REPORT: Mixture of Experts (MoE) Model for Image Processing

Carlos Garcia

October 21, 2024

1 Introduction

Mixture of Experts (MoE) models have gained significant attention due to their ability to combine multiple specialized models ("experts") to improve the performance of deep learning architectures. Although MoE models are traditionally applied in natural language processing (NLP) [1], this report explores their adaptation to image processing tasks. Specifically, we implement and evaluate a transformer-based MoE model for image classification.

2 Mixture of Experts (MoE) Overview

MoE is a concept where different "experts" are selected dynamically based on the input data. In the MoE architecture, each input is routed to a subset of these experts, typically using a gating mechanism, ensuring that only the most relevant experts are utilized at any given time. This approach not only reduces computational cost, but also increases flexibility and specialization in the model [2].

For this project, the transformer architecture [3] is adapted, where the feed-forward layers in the transformer blocks are replaced with expert layers. The MoE mechanism for image processing introduces unique challenges compared to its application in NLP, particularly in handling the two-dimensional structure of images.

3 Model Implementation

The model is implemented in PyTorch [4], utilizing the MoE design to replace the feed-forward layers in a transformer encoder block with specialized expert layers. We use the MNIST dataset [5], which is widely used to benchmark image classification models. The implementation consists of:

- **Experts**: Each expert is a simple network composed of two linear layers with a GELU activation function [6].
- **Gating Mechanism**: A learned linear layer is used to route input data to a selected subset of experts based on the top-k gating strategy.
- **Transformer Architecture**: A traditional transformer encoder is employed, where MoE layers are integrated. The transformer processes image patches extracted from the MNIST dataset as in the Vision Transformer (ViT) [7].
- **Training Pipeline**: The model is trained using the Adam optimizer and a standard cross-entropy loss function. The training pipeline is tracked using MLFlow [8] and DAGsHub [9].

- **Hyperparameter Tuning**: Key hyperparameters such as the number of experts, top-k selection, patch size, and expert placement are tuned to optimize the model's performance.

4 Training and Experimentation

Given the constraints on computational resources, training time was a critical consideration during the experiments. The process was divided into five key stages:

- **Number of Experts and Top-k Selection**: We evaluated combinations of parameters $num\_experts = [2, 4, 8, 16, 32]$ and $top - k = [2, 4, 8]$. The optimal balance between accuracy and computational cost was achieved with $num\_experts = 4$ and $top - k = 2$ (Figure 1), which served as the base configuration for subsequent experiments.

- **Patch Size**: The impact of patch size on model performance was systematically assessed. Although smaller patch sizes allow the model to capture finer details, they also significantly increase computational costs and may hinder convergence. We tested patch sizes of $patch\_size = [4, 8, 16, 32]$. The best trade-off between accuracy and computational cost was found with a patch size of 16×16 (Figure 2).

- **Number of Layers and Expert Placement**: Initial tests were performed on the number of layers, with values $n\_layers = [2, 3, 4, 5]$. The optimal balance was found with $n\_layers = 3$. Subsequently, we evaluated the placement of experts within the transformer layers, testing $expert\_placement = ['all', [0], [1], [2], [0, 2]]$. The best results were obtained when the experts were placed in the middle of the transformer layers ($expert\_placement = [[2]]$) (Figure 3). Generally, the model performed better when experts were neither too early nor too late in the transformer layers, and not in all layers.

- **Learning Rate**: The learning rate was fine-tuned to identify the optimal value for the model. We tested $learning\_rate = [0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005]$. The best performance was achieved with a learning rate of 0.0001 (Figure 4).

- **Long Run**: A long-term training run of 50 epochs was conducted to observe the model's convergence and assess potential overfitting. The results, shown in Figure 5, indicate that the test accuracy curve initially drops but then recovers rapidly, followed by steady improvement until it plateaus around 99% after 30 epochs. This suggests effective convergence without overfitting, with minimal gains beyond 30 epochs.

5 Results

The results of the experiments are summarized below.

- **Optimal Configuration**: The optimal configuration for the MoE model was found to be:
  - Number of Experts: 4
  - Top-k Selection: 2
  - Patch Size: 16×16
  - Number of Layers: 3
  - Expert Placement: Middle of the transformer layers
  - Learning Rate: 0.0001

- **Accuracy**: The model achieved a test accuracy of **99.33%** after 50 epochs in the long run experiment.
- **Scalability**: The MoE model demonstrated scalability with the ability to handle larger datasets and more complex tasks efficiently.
- **Resource Allocation**: The dynamic selection of experts allowed for efficient resource allocation, balancing computation, and performance even with limited resources (most of the experiments were conducted on Google Colab with no GPU).
- **Convergence**: The model showed effective convergence without overfitting, with minimal gains beyond 30 epochs.

6 Baselines

To evaluate the performance of the MoE model, we compared it with two baseline models: a Convolutional Neural Network (CNN) (See file 'nb_baseline_cnn.ipynb' in the repository) and a standard Transformer model (See file 'nb_baseline_transformer' in the repository). The CNN model consists of multiple convolutional layers followed by fully connected layers, while the standard Transformer model uses self-attention mechanisms to process the image patches.

Figure 6 shows that the MoE model (red) takes longer to converge compared to the CNN (brown), but it eventually matches its accuracy at around 99%. The Transformer model (yellow), while improving early on, fluctuates significantly and underperforms compared to both the CNN and MoE models.

The strength of MoE lies in its scalability; it can handle larger datasets more efficiently by distributing the computational load across multiple experts, making it ideal for complex image processing tasks. As datasets grow, MoE models can provide better resource allocation and adaptivity, leading to better performance with high-dimensional data.

7 Conclusion

The Mixture of Experts model demonstrates promising results for image classification tasks. By dynamically selecting experts, the model can balance computation and performance, making it an efficient choice for resource-constrained scenarios. Future work could explore applying this model to larger and more complex datasets, as well as optimizing the training pipeline for faster convergence and to include more hyperparameters during experiments such as batch size, dropout, and weight decay. Additionally, we could include a load balancing loss term to encourage a more even use of experts and apply further visualization techniques to analyze the behavior of different experts for different types of input.

8 Resources

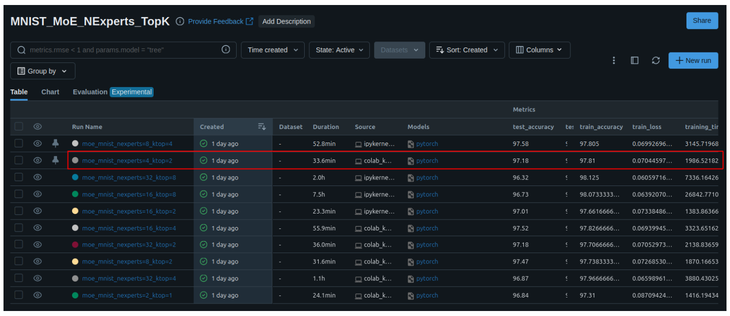
- **Code Repository**: https://github.com/CarlosJesusGH/moe_content_processing/
- **Main Model and training notebook**: https://github.com/CarlosJesusGH/moe_content_processing/blob/main/train/nb_moe.ipynb
- **DAGsHub Project**: https://dagshub.com/s.carlosj.28/moe_image_class.mlflow
- **MNIST Dataset**: https://yann.lecun.com/exdb/mnist/

9 Inspiration

These are some of the materials that inspired the approach taken in this project:

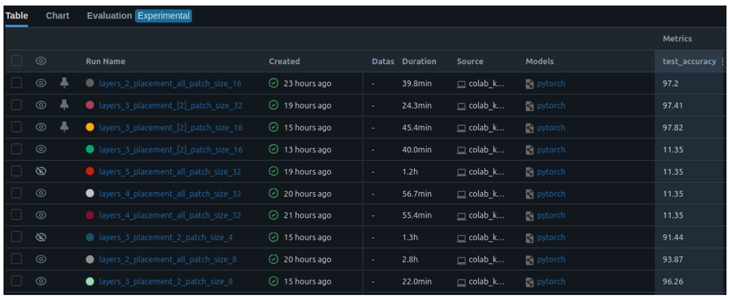
- **Mixture of Experts Explained**: https://huggingface.co/blog/moe
- **makeMoE_from_Scratch**: https://colab.research.google.com/github/AviSoori1x/makeMoE/blob/main/makeMoE_from_Scratch.ipynb
- **makemore**: https://github.com/karpathy/makemore/tree/master
- **Fine-Tune ViT for Image Classification with Transformers** : https://huggingface.co/blog/fine-tune-vit
- **Vision Transformer (ViT)**: https://huggingface.co/docs/transformers/en/model_doc/vit

10 Appendix



						Metrics				
Run Name	Created	Dataset	Duration	Source	Models	test_accuracy	te	train_accuracy	train_loss	training_tir
moe_mnist_nexperts=8_ktop=4	1 day ago	-	52.8min	ipykerne...	pytorch	97.58	[illegible]	97.805	0.06992696...	3145.71968
moe_mnist_nexperts=4_ktop=2	1 day ago	-	33.6min	colab_k...	pytorch	97.18	[illegible]	97.81	0.07044597...	1986.52182
moe_mnist_nexperts=32_ktop=8	1 day ago	-	2.0h	ipykerne...	pytorch	96.32	[illegible]	98.125	0.06059716...	7336.16426
moe_mnist_nexperts=16_ktop=8	1 day ago	-	7.5h	ipykerne...	pytorch	96.73	[illegible]	98.0733333...	0.06392070...	26842.7710
moe_mnist_nexperts=16_ktop=2	1 day ago	-	23.3min	ipykerne...	pytorch	97.01	[illegible]	97.6616666...	0.07338486...	1383.86366
moe_mnist_nexperts=16_ktop=4	1 day ago	-	55.9min	colab_k...	pytorch	97.52	[illegible]	97.8266666...	0.06939945...	3323.65162
moe_mnist_nexperts=32_ktop=2	1 day ago	-	36.0min	colab_k...	pytorch	97.18	[illegible]	97.7066666...	0.07052973...	2138.83659
moe_mnist_nexperts=8_ktop=2	1 day ago	-	31.6min	colab_k...	pytorch	97.47	[illegible]	97.7383333...	0.07268530...	1870.16653
moe_mnist_nexperts=32_ktop=4	1 day ago	-	1.1h	colab_k...	pytorch	96.87	[illegible]	97.9666666...	0.06598961...	3880.43025
moe_mnist_nexperts=2_ktop=1	1 day ago	-	24.1min	colab_k...	pytorch	96.84	[illegible]	97.31	0.08709424...	1416.19434

Figure 1: Accuracy vs. Number of Experts and Top-k Selection



					Metrics
Run Name	Created	Datas	Duration	Source	Models	test_accuracy
layers_2_placement_all_patch_size_16	23 hours ago	-	39.8min	colab_k...	pytorch	97.2
layers_3_placement_[2]_patch_size_32	19 hours ago	-	24.3min	colab_k...	pytorch	97.41
layers_3_placement_[2]_patch_size_16	15 hours ago	-	45.4min	colab_k...	pytorch	97.82
layers_3_placement_[2]_patch_size_16	13 hours ago	-	40.0min	colab_k...	pytorch	11.35
layers_5_placement_all_patch_size_32	19 hours ago	-	1.2h	colab_k...	pytorch	11.35
layers_4_placement_all_patch_size_32	20 hours ago	-	56.7min	colab_k...	pytorch	11.35
layers_4_placement_all_patch_size_32	21 hours ago	-	55.4min	colab_k...	pytorch	11.35
layers_3_placement_2_patch_size_4	15 hours ago	-	1.3h	colab_k...	pytorch	91.44
layers_2_placement_all_patch_size_8	20 hours ago	-	2.8h	colab_k...	pytorch	93.87
layers_3_placement_2_patch_size_8	15 hours ago	-	22.0min	colab_k...	pytorch	96.26

Figure 2: Accuracy vs. Patch Size

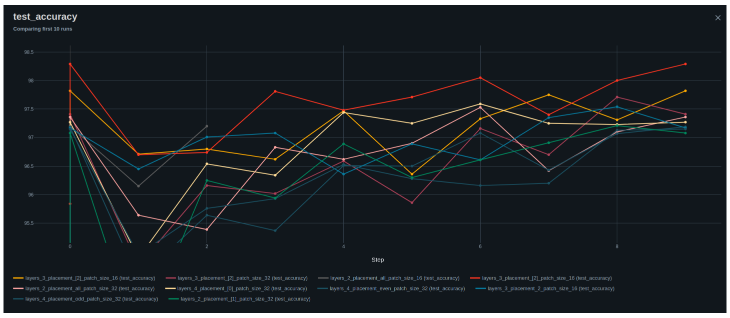


Figure 3: Accuracy vs. Number of Layers and Expert Placement

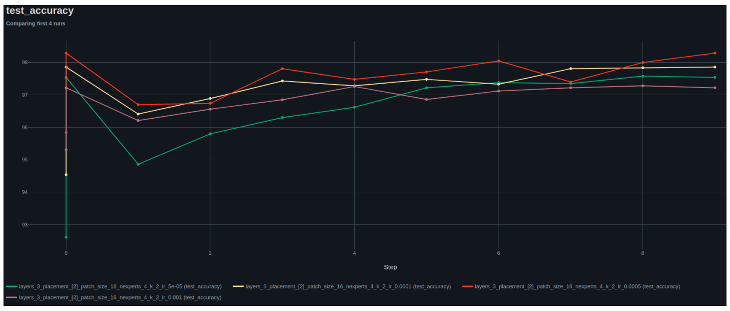


Figure 4: Accuracy vs. Learning Rate

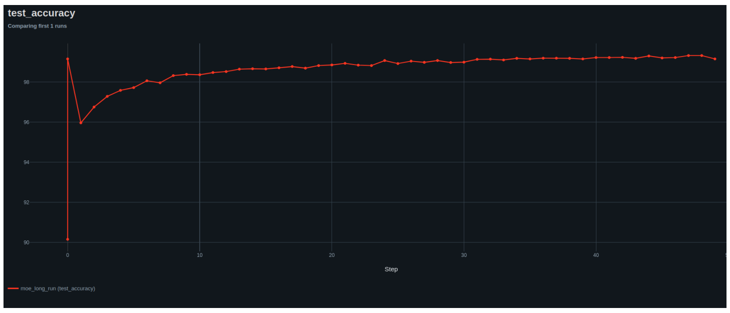


Figure 5: Accuracy vs. Epochs (Long Run)

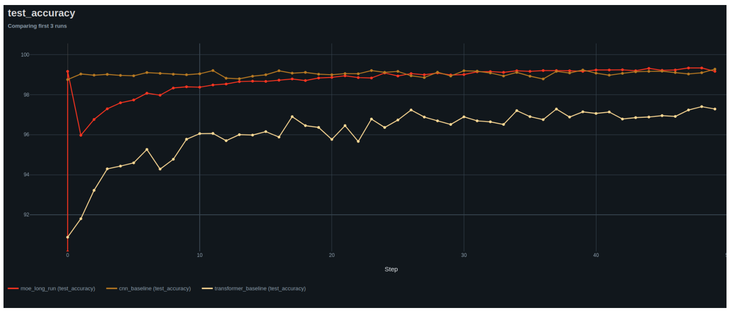


Figure 6: Comparison of MoE Model with CNN and Standard Transformer

11 References

[1] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[2] "Mixture of Experts Explained — huggingface.co." https://huggingface.co/blog/moe. [Accessed 20-10-2024].

[3] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] "PyTorch." https://pytorch.org/. [Accessed 20-10-2024].

[5] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[6] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelu)," *arXiv preprint arXiv:1606.08415*, 2020.

[7] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] "MLFlow." `https://mlflow.org/`. [Accessed 20-10-2024].

[9] "DAGsHub." `https://dagshub.com/`. [Accessed 20-10-2024].