More Attention Heads Help With Training Under Large Learning Rates

Yizhou Liu*

Dec 10, 2025

Abstract

Multiple attention heads play an important role in current large language models (LLMs). Yet, the origin of the benefits of many heads is not clear. We explore this question by studying the performance (measured by loss) of decoder-only Transformers on arithmetic tasks as a function of the embedding dimension and the number of attention heads. We found that, at least for this simple task, more heads do not help with the expressivity of the model (one head is strong enough), but do help with the trainability – fixing the embedding dimension (expressivity or representation capacity), more heads lower the head dimension, leading to trainability at a larger learning rate. We anticipate our results may inspire new hyperparameter transfer ideas – scaling query and key matrices with head dimension rather than the total embedding dimension – which may lead to more efficient LLM training.[1]

1 Introduction

Multi-head attention is one of the key innovations behind the success of the Transformer architecture (Vaswani et al., 2017). We want to understand the benefits of multiple attention heads scientifically, with the hope of choosing hyperparameters better or improving the architecture.

Multi-head attention was introduced to allow the model to attend to different parts of the input and different features simultaneously. In the original paper (Vaswani et al., 2017), multiple heads were described to be able to focus on different positions. Each head can attend to a different token or phrase sharply, allowing the model to consider multiple relevant pieces of context at once. Multiple heads also separate the large embedding space into small subspaces. This may implicitly encourage the model to extract different information from the same token. These design ideas suggest that multiple heads help with the expressivity of the model.

Later, both experimental (Michel et al., 2019) and theoretical (Bordelon et al., 2024) findings suggest that different heads may perform similar operations. It is shown that a large percentage of heads could be removed after training without significantly impacting accuracy (Michel et al., 2019), suggesting redundancy. A picture to interpret this fact is that the heads all perform similar tasks and form an ensemble to cancel each other's errors, thereby improving the performance (Bordelon et al., 2024). This mechanism is different from the design idea, yet still suggests that many heads improve model expressivity.

Aside from expressivity, some works suggest that many heads practically improve trainability. One practical reason for using multiple heads is that it's easier to train a wide (multi-head) model

*Group 25, 9.520 Course Project 18. Email: liuyz@mit.edu

[1]Code is available at https://github.com/liuyz0/MultiHeads

than an extremely deep single-head model. A recent empirical comparison (Liu et al., 2021) showed that a Transformer with one head per layer can theoretically achieve the same multi-position attention coverage by stacking many more layers – but this makes the network very deep and hard to train. Multiple heads allow transformers to be shallower and more trainable while still attending to many different aspects of the input.

An important takeaway from the above review is that the design principles of something may not be the real reason for it to work. A profound scientific understanding of the reason why something can work will help us reduce redundancy or design better architectures. We are therefore motivated to explore or verify the benefits of many attention heads ourselves. Note that a higher embedding dimension may also help with expressivity. To decouple the effect of the number of heads and the embedding dimension, we ask the specific question

Question:

Varying the number of heads and the embedding dimension, how will the model performance change, and what are the mechanisms behind?

Limited by the computing resources, we are not able to train large language models (LLMs) on natural language. We choose to train decoder-only Transformers on specific arithmetic problems, which have a small context length, a small vocabulary size, and a clear human understanding of which digit the attention should focus on. The generality of our results is limited. We anticipate our results to provide different insights and hope future studies can test them on other tasks and generalize them.

The rest of the paper is organized as follows. In Section 2, we introduce the model architecture, the dataset, and our training process in detail. In Section 3, we explain our observations and simple explanations. We found that many heads help with training, not expressivity. The trainability is determined by learning rates and head dimension. A smaller head dimension (more head at a fixed embedding dimension) enables good training with larger learning rates. We further connect our results to previous observations in Section 4. At the end, we discuss the limitations, future directions, and implications in Section 5.

2 Methodology

We follow (Bai et al., 2025) to train a two-layer Transformer model with trainable position embeddings from scratch. The embedding dimensions m are 24, 48, 96, 192, and 384. For each embedding dimension, I scan 5 numbers of heads n_H: 1, 2, 3, 6, and 12. So, in total, I have 25 configurations.

Key parameter definitions

We use m for embedding dimension, n_H for the number of heads, $m_H = m/n_H$ for head dimension, and η for learning rate.

The task is three-digit addition from (Chen et al., 2023; Kangaslahti et al., 2025). Each sample has the form

$$\text{Input}: A = a_2a_1a_0 + b_2b_1b_0,\ Ai =?\ \text{Output}: A_i, \tag{1}$$

where a_j and b_k ($j, k \in \{0, 1, 2\}$) are integers from 0 to 9, A_i ($i \in \{0, 1, 2\}$) is the digit with index i

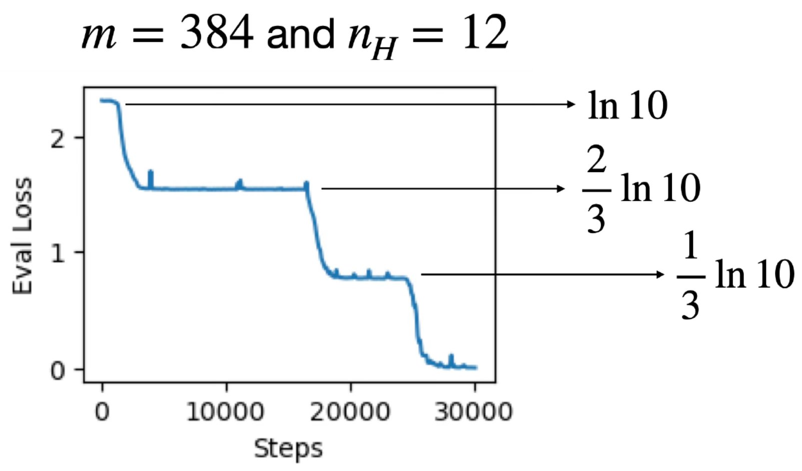

Figure 1: Loss drop is step-wise, all models can learn that the answer is in $\{0, 1, ..., 9\}$.

(0-indexed from the right) of the number A. An example of such samples is

$$\text{Input} : A = 445 + 404,\ A2 =?\ \text{Output} : 8. \tag{2}$$

In the above example, $A = 445 + 404 = 849$ and so $A_0 = 9$, $A_1 = 4$, and $A_2 = 8$.

I first trained the model with next-token prediction on all tokens, which appears to be slow. Then I shift to train the models to only predict the last token. My training set contains 192000 samples, and the test set has 576. I trained the models with a batch size of 64 on the training set for 10 epochs (i.e., 30000 steps). I used the Adam optimizer and tested different learning rates η from `1e-5` to `1e-3`.

3 Results

First of all, we show the loss dynamics of the models (Figure 1). The example in Figure 1 is obtained at embedding dimension $m = 384$, number of heads $n_H = 12$, and learning rate $\eta = 5e - 5$. Yet, this pattern is general for all other configurations. We found that the loss drop is not continuous but step-wise. All the models are trained in a sense that the highest loss is $\ln 10$, which is below the random initialization value ln(vocabulary size). And there is a natural interpretation of $\ln 10$: the model can know the answer must be one number in the 10 numbers $\{0, 1, ..., 9\}$, but is randomly guessing a number. The other two plateaus in loss are close to $\frac{2}{3} \ln 10$ and $\frac{1}{3} \ln 10$, respectively – meaning the model learned addition at 1 digit and 2 digits. After learning all three digits, the model reached a loss close to 0. For models that cannot reach a loss 0, they will stop at $\ln 10$, $\frac{2}{3} \ln 10$, or $\frac{1}{3} \ln 10$.

We next check the effect of the embedding dimension and the number of heads on the final performance of the model. In Figure 2, the color refers to the final test loss. We can see that even with just one head and a minimum embedding dimension, the model can reduce the loss to almost zero. This indicates at least for this task, the models are strong enough. If we fix the number of heads to be 1 but increase the embedding dimension, we saw that the loss increases. This is counterintuitive since a higher embedding dimension should have stronger expressivity and should be able to reduce loss to 0. The reason those models have a large loss is then probably trainability – a higher dimension is hard to train. However, for a large embedding dimension, once we increase

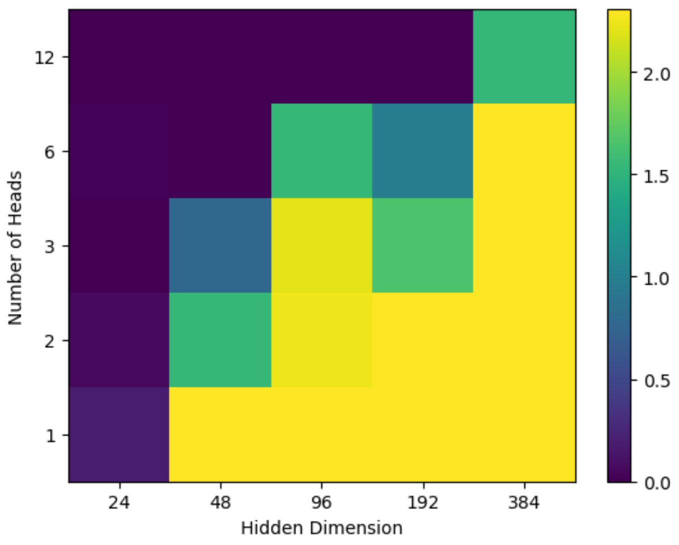

Figure 2: Final test losses (colorbar) as a function of embedding or hidden dimension and number of heads. Obtained when learning rate is `1e-3`. All the training trajectories can be found in Figure 5.

the number of heads, the model can again have zero loss. We therefore claim that, at least for this simple task, more heads merely help with trainability.

We next study the reason more heads can help with trainability, or more generally, what determines trainability. We replot the results from Figure 2 in several ways, trying to find a pattern. If we plot the final loss as a function of the hidden dimension, we can see that larger embedding dimensions tend not to be trainable, yet the transition to be trainable is different for different n_H (Figure 3 left). After plotting the losses as a function of head dimension $m_H = m/n_H$, we found surprisingly that the lines collapse, suggesting that trainability is only a function of head dimension given this training process (Figure 3 right). We therefore conclude that trainability is associated with head dimension.

The question is then what the critical head dimension depends on. Since it is a trainability issue, the learning rate is certainly one factor. In other words, the head dimension and the learning rate need to satisfy some conditions to make the model trainable. We hypothesize that for the loss to decay, the parameters need to squeeze into a narrower region – a smaller learning rate will make the system more trainable. To determine the rigorous relationship between m_H and η, it requires us to analyze the loss landscape, which has not be done. We adopt a simple analysis based on an equivalence mapping between two models to determine the scaling. Say one model with head dimension m_H has the trainability transition happening at learning rate η. Another model with head dimension m'_H needs to reach the same logit value, and each parameter in the query and key matrices should be $\sqrt{m_H/m'_H}$ times of the first model, and the loss landscape is rescaled such that the new learning rate leading to transition should be $\eta' = \eta\sqrt{m_H/m'_H}$. In other words, the trainability transition happens when $\eta\sqrt{m_H}$ is some constant, and being trainable means $\eta\sqrt{m_H}$ is smaller than that constant.

Our results in Figure 3 do not reject this analysis, since we fixed η and observe that a smaller m_H makes the models trainable. But to further test the simple theory, we need to scan learning rates for different models. For one model, when we scan the learning rate, the typical behavior we can see is that the model is not very trainable (guessing the answer from the 10 numbers) when learning rates are large (Figure 4). And the model is trainable (reaching a loss near 0) when the

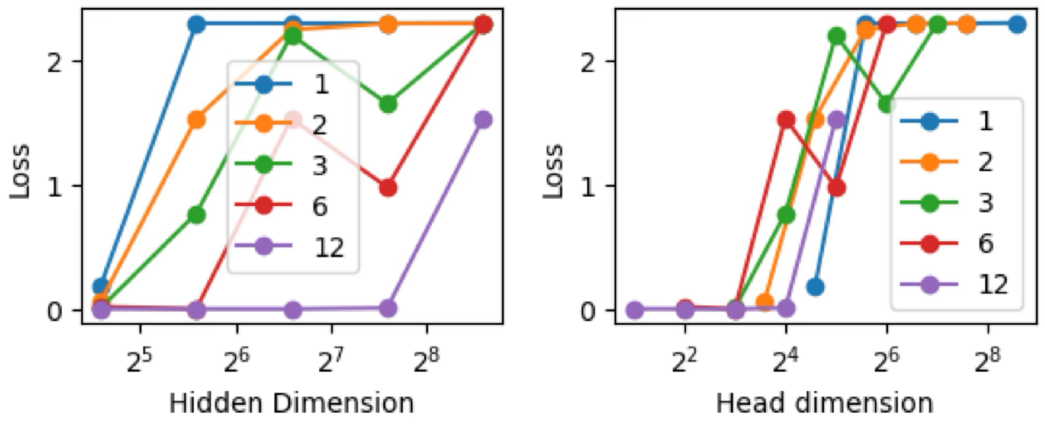

Figure 3: Trainability is associated with head dimension. The color of lines refers to the number of heads n_H. We here simply replot data from Figure 2 in different ways.

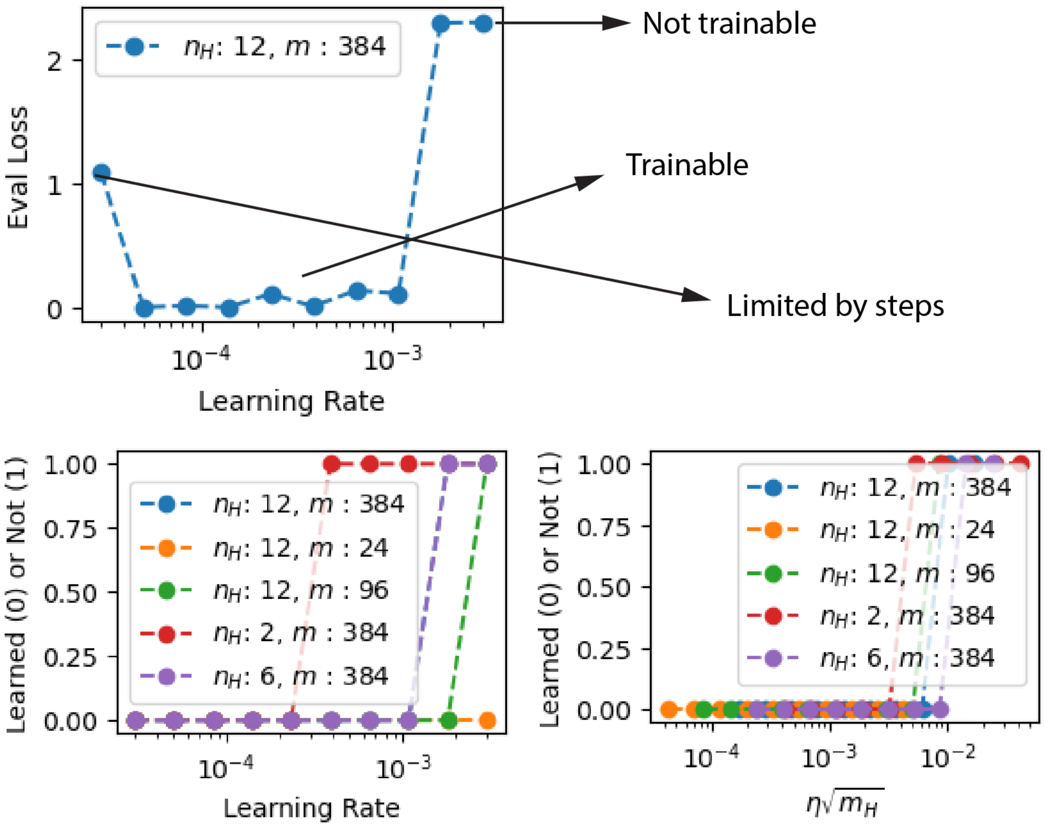

Figure 4: We verified the predicted relationship by varying learning rates. We define learned as 0 and not learned as 1 in the lower panels, where "learned" means the final loss is lower than $\ln 10$ (random guess).

learning rate is lower than some value. But for a too small learning rate, we saw that the loss is non-zero, which is due to finite training steps – the model can reach 0 loss, but has not because the total update is small. For the clarity of visualization, we define learned as 0 and not learned as 1 in Figure 4 lower panels, where learned means the loss is lower than $\ln 10$ (random guess). When we plot different model configurations changing the learning rates, they have trainability transitions at different places. But when we use $\eta\sqrt{m_H}$ as the x-axis, the transitions are relatively closer.

Takeaway

We therefore conclude that the simple prediction "$\eta\sqrt{m_H}$ smaller than some constant leads to trainability" is roughly correct.

4 Related works

As has been explained in Section 1, previous works in explaining the advantage focus on expressivity gain – focusing on more places at the same time (Vaswani et al., 2017) and the head ensemble enables error cancellation (Bordelon et al., 2024), or expressivity-trainability trade-off – if we want to keep the same expressivity with fewer heads, we need to increase the depth, making it difficult to train (Liu et al., 2021). Our work provides a new perspective, at least for simple tasks, that more heads help with purely with trainability, not affecting expressivity, as more heads may reduce the head dimension and enable good training for larger learning rates.

The loss drop we saw is related to saddle-to-saddle dynamics Kunin et al. (2025) and may also be related to the emergence of sparse attention (Zucchet et al., 2025), which are interesting to study, especially if we want to do further theoretical analysis.

This finding may be related to a recent observation that LLMs can use larger learning rates than expected (Haas et al., 2025). The expectation is based on embedding dimension, yet what we found the trainability is more about head dimension, which is kept as a constant 128 in most LLMs.

Another line of work that may be related is about hyperparameter transfer. It seems that people are thinking about changing learning rates based on the embedding dimension and the number of layers, yet not the head dimension (Dey et al., 2025). There might be something interesting and new we should consider. The optimal learning rates swept in experiments in (Liu et al., 2023) indeed decrease more slowly with model size than the theoretical expectation.

5 Discussion

Our work is definitely limited by shallow theoretical analysis. Several claims, like too small learning rates lead to non-zero loss in Figure 4, are not validated due to limited time. And our conclusion can only apply to the simple arithmetic tasks. It remains open how general our insights can be. Naturally, future works can focus on better theoretical analysis, more detailed experiments for validation of many claims here, mechanistic interpretation of the hidden states and attention, and other more difficult tasks.

Our biggest implication is about training LLMs and hyperparameter transfer. If we are correct, we should scale the learning rate of the query and key matrices based on the head dimension. This is different from current practice, as far as I know, where people scale all non-embedding matrices based on the embedding dimension. Again, if our result is correct and can be generalized to broad natural language tasks, we expect the new insights can make LLM training faster.

References

Bai, X., Pres, I., Deng, Y., Tan, C., Shieber, S., Viégas, F., Wattenberg, M., and Lee, A. Why can't transformers learn multiplication? reverse-engineering reveals long-range dependency pitfalls. *arXiv preprint arXiv:2510.00184*, 2025.

Bordelon, B., Chaudhry, H., and Pehlevan, C. Infinite limits of multi-head transformer dynamics. *Advances in Neural Information Processing Systems*, 37:35824–35878, 2024.

Chen, M., Roberts, N., Bhatia, K., Wang, J., Zhang, C., Sala, F., and Ré, C. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36:36000–36040, 2023.

Dey, N., Zhang, B. C., Noci, L., Li, M., Bordelon, B., Bergsma, S., Pehlevan, C., Hanin, B., and Hestness, J. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

Haas, M., Bordt, S., von Luxburg, U., and Vankadara, L. C. On the surprising effectiveness of large learning rates under standard width scaling. *arXiv preprint arXiv:2505.22491*, 2025.

Kangaslahti, S., Rosenfeld, E., and Saphra, N. Hidden breakthroughs in language model training. *arXiv preprint arXiv:2506.15872*, 2025.

Kunin, D., Marchetti, G. L., Chen, F., Karkada, D., Simon, J. B., DeWeese, M. R., Ganguli, S., and Miolane, N. Alternating gradient flows: A theory of feature learning in two-layer neural networks. *arXiv preprint arXiv:2506.06489*, 2025.

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

Liu, L., Liu, J., and Han, J. Multi-head or single-head? an empirical comparison for transformer training. *arXiv preprint arXiv:2106.09650*, 2021.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1905.10650.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.03762.

Zucchet, N., d'Angelo, F., Lampinen, A. K., and Chan, S. C. The emergence of sparse attention: impact of data distribution and benefits of repetition. *arXiv preprint arXiv:2505.17863*, 2025.

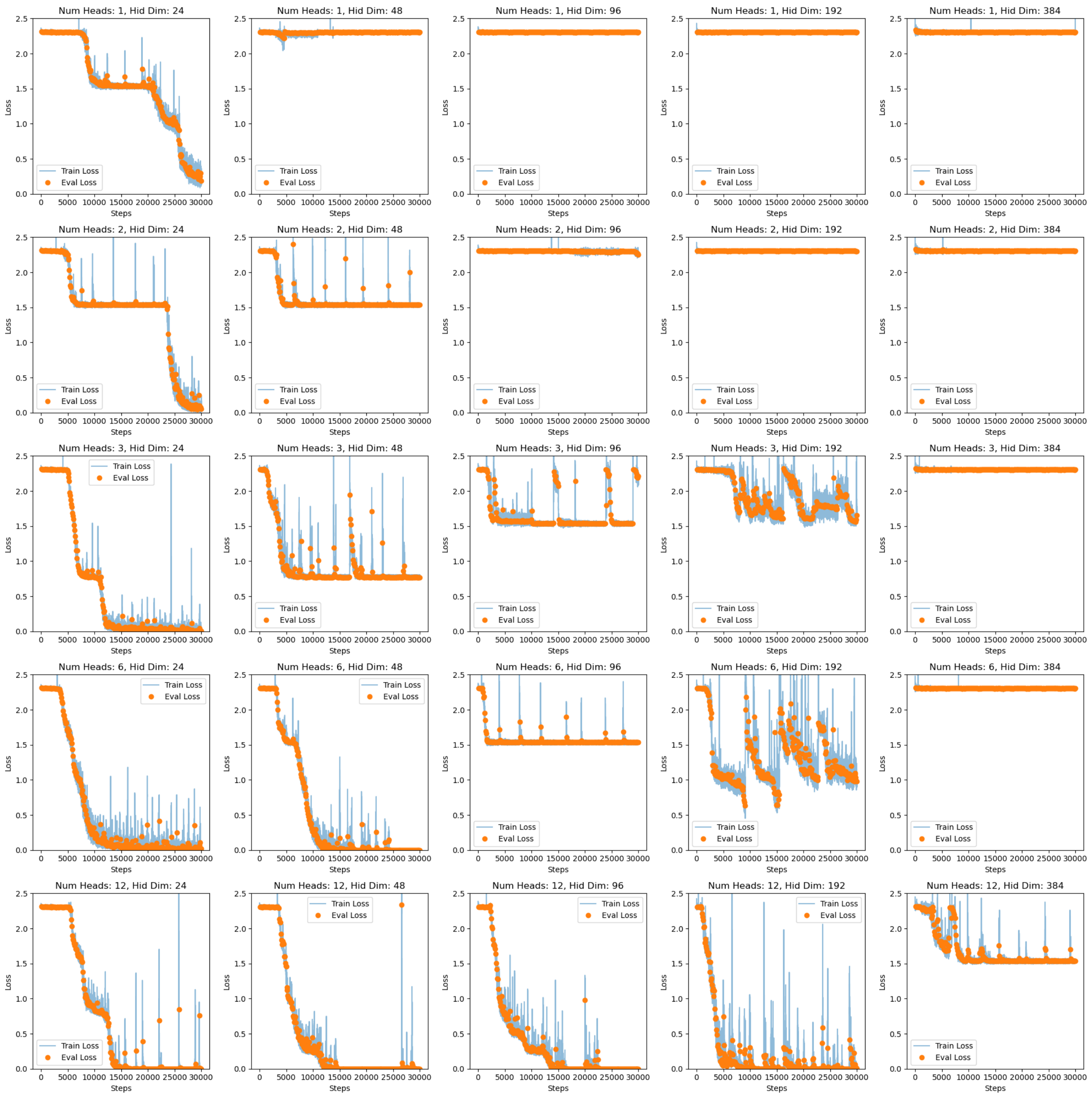

Figure 5: Training and test losses during training. Obtained when learning rate is `1e-3`.