Mixture of Tokens: Continuous MoE through Cross-Example Aggregation

Szymon Antoniak [*]

Michał Krutul [*]
IDEAS NCBR
University of Warsaw

Maciej Pióro
IDEAS NCBR
Polish Academy of Sciences

Jakub Krajewski
IDEAS NCBR
University of Warsaw

Jan Ludziejewski
IDEAS NCBR
University of Warsaw

Kamil Ciebiera
IDEAS NCBR
University of Warsaw

Krystian Król
IDEAS NCBR
University of Warsaw

Tomasz Odrzygóźdź
IDEAS NCBR

Marek Cygan
University of Warsaw
Nomagic

Sebastian Jaszczur [*]
IDEAS NCBR
University of Warsaw

Abstract

Mixture of Experts (MoE) models based on Transformer architecture are pushing the boundaries of language and vision tasks. The allure of these models lies in their ability to substantially increase the parameter count without a corresponding increase in FLOPs. Most widely adopted MoE models are discontinuous with respect to their parameters - often referred to as *sparse*. At the same time, existing continuous MoE designs either lag behind their sparse counterparts or are incompatible with autoregressive decoding. Motivated by the observation that the adaptation of fully continuous methods has been an overarching trend in Deep Learning, we develop Mixture of Tokens (MoT), a simple, continuous architecture that is capable of scaling the number of parameters similarly to sparse MoE models. Unlike conventional methods, MoT assigns mixtures of tokens from different examples to each expert. This architecture is fully compatible with autoregressive training and generation. Our best models not only achieve a $3\times$ increase in training speed over dense Transformer models in language pretraining but also match the performance of state-of-the-art MoE architectures. Additionally, a close connection between MoT and MoE is demonstrated through a novel technique we call *transition tuning*.

1 Introduction

Transformer-based Large Language Models (LLMs) make up one of the most active fields in AI, exhibiting human-level performance across a variety of tasks, including translation, language understanding, reasoning, and code generation [1, 2, 3]. The exorbitant sizes of all state-of-the-art language models are integral to their success, with parameter counts reaching tens or even hundreds of billions. This phenomenon aligns with findings from [4, 5], where the latter suggests that the optimal model size grows proportionally to the available computational budget. Given that hardware efficiency has been steadily increasing over the past decade [6, 7], these findings imply that scaling will continue to be a vital component in training increasingly capable models.

[*]Equal contribution. Work done while at IDEAS NCBR and University of Warsaw.
Detailed authors' contributions are listed in Appendix F.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

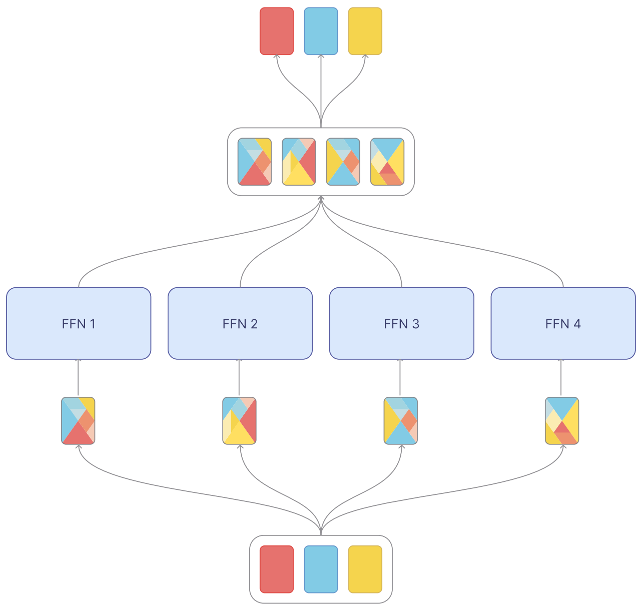

Figure 1: Mixture of Tokens: Each expert receives a unique mixture of tokens in the group. Mixing weights are determined by the controller, which is a fully connected layer (omitted for clarity). For a given token, its update is a linear combination of expert outputs, with the coefficients equal to the token's original mixing weights for each expert.

However, model scaling invariably comes at a cost. Larger models execute more Floating Point Operations (FLOPs) per token, resulting in both training and inference becoming slower and more expensive [8, 9]. Mixture of Experts [10] architectures offer an attractive alternative to standard Transformers by drastically increasing the number of parameters. The core idea is to have multiple *experts*, each specializing in a different part of the input space. Currently, state-of-the-art models based on MoE leverage sparsity by activating only a fraction of the parameters for each token [11]. This allows the networks to increase the number of parameters by an order of magnitude while keeping the FLOPs per token roughly constant. In this work, we use MoE to signify sparse Mixture of Experts architectures unless explicitly stated otherwise.

The aforementioned sparsity is made possible with a *router*, a small network that selects the best experts for each token. This makes the output of an MoE layer discontinuous with respect to its parameters, as only a subset of the experts is chosen for each token (this is typically done with a discrete *top-k* operation). The discontinuity and the resulting fluctuations of the router's decisions have been shown to hurt training efficiency [12, 13] and are hypothesized to be a source of training instability in large MoE models [14, 15]. Conversely, existing continuous MoE architectures involve trade-offs, including the inability to scale [16, 17], or incompatibility with autoregressive decoding [15].

This paper introduces Mixture of Tokens, a novel, continuous Transformer architecture closely related to sparse Mixture of Experts. Similar to MoE, it can support large parameter counts without significant costs in FLOPs. The core idea behind our design is for each expert to process not individual tokens separately, but their combined representation.

This technique results in a continuous model that avoids the top-k operation. It requires no additional techniques commonly required in existing MoE designs (both sparse and continuous), such as load balancing losses, calculating solutions to optimization problems, or non-homogeneous training schedules [17, 18, 12]. It is capable of scaling the parameter counts akin to sparse MoEs and is compatible with autoregressive language modeling and generation. Our analysis demonstrates a $3\times$ speedup over a dense baseline and improved stability over MoE.

In summary, our contributions are the following:

- Introducing the novel Mixture of Tokens (MoT), a continuous Mixture of Experts architecture that mixes tokens from different examples for joint processing.
- An analysis of scaling properties of Mixture of Experts models on multiple scales.
- Introducing transition tuning, allowing a pretrained MoT model to be tuned for sparse MoE inference if desired.

2 Background and Related Work

In this section, we provide an overview of the approaches related to our work and discuss the differences between various MoE designs. We will introduce the Mixture of Tokens architecture in Section 3 and provide a detailed comparison between MoT and related methods in Section 3.4.

2.1 Large Language Models

Transformer scaling has been shown to be a critical factor in achieving state-of-the-art results in language and vision tasks [4, 19], with the largest disclosed parameter counts in dense models reaching hundreds of billions of parameters [20, 3, 2]. These large models exhibit impressive abilities not present in their smaller counterparts [21]. [4] and [5] have demonstrated that the final model performance is predictable and correlates directly with the model size and the amount of training data. However, increasing model sizes raises the demand for computational resources during both training and inference [22].

2.2 Mixture of Experts

Mixture of Experts (MoE) was first introduced by [10, 23] as an ensemble-like neural network comprised of separate sub-networks called experts. The original design used a gating network to select a soft assignment of experts for each input. In the context of Deep Learning, the notion of an MoE *layer* was introduced in [24]. [25] combined a sparse version of MoE layers with an LSTM to train a model with over 100 billion parameters, which was unprecedented at the time. The design, similar to state-of-the-art MoE models today, used a small routing network to decide the top-k best experts for each input. By choosing only a subset of the experts, they were able to increase the size of the network while keeping FLOPs per token roughly constant. The Transformer was first combined with MoE layers in [26], where it replaced the Feed-Forward layer. The design was further simplified in [27], which trained a model with 1.6 trillion parameters using top-1 routing. Since then, a number of studies have investigated different sparse MoE designs [28, 29, 30, 31, 32]. A comprehensive analysis of scaling properties of sparse MoE architectures can be found in [33]. [34] introduced the notion of granularity, which in spirit is similar to the number of groups in MoT, described in Section 3.2.

2.3 Continuous Mixture of Experts

Continuous architectures serve an important role within the field due to their flexible and efficient nature. [17] were pioneers in introducing them in MoE by presenting continuous techniques for calculating encodings of the choice of an expert. In another approach, [16] proposed a method in which they merge experts based on the weights of the router network. In a recent advancement, [15] proposed a continuous variant of MoE for the Vision Transformer, where patches are mixed only within each image.

2.4 From Hard to Soft Methods

From the very beginning of the Deep Learning field, there has been a shift from discrete functions toward continuous ones. The first perceptron [35] used "all-or-none" activation, supposedly to align with propositional logic. This was later improved with soft activation functions, enabling gradient descent and multi-layer neural networks. Similarly, soft attention, introduced in [36], enabled RNNs to look at arbitrary input from the past while maintaining the ability to learn the selection with standard gradient descent. This is in contrast to hard attention, which requires, e.g., reinforcement

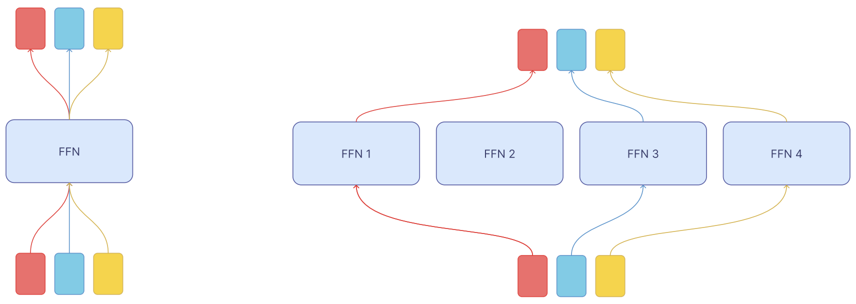

Figure 2: **(Left).** Diagram of a standard Feed-Forward layer featured in the Transformer architecture: each token is processed with the same MLP independently of other tokens. **(Right).** Diagram of a Token Choice layer, where each token decides which expert to choose. In this way, different experts process a different number of tokens. If one expert is chosen by too many tokens, a portion of the tokens is dropped — they receive no update.

learning techniques. While hard attention could perform on par with soft attention [37, 38], soft attention, with its simplicity of training, offered better trade-offs and was later used as the basic building block of the Transformer [39].

Mixture of Experts, introduced into Deep Learning by [10, 23, 25], appears to be inherently a discrete function—after all, the expert either processes a given token or it does not. However, similar to the transition from hard to soft attention, an expert in MoE can "attend" to a combination of tokens, taken as a weighted average. This results in a smooth, continuous model and facilitates more stable training.

3 Mixture of Tokens

The goal of this work is to develop an efficient, continuous architecture that retains the scalability of Mixture of Experts, while simultaneously omitting the top-k operation, which limits a token's exposure to different experts. An intuitive approach to achieving this is to route all tokens to all experts, but this is computationally infeasible for large-scale pretraining. To overcome this constraint, the method explored in this work considers what happens not to an individual token but to a whole group of tokens instead. The main contribution of this work is the observation that allowing an expert to dynamically produce a continuous representation of the entire group of tokens, which is more lightweight to process than each token individually, yields positive results.

Algorithm 1 Mixture of Tokens layer

```
1: for each $E$ in experts do:
2:     weights$_E$ = Softmax(Linear(tokens))
3:     mix $= \sum_i$ token$_i$ $*$ weights$_{i,E}$
4:     output$_E$ $= E$(mix)
5: for each i do
6:     for each E do
7:         update$_i$ $= \sum_E$ output$_E$ $*$ weights$_{i,E}$
```

More specifically, in our design, an input batch is divided into groups of tokens, and each group is processed independently. Given a group and a single expert, a scalar weight is produced for each token. The weights are then normalized and used to compute a linear combination of the tokens, which is used as the expert's input. The experts' outputs are used for token updates as follows: for each input token, its update is a linear combination of expert outputs, with the token's mixing weights for each expert as coefficients[2]. A diagram of our method is presented in Figure 1.

[2]The authors note that an MoT layer admits an efficient vectorized implementation, where all meaningful computations are done with batched matrix multiplications.

To see why this method is scalable, it is helpful to examine the relationship between the number of tokens in a group and the number of experts. Essentially, if these two quantities are equal, the total computation performed by the experts is the same as in the case of top-1 routing. This allows MoT to benefit from the same parameter scaling as seen in MoE, which we confirm empirically in Section 4.2.

3.1 Intuition Behind Our Method

As we mix tokens from multiple unrelated sequences, we do not expect the model to meaningfully use the information from one sequence to improve prediction in a different sequence. However, we hypothesize that such mixing (1) provides richer feedback (gradients) to train the model, especially the router, and (2) results in a smoother loss landscape, which is resistant to small perturbations in inputs and weights.

Intuitively, for a given expert, from the perspective of each token, the token receives a certain amount of update to its representation (in the residual stream) based on:

- Itself, producing a proper signal expected to improve the token representation.
- Tokens other than itself, which are essentially random tokens from unrelated sequences. As these sequences are randomly sampled from the dataset, the impact from these tokens will point in random directions and, essentially, just add some amount of noise to the token update.

We generally expect neural networks to be resistant to a certain amount of noise added to them. Moreover, while the signal-to-noise ratio worsens for tokens with low expert weight, the expert weight also modulates the magnitude of the update. Therefore, the amount of noise added to the representation is limited.

We stipulate that MoT experts will learn to focus on a single token or a small number of tokens, thereby minimizing noise and approximating sparse MoE when optimal. However, other tokens will be assigned nonzero weight, allowing some information to flow to the router for each and every token-expert pair, unlike sparse MoE. Additionally, the output of MoT is more continuous, with small perturbations of input/weights corresponding to small changes in the output rather than large discrete jumps, as may occur in the case of sparse MoE.

3.2 More Mixtures per Expert

Building on the design described above, we experiment with feeding more than one mixture into each expert. Without further modifications, this approach would result in a linear increase in computational costs for each additional mixture processed. To circumvent this added cost, MoT uses more experts, but each expert has a proportionally reduced hidden dimension. This way, each mixture is processed by a smaller expert, and the layer's total number of parameters, as well as the number of FLOPs used by all experts, remains approximately the same. We find that this design consistently improves MoT as the number of processed mixtures increases, just as it does in sparse MoE [34]. Likewise, the optimal granularity aligns roughly with the number of mixtures in this work.

3.3 Token Groups in Mixture of Tokens

The question of how token *groups* are decided within a batch is crucial for compatibility with autoregressive training and inference. The main insight here is that tokens from the same sequence cannot be placed in a single group, as the mixing operation would result in an information leak. Due to this restriction, MoT groups tokens from different examples based on their position in the sequence. Thus, all tokens within a group share the same position in their respective sequences. As previously mentioned, to maintain a constant number of FLOPs per token, an increase in the number of experts means an equal increase in group size. An illustration of how grouping is done within a batch of tokens is shown in Figure 3.

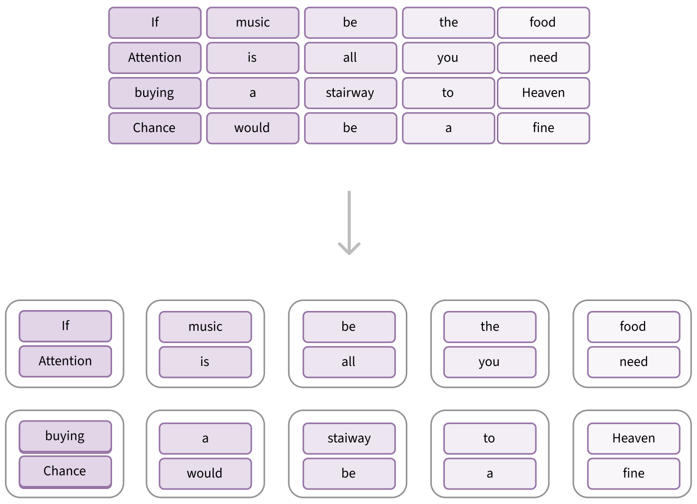

Figure 3: Each group consists of tokens with the same position in a sequence. In this example, the group size is 2. Note that the maximum possible group size equals the size of the batch.

3.4 Comparison with Other Mixture of Experts Architectures

Scaling The technique featured in [17] is based on a continuously differentiable sparse top-k router, which is a major advantage compared to the common top-k gating. However, this approach requires that all experts are utilized in a portion of training, rendering it computationally prohibitive for models with large numbers of experts. The architecture based on merging experts proposed in [16] also offers an attractive, continuous alternative to top-k gating, yet the cost of merging all experts again scales linearly with the number of experts. To address this, the technique is applied once per sequence, which limits the expressive power of the final model.

Training stability [26] reported instabilities during the training of large MoE models, stemming from the inaccuracies when calculating router weights in low precision. To stabilize the training, they resorted to using full precision. [27] made progress in using mixed precision when training MoE by using selective high precision for gating. When comparing [26, 27] to MoT, an advantage of our technique emerges - it is more robust to training in lower precision than other methods. We conjecture, that this is due to the merging mechanism being less susceptible to rounding errors than gating in sparse MoEs.

Token dropping Token dropping is a phenomenon where tokens do not receive an update from any expert. This can happen when the expert was selected by too many tokens in a batch [27, 26, 40] or, in the case of routing experts to tokens, when a token is not selected by any expert [30]. Existing techniques to combat this phenomenon offer a partial solution, yet the problem persists. In contrast, tokens in MoT are part of every mixture produced within their group; hence, they always receive an update.

Auto-regressive decoding Mixture of Tokens is based on the concept of merging tokens before they are processed by an expert. An encoder-only design of a similar nature is featured in concurrent work [15]. The technique is based on merging patches within an image for vision models, i.e., within a single sample. This should be contrasted with MoT, which merges tokens from different sequences within a batch. This crucial difference allows MoT to be compatible with autoregressive training and inference.

Time complexity The time complexity of our approach is identical to that of the Token Choice and Expert Choice methods. In all cases, the cost of computing routing logits is of order $O(d_{model} \cdot N_{experts} \cdot N_{tokens})$.

4 Experiments

The focus of this work is to investigate the efficiency of Tokens on autoregressive language modeling. To measure model quality, we pretrain models for a fixed number of tokens and compare final perplexity in accordance with existing MoE literature [28, 27]. In all experiments, the models are trained on the C4 dataset[3] [41] and use the GPT-2 tokenizer. Unless specified otherwise, we use mixed precision, where all heavy computation is done in bfloat16, whereas the optimizer state and weights are kept in full precision. To study the stability of our model, we experiment with training fully in reduced precision.

Our main result is a substantial speed-up of MoT models compared to dense Transformers (Figure 7) and results comparable to sparse MoEs (Figure 6). What follows is the analysis of the scaling properties of the MoT architecture with respect to the number of parameters (Figure 4) and the number of mixtures sent to each expert (Figure 5). We investigate the model's performance in low precision in order to simulate training instability and find that MoT is less susceptible to instabilities arising from low-precision training. Lastly, we show the connection between MoT and MoE, by spending an additional fraction of pretraining compute to effectively transform a MoT model into a Token Choice model (Section 4.4).

4.1 Model Architecture

The base of our experiments is a decoder-only Transformer based on GPT-2 [42]. We conduct experiments on two model scales: a 77M Medium model and a 162M Base model (refer to Appendix A for hyperparameters and training details). To obtain a Mixture of Tokens model, we replace the second half of the Feed-Forward layers in the Transformer with MoT layers. Because, similar to MoE models, the FLOPs and parameter counts in MoT are decoupled, we indicate the model architecture by its dense counterpart in terms of the number of FLOPs and, separately, the number of experts (or equivalently, group size). To this end, a MoT-Medium/32E model is one that uses the same number of FLOPs as a Medium (77M) Transformer model but uses 32 experts in MoT layers.

As outlined in Section 3.2, Medium/32E/4 signifies a model employing MoT layers with $32 \cdot 4$ small experts, which add up to the same number of parameters as 32 normal experts.

In addition to using the Transformer as a baseline, we also compare against Token Choice [27] and Expert Choice [30] as sparse MoE baselines. Given that Expert Choice is sensitive to the size of the batch, in order to avoid discrepancy between training and inference, we group tokens prior to routing in training Expert Choice models.

4.2 Scaling Results

Mixture of Tokens models demonstrate strong scaling properties with respect to the number of parameters. As seen in Figure 4, increasing the number of experts in MoT layers while using the same compute budget yields consistent improvements. All MoT models are a strict improvement over the Transformer. The figure also includes an ablation experiment, where the mixing weights are fixed to $1/n$, where n is the group size. This corresponds to a uniform mixing strategy; the performance of that model clearly suffers, confirming that MoT layers learn non-trivial mixing strategies.

The increased number of token mixtures described in Section 3.2 represents another axis of scaling for MoT models, once again exhibiting consistent improvements. We hypothesize that this phenomenon is due to two mechanisms: First, the model becomes more expressive with a larger number of smaller experts. Second, the model can allocate its focus (the mixing weights) more flexibly to more important tokens while reducing the updates for trivial ones.

4.3 Comparison with the Transformer and Sparse MoEs

Crucially, the performance of Mixture of Tokens is comparable to that of the strong Mixture of Experts baselines (Figure 6). An increased number of mixtures allows it to compete with both Expert Choice and Token Choice architectures. As the sparse routing is hypothesized to contribute to training instabilities in large sparse models, Mixture of Tokens, being continuous, presents a promising

[3]https://huggingface.co/datasets/c4, dataset licensed under ODC-By.

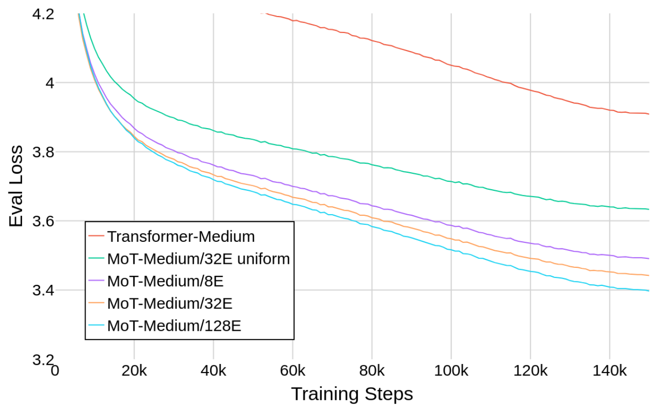

Figure 4: Scaling with respect to the number of parameters. Also featured are the Transformer baseline and an MoT model with a non-learnable, uniform routing strategy.

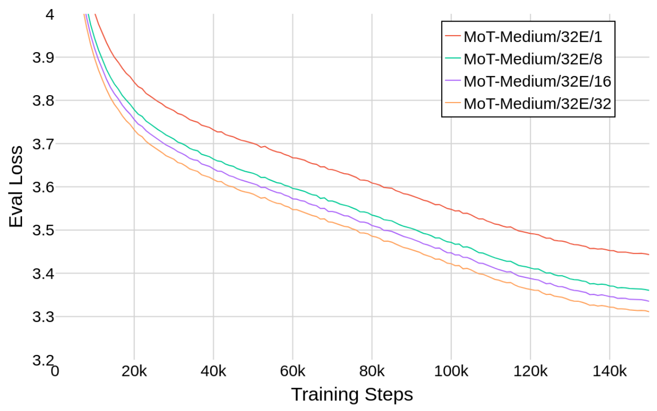

Figure 5: Scaling with respect to the number of token mixtures.

alternative. To investigate training instabilities at the scale we experiment on, we trained models entirely in bfloat16, as opposed to the mixed precision used in all other experiments. The results confirm that MoT is more resistant to lower precision training: as the precision of training decreases, the performance of Expert Choice drops below that of Mixture of Tokens, despite the former attaining better perplexity using mixed precision. We find this to be evidence of the architecture's potential for stable training at higher model scales. See Table 1 for details.

Finally, we combine our findings on MoT scaling properties to train our most efficient MoT model and compare it to the Transformer baseline (Figure 7). The result is a model that achieves the final loss of the baseline in one-third of the training steps. This represents a 3× improvement in terms of the compute budget.

Table 1: Comparison of training result loss comparison. MoT performs better in the bfloat16-only setting. Learning rates were separately tuned in lower precision for both EC and MoT. Results are averaged over 3 random seeds.

	MoT-Medium/32E	Expert Choice-Medium/32E
Mixed Precision	3.442 ($\pm$ 0.002)	**3.420** ($\pm$ **0.002**)
bf16 only	**3.661** ($\pm$ **0.007**)	3.728 ($\pm$ 0.044)

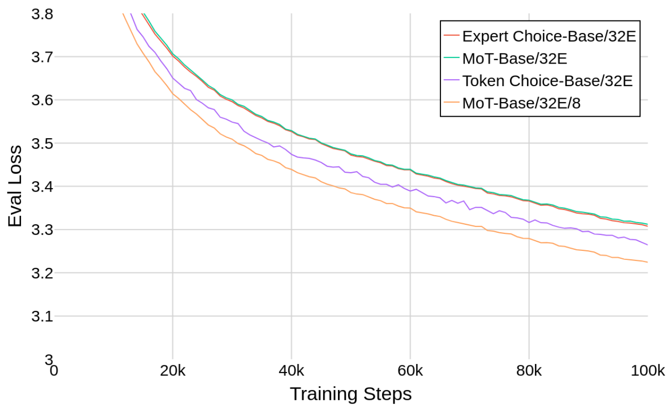

Figure 6: Comparison of MoT and sMoE architectures. An increased number of smaller experts allows MoT to match the performance of the best sMoE model. Due to computational constraints, the models were trained for 100K steps.

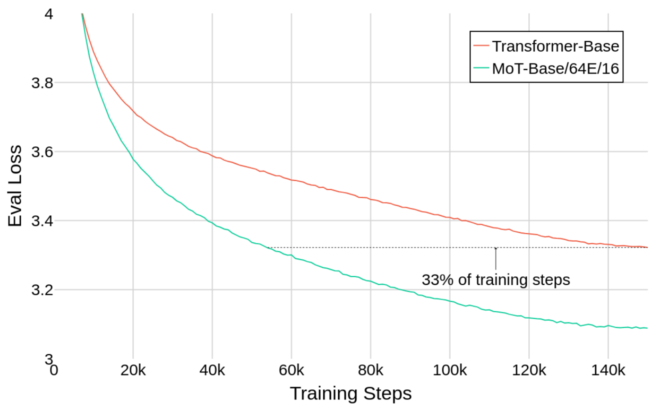

Figure 7: Our best MoT model reaches the final loss of the baseline in just 33% of the compute budget.

4.4 Transition Tuning

Mixture of Tokens suffers from a drawback common to MoEs, namely, it does not support unbatched inference. This is a direct consequence of its design - in the forward pass, it groups several tokens from different examples in the batch. With the growing adoption of Large Language Models on consumer hardware [43, 44], this lack of support could hinder the architecture's wider adoption. While a Mixture of Tokens with a group size of one is technically possible, in order to keep FLOPs constant, the layer would need to trivially reduce to a standard Transformer MLP.

To address this issue, we demonstrate that the weights learned by the Mixture of Tokens can be used to directly initialize a Token Choice model of the same specifications (number of experts and expert size). The layer responsible for producing mixing weights is utilized to initialize the sparse router. In order to mitigate the difference in performance that is caused by this change in architecture, we train the entire new model (no weights are frozen) for 10% of the total pretraining steps of the original model in order to recover the original model's performance (measured in eval loss). We call this technique *transition tuning*. This way, it is possible to train with Mixture of Tokens and enjoy unbatched generation at inference time. We hypothesize that this pipeline would be especially attractive in setups where having parts of the model train in higher precision is impossible, e.g., on specialized, low-precision hardware. The results are presented in Figure 8.

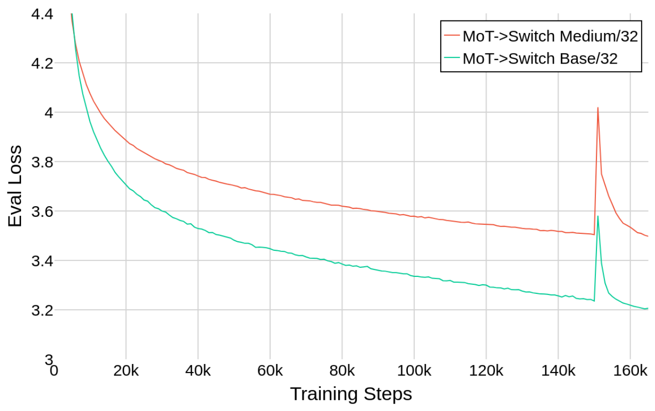

Figure 8: Transition tuning: The first 150K steps of the model are completed using the Mixture of Tokens architecture. Then, a new Token Choice model is initialized with weights from the MoT model, and the model trains for an additional 15K steps to recover performance. The spike in loss results from the sudden change of architecture.

5 Limitations and Future Work

With the strong performance of MoT on medium-sized models, an obvious next step is to train larger models. This would present an opportunity to validate the stability results on larger models, where training instabilities are more common.

As with most Mixture of Experts models, the memory footprint of MoT layers is substantial. Scaled models require large amounts of RAM on specialized hardware for training, making their adoption expensive. To this end, an attractive future direction would be to investigate model distillation with Mixture of Tokens models.

In this work, we experimented only with text modality in an autoregressive manner. Other modalities, such as vision, heavily overlap with the approach presented in work concurrent to ours [15].

Lastly, both training and inference with MoT involve mixing different examples within a single batch. This mixing of tokens from different sequences and the requirement of performing batched inference may be undesirable in some use cases. While performing unbatched inference is always inefficient with LLMs, as the memory throughput to access model weights becomes the bottleneck, unbatched inference still finds its uses. Even though transition tuning solves this problem, exploring different inference strategies might bring new insights.

6 Conclusions

In this work, we presented the Mixture of Tokens, a novel continuous Mixture of Experts architecture compatible with autoregressive decoding. This architecture scales to model sizes similar to sparse Mixture of Experts models, matches their performance, and is more resistant to training instabilities due to lower precision training. Moreover, we introduced transition tuning, a technique for initializing an MoE model with another pretrained MoE model of a different architecture, and showed that the new model achieves the performance of the original one using a fraction of the compute budget.

Acknowledgements

We would like to express our sincere gratitude to Piotr Padlewski and Tomasz Trzciński for their general feedback and Dagmara Rudzińska for her invaluable support with graphic design and to Piotr Sankowski for providing an excellent scientific environment.

This work was funded by IDEAS NCBR, which also provided significant computational resources. Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0392/17-00. The research was supported by PL-Grid infrastructure (grants PLG/2023/016148, PLG/2024/017060). We also benefited from the Entropy cluster (hosted at the Faculty of Mathematics, Informatics, and Mechanics of the University of Warsaw) funded by NVIDIA, Intel, the Polish National Science Center grants UMO-2017/26/E/ST6/00622 and 2022/45/N/ST6/02222, and ERC Starting Grant TOTAL. Part of the evaluations was done using computational resources provided by Writer.

References

[1] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila

Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

[2] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023.

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[4] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

[6] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2022.

[7] Malte J. Rasch, Charles Mackin, Manuel Le Gallo, An Chen, Andrea Fasoli, Frederic Odermatt, Ning Li, S. R. Nandakumar, Pritish Narayanan, Hsinyu Tsai, Geoffrey W. Burr, Abu Sebastian, and Vijay Narayanan. Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators, 2023.

[8] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference, 2023.

[9] Joseph McDonald, Baolin Li, Nathan Frey, Devesh Tiwari, Vijay Gadepally, and Siddharth Samsi. Great power, great responsibility: Recommendations for reducing energy for training language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, 2022.

[10] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[11] Omar Sanseviero, Lewis Tunstall, Philipp Schmid, Sourab Mangrulkar, Younes Belkada, and Pedro Cuenca. Mixture of experts explained, 2023.

[12] Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts, 2022.

[13] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. On the representation collapse of sparse mixture of experts, 2022.

[14] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts, 2022.

[15] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts, 2023.

[16] Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing, 2023.

[17] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed H. Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning, 2021.

[18] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Łukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers, 2021.

[19] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2022.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[21] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[22] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien

de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2022.

[23] Robert A. Jacobs, Michael I. Jordan, and Andrew G. Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250, 1991.

[24] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts, 2014.

[25] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[26] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

[27] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022.

[28] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

[29] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[30] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022.

[31] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models, 2021.

[32] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models, 2021.

[33] Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models, 2022.

[34] Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts, 2024.

[35] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[37] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[38] Zanyar Zohourianshahzadi and Jugal K. Kalita. Neural attention for image captioning: review of outstanding methods. *Artificial Intelligence Review*, 55(5):3833–3862, November 2021.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[40] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[44] Christophe Cerisara. *SlowLLM: large language models on consumer hardware*. PhD thesis, CNRS, 2023.

[45] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019.

[46] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[47] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

A Training Hyperparameters

All models were trained using mixed precision unless explicitly stated otherwise. We conducted all experiments with a batch size of 256 and a context length of 256 for 150K training steps (unless explicitly stated), resulting in a total of 10B training tokens. We used the AdamW optimizer with default hyperparameters. When necessary, we adopted a Fully Sharded Data Parallel approach from PyTorch to parallelize training across multiple machines. Learning rates were tuned separately based on model size and architecture. The optimal learning rate for Transformers was 1e-3 for Medium models and 4e-4 for Base models, while for both MoT and MoE, they were 7e-4 for Medium models and 2e-4 for Base models.

Table 2: Training hyperparameters. The table provides example models featured in our experiments. All remaining models can be derived from this table.

Model	Experts	Expert size	Group size	Total params	Blocks	d_{model}	d_{ff}	#att. heads
Transformer-Medium	-	-	-	77M	8	512	2048	8
MoT-Medium/32E	32	2048	32	336M	8	512	-	8
MoT-Medium/32E/8	256	256	32	337M	8	512	-	8
Transformer-Base	-	-	-	162M	12	768	3072	12
MoT-Base/32E	32	3072	32	520M	12	768	-	12
MoT-Base/64E/16	1024	192	64	977M	12	768	-	12

B Downstream Evaluation

When trying to predict how specific changes to the model architecture will impact large-scale models, comparing perplexity can provide a reliable indication of model improvements. However, for completeness, we also measured performance on several downstream tasks relevant at this model scale, comparing MoT-Medium to Transformer-Medium, without fine-tuning, in a zero-shot setting. In these evaluations, for MoT, a single evaluation query is included in a batch of 32, with the remainder of the batch comprised of random sequences from the C4 training dataset, ensuring it remains zero-shot. We observe predictable improvements with the Mixture of Tokens on tasks PIQA [45], HellaSwag [46], and ARC-e [47], see Table 3.

Table 3: Performance of a medium-sized model on downstream benchmarks.

	Transformer-Medium	MoT-Medium/32E/1	MoT-Medium/32E/16
PIQA	60.2	62.4	**65.8**
HellaSwag	27.3	31.1	**33.3**
ARC-e	35.5	37.3	**39.6**

C Reproducibility

The code and configuration files used to produce the results described in this work are available in our public repository at `https://github.com/llm-random/llm-random`.

D Socio-Economic Impacts

The goal of this paper is to advance the field of Transformer training and Machine Learning in general, language modeling in particular. With Large Language Models exhibiting the most impressive results in Machine Learning to date, we believe that this work is able to help advance the model capabilities even further. As to the potential socio-economic consequences of our work, it shares the common potential impact of all work done on training efficiency.

E Compute Resources

Table 4: Compute resources used for each experiment. All models were trained on NVIDIA A100 GPUs, with either 40 or 80 GB of RAM.

Model	GPU RAM	Time	GPUs
Transformer-Base	40GB	32h 20m	1
MoT-Base/64E/16	40GB	33h 12m	2
MoT-Medium/128E	40GB	26h 36m	1
MoT-Medium/32E	40GB	23h 9m	1
MoT-Medium/8E	40GB	22h 31m	1
MoT-Medium/32E	40GB	20h 17m	1
Transformer-Medium	40GB	18h 48m	1
MoT->Switch Medium/32	40GB	35h 11m	1
MoT->Switch Base/32	40GB	17h 20m	4
MoT-Medium/32E/1	40GB	23h 9m	1
MoT-Medium/32E/8	40GB	24h 13m	1
MoT-Medium/32E/16	40GB	25h 36m	1
MoT-Medium/32E/32	40GB	28h 20m	1
Expert Choice-Base/32E	80GB	21h 12m	2
MoT-Base/32E	80GB	19h 38m	2
MoT-Base/32E/8	40GB	22h 10m	2
Token Choice-Base/32E	80GB	20h 19m	8

F Contributions

Szymon implemented a PoC and different variants of MoT, together with running experiments and optimizations. Michał implemented and experimented with various MoT designs and contributed to the infrastructure design and implementation. Sebastian provided the initial idea, research intuitions, and direct project supervision. Maciej was responsible for parts of evaluation and significant engineering. Jakub implemented MoE baselines, Jan stabilized Mixture of Experts training, while both helped with MoE hyperparameter tuning. Tomasz consulted ideas and helped with cluster infrastructure. Kamil and Krystian contributed to general engineering. Everybody above contributed to the infrastructure of the project. Marek provided scientific advice and high-level supervision.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims in the abstract and the introduction are substantiated by experiments and results shown in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 provides detailed limitations to the introduced technique.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper doesn't contain theoretical results or theorems, as it relies on experimental data.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper comes with a complete training and evaluation code, the settings of experiments are contained in the paper, and the dataset is publicly available. Therefore, the paper provides all the information needed to reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Complete training and evaluation code is provided and open-sourced. The datasets used are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides all settings necessary to understand the results in Appendix A, and training/test setup is available in the source code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: With the work's budget, running multiple experiments to get statistical significance was not feasible for most experiments. Experiments with multiple reruns showed stability of reported results over multiple reruns compared to reported differences between models (see Table 1, row Mixed Precision).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The utilized computational resources are summarized in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Authors confirm adherence to the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The societal impact of this paper is no different from any other work improving the efficiency of LLMs. This is described in more detail in the Appendix D.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any pretrained models and use publicly available data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We are the creators of the code used in the paper. We cited the original paper of the dataset and included the URL to a version that was used in the training setup. We also mentioned the license of the dataset, which we properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: With the paper submission, we provide authored code on Apache License Version 2.0. The repository contains files with instructions on how to set up the environment and run experiments. The code is well documented with emphasis on our main algorithm.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The work did not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.