads. The model is a checkpoint trained by the original paper on geospatial data. After pretraining, it is a regular transformer and loads the same as any other vision transformer. For loading the model, we use the timm library (Wightman, 2019).

Tiny image net model This model has the same structure as the SatMAE model. It is a vision transformer with 12 blocks, each with an attention layer and an mlp layer. As discussed before, the only difference is the pretraining. The library again uses timm.

5.3 Experimental setup

There are several settings for the training process. The models are large. Therefore, we set the batch size to 8 to fit the training process on the available GPUs. The learning rate performed best at 10^{-5} so we chose to use that.

The expert model had three different hyperparameters. The number of experts is 4. Four was the number of experts that fit on the CPU/GPU. A few tests were possible on a bigger model, which showed that eight experts also did not show improvement. Hence, the need of more than four experts was not necessary. Other papers used this order of magnitude of experts before as well, with good performance (X. Wu et al., 2024),(Jia et al., 2025). To select experts, we used top-k selection. We chose the top 2 experts and combined them by multiplying the softmaxed probabilities by their output. We chose two instead of taking the top expert only since it performed better.

For balancing, we tuned the two methods from the related work. The first was the auxiliary loss (Lepikhin et al., 2020),(Fedus et al., 2022). Auxiliary loss outperformed the other method, which used an additive bias value to the gate output (Wang et al., 2024). The auxiliary loss had a loss scaling factor, which we set to 0.1 since this performed the best. We did not add noise to the expert model's routing output, as this showed no difference.

For the final model with mixture of experts and pruning there was a design choice. Which MLP layers should be replaced with experts? The analysis looks at three different settings: replacing all layers, only replacing even layers, and only replacing the first layer. Analyzing the data and the experts revealed some behavioral differences in experts, which is why we decided to analyze these differences.

Additionally, we used some extra alternative models. First, we used a warm-up on the experts by training them on a part of the label classes for 10 epochs. Warm-up trains the individual expert for each step and the attention layer. Freezing the attention layer during the expert warm-up performed worse. Therefore, we only analyse without freezing. With warm-up, the experts diverge more at the start. The second alternative, was to add random Gaussian noise to the expert initialization. As they start out the same, the noise should make them diverge more. Third, we ran a model with different lambda values based on tests we did, which are explained more in the results. For this setup, we changed lambda for the first six transformer blocks to 0.01 and for the last six transformer blocks to 0.05, which was the best performing combination after some slight tuning. Last, for pv4 since it has only two labels we used an alternative with two experts instead of four so it has only one label per class for the pruning.

We ran each experiment (our method and the baselines) three times for validation with different seeds for randomization. The dataset splits are based on default splits used in Geobench and the standard Mit dataset. For performance comparison, we used two metrics. The first metric is prediction performance, which we measure with the model's accuracy. The second is model size, which we measure with the maximum memory usage of the VRAM on the GPU. These two metrics tell us the total performance of the models.

Due to hardware limitations the experiments where split. The experiments for the geobench datasets were ran on a different hardware setup than the one for the mit scene classification dataset. The geobench datasets used a nvidia GeForce RTX 4070 Ti for the GPU with an Intel Core i7 14700K 2.5GHz 20 cores CPU, and the Mit scene classification dataset used a nvidia GeForce GTX 1080 Ti with Intel Xeon E5-2620 2.10GHz 16 cores CPU.

6 Results and discussion

The first three research questions examine the prediction accuracy of the models. Hence, we start examining these three together. For the second subsection, we show the results for research question 4, which shows an analysis based on visual images.

6.1 How does the default model compare to using mixture of expert, pruning or both

The first part is the actual performance of the model as shown in Table 1.

Model	Memory Usage	Prediction Accuracy
0% pruned baseline	1939±0 MB	93.5 ± 0.479
95% pruned baseline	524±22 MB	93.2 ± 0.572
95% pruned moe	938±8 MB	93.2 ± 0.125
0% pruned moe	4686±7 MB	93.6 ± 0.375
2 experts 95% pruned	682±16 MB	93.0 ± 0.309

Table 1: Top average performance for each model on the pv4 dataset from geobench on validation

In Table 1, we can observe the performance for the dataset pv4. From the table, we can observe that the performance for all models is very similar. With only pruning, we can still get accuracy close to the default model without pruning, while reducing the memory size of the model. Hence, to answer research question two, performance is close to the original performance with less memory use. With only mixture of experts, there is no significant improvement in the model performance over the baseline. The model performance is 0.1% better but with a significant standard deviation. The memory is worse than if we only prune, which we expected, since the mlp layer is copied multiple times. Compared to just pruning, the combined model has a larger size while predicting less accurately. The two metrics clearly show that the model performs worse. The dataset is binary, but even with only two experts, there is worse performance. The model uses slightly less memory, but accuracy is slightly worse. The slight dip in performance could be because there are fewer parameters, even when the two experts fit more to two classes.

Model	Memory Usage	Prediction Accuracy
0% pruned baseline	1938±0 MB	52.2 ± 0.859
95% pruned baseline	511±6 MB	51.8 ± 0.460
95% pruned moe	931±15 MB	51.1 ± 0.700
0% pruned moe	4696±2 MB	52.3 ± 0.126
95% pruned moe with noise	1031±10 MB	51.4 ± 0.909
95% pruned warmup	931±20 MB	51.1 ± 0.504
0% pruned warmup	4682±3 MB	51.6 ± 1.174
95% pruned moe first layer	524±21 MB	50.3 ± 0.787
95% pruned moe even layers	697±13 MB	51.1 ± 1.111
95% pruned moe with different lambda	964±18 MB	51.9 ± 0.672
0% pruned moe with different lambda	4686±4 MB	51.8 ± 0.239

Table 2: Top average performance for each model on the forestnet dataset from geobench

From the results in Table 2, we can observe a similar conclusion to the pv4 dataset. Pruning performs closely to the default model without changes while reducing memory usage by over a factor of 3. Mixture of experts performs about the same as the baseline, but with more memory. Combining pruning and mixture of experts does not improve over the pruned baseline, but has worse performance. In this case, we tested a few additional setups. Warming up the experts as described in the experimental setup shows the same performance as the same level of just pruning, and in the case of not pruning, mixture of experts has worse performance. Adding noise also shows nearly the same performance. The results show that the addition of different techniques to differentiate the experts before the full training process does not provide benefit.

When only using mixture of experts in the first layer, predictions are less accurate as well, but memory use is better since we are using fewer expert layers. Using expert layers in every even layer shows no difference, but since memory use is better, it is an improvement over using mixture of experts in every layer.

Using different balancing values for the mixture of experts layers in the first six transformer blocks and from the last six transformer blocks shows improvement over the default mixture of experts approach. However, entropy shows more randomness, as we will discuss later. Hence, the increase in performance could mean that its routing is not necessarily improving, but is getting worse. The improved performance could be due to the increased number of parameters. We observe the same results by looking at the same model with no pruning. Predictive accuracy is worse in that case than the default model. However, more research would be required to determine why we see an improvement in performance here.

Model	Memory Usage	Prediction Accuracy
0% pruned baseline	1939±0 MB	58.7 ± 0.353
95% pruned baseline	814±12 MB	43.7 ± 0.464
50% pruned baseline	1133±3 MB	53.1 ± 1.027
95% pruned moe	1436±50 MB	49.7 ± 1.075
50% pruned moe	2657±11 MB	55.5 ± 0.889
20% pruned moe	3848±8 MB	56.9 ± 0.918
0% pruned moe	4706±1 MB	59.0 ± 1.216

Table 3: The top average performance of each model for the mit scene classification dataset for validation

Table 3 shows the same conclusions as for forestnet. When comparing the base model and mixture of experts, both with 95% pruning, then mixture of experts performs much better. However, if we use pruning 50%, it performs way better for less memory than mixture of experts. The reasoning is that in this case, the better routing of tokens does not help improve the accuracy, but just the addition of extra parameters. We can see this using the standard expert mixture without pruning compared to the base model. The performance is only 0.3% better, which is only a small amount for more than double the memory usage. There is a difference between geospatial foundation models and basic natural imaging foundation models. The difference between pruning and not pruning is much bigger, and adding parameters using mixture of experts shows a bigger impact. The difference could result from how pretraining happens. Satmae uses masked auto encoders which is an actual specialised pretraining task that learns a representation of images instead of labels. The tinyimage net model uses supervised learning on a lot of different labels to learn the representation. Hence, it might have different network parts that might be grouping information. If the information is in groups, removing close weights can impact performance more. If weights related to some information are scattered, then there is less difference if pruning removes a group of weights. However, this needs more research.

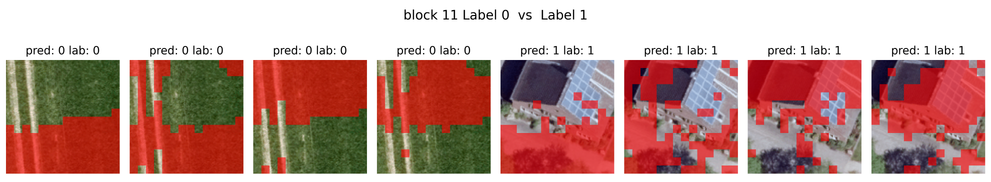
Generally, the model performs the same as the baseline with the same level of pruning. Showing that any improvement most likely comes from the extra parameters. Adding different approaches before finetuning does not help improve prediction accuracy either. Hence, the bad performance cannot be due to the experts being different enough.

6.2 How does the router differentiate experts after training?

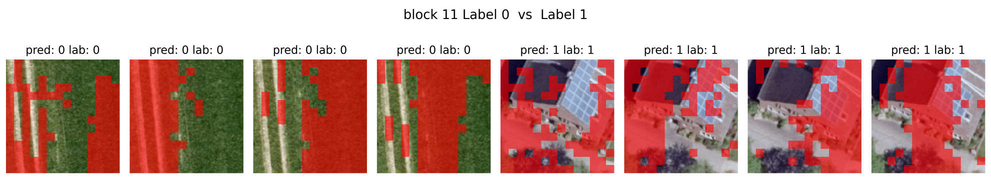
The experts should work differently so that they learn different aspects of the data. To analyze whether there are differences other than just looking at performance metrics, we also look at patch distributions between the experts and visualize some of the data.

We divide images into 14 by 14 patches, which is 196 patches when flattened. These patches are embedded with feature vectors, which go through the network. The gating network decides which expert receives which patch. The distribution of how patches are divided can differ between classes and experts. To understand how the experts work, we look at the gradcams of the images and with a mask dependent on the gating network, we visualize how the gradcam changes for each expert.

pv4 By looking at which patches of the images route to which expert, we can visualize the mask for the models and observe any peculiarities. Fig. 2 shows two visualizations. One for both the 95% pruned (top) and 0% pruned (bottom) models for the mixture of experts. There are four experts and two possible labels, so these are only one example of each label. The left four are the four expert masks for label 0, which is no solar panel. The right four are the same experts, but for label 1, which is that there is a solar panel in the image. From both we can observe the same thing. Namely, the model is not learning to route specific features to the experts. Instead, the router learns to route the same patch position to each expert (with some noise). The top model shows that the first two experts mainly focus on the bottom half for both labels, while the last two experts focus primarily on the top half of the images.



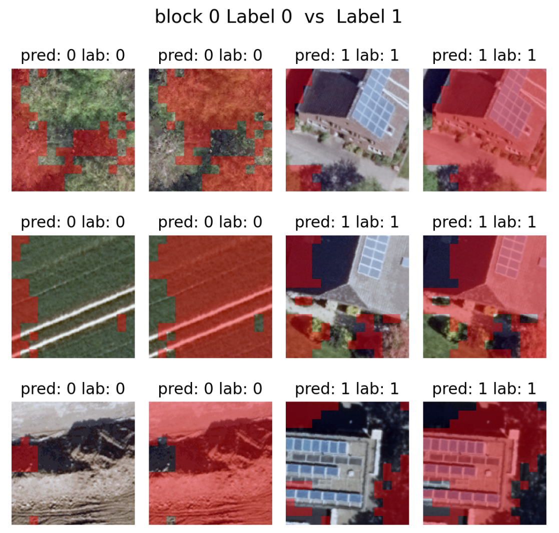
(a) This image shows the masks for the pruned expert model with 95% pruning.



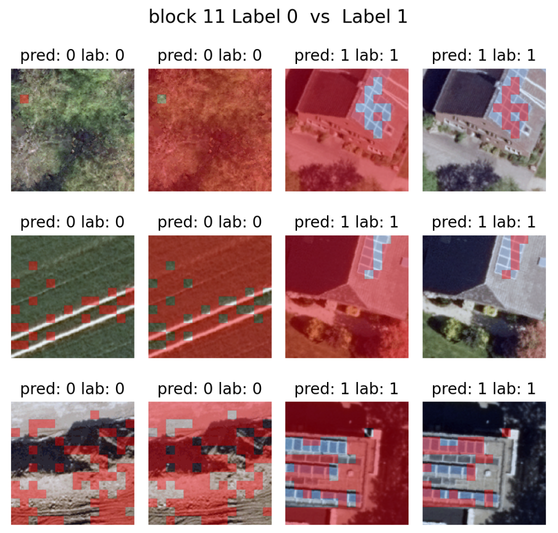
(b) This image shows the masks for the mixture of expert model that was not pruned.

Figure 2: This shows the routed patches for four experts for 1 image of both labels. The top sequence is for the 95% pruned model and the bottom sequence is for the mixture of expert model without pruning.

There are only two labels. Hence, we tested another model with two experts for this dataset. Even though accuracy did not improve, we see a difference in the visualization. Fig. 3 shows four images in each sequence, the two left images are the two experts predicting one example of label 0, and the two right images do the same for label 1. In this case, there are three examples for both labels. We have two different visuals. We visualize the last transformer layer and the first transformer layer. Both experts behave differently in the last transformer layer. For label 0, only one expert gets the majority of the tokens, which is the second expert on the right. For label 1, mainly expert 1 receives the tokens. However, there is a slight difference. Expert 1 has a small part, it does not receive patches from, which are patches that contain the solar panel. These are routed to the other expert. Hence, for two experts, there is a difference. However, this does not change the accuracy of the model. So even though the experts are showing better differentiation it does not seem to help.



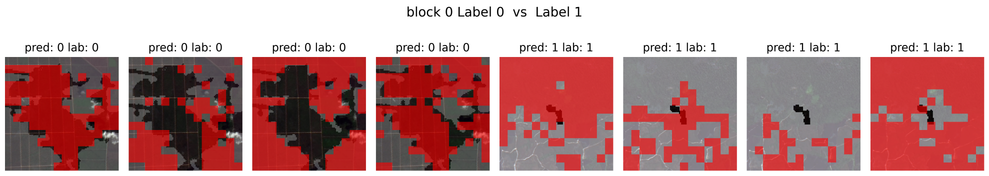
(a) This is the image mask for the experts in layer 1 for the model with two experts.



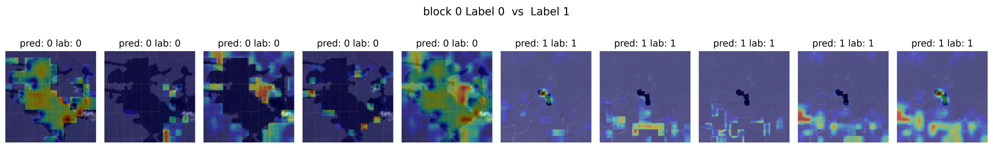
(b) This is the image mask for the experts in layer 12 for the model with two experts.

Figure 3: This image shows the masks for the pruned expert model with 95% pruning but with only 2 experts instead of 4. It shows the images for two transformer blocks. Left is the transformer block 1 and on the right is transformer block 12.

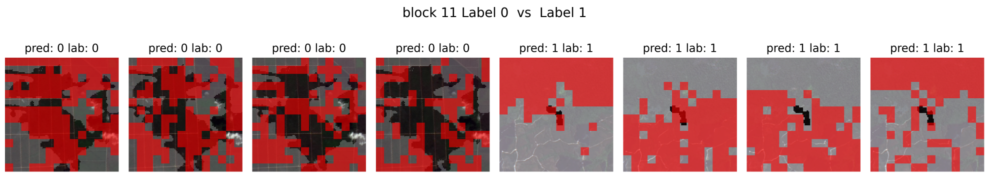
Forestnet For forestnet we can do the same as we did for pv4. We can look at the masks for the experts again. Forestnet has 12 labels. However, we pick an example from two labels to keep it simple. Fig. 4 shows three sequences. (a) shows the mask for the first transformer layer of the pruned mixture of expert model, (b) the GradCAM for the first layer, and (c) the mask for the last layer of experts. The masks are the same as before for the pv4 dataset. GradCAM computes the importance based on gradients for the whole image. The GradCAM images show the value that GradCAM gives, but multiplied by the masks. As the data description mentioned, Forestnet outlines where the old forest used to be. The left image example in Fig. 4(a) shows that one or two experts look at this outline in the middle. The other two experts are looking at the area outside that. For the right image, this is not the case. The differences indicate that there is some differentiation that is not just random, similar to pv4. However, if the forest area is too small, like in the right image, then it is random. From (c), we observe that the experts are more random for the last layer. The gradcam images show that the “important" gradients are in the forest area outline, which makes sense since that is what we are trying to classify. However, it could mean that only one or two experts are focusing on these parts because the gradients are higher there. So, the router only uses one or two experts, and the others do nothing important.



(a) First image



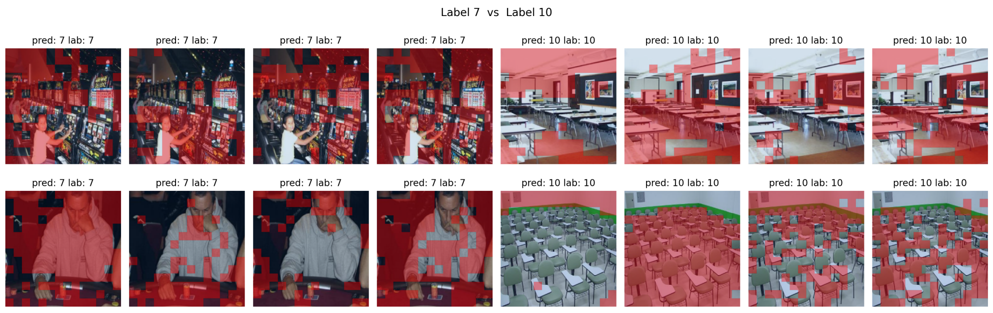
(b) Second image



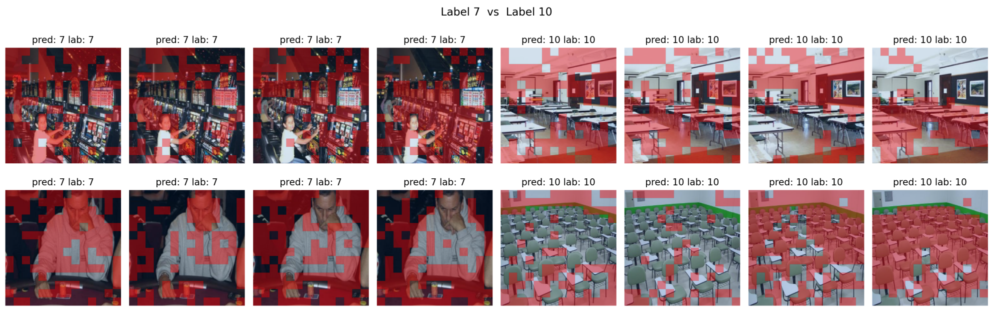
(c) Third image

Figure 4: Three images stacked vertically

Mit scene classification For the MIT scene classification dataset, we did another analysis on the image masks for the experts. Fig. 5 shows the masks for two models. One was mixture of experts with 95% pruning (a), and the other was mixture of experts without pruning (b). There are four examples: two examples from class 7 on the left four columns and class 10 on the right four columns. Both models correctly predicted both of these. The examples for class 10 show that for the pruned model, expert 1 focuses more on the ceiling, while expert 2 focuses on the rest. Experts 3 and 4 are similar but with more noise added. For the model without pruning, this is not as visible. Expert 4 shows a boundary between the wall/ceiling and the chairs in example two, but there is a lot of seemingly random noise. Both models are more random for class 7, where there is no clear ceiling. The only thing the pruned model seems to learn is to differentiate between ceilings and the rest. However, this is difficult to observe since the boundary between the ceiling and the floor is sometimes unclear. The differences mostly happen for the pruned model, even though it is performing worse by a significant amount. So even if it does differentiate in this way, it is not helping prediction accuracy.



(a) First image



(b) Second image

Figure 5: Two images

In general, the experts can focus on different parts of the image, even if it is a tiny bit. The differentiation is in some cases random. The models that do not differentiate randomly are not performing better so the effectiveness of how the router differentiates does not seem to have immediate influence on the performance. However, more research would be necessary. As the routers do differentiate experts, there is no reason to believe that the mixture of experts is not working.

Entropy For some of the models, we calculated the entropy for the experts. In the case of four experts, there are four options, so the upper limit is log 4, which is about 1.4. The lower limit is always 0. 6 shows the average entropy for the 12 transformer blocks in the model for three different models. One model for the pv4 task and two models for the forestnet task. The pv4 task (blue) and the default model for forestnet (orange) have similar conclusions, even though the effect is more apparent for forestnet. The figure shows that the first few transformer layers have an entropy value that is too high. The further it gets the lower the entropy is. 0.3-0.8 is a pretty good range since its not too small but not too high. The problem is that the first transformer blocks are too high, meaning they are too random, while the last six specialize more. Hence, the last six blocks have good entropy, but the first six do not.

The other model in green is the mixture of experts model, but with a different lambda value for the last six blocks than for the first six blocks. After some small tuning, we got a model that is also in Table 2. The performance is better than the default mixture of experts model but not than the baselines. The entropy is also higher and not lower. The lambda values for the last six experts are the same or similar, but the first six blocks had a lower lambda value to make them specialize more and lower entropy. Instead, the last six blocks became more random as indicated by the higher entropy. The first six blocks have close to the same value, but the last six for the green model are higher. The increase shows that the effect of the interplay of different lambda values is difficult to predict, and tuning them is difficult.

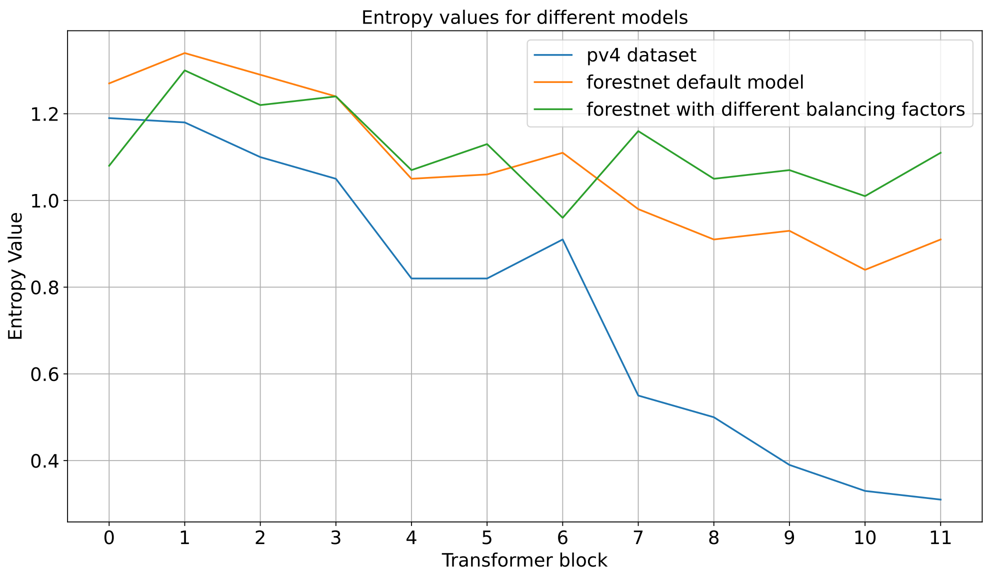


Figure 6: This figure shows the average entropy in each transformer block for a few models that use mixture of expert.The max possible entropy is 1.4 the lowest is 0 and we expect a value in the middle. The models have lower entropy for transformer blocks further into the model. Trying to lower the entropy in the first half of the model instead makes the entropy higher in the later half.

7 Conclusions

In this thesis, we explored mixture of expert and Taylor pruning. We tested whether combining mixture of expert and Taylor pruning can improve finetuning of pretrained vision transformers. The goal was to improve memory or predictive accuracy compared to the default pretrained vision transformer.

As expected, pruning alone showed improved memory use compared to the default finetuned model without pruning, while maintaining predictive accuracy. Implementing mixture of experts during the finetuning stage showed similar predictive accuracy or worse compared to the model with the same level of pruning. However, the mixture of expert approach showed worse memory use. We used three datasets, one from the natural image domain and two from the geospatial image domain. The different domains helped show that the results are valid for more than one type of image. The varying number of classes showed that it does not matter whether there are few or many classes, the method does not show different performance.

From the testing, we learned that in some cases, there are improvements for a mixture of experts, but any improvement can most likely be attributed to just the extra parameters and not the better routing method.

Furthermore, we found differences in how tokens are distributed among experts from visual analysis of the token masks provided by looking at the router. These differences are, in some cases, random. However, we found that even when they are not random, they do not provide any improvement and sometimes even a decrease in predictive accuracy. The visual analysis does show that the method works as we expect in some cases, even if it is not improving performance.

Lastly, we looked at the entropy which showed values that are too high. We tested some models to improve the entropy, which in turn made it worse. Hence, these tests show that it is difficult to force entropy to go down.

There are also some interesting avenues for future work to explore. First, it is unknown why the model shows some differences between experts but still shows no improvement in performance. Second, it is unknown why lowering the expert count shows lower performance for pv4 but more visual differences for classes in the masks.

Furthermore, this thesis only looked at classification tasks, but other types of tasks might show different levels of effectiveness, such as segmentation and object detection. This thesis did not go deep into hyperparameter tuning for the number of experts, top-k value, balancing parameters, and batch number. These parameters did show an effect, but it is challenging to find how much without more testing. Lastly, we saw that entropy is difficult to control. More research could help to find out how to lower entropy to more reasonable values, which could help increase performance.

In sum, mixture of experts did not outperform basic pruning and pruning alone is still a strong baseline for memory reduction. Mixture of expert might work well for pretraining large models or multi-task applications, but for finetuning single downstream tasks it is harder to find a good use.

References

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., … others (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).

Cong, Y., Khanna, S., Meng, C., Liu, P., Rozi, E., He, Y., … Ermon, S. (2022). Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery. *Advances in Neural Information Processing Systems*, *35*, 197–211.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fan, Z., Sarkar, R., Jiang, Z., Chen, T., Zou, K., Cheng, Y., … others (2022). M^{3}vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, *35*, 28441–28457.

Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, *23*(120), 1–39.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, *28*.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 16000–16009).

He, Y., & Xiao, L. (2023). Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *46*(5), 2900–2919.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., … others (2022). Lora: Low-rank adaptation of large language models. *ICLR*, *1*(2), 3.

Ilhan, F., Su, G., Tekin, S. F., Huang, T., Hu, S., & Liu, L. (2024). Resource-efficient transformer pruning for finetuning of large models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 16206–16215).

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, *3*(1), 79–87.

Jia, Y., Marsocci, V., Gong, Z., Yang, X., Vergauwen, M., & Nascetti, A. (2025). Can generative geospatial diffusion models excel as discriminative geospatial foundation models? *arXiv preprint arXiv:2503.07890*.

Lacoste, A., Lehmann, N., Rodriguez, P., Sherwin, E., Kerner, H., Lütjens, B., … others (2023). Geo-bench: Toward foundation models for earth monitoring. *Advances in Neural Information Processing Systems*, *36*, 51080–51093.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., … Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Liu, S., Cheng, Y., Zha, F., Guo, W., Sun, L., Bing, Z., & Yang, C. (2024). Learning effective pruning at initialization from iterative pruning. *arXiv preprint arXiv:2408.14757*.

Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., … Li, H. (2024). Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11264–11272).

Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *2009 ieee conference on computer vision and pattern recognition* (pp. 413–420).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211-252. doi: 10.1007/s11263-015-0816-y

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., … others (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wang, L., Gao, H., Zhao, C., Sun, X., & Dai, D. (2024). Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*.

Wightman, R. (2019). *Pytorch image models.* `https://github.com/rwightman/pytorch-image-models`. GitHub. doi: 10.5281/zenodo.4414861

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., … Vajda, P. (2020). *Visual transformers: Token-based image representation and processing for computer vision.*

Wu, X., Huang, S., & Wei, F. (2024). Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Xie, Y., Zhang, Z., Zhou, D., Xie, C., Song, Z., Liu, X., … Xu, A. (2024). Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*.

Yu, Z., Shen, L., Ding, L., Tian, X., Chen, Y., & Tao, D. (2024). Sheared backpropagation for fine-tuning foundation models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 5883–5892).

Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A., & Beyer, L. (2022). Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 18123–18133).

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., … others (2024). A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *International Journal of Machine Learning and Cybernetics*, 1–65.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., … Laudon, J. (n.d.). Mixture-of-experts with expert choice routing, 2022. *URL https://arxiv. org/abs/2202.09368*.