A multilevel approach to accelerate the training of Transformers

Guillaume LAUGA[1] Maël CHAUMETTE†[1] Edgar DESAINTE-MARÉVILLE†[1] Étienne LASALLE†[1] Arthur LEBEURRIER†[1]

[1]ENS de Lyon, CNRS, Inria, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France

Résumé – Dans cet article, nous étudions le potentiel des approches multiniveaux pour accélérer l'entraînement des Transformers. En utilisant une interprétation de ces architectures sous forme d'équations différentielles ordinaires (EDO), nous proposons une manière appropriée de varier la discrétisation de l'EDO décrivant ces transformers afin d'accélérer l'apprentissage. Nous validons le potentiel de notre approche en la comparant à un apprentissage standard.

Abstract – In this article, we investigate the potential of multilevel approaches to accelerate the training of transformer architectures. Using an ordinary differential equation (ODE) interpretation of these architectures, we propose an appropriate way of varying the discretization of these ODE Transformers in order to accelerate the training. We validate our approach experimentally by a comparison with the standard training procedure.

1 Introduction

Transformer architectures have become ubiquitous since their introduction [26]. They are state-of-the-art on a large class of problems including image recognition [9, 20], speech processing [4, 21] and large language models [24, 3, 25]. The performance of these models, which increases with their dimension, is balanced by training and inference costs. Hence, it is crucial to propose new training methods to better handle their cost. In this paper, we propose to accelerate the training of transformers models by relying on an ODE interpretation of transformer-decoder architectures and multilevel optimization, in a similar fashion as in [15, 10] for ResNet models.

Transformer-decoder networks are composed of an input layer that transforms tokens into embeddings and passes these embeddings to a sequence of transformer blocks which are attention layers followed by feed-forward networks, intertwined with normalization layers. The output is then transformed from embeddings to tokens. Similar to previous deep neural architectures, transformers become progressively harder to train as their depth increases, primarily due to large-scale optimization and dependencies on residual connections that makes training unstable, since it amplifies small parameter perturbations [19]. Reducing the number of parameters is a direct remedy to this problem, at the expense of the network's expressivity. Therefore it becomes more and more interesting to find training approaches able to manage the growth in the number of parameters without reducing performance.

A possible framework to manage this growth is derived from an ODE interpretation of transformer networks [18, 2]. The success of the ODE interpretation of ResNet networks [6, 1] motivated the community to extend such study to other residual networks such as transformer-decoder architectures [18, 2], at the heart of modern large language models [24, 3].

Our approach is inspired by the work of [15, 10], whose authors investigate the impact of varying depths of ResNet networks based on the ODE formalism, and train a deep ResNet in a multilevel fashion. In order to accelerate the optimization, multilevel algorithms exploit a hierarchy of approximations of the objective function. Reducing the problem dimension can lead to significant gains in convergence speed [17, 16].

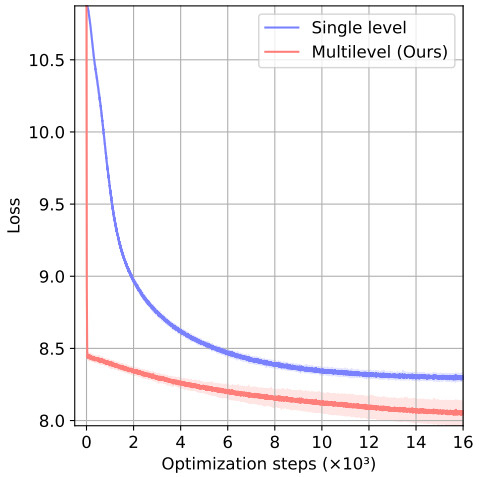

Figure 1 – Training loss of the single level algorithm (standard method) in blue and of the multilevel algorithm (our proposed approach) in red with respect to the number of optimization steps (at fine level). The curves are averaged over 6 seeds. In lighter colors we display the standard deviation of the 6 training runs.

Related works. A lot of effort has been dedicated to accelerating the training of transformers. A common idea is to increase the size of the network during training, starting from a smaller network. We can divide these approaches depending on the proposed growth method: depth [11], width [13] or both [7, 5, 27, 8].

In [11], the authors propose a method to transfer knowledge from a shallow model to a deeper one by progressively stacking layers and doubling its depth. This strategy is motivated by observing that lower and upper layer representations exhibit similar distributions. A similar perspective is adopted in [13],

†: Equal contributions. We acknowledge the support of the Centre Blaise Pascal's IT test platform at ENS de Lyon (Lyon, France) for its computing facilities. The platform operates the SIDUS solution [23] developed by Emmanuel Quemener.

where the authors stack weights and double the width of the model, ensuring that the output remains preserved after the expansion.

In [7, 5, 8], authors propose techniques for transferring information stored in a neural network into another neural network with increased width or depth. For instance, the Net2DeeperNet method in [7] transforms one layer into two layers by initializing the added layer's weights as the identity. In [8], the small network is initialized as a pruned dense network, that gradually extends into the dense network. Authors in [27] also propose a method to simultaneously extend the width and the depth. Their algorithm learns an expansion matrix from a given dataset. All these methods change the size of the network during training, and are therefore not comparable to our proposed approach.

In an adjacent direction, authors of [28] propose to speed up the training by sharing weights across all transformer blocks up until a certain point in the training. This procedure allows the blocks to quickly learn common shared features, therefore bringing the weights closer to the optimal solution faster.

The idea of training neural networks using multilevel techniques has been investigated in several papers [10, 15, 30, 12]. Authors of [10, 15] start from the ODE interpretation of ResNets to create a hierarchy of smaller networks then used in a multilevel fashion by alternating the training of the smaller networks and of the target network, achieving great acceleration. In [12], the context of PDE solving allows the network to be divided into blocks that can be trained separately. Each block targeting different frequency components of the solution, the training is accelerated.

Lastly, a multilevel approach reducing width and depth of transformer network during training was recently proposed in [30], saving up to 20% of the total computational cost. The authors introduce a construction of operators that allow to go from one network to the other, reducing (resp. increasing) the depth and the width sequentially.

Contributions. In this paper, we propose a multilevel approach to train an ODE transformer neural network for sequence generation. By varying the depth, we obtain smaller networks that are easier to train, and we propagate these smaller networks to the fine network, thus accelerating its training. In contrast to [30], we do not use operators to reduce the width and the depth of the network. We rely on the ODE interpretation to reduce the dimension without any operators.

Outline. In Section 2, we present the ODE formalization of transformer networks, and we then introduce our multilevel algorithm for the training. In Section 3, we compare our algorithm to a single level training on a sequence generating task. Finally, we discuss our findings and potential improvements.

2 Multilevel training for ODE transformers

In this section, we present the interpretation of transformer-decoder networks using an ODE [18]. The presentation is slightly different than in [18], but principles remain similar. This interpretation will then allow us to derive a multilevel algorithm to train transformer networks.

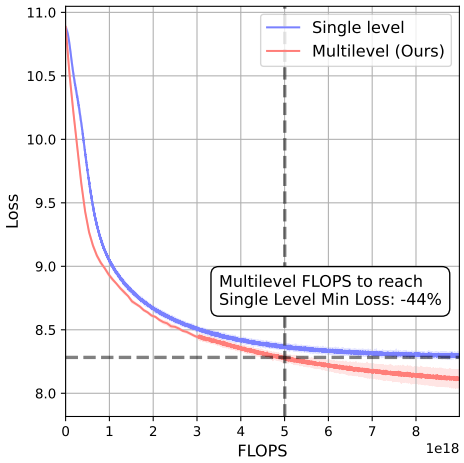

Figure 2 – Training loss of the single level algorithm (standard method) in blue and of the multilevel algorithm (our proposed approach) in red with respect to FLOPS. The curves are averaged over 6 seeds. In lighter colors we display the standard deviation of the 6 training runs.

2.1 ODE Transformers.

Like ResNets, transformer networks use residual connections. The ODE interpretation is well known for ResNets [6, 1], and is a consequence of these residual connections.

A transformer layer is composed of a self-attention (SA) block followed by a feed-forward (FF) network. The input of these blocks is passed through a Layer Norm (LN) operation, and a residual connection. Denote by $\mathbf{x}_t$ the input of a transformer layer at time t. The output $\mathbf{x}_{t+1}$ is given by:

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t + \mathrm{SA}_t(\mathrm{LN}_t(\mathbf{x}_t)) + \mathrm{FF}_t(\mathrm{LN}_t(\mathbf{x}_t + \mathrm{SA}_t(\mathrm{LN}_t(\mathbf{x}_t)))) \\ &= \mathbf{x}_t + \mathrm{F}(\mathbf{x}_t, \theta_t) \end{aligned} \tag{1}$$

where $\mathrm{F}(\cdot,\cdot) = \mathrm{SA}_t(\mathrm{LN}_t(\cdot)) + \mathrm{FF}_t(\mathrm{LN}_t(\cdot + \mathrm{SA}_t(\mathrm{LN}_t(\cdot))))$. At time t, F will depend on parameters θ_t. This equation can be seen as the Euler discretization of the following ODE

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathrm{F}(\mathbf{x}(t), \theta(t)) \tag{2}$$

with a discretization step $\Delta t = 1$. Our idea is to use different levels of discretization of this ODE to define a hierarchy of networks. The fine level network will have the desired N (even) number of transformer layers F_i with parameters θ_i, for $i = 1, \ldots, N$. Coarser networks will use less and less layers corresponding to their discretization level.

2.2 Multilevel training of transformers.

For the clarity of the presentation, we limit ourselves to two levels. We will denote by h elements of the fine level network, and by H elements of the coarse level network.

In the following, coarse and fine level networks will share input and output layers that respectively transform tokens into embeddings and embeddings into tokens. It is standard practice for input and output layers to share the same weights [24, 3].

Construction of coarse models. To coarsen our fine level network, we will divide by 2 the number of transformer layers.

The coarse model has therefore $N/2$ layers F_i^H with parameters θ_i^H for $i = 1, \dots, N/2$. These layers are the layers of the fine level network, linked by the following relationship

$$\forall i \in \{1, \dots, N/2\}, \quad \theta_i^H = \theta_{2i}^h. \tag{3}$$

At any point during the training, this relationship holds (without additional memory footprint). After K steps of coarse optimization, we obtain a new set of parameters $(\tilde{\theta}_i^H)_{1 \leq i \leq N/2}$. We then update fine level parameters by prolongating the coarse parameters. For all $i \in \{1, \dots, N\}$

$$\begin{aligned} \tilde{\theta}_{2i}^h &= \tilde{\theta}_i^H, \\ \tilde{\theta}_{2i+1}^h &= (1-\delta)\theta_{2i+1}^h + \delta\tilde{\theta}_i^H, \end{aligned} \tag{4}$$

where $\delta \in [0, 1]$ is an averaging constant. Here we follow the example of [30]. The simplest setting would be to set $\delta = 1$, and would be substantiated by the literature on numerical solution of ODEs (see [15, 10] and references therein). However, due to the low number of layers in our context (a dozen versus a few thousands in [15, 10]), we would lose too much information at the fine level. Moreover, one can see that only training even layers leads the training to be quite asymmetrical. Therefore we propose to *train two coarse models*. The second coarse model is linked to the fine model by

$$\forall i \in \{1, \dots, N/2\}, \quad \theta_i^H = \theta_{2i-1}^h, \tag{5}$$

with the symmetric of Equation 4 to prolongate the coarse parameters to the fine levels. We display in Figure 3 an example of a fine level network with 4 layers and its two coarse models. One training step on a coarse model costs less than an training

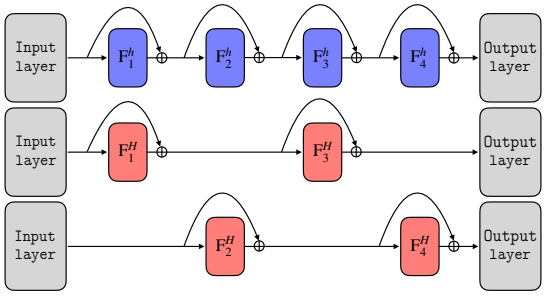

Figure 3 – Scheme of our proposed approach on a network with 4 transformer layers. The fine level network (blue blocks) is decomposed into two coarser networks (red blocks) that contains even-indexed (resp. odd-indexed) layers. Input and output layers (gray blocks) are shared across all networks.

step on the fine model and training a smaller network is easier. Hence, the loss of expressivity by using smaller networks should be outweighed by the computational and training gains.

Optimization parameters at coarse level. It remains now to address the training of the coarse models. Both coarse models see the same data points as the fine level, in the same order. We tried to vary the data points and the order, without any noticeable effect. The number of tokens seen at each optimization step also remains the same. In a few words, we train each coarse model like the fine model. After training both coarse models, we always compute one optimization step for the fine model.

A notable difference with standard multilevel approaches is that we do not share gradient information between levels. Standard approaches impose coherence between levels using this information [17]. Due to the stochastic nature of the gradient, in our context, adopting this coherence leads to be coherent with noisy information.

3 Numerical experiments

In this section, we numerically compare the training speed of our approach against the standard single level approach.

Dataset. We use a portion of the FineWeb-Edu dataset: a billion of GPT-2 tokens, sampled from the 10 billions smaller version [22]. This dataset contains educational pages of the FineWeb dataset.

Architecture choice. For our experiments we consider accelerating the training of a transformer-decoder architecture that takes as input sequences 256 tokens obtained with the GPT-2 tokenizer [1] in a vocabulary of size 50257. The input layer embeds the tokens into a space of dimension 256, after which follows 12 transformer blocks with 8 attention heads each. The output layer then reverts the embedding operation. Across all levels, the input and output layers remain untouched, and we only reduce the number of transformer blocks to define our coarse models. The number of parameters of this architecture is $22,368,512$, which allows training the fine model and the coarse models on a single GPU with 16 GB of RAM.

Hyperparameters for training. Each model was trained with Stochastic Gradient Descent (SGD). This choice is not the state-of-the-art for training transformer models for sequence generation, but is easier to handle when dealing with the training of several related networks (see the discussion at the end). 16000 steps are computed with a batch size of 32 sequences of length 256, accumulated to reach a total batch size of 262144 tokens. Therefore, the network is trained for a little bit more than 4 epochs. All training runs follow a learning rate schedule with 715 steps of linear warm-up followed by a cosine decay to zero. Minimum and maximum learning rates were set to 1.2×10^{-4} and 1.2×10^{-3} respectively [24, 3]. Hyperparameters for the coarse models are identical, except for the learning rate that remains constant equal to 1.2×10^{-3}.

Multilevel hyperparameters. We use two coarse models: one that trains odd-indexed transformers blocks (1,3,5,7,9,11), the other that trains even-indexed (2,4,6,8,10,12) transformer blocks. Each coarse model has $17,649,920$ parameters. After one step of fine level optimization, we use the coarse models for 35 steps. During these steps, each coarse model is trained for 100 steps, before propagating their weights to the fine level network. Both coarse networks are trained using the same optimizer as the fine network. For both coarse models, we set the averaging constant δ to 0.25 following [30] and our own experiments. After these 35 steps, we only train the fine level network. In practice, when using coarse levels long after these starting steps, we observed diminishing returns with respect to the computational cost, therefore we only used them at the start.

Comparison metrics. In order to show that a multilevel approach can accelerate the training, we compare the training

1. tiktoken used with `tiktoken.get_encoding("gpt-2")`

losses with respect to the iterations of our approach and of the single level approach. Such comparison is at our advantage since one iteration of our algorithm encompasses several coarse iterations. Thus, to be fair to the single level algorithm we also compare the training loss with respect to the number of Floating Point Operations (FLOP)s computed per iteration. We estimate the number of FLOPs of one iteration by estimating the number of FLOPs required to do one forward pass of the networks. Then we follow similar works that estimate that one training step requires in FLOPs the equivalent of 3 forward pass (cf this report [2] or [14]).

Results. Training curves are averaged over 6 training runs using 6 different seeds. We display in Figure 1 the training loss of our multilevel approach and of the single level approach with respect to the number of optimization steps. In Figure 2, we display losses with respect to the number of FLOPs. Our algorithm achieves the same loss as the single level training in 16000 steps, while reducing the total number of FLOPs by 44%, demonstrating the efficiency of our multilevel method in accelerating the training of transformer networks.

Discussion. As mentioned earlier, [30] performs multilevel on a transformer-decoder. In order to compare to their method, we need to train larger architectures such as GPT-2 [24], as they did. Due to the higher number of hyperparameters requiring tuning for their method to work (compared to ours), we could not afford to do this expensive tuning yet. Hence, we leave a comparison for later works.

However, their coarse level is obtained by reducing the width of each layer and the depth of the network while our multilevel approach focuses on the depth of a transformer-decoder as this method respects the ODE interpretation detailed in Section 2. We are aware that SGD is not the preferred algorithm for language tasks [29]. Showing that our approach can accelerate the training with the vanilla algorithm is a first but necessary step towards the acceleration of more complex algorithms. Notably, one question needs to be solved in order to be competitive with Adam or AdamW: the interaction between the momentum at fine and coarse level. In a convex setting, this momentum does not seem to be a problem [17], but in deep learning settings it is [12]. To the best of our knowledge, no multilevel algorithm handle it in a satisfying manner. The question is either omitted [30, 10, 15] or circumvented using restarting techniques [12].

4 Conclusion

In this article we proposed a multilevel approach to accelerate the training of transformer networks. Our approach is based on an ODE interpretation of these networks, which allows us to vary their size by varying the discretization that solves the associated ODE. We obtained impressive savings in FLOPs with respect to the single level training. At the moment, the setting of these experiments is limited. It calls for more experiments, and also more theoretical developments to better understand interactions between the training of finer and coarser networks, notably the momentum. We also need to compare the robustness of the trained network with respect to test tasks to completely validate the benefit of our approach.

2. https://epoch.ai/blog/backward-forward-FLOP-ratio

References

[1] B. Avelin and K. Nyström. Neural ODEs as the deep limit of ResNets with constant weights. *Analysis and Applications*, 19(03):397–437, 2021.

[2] A. Baier-Reinio and H. De Sterck. N-ode transformer: A depth-adaptive variant of the transformer using neural ordinary differential equations. *Preprint arXiv:2010.11358*, 2020.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[4] X. Chang, W. Zhang, Y. Qian, J. Le Roux, and S. Watanabe. End-to-end multi-speaker speech recognition with transformer. In *ICASSP 2020*, pages 6134–6138. IEEE, 2020.

[5] C. Chen, Y. Yin, L. Shang, X. Jiang, Y. Qin, F. Wang, Z. Wang, X. Chen, Z. Liu, and Q. Liu. bert2bert: Towards reusable pretrained language models. *Preprint arXiv:2110.07143*, 2021.

[6] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations, December 2019. arXiv:1806.07366.

[7] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *Preprint arXiv:1511.05641*, 2015.

[8] N. Ding, Y. Tang, K. Han, C. Xu, and Y. Wang. Network expansion for practical training acceleration. In *IEEE/CVF CVPR*, pages 20269–20279, 2023.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Preprint arXiv:2010.11929*, 2020.

[10] L. Gaedke-Merzhäuser, A. Kopaničáková, and R. Krause. Multilevel minimization for deep residual networks. *ESAIM: Proceedings and Surveys*, 71:131–144, August 2021.

[11] L. Gong, D. He, Z. Li, T. Qin, L. Wang, and T. Liu. Efficient training of bert by progressively stacking. In *ICML*, pages 2337–2346. PMLR, 2019.

[12] S. Gratton, V. Mercier, E. Riccietti, and P. L. Toint. A block-coordinate approach of multi-level optimization with an application to physics-informed neural networks. *Computational Optimization and Applications*, 89(2):385–417, 2024.

[13] X. Gu, L. Liu, H. Yu, J. Li, C. Chen, and J. Han. On the transformer growth for progressive bert training. *Preprint arXiv:2010.12562*, 2020.

[14] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling Laws for Neural Language Models. *CoRR*, abs/2001.08361, 2020.

[15] A. Kopaničáková and R. Krause. Globally Convergent Multilevel Training of Deep Residual Networks, June 2022.

[16] G. Lauga. *Multilevel proximal methods and application to image restoration*. phdthesis, École Normale Supérieure de Lyon, December 2024.

[17] G. Lauga, E. Riccietti, N. Pustelnik, and P. Gonçalves. IML FISTA: A Multilevel Framework for Inexact and Inertial Forward-Backward. Application to Image Restoration. *SIAM Journal on Imaging Sciences*, 17(3):1347–1376, 2024.

[18] B. Li, Q. Du, T. Zhou, Y. Jing, S. Zhou, X. Zeng, T. Xiao, J. Zhu, X. Liu, and M. Zhang. ODE Transformer: An Ordinary Differential Equation-Inspired Model for Sequence Generation, March 2022. arXiv:2203.09176.

[19] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han. Understanding the difficulty of training transformers. *Preprint arXiv:2004.08249*, 2020.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF ICCV*, pages 10012–10022, 2021.

[21] L. Lu, C. Liu, J. Li, and Y. Gong. Exploring transformers for large-scale speech recognition. *Preprint arXiv:2005.09684*, 2020.

[22] G. Penedo, H. Kydlíček, L. Ben allal, A. Lozhkov, M. Mitchell, C. Raffel, L. Von Werra, and T. Wolf. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale, 2024.

[23] E. Quemener and M. Corvellec. SIDUS—the Solution for Extreme Deduplication of an Operating System. *Linux Journal*, 2013.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[25] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.A. Lachaux, et al. Llama: Open and efficient foundation language models. *Preprint arXiv:2302.13971*, 2023.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. 2017.

[27] P. Wang, R. Panda, L. T. Hennigen, P. Greengard, L. Karlinsky, R. Feris, D. D. Cox, Z. Wang, and Y. Kim. Learning to grow pretrained models for efficient transformer training. *Preprint arXiv:2303.00980*, 2023.

[28] S. Yang, L. Hou, X. Song, Q. Liu, and D. Zhou. Speeding up deep model training by sharing weights and then unsharing. *Preprint arXiv:2110.03848*, 2021.

[29] R. Zhao, D. Morwani, D. Brandfonbrener, N. Vyas, and S. Kakade. Deconstructing what makes a good optimizer for language models. *Preprint arXiv:2407.07972*, 2024.

[30] L. Zou, H. Zhang, and Y. Deng. A Multi-Level Framework for Accelerating Training Transformer Models, April 2024. arXiv:2404.07999.