Multi-Headed Lory: Fine-Grained Differentiable Mixture-of-Experts

Michał Maszkowski, Antoni Janowski, Miriam Lipniacka
m.maszkowsk2, m.lipniacka@student.uw.edu.pl, aj421072@students.mimuw.edu.pl
Supervisor: Jan Ludziejewski

July 1, 2024

Abstract

Recently, two different MoE architecture improvements were introduced: (1) Lory (Zhong et al., 2024), which efficiently scales a fully differentiable MoE architecture to autoregressive language modeling pretraining, and (2) MH-MoE (Wu et al., 2024), which employs a multi-head mechanism to split tokens into subtokens that are processed in parallel before being reintegrated, allowing for more fine-grained analytical capabilities. In this paper, we present MH-Lory, the first architecture combining both of these advancements.

Experimental results suggest that our model achieves a slight improvement over the original Lory while maintaining all its benefits. Our work highlights the potential of merging Lory and MH-MoE architectures and encourages further investigation of this issue with more computational resources. Our code is available at https://github.com/MichalMaszkowski/Multi-Headed_Lory.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) introduced a parallelizable method to combine information from distant parts of sequences, allowing for effective language modeling. It was further improved by the Mixture-of-Experts (MoE) approach (Shazeer et al., 2017), which enabled the training of models with significantly more parameters without a substantial increase in training or inference costs. However, due to non-differentiable discrete routing mechanisms, such as top-k expert-activation routing, MoE faces challenges in learning effective routing strategies (Muqeeth et al., 2023). It also suffers from low expert activation and lacks fine-grained capabilities to analyze multiple semantic concepts within individual tokens (Wu et al., 2024).

Recently, several works have attempted to address the non-differentiability of expert choice in the MoE architecture (Puigcerver et al., 2023, Muqeeth et al., 2023). In (Muqeeth

et al., 2023), each token is processed by a "merged expert," whose parameters are a sum of experts' parameters weighted uniquely for every token according to the router. This operation is differentiable but increases computational complexity and memory requirements, scaling linearly with the number of experts, which makes it impractical for autoregressive tasks. This problem is mitigated by Lory (Zhong et al., 2024), which recomputes the "merged expert" only once per segment (comprising a few hundred tokens) instead of for each token, while avoiding information leakage.

A related problem of low expert activation is addressed by Multi-Head Mixture-of-Experts (MH-MoE) (Wu et al., 2024). It also addresses the lack of fine-grained capabilities to analyze multiple semantic concepts within individual tokens. MH-MoE achieves this by splitting tokens into sub-tokens (corresponding to the number of "heads") and then assigning and processing these sub-tokens in parallel with experts before reintegrating them. The naming convention resembles that of multi-head attention, though it is worth noting that the number of heads in these two parts does not need to be related.

In this work, we propose to apply both techniques within a single architecture: combining Lory (Zhong et al., 2024) and Multi-Head Mixture-of-Experts (MH-MoE) (Wu et al., 2024). We introduce a novel MoE architecture, MH-Lory, that is fully differentiable and enables fine-grained analysis of different semantic meanings for each token. MH-Lory is a Lory baseline with the addition of heads, as precisely described in section 3 of this paper.

2 Related work

2.1 MH-MoE

In (Sparse) MoE (Shazeer et al., 2017), which serves as the baseline for the authors of MH-MoE, each MoE block consists of E independent feed-forward networks (FFNs) called Experts (denoted as e_i) and a gating function (router), g, used to calculate weights over these experts' outputs, with k of them (activated according to top-k gated values) processing any token. For any input token's hidden representation $h \in \mathbb{R}^d$, the output of the MoE block is computed as follows:

$$\text{output}(h) = h + \sum_{i=1}^{i=k} e_i(h) \cdot g(h, e_i)$$

MH-MoE uses a multi-head mechanism to split each input token into sub-tokens, then processes them in parallel by diverse experts, and reintegrates them into the original token form. Let us denote the number of heads as H and the token's hidden representation $h \in \mathbb{R}^d$ split into H sub-tokens as $h_1, h_2, ..., h_H \in \mathbb{R}^{\frac{d}{H}}$. Then given p-th sub-token the output of the MH-MoE block is computed as follows:

$$\text{output}(h_p) = h_p + \sum_{i=1}^{i=k} e_i(h_p) \cdot g(h_p, e_i)$$

Additionally, before splitting tokens, they are multiplied by a "multi-head projection" matrix, and similarly, they are multiplied by a "merge" matrix after reintegration.

Thanks to this technique, the average volume of data routed to a specific expert is increased by a factor of H. Moreover, while current tokenization patterns limit a model's ability to capture multiple semantic interpretations of tokens, MH-MoE enables capturing information from diverse feature spaces across these experts (Wu et al., 2024).

2.2 Lory

In Lory architecture (Zhong et al., 2024) authors aim to create a fully differentiable and computationally feasible MoE. To process a sequence of L tokens, with hidden representations $h_1, h_2, \dots h_L$ it is divided into N segments of length $T = \frac{L}{N}$. Each segment will be processed by a single "merged expert" that is created based on the information from the previous segment. Let's suppose that we currently want to start to process segment $k > 1$ (so we want to process tokens $h_k, h_{k+1} \dots, h_{k+T-1}$, while previous segment consisted of tokens $h_{k-T}, h_{k-T+1} \dots, h_{k-1}$). Assuming that a Lory block has E feed-forward expert networks, each parameterised by weights θ_i its output is calculated as follows:

$$\text{output}(h_{k+j}) = (h_{k+j} + \text{FFN}(h_{k+j}, \sum_{i=1}^{E} \theta_i w_i), \text{ where } w_i = \text{Softmax}(g(\frac{1}{T}\sum_{q=0}^{T-1} h_{k-T+q}), e_i)$$

where $\text{FFN}(a, b)$ means an expert (feed-forward networks) with weights b applied to input a and g the gating function (router). For processing the first segment, the authors propose to create an expert matrix by using all tokens from the first segment, and adding a stop gradient operation on top of the router. This approach allows to compute weighted sum of experts only once per segment making it suitable for auto-regressive tasks. It prevents information leakage and keeps the causal nature of the model. At inference the prompt is used to calculate one merged expert used throughout generation.

The authors also used similarity-based data batching technique introduced by (Shi et al., 2023). It ensures that batches contain similar documents, so even when one document ends and other start inside one segment, routing decision made based on the previous document can still be relevant to the next document, which enhances expert specialization in later layers.

3 Our architecture: MH-Lory

Both Lory and MH-MoE report promising gains in performance, while using techniques that we believe can be beneficially combined into MH-Lory.

Combining Lory and MH-MoE architectures admittedly poses a challenge as their strategies aren't orthogonal but diverge slightly. Lory achieves full activation of experts by merging them in pursue of full differentiability. Keeping this feature in the MH-Lory makes one of two advantages of MH-MoE irrelevant as introducing heads can't increase expert activation further. What's more, in Lory whole segments are processed by the same "merged expert" while MH-MoE goes in different direction, routing mere parts of tokens to different experts. Thus Lory decreases the variability of assigned experts between tokens compared to traditional MoE while MH-MoE does the opposite.

The hope is to make an improvement on Lory by reaping the rewards of allowing more fine-grained understanding of multiple semantic concepts within a token as happens in MH-MoE. As an additional benefit by introducing expert heads to Lory architecture we decrease the number of parameters per expert allowing to increase the number of experts. This allows for more diverse routing strategies and might increase expert specialization.

The most natural way to combine Lory and MH-MoE is to use the Lory model as a baseline and modify it by adding heads. More precisely, after the input is passed through the Multi-Head linear layer it's divided into N segments of length $T = \frac{L}{N}$ and then further divide each token into H sub-tokens where H denotes the number of MH-Lory heads. For each segment n_i and for each head h_j, MH-Lory calculates the gating distribution over E experts. This distribution is then used to create a unique merged expert, which is responsible for processing that particular head from the given segment. Finally, the tokens are reassembled into their original shape and passed through the linear Merging layer. Similar to MH-MoE, the Multi-head layer and the Merging layer ensure that information from a token is effectively divided into sub-tokens and then accurately merged back together.

For an input token t_k, assuming that MH-Lory block has E feed-forward expert networks, each parameterised by weights θ_i its output is computed as follows:

$$\text{output}(t_k) = \text{Merge-Layer}(\text{Concatenation}(t_k^1, \dots t_k^H))$$

where

$$t_k^i = \text{FFN}(s_k^i, \sum_{i=1}^{E} \theta_i w_i)$$

and

$$s_k^1, \dots, s_k^H = \text{Divide}(\text{Multi-Head-Layer}(t_k))$$

and

$$w_i = \text{Softmax}(g(A))$$

where A represents the average embedding of the sub-tokens corresponding to a given head from previous segment, g represents the Router gating function and Divide is the operation of splitting a token into H sub tokens along the hidden dimension.

4 Experiments

Our main goal was to compare four MoE Transformer models: Lory, (Sparse) MoE, MH-MoE and MH-Lory, our architecture, in terms of perplexity on a language modelling task with equal number of parameters. We also compare the inference and training efficiency of these models.

Parameter	**MoE**	**MH-MoE**	**Lory**	**MH-Lory**
Transformer blocks	12	12	12	12
Hidden size	256	256	256	256
Number of experts	3	8	3	8
Number of heads	-	4	-	4
Number of segments T	-	-	2	4
Load balancing coefficient α	0.01	0.01	-	-
Capacity factor c	3	3	-	-

Table 1: Models' hyperparameters

4.1 Hyperparameters

The number of heads is specific to the MH-MoE and MH-Lory models, affecting their ability to attend to multiple positions simultaneously. The number of segments T is a unique parameter for the Lory and MH-Lory models, determining the sequence's segmentation granularity. The load balancing coefficient α and capacity factor c play crucial roles in distributing computational load among experts. For MoE and MH-MoE models, the training loss is a sum of a primary cross-entropy loss related to the target task (only this loss is logged) and an auxiliary load balancing loss (Wu et al., 2024). The load balancing loss is computed as follows:

$$L_{balance} = \frac{N}{|\ddot{X}|} \sum_{p=1}^{N} \sum_{x_i^j \in \ddot{X}} t_p \cdot g(f_p^{FFN})$$

where N denotes the number of experts, $|\ddot{X}|$ is the number of sub-tokens contained in $\ddot{X}$. $g(f_p^{FFN})$ is the value of routing a certain sub-token $x_{i,j}$ into the p^{th} expert.

4.2 Experimental Set-up

Constrained by limited computational resources we trained and validated the aforementioned models on the pre-processed "20220301.simple" subset from Wikipedia dataset (Foundation, n.d.). The dataset was partitioned using an 80-10-10 train-validation-test split. Each model was trained for 10 epochs using Cross-Entropy loss and the AdamW optimizer with a learning rate of 0.006, betas set to $(0.9, 0.95)$, and a batch size of 20. In each model we used a BERT tokenizer (Devlin et al., 2018) with a vocabulary size of 30522 tokens. We also used a Rotary Positional Embedding (Su et al., 2024) in each attention layer to capture the positional information in input sequences. For comparison purposes, the number of parameters in each model was standardized to 0.1 billion. The number of active parameters differs between the 4 models. In the Lory model all experts' embeddings are utilized during both the forward and backward passes as they are "merged" into weighted experts. This merging operation increases training times for both the Lory and MH-Lory models.

5 Results

Figure 1 presents the validation losses of our best models, illustrating their performance over the epochs. The plot indicates that all models exhibit comparable performance. Despite our efforts, training a highly effective Lory model proved challenging. Nevertheless, our approach demonstrates behavior akin to sparse MoE and MH-MoE models, outperforming the simple Lory model.

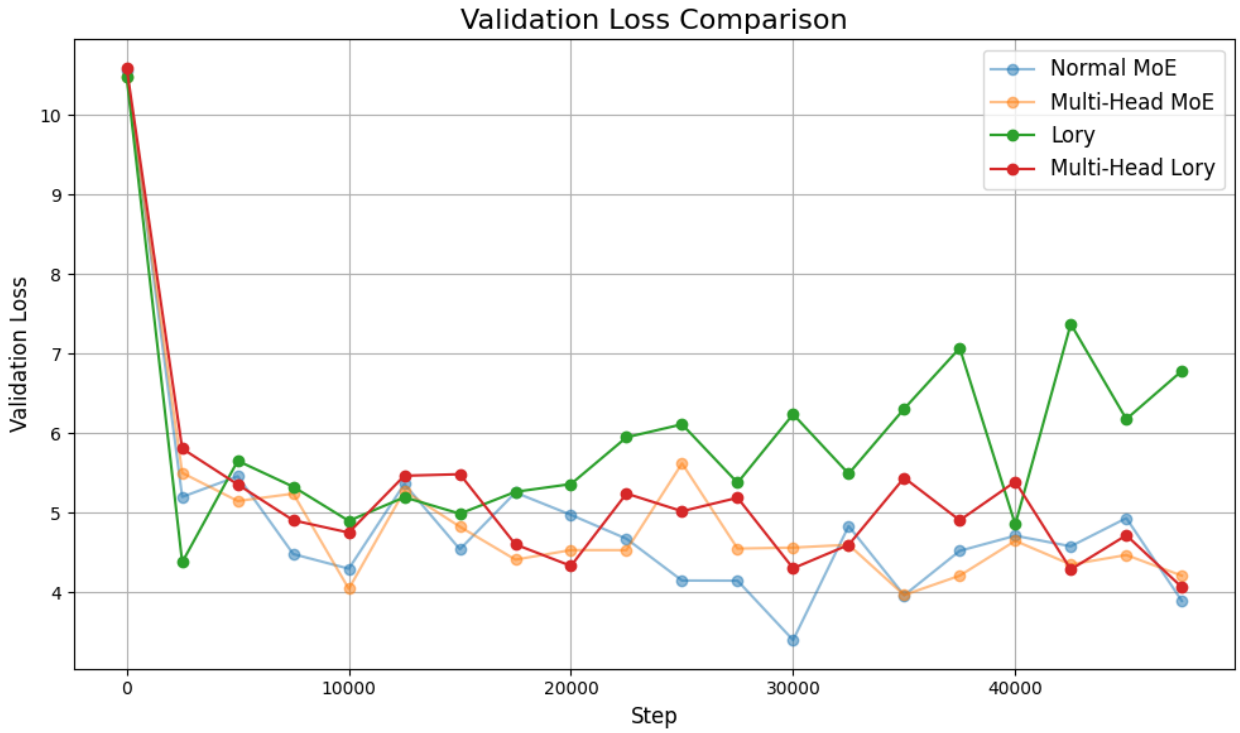

Figure 1: Validation Loss Comparison for Models

The validation and training losses for our best models are illustrated in Figure 2, showcasing their performance over the epochs. The growing discrepancy between the validation and training losses suggests that the models may begin to overfit to the training data. This overfitting is likely due to the relatively small size of the dataset used.

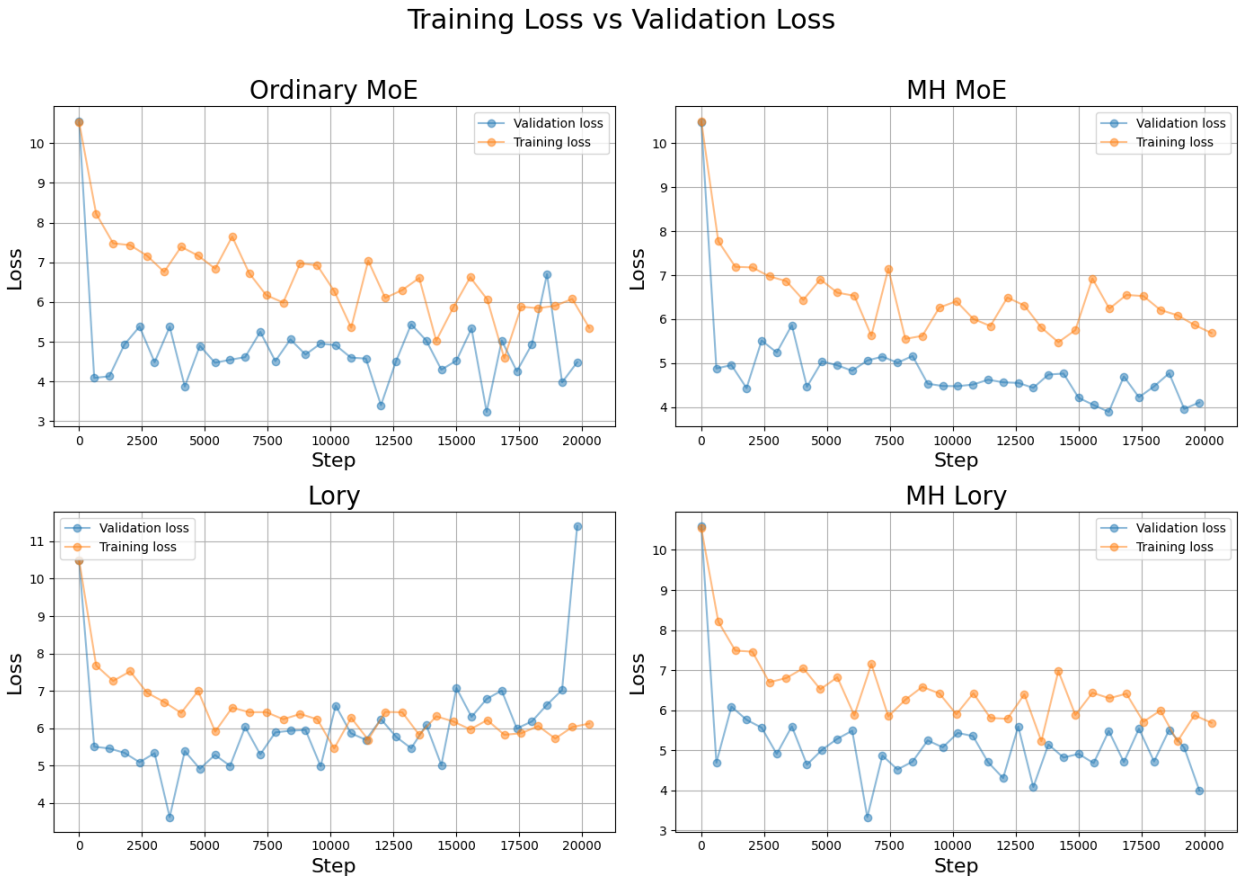

Figure 2: Training and Validation Loss Comparison

6 Discussion

6.1 Limitations

Results of our experiments suggest that when it comes to perplexity MH-Lory achieves slightly better results than original Lory, although due to small size of the used dataset and length of training this isn't a conclusive result as all 4 models achieve comparable perplexity on the dataset we used. Because of our limited resources, we were not able to perform an extensive search for the best model. This uncertainty can be resolved by using much greater computational resources than those at our disposal.

The superior performance of our approach compared to Lori warrants further investigation. However, given the modest size (0.1B) of all models, this is not definitive proof of our model's superiority. Nonetheless, it suggests that our model may be more stable than Lori, which is an improvement.

Nevertheless our results suggest that the model performance doesn't suffer because of the combination of techniques that we propose. This encourages its further investigation as its performance potential may become visible only at greater scales, especially as theoretical considerations suggest that it should yield benefits over original Lory, which is the state of

the art fully-differentiable architecture, as MH-Lory allows for greater number of experts and more fine-grained understanding of multiple semantic concepts within a token as happens in MH-MoE.

6.2 Further research

All of our models were initially trained on a fraction of the Wikipedia dataset, which poses a significant limitation due to the reduced diversity and volume of training data. This can result in sub-optimal performance and a lack of generalization to new data. To address this issue, we plan to re-train the models using the entire Wikipedia dataset, thereby enhancing their robustness and accuracy. Furthermore, we intend to implement the similarity-batching technique introduced by Shi et al. (2023) during pre-processing, as this technique is most effective when applied to the full dataset. By doing so, we anticipate improved model performance through more efficient training and better generalization.

It may also be interesting to experiment with an alternative version (v2) of MH-Lory and compare it with the original (v1). In the alternative one (v2) each head would have a separate set of experts from which a "merged expert" is computed, instead of sharing one set of experts between all heads as in v1. Intuitively MH-Lory-v1 has advantage of increasing the number of examples (sub-tokens) per expert and having less parameters (there would be $N \cdot H$ experts in MH-Lory-v2 compared to N experts in MH-Lory-v1). But on the other hand MH-Lory-v2 could allow for greater specialization of heads, potentially leading each head to focus on a particular aspect of token meaning.

7 Conclusions

We have introduced MH-Lori, a novel integration of two state-of-the-art architectures. Through extensive training and testing, we have shown that our approach performs on par with, and potentially better than, existing alternatives. We leave it for a future research to test how these models scale, and whether they improve if the dataset is pre-processed using similarity-batching technique introduced by (Shi et al., 2023). We hope to pursue these questions further in the future.

References

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Muqeeth, M., Liu, H., & Raffel, C. (2023). Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*.

Puigcerver, J., Riquelme, C., Mustafa, B., & Houlsby, N. (2023). From sparse to soft mixtures of experts. *ArXiv*, *abs/2308.00951*. https://api.semanticscholar.org/CorpusID:260378993

Shi, W., Min, S., Lomeli, M., Zhou, C., Li, M., Lin, V., Smith, N. A., Zettlemoyer, L., Yih, S., & Lewis, M. (2023). In-context pretraining: Language modeling beyond document boundaries. *arXiv preprint arXiv:2310.10638*.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, *568*, 127063.

Wu, X., Huang, S., Wang, W., & Wei, F. (2024). Multi-head mixture-of-experts.

Zhong, Z., Xia, M., Chen, D., & Lewis, M. (2024). Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training.

Foundation, W. (n.d.). *Wikimedia downloads*. https://dumps.wikimedia.org

A Additional Details

The code for data processing, models and their training can be viewed on our github repository under https://github.com/ MichalMaszkowski/Multi-Headed Lory. We included not-vectorized version of MH-MoE, Lory and MH-Lory which we implemented to test our vectorized versions.