Cup Curriculum: Curriculum Learning on Model Capacity

Luca Scharr LUCA.SCHARR@UNI-BONN.DE
Vanessa Toborek TOBOREK@CS.UNI-BONN.DE
University of Bonn, Germany

Abstract

Curriculum learning (CL) aims to increase the performance of a learner on a given task by applying a specialized learning strategy. This strategy focuses on either the dataset, the task, or the model. There is little to no work analysing the possibilities to apply CL on the model capacity in natural language processing. To close this gap, we propose the *cup curriculum*. In a first phase of training we use a variation of iterative magnitude pruning to reduce model capacity. These weights are reintroduced in a second phase, resulting in the model capacity to show a cup-shaped curve over the training iterations. We empirically evaluate different strategies of the cup curriculum and show that it outperforms early stopping reliably while exhibiting a high resilience to overfitting.

Keywords: Curriculum Learning, Natural Language Processing, Iterative Magnitude Pruning

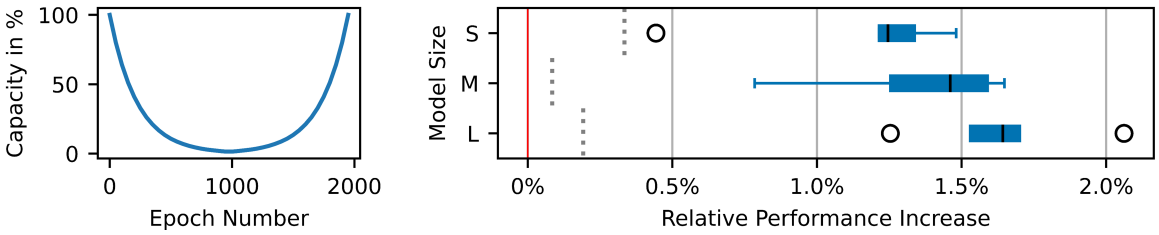

Figure 1: (Left) Model capacity throughout training with the cup curriculum. (Right) Relative performance increase of the best performing cup curriculum strategy observed over the best performance of 20 early stopping runs per model size. The red line marks early stopping, the grey line the relative performance increase of the best model found by IMP.

1. Introduction

Curriculum Learning (CL) is a well established learning strategy for state of the art approaches in multiple areas of supervised learning including image recognition and natural language processing (NLP) [17, 19, 23]. CL approaches can specify training strategies either for the training data, the task, or the model capacity, where most popular approaches focus on the training data [19]. Although the seminal work of Elman [7] stresses the importance of CL for model capacity, this seems not have been explicitly addressed in literature for NLP so far, especially the reintroduction of capacity. We therefore in an NLP scenario investigate iterative pruning strategies reducing model capacity followed by iterative reintroduction of model capacity (see Fig. 1 left). We formally formulate this as *cup curriculum*. In experiments we see that this approach can significantly and reliably improve upon early stopping as well as existing iterative magnitude pruning (IMP) strategies (see Fig. 1 right).

© L. Scharr & V. Toborek.

Related Work It is common knowledge, that children learn exceptionally well. This is, at least partly, attributed to the large changes in synaptic density in the human brain observed throughout its development in infancy and adolescence [10]. These changes in synaptic density precisely constitute CL for model capacity. Therefore, Elman [7] points out the need to apply this variation of CL in the field of NLP. Bengio et al. [2] introduced CL to computer science but focus on the training data. Their pioneering work shaped the field of CL in computer science, which largely focuses on the training data [19]. Few work utilize CL for model capacity such as Morerio et al. [15] and Sinha et al. [18]. The former achieves it by utilising dropout and progressing the dropout probability of neurons. The latter gradually reduces noise to impair the model capacity during the training. However, none of these works are in the field of NLP. The idea of pruning parts of a neural network and introducing equivalent parts in terms of expressive capacity later in the training procedure was presented by Prakash et al. [16]. However, they studied this idea for Convolutional Neural Networks (CNN) and only for a few filters at a time.

Our training strategy requires the reduction of model capacity, which makes a suitable pruning algorithm paramount. We decided on a variation of layer wise IMP, as proposed by Zhou et al. [24]. This pruning algorithm is very similar to those used in Frankle and Carbin [8] and Lin et al. [11]. We decided to stay closer to the former (see Algorithm 1), as it was applied successfully to the transformer architecture before [3, 22]. During the second part of our training strategy, the model capacity is increased. To do so we select the introduction scheme promoted by Prakash et al. [16] and tailor it for our problem setting (see Algorithm 2).

Contributions To the best of our knowledge this is the first paper providing an analysis of CL for model capacity in NLP. We formulate a new CL approach for the model capacity, the cup curriculum, and apply it to the transformer architecture. The cup curriculum is characterized by two sequential training phases: first, we create a curriculum by reducing the expressiveness of a given model, then we increase the capacity by reintroducing the weights during training. The transformer architecture is selected for our analysis due to its importance for the field of NLP [1, 4–6, 20, 21]. In our hyperparameter search we were able to find promising cup curricula, which are able to reliably (confidence of 99%) improve over the performance reached by early stopping. Summarizing, we (i) introduce a model curriculum (Section 3), (ii) diligently test different strategies (Section 4), and (iii) provide easy to use code (GitHub).

2. Curriculum Learning for Model Capacity

We say a model learns from data with respect to some task if its performance as measured by some performance measure improves over time [14]. CL for model capacity can be characterized by a manipulation of model capacity (measured by some capacity measure) during training with the aim of improving the model's performance [19].

Problem Formulation Similar to multiple other machine learning approaches, the goal of our training strategy is the reduction of loss. This naturally coincides with an improvement in the performance measure used for training, in our case the model's perplexity per word. Given a task and dataset, our goal is to find a parametrization Θ_{CL} of a given model architecture, such that it performs better than the parametrization $\Theta_{\text{No CL}}$ retrieved without CL. In contrast to $\Theta_{\text{No CL}}$, Θ_{CL} may have more, or less, parameters than the initial state Θ_0. We use IMP to specify the number of model parameters available throughout training.

Curriculum Generation We utilize the iterative nature of IMP to generate a curriculum for the model capacity. IMP is based on pruning cycles which span a set number of epochs. After the training in each cycle a pruning criterion is used to prune a percentage of weights individually. By specifying the number of pruning cycles, training epochs per pruning cycle, and the pruning criterion we receive a curriculum. Based on the analysis by Zhou et al. [24], we decided to use the magnitude change c as the pruning criterion ($c = \|w_c\| - \|w_i\|$, with current weight w_c and initial weight w_i). We utilize our cup curriculum framework to refine the received curriculum.

3. Cup Curriculum

Utilizing IMP and the insights of Prakash et al. [16] about the reduction and reintroduction of model capacity, we are able to formulate the *cup curriculum*:

Definition 1: Cup Curriculum Let M be a learner, D a dataset, P a performance measure, C a capacity measure, and let C_M denote the capacity of M according to C. Specify a pruning algorithm, a number of pruning cycles n, and a number of growth cycles m. For each of those cycles specify the number of training epochs e_i and the resulting model capacity c_i, where $i \in \{1, \ldots, n+m\}$. We also require $c_1 > c_2 > \cdots > c_n$ and $c_{n+1} < c_{n+2} < \cdots < c_{n+m}$.

Together, the above define a *cup curriculum*. Training M with it works as follows:

Pruning Phase For $i \in \{1, \ldots, n\}$ do:

1. Train M for e_i epochs.
2. Prune M, such that $C_M = c_i$ after completion of the pruning.
3. Optionally, rewind the unpruned parameters of M to an earlier state.

Growth Phase For $i \in \{n+1, \ldots, n+m\}$ do:

1. Introduce capacity to M, such that $C_M = c_i$ after completion of the introduction.
2. Train M for e_i epochs.

Report Version of M performing best according to P.

Definition 1 specifies a family of curricula for the model capacity. In our experiments we fix D to WikiText2 [13], P to cross-entropy loss, C to percentage of trainable weights, n and m to 20, e_i to 50 for $i \in \{1, \ldots, n+m\}$, and the pruning algorithm to IMP. Additionally, we decay c_i by 20% for $i \in \{1, \ldots, n\}$ and set $c_i = c_{n+m-i+1}$ for $i \in \{n+1, \ldots, n+m\}$. We evaluate different rewinding strategies for the pruning phase. The features extracted during this phase are considered to be essential for the model, as IMP aims to preserve the subspaces of the function space which are most important in the neural network [8]. Thus, we reintroduce the pruned capacity to the network based on their pruning order (last in first out) aiming to preserve these very features. We evaluate different initialization schemes which specify the value of reintroduced weights. Further, we analyze update schemes for the weight update depending on the introduction time of the weight, as further measures may be required to preserve important weights.

Overall, we evaluate the effects of the selected rewinding, initialization, and update scheme on the model performance and generalisation for three different model sizes. All results are compared to early stopping and among each other.

Model Sizes We consider three different model variations of the transformer differing in total parameter count. They are referred to as the *Small*, *Medium*, and *Large* model (see Table 3 for the exact number of attention heads and encoder layers).

Table 1: All rewinding schemes applied during the pruning phase.

Rewinding Scheme	Rewinding State
Initial	Initial State Θ_0
Warm	Warm-up State Θ_{warm}
Best	Best State $\Theta_{\text{best, cycle}}$
No	No Rewinding done

Table 2: All initialization schemes for reintroduced weights.

Initialization Scheme	Initialization State
Original	Initial State Θ_0
Random	Randomised State $\hat{\Theta}$
Old	Pruning State $\Theta_{\text{last, cycle}}$
Top	Best State $\Theta_{\text{best, cycle}}$

Rewinding Schemes We test four different rewinding schemes (see Section 3). The naive *Initial* scheme implements the original IMP [8], which rewinds weights repeatedly to their initial state Θ_0. In accordance with other results regarding the training of transformer models [3, 22], we test the *Warm* scheme that rewinds weights to a warm-up state $\Theta_{\text{warm}} = \Theta_3$. We select this model state according to the relative limits given by Frankle et al. [9]. The third scheme *Best* rewinds to the weights of the best performing model of the current cycle $\Theta_{\text{best, cycle}}$. Lastly, we compare all schemes to a baseline of no rewinding (*No*) with a fixed number of 50 iterations.

Initialization Schemes The second training phase of the cup curriculum is based on the work of Prakash et al. [16], but their findings are not fully applicable to our settings as we prune single weights instead of entire CNN filters. We thus explore four different initialization schemes (see Section 3). The *Random* scheme rewinds the weights by reevaluating the random distribution used to initialize Θ_0. We also tested initialization schemes based on the pruning criterion c employed during the first training phase ($c = \|w_c\| - \|w_i\|$) [24]. The *Old* scheme reintroduces the current weight w_c and the *Original* scheme the initial weight w_i. Both are motivated through their usage in c. Lastly, the *Top* scheme initializes each reintroduced weight with its value in $\Theta_{\text{best, cycle}}$, i.e. the model which performed best during the pruning cycle which pruned the respective weight.

Update Schemes We analyze the effect of three update schemes in the experiments, being the *Freezing*, *Identical*, and *Dynamic* update scheme. Here the *Identical* update scheme denotes the standard backpropagation update. We focus our analysis on it, as it produces the best results. For an in depth overview of the update schemes analyzed in the experiments see Appendix C (or Table 4).

4. Experiments

We trained a transformer architecture [21] on the WikiText2 dataset [13]. We ran each experiment with 5 unique seeds, a fixed learning rate scheduler, and started the learning rate at 5.0. The reported results required $1\,080$ wall clock GPU (Nvidia Geforce GTX 1080 Ti 11 GB) hours.

Performance Figure 2 shows the relative performance difference of all tested curricula to the best performance of 20 early stopping runs for the *Small* model size. The best performing strategy is *Best* rewinding with *Random* initialization and *Identical* updating. It shows a relative performance increase over early stopping of at least 0.5%. However, we also observe improvements of over 1.5%. Figure 1 (right) shows that these improvements increase the larger the model size becomes, even showing an improvement of over 2%. Additionally a median relative performance increase of 1% to 1.5% over the best performance of 5 IMP runs is shown. All of these improvements are obtained reliably with a confidence of 99% ($\alpha = 0.01$), as measured by the Wilcoxon-Mann-Whitney test [12]. It is

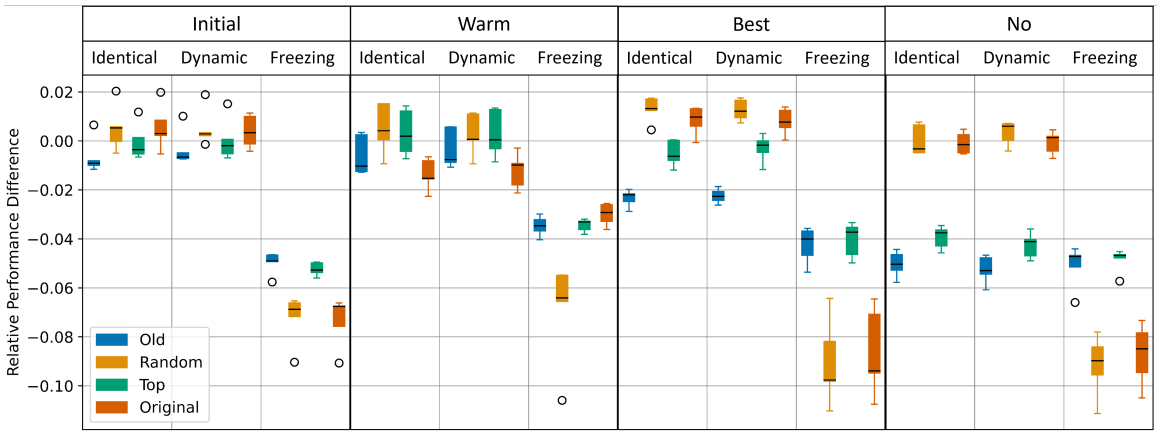

Figure 2: Relative performance difference to the best performing model found by early stopping in 20 runs averaged over all seeds.

surprising, but convenient, that the simplest considered version of the cup curriculum outperforms more complex ones.

Of the results seen in Fig. 2, it is particularly noticeable that strategies which use the *Identical* and the *Dynamic* update scheme significantly outperform those using the *Freezing* update. Furthermore, the *Random* and *Original* initialization work best for all rewinding schemes but *Warm* rewinding, where the *Random* and *Top* initialization work best. Additionally, for *Best* and *No* rewinding the *Old* and *Top* initialization perform much worse than the other initialization schemes (not counting the *Freezing* update). Despite this interaction, no significant difference between the *Identical* and the *Dynamic* update scheme can be observed for any of the experiments.

As an example of a training curve, Fig. 3 shows the training and validation loss of the best performing cup curriculum strategy found in the experiments. It exhibits resilience to overfitting in addition to the improvements over early stopping and IMP. While other strategies achieve different performance, all show similar resilience to overfitting (see Fig. 5). In contrast to early stopping, which prevents overfitting, the cup curriculum could further improve achieved performance with additional training.

5. Conclusion

In this work we highlight the existence of CL strategies for model capacity in the field of NLP and derive a framework for it: the cup curriculum. In addition to this proof of concept we analyze multiple strategies of the cup curriculum. This includes a strategy which reliably (confidence 99%) outperforms the best performance found by early stopping over multiple runs by an observed magnitude of 0.5% to 2% while staying resilient to overfitting with the prospect of increasing performance with additional training. Future work should analyze the impact of different learning rate schedulers on the cup curriculum given the insights of Touvron et al. [20] and test our curriculum on state-of-the-art LLMs with the prospect of circumventing the high number of checkpoints and their connected costs as mentioned by Chowdhery et al. [5].

Acknowledgements VTs research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence Lamarr22B.

References

[1] 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. *arXiv: 2305.10403*, 2023. URL https://arxiv.org/abs/2305.10403.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. In *ICML*, page 41–48. Association for Computing Machinery, 2009. URL https://doi.org/10.1145/1553374.1553380.

[3] Christopher Brix, Parnia Bahar, and Hermann Ney. Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture. In *ACL*, pages 3909–3915. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.acl-main.360.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *NeurIPS*, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] 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. *CoRR*, 2022. URL https://arxiv.org/abs/2204.02311.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1423.

[7] Jeffrey L. Elman. Learning and development in neural networks: the importance of starting small. In *Cognition*, pages 71–99. Elsevier, 1993. URL https://www.sciencedirect.com/science/article/pii/0010027793900584.

[8] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*. OpenReview.net, 2019. URL https://openreview.net/pdf?id=rJl-b3RcF7.

[9] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. The Lottery Ticket Hypothesis at Scale. *arxiv: 1903.01611*, 2019. URL http://arxiv.org/abs/1903.01611.

[10] Peter R. Huttenlocher. Synaptic density in human frontal cortex — Developmental changes and effects of aging. In *Brain Research*, pages 195–205, 1979. URL https://www.sciencedirect.com/science/article/pii/0006899379903494.

[11] Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic Model Pruning with Feedback. *arXiv: 2006.07253*, 2020. URL https://arxiv.org/abs/2006.07253.

[12] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. In *The Annals of Mathematical Statistics*, pages 50–60. Institute of Mathematical Statistics, 1947. URL http://www.jstor.org/stable/2236101.

[13] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *ICLR*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

[14] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. URL https://www.worldcat.org/oclc/61321007.

[15] Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, Rene Vidal, and Vittorio Murino. Curriculum Dropout. In *ICCV*, pages 3564–3572. Proceedings of the IEEE International Conference on Computer Vision, 2017. URL https://ieeexplore.ieee.org/document/8237645.

[16] Aaditya Prakash, James Storer, Dinei Florencio, and Cha Zhang. RePr: Improved Training of Convolutional Filters. In *CVPR*, pages 10658–10667. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. URL https://ieeexplore.ieee.org/document/8953322.

[17] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training Region-based Object Detectors with Online Hard Example Mining. *arXiv: 1604.03540*, 2016. URL https://arxiv.org/abs/1604.03540.

[18] Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum By Smoothing. In *NeurIPS*, pages 21653–21664. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f6a673f09493afcd8b129a0bcf1cd5bc-Paper.pdf.

[19] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum Learning: A Survey. *arXiv: 2101.10382*, 2021. URL https://arxiv.org/abs/2101.10382.

[20] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv: 2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NeurIPS*, pages 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[22] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. *arXiv: 1906.02768*, 2020. URL https://arxiv.org/abs/1906.02768.

[23] Peilin Zhao and Tong Zhang. Stochastic Optimization with Importance Sampling. *arXiv: 1401.2753*, 2015. URL https://arxiv.org/abs/1401.2753.

[24] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. In *NeurIPS*. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

Appendix A. Algorithms

The pruning procedure constitutes the first phase of the cup curriculum. In our experiments it consists of multiple pruning cycles. Each cycle starts by training the model for a set number of epochs with the possibility of employing early stopping. After training, a set number of weights is pruned according to their magnitude which is measured by the selected pruning criterion c. Depending on the selected training strategy, the remaining weights are rewound to an earlier state.

The data saved throughout Algorithm 1 is included as it is relevant for Algorithm 2.

Algorithm 1: Pruning Procedure

```
Data: neural network; training data; evaluation data; number of pruning cycles; percentage to
      prune; number of epochs; rewinding scheme; pruning version
Result: For each pruning cycle: mask, best and final state; After all pruning cycles: model state
model ← neural network;
list of best states ← [] ;                                  [] denotes an empty list
list of final states ← [];
list of masks ← [];
for cycle in number of pruning cycles do
    best performance ← worst possible performance;
    for epoch in number of epochs do
        state of model ← train model(training data);
        performance ← evaluate model(evaluation data) ;          evaluates the model
        if performance better than best performance then
            best performance ← performance;
            best state ← state of model;
        end
    end
    append best state to list of best states ;       best performing model is stored
    append state of model to list of final states ;     last reached model is stored
    for weight in weights of the model do
        w_c ← current value of the weight;
        w_i ← initial value of the weight;
        pruning criterion of weight ← ||w_c|| − ||w_i||;
    end
    model ← rewind weights(rewinding scheme) ;                rewinds the weights
    model ← prune weights(pruning criterion, percentage to prune, pruning version);
    mask ← positions remaining;
    append mask to list of masks;
end
Return model, list of best states, list of final states, list of masks;
```

The growth phase is the second phase of the cup curriculum. In our experiments it consists of multiple introduction steps. Each one starts by reintroducing capacity, i.e. a number of weights, to the model. The weight value of each reintroduced weight depends on the introduction scheme

chosen. After this introduction of capacity, the model is trained for a set number of epochs. The weight update during this training depends on the update scheme chosen.

Just like in Algorithm 1 we include data used by Algorithm 2 to improve clarity.

Algorithm 2: Growth Procedure

```
Data: training and evaluation data; model state, list of masks, list of best states and list of final
      states returned by pruning procedure (Algorithm 1); number of introduction steps; number
      of epochs; initialization scheme; update scheme;
Result: A list containing the best model per capacity
model ← model state;
reverse the lists received;
list of best states ← [] ;                                  [] denotes an empty list
for step in number of introduction steps do
    mask ← list of masks at position step;
    weights to introduce ← mask where the model has no active weight;
    introduce weights(initialization scheme, weights to introduce);
    best performance ← worst possible performance;
    for epoch in number of epochs do
        state of model ← train model(training data, update scheme);
        performance ← evaluate model(evaluation data) ;        evaluates the model
        if performance better than best performance then
            best performance ← performance;
            best state ← state of model;
        end
    end
    append best state to list of best states;
end
return list of best states
```

Appendix B. Details on Training Curves

Throughout the pruning phase of each considered training scheme regular spikes in training and validation loss can be observed (see Figs. 3 to 5). These spikes occur every 50 epochs and are due to weight rewinding and subsequent pruning. Their magnitude depends on the rewinding strategy used. *Initial* and *Warm* rewinding results in larger spikes, as their rewinding state is separated from the current state by more training epochs than the rewinding state of *Best* and *No* rewinding. Similar spikes are present in the growth phase due to the reintroduction of weights. However, here they are less pronounced.

Due to the usage of dropout throughout the training procedure the model capacity used to measure the training and validation loss differ (80% compared to 100%). This relative difference in model capacity is our explanation for the validation loss being substantially smaller than the training loss at smaller model capacities (see Figs. 3 to 5).

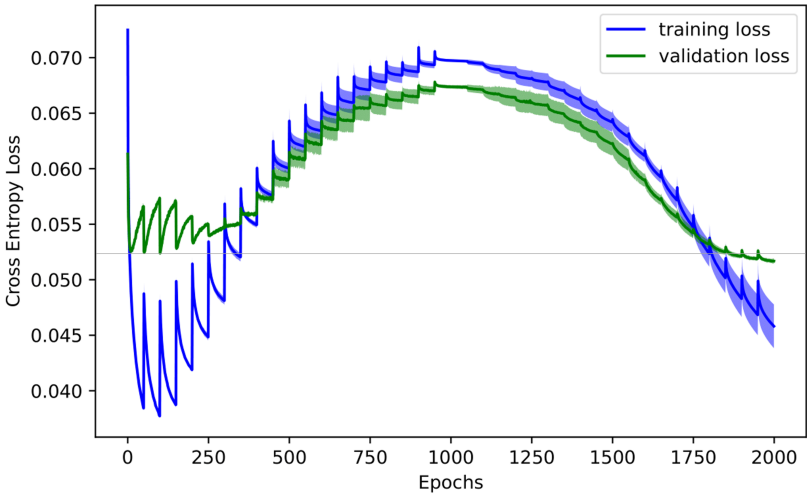

Figure 3: Number of training epochs vs. training and validation loss of the cup curriculum strategy using *Best* rewinding, *Random* initialisation, and *Identical* updating per seed. The grey line shows the best validation loss of the pruning phase.

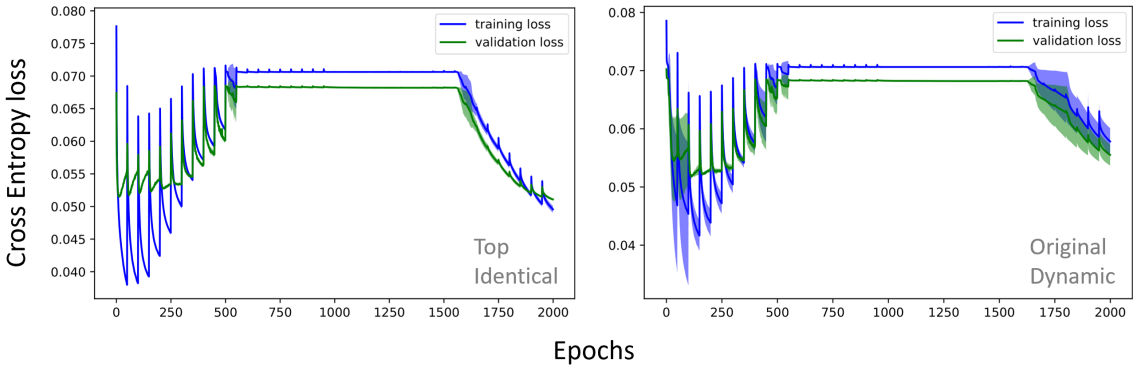

Figure 4: Exemplary training curves for the medium (left) and large (right) model.

For the *Small* model size all initialization schemes but the *Old* scheme show resilience to overfitting across multiple initialization and update scheme combinations (see Fig. 5). Our explanation for this circumstance is the unaddressed overfitting throughout the early pruning cycles of training schemes utilizing the *Old* scheme. For the *Small* model size the training and validation loss of strategies using the other initialization schemes in combination with the *Identical* or *Dynamic* update scheme are shaped like a bell. This differs when changing the model size. For the *Medium* and *Large* model sizes the loss plateaus after the 11-th pruning cycle and only decreases with the beginning of the 11-th and 12-th introduction step (see Fig. 4). All initialization schemes but the *No* scheme show resilience to overfitting across multiple rewinding and update scheme combinations.

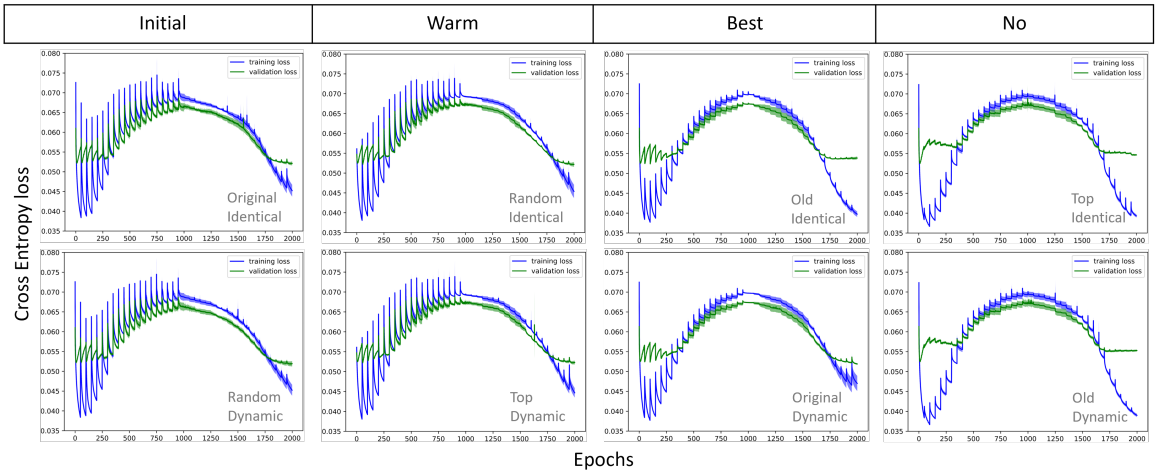

Figure 5: Overview over different training curves of different strategies of the cup curriculum on the small model.

Appendix C. Implementation Details

Here the remaining parameters mentioned in the paper are explained in more detail.

Table 3: Transformer architecture details by model variation.

Model	Attention Heads	Encoder Layers	Parameter Count
Small	2	2	13 829 280
Medium	4	4	14 313 280
Large	8	8	15 281 280

Table 4: All update schemes applied during the growth phase.

Update Scheme	Weight Update
Freezing	only last introduced weights are updated
Identical	same LR for all weights
Dynamic	LR depends on introduction time

Model Sizes We consider three model sizes in the experiments to explore the impact of parameter count, number of attention heads, number of encoder layers and dimensionality of attention heads on the training strategies considered. The *Small* model consists of 2 attention heads and 2 encoder layers. The *Medium* model consists of 4 attention heads and 4 encoder layers. Lastly, the *Large* model consists of 8 attention heads and 8 encoder layers. The dimension of the projection computed in each attention head scales inversely with the number of attention heads, therefore these model sizes also explore this effect on the training strategies considered. Table 3 lists the considered model sizes.

Update Schemes In the experiments we analyze the effect of three different update schemes. First, the *Freezing* update scheme is considered. It freezes all but the last introduced weights of the model. Additionally, we consider the usual weight update (*Identical*) which is plain backpropagation. Lastly, a weight update based on the individual weights age in the network (*Dynamic*) is considered. The *Dynamic* update scheme multiplies the weight update for all weights introduced in the n-th introduction step with f^n, where $f \in \mathbb{R}_0^+$ and weights which were not pruned during the pruning phase are considered to be from the 0-th introduction step. This extends the idea of freezing the update (by applying update masks with elements in $\{0; 1\}$), to update masks with elements in $\mathbb{R}_0^+$. The update schemes analyzed in the experiments are listed in Table 4.