GUIDE: GUIDED INITIALIZATION AND DISTILLATION OF EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Algorithmic efficiency techniques such as distillation (Hinton et al. (2014)) are useful in improving the model quality without increasing serving costs, provided a larger teacher model is available for a smaller student model to learn from during training. Standard distillation methods are limited to forcing the student to match only the teacher's outputs. Given the costs associated with training a large model, we believe that we should be extracting more useful information from a teacher model than just making the student match the teacher's outputs.

In this paper, we introduce GUIDE (Guided Initialization and Distillation of Embeddings). GUIDE can be considered an initialization technique (a special distillation technique) that forces the student to match the teacher in the parameter space. Using GUIDE we show a 25-26% reduction in the teacher-student quality gap when using large student models (400M - 1B parameters) trained on $\approx$ 20B tokens. We also present a detailed analysis that shows that GUIDE can be combined with knowledge distillation with near additive improvements. Furthermore, we show that applying GUIDE alone leads to substantially better model quality than applying knowledge distillation by itself.

Most importantly, GUIDE introduces no training or inference overhead, and hence any model quality gains from our method are virtually free.

1 INTRODUCTION

With the advent of Large Language Models (LLMs), a naive way to scale model performance is via increasing the number of model parameters (increasing width and/or depth), and/or training on more tokens. Better model quality achieved through such methods comes at the cost of increased model footprint metrics such as inference latency, memory, etc., and training costs.

Such a naively scaled up model might perform strongly on benchmarks, but could also have prohibitively large latency and/or memory requirements, which might make it infeasible to be trained or deployed at large scale. Therefore, it is of crucial to pay attention to *model efficiency* at the same time as model quality. Many recent works cover various aspects of model efficiency such as algorithmic efficiency techniques (Menghani (2023)), hardware support (Sze et al. (2017)), and best practices (Dehghani et al. (2022)).

For the purpose of this work, we are specifically interested in algorithmic efficiency techniques like Distillation Hinton et al. (2014), which helps a smaller *student* model learn from a larger *teacher* model during training, without having any significant impact on the model's footprint metrics (parameters, latency, memory, etc.). Since its introduction, distillation has found widespread use across models of various kinds (Kim et al. (2023); Sanh et al. (2019)).

In this paper, we introduce *Guided Initialization and Distillation of Embeddings*, or GUIDE, which is a novel approach to improve model quality without any impact on the model's footprint metrics. We know that standard knowledge distillation leads to progressive transfer of knowledge from the teacher to the student, by forcing the student to match the teacher's predictions. On the other hand, GUIDE is designed so that the student also utilizes the teacher model's parameters to have a strong initialization *guided* by the teacher model. We can view GUIDE as distillation in the parameter space, without the need for the teacher model to label a dataset.

2 Related Work

GUIDE is an algorithmic efficiency technique, as mentioned earlier, closely related to and combinable with distillation Hinton et al. (2014) and its variants Sanh et al. (2019). However, unlike distillation, GUIDE does not change the loss function and neither does it need the teacher model to label its training dataset.

A closely related work Xu et al. (2023) pitches a very similar idea on very small models (5M parameter student and 22M parameter teacher models). However, we show that their method does not scale to large models of size up to 1B parameters.

Transferring knowledge in the parametric space from the teacher to the student has been attempted in Sanh et al. (2019); Shleifer & Rush (2020) with positive results in LLMs. Unfortunately, in these works, the authors assume that both the teacher and the student share the same model's embedding dimension, which allows them to let the student copy whole chosen layers from the teacher. This requirement is too restrictive because very large teacher models often have a much larger embedding dimension than the student.

In Zhong et al. (2024), the authors introduce a sensitivity-based technique to extract and align parameters between teacher and student LLMs. However, their approach requires adding another LoRA module and training that module to select the teacher's relevant weights. Similarly, in Xia et al. (2024), the proposed pruning method tries to solve a constrained optimization problem, requiring us to learn pruning masks in order to decide which parameters should be kept from the teacher model. Lin et al. (2021) propose that a "Parameter Generator" can be trained to first predict the student's weights. Then we can continue finetuning the student model. However, this incurs an inference cost from the teacher while training the generator. Thus, all of these methods are quite expensive in practice. On the other hand, GUIDE does not require any new parameters or additional training.

There have also been attempts to use the weights of smaller students to initialize and train a larger teacher model Chen et al. (2022); Samragh et al. (2024). In this work, we only transfer knowledge from a trained teacher to a smaller student.

Frankle & Carbin (2019) shows that well-trained large models may contain *subnetworks* that can perform almost as well as the original larger models. They provide a recipe that can be followed to extract smaller subnetworks from simple fully connected networks and convolutional neural networks.

They also demonstrate that when trained from scratch, the subnetworks do not attain the same performance as when extracted from larger models. This aligns with our results, where we empirically confirm that models using GUIDE to parametrically distill from a larger teacher perform significantly better than initializing from scratch.

Another method Wortsman et al. (2022) proposes averaging the weights of multiple models with the same architecture, when fine-tuned on the same dataset and initializing from the same pre-training checkpoint (referred to as 'Model Soup'). This is akin to ensembling in the parameteric space.

Devvrit et al. (2024) proposes a method that can be used to train nested smaller models within a larger model. This can be useful when a new larger model is being trained froms scratch, but is applicable for leveraging an existing large model that was trained without this technique.

Other common efficiency techniques such as quantization Krishnamoorthi (2018); Jacob et al. (2018), sparsity Gale et al. (2019), architectural techniques Menghani et al. (2025), etc. are orthogonal to our work can be applied in conjunction with GUIDE without any adverse effects.

3 Guided Initialization and Distillation of Embeddings (GUIDE)

In this section, we introduce GUIDE, a simple and fast approach which directly transfers knowledge from a teacher model to the student model by using the pretrained teacher to initialize the student model's parameters.

3.1 PRELIMINARIES

In the standard distillation setting, we have a teacher model T and we want to distill T into a smaller student model S. Generally speaking, the teacher has more parameters and layers and performs better than the student. Here we assume that S and T are traditional transformers to be trained on the same dataset with the same context length. For ease of notation, we shall use S and T as subscripts of the relevant variables, tables, and weight matrices to indicate whether they are from the student or teacher model. Let d, n, h, ℓ, and f be the model dimension, the number of transformer blocks, the number of attention heads, the key/query/value dimension, and the MLP inner dimension, respectively. Recall that $d_S = h_S \ell_S$ and $d_T = h_T \ell_T$. In this work, we also assume that $d_S \leq d_T, h_S \leq h_T$, and $\ell_S \leq \ell_T$.

Next, let us introduce some notation needed to describe our method and briefly recap how transformer-like models work. Note that we only describe the relevant details here while skipping others (e.g. layer normalization, causal mask, etc.). The reader is referred to Vaswani et al. (2017) for a full description of the transformer architecture. Let $\boldsymbol{E}$ and $\boldsymbol{P}$ be the embedding table and the positional encoding table of size $m \times d$ where m is the vocabulary size. Given an input sequence of token indices $\boldsymbol{z}$ of length L (i.e., context length), the input embedding $\boldsymbol{X} \in \mathbb{R}^{L \times d}$ is computed by looking up tokens from $\boldsymbol{E}$ and adding the positional encodings from $\boldsymbol{P}$:

$$\boldsymbol{X}_i := \boldsymbol{E}_{\boldsymbol{z}_i} + \boldsymbol{P}_i.$$

Then $\boldsymbol{X}$ is passed through a sequence of n multi-head self-attention blocks where the output of one block is added to the original input and becomes the input of the next block. Each block has h attention heads and an MLP layer that work as follows. Slightly abusing notation, for each attention head $i \in \{1, \ldots, h\}$, define learnable projection matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{d \times \ell}$. We then project $\boldsymbol{X}$ onto these to get the query, key, and value matrices: $\boldsymbol{Q}_i = \boldsymbol{X}\boldsymbol{W}_i^Q, \boldsymbol{K}_i = \boldsymbol{X}\boldsymbol{W}_i^K, \boldsymbol{V}_i = \boldsymbol{X}\boldsymbol{W}_i^V$. The attention scores is computed as

$$\boldsymbol{A}_i := \text{softmax}\left(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^T}{\sqrt{\ell}}\right) \in \mathbb{R}^{L \times L}.$$

The scores are then used as weights to compute the contextual embeddings:

$$\boldsymbol{H}_i := \boldsymbol{A}_i \boldsymbol{V}_i \in \mathbb{R}^{L \times \ell}.$$

The head outputs are then concatenated and projected onto a learnable output projection matrix $\boldsymbol{W}^O \in \mathbb{R}^{d \times d}$:

$$\boldsymbol{O} := \text{Concat}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_h)\boldsymbol{W}^O \in \mathbb{R}^{L \times d}.$$

Finally, $\boldsymbol{O}$ is pushed through an MLP layer giving the block output:

$$\boldsymbol{B} := \phi(\boldsymbol{O}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)})\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} \in \mathbb{R}^{L \times d},$$

where $\boldsymbol{W}^{(1)} \in \mathbb{R}^{d \times f}, \boldsymbol{W}^{(2)} \in \mathbb{R}^{f \times d}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^f$, and $\boldsymbol{b}^{(2)} \in \mathbb{R}^d$ are learnable, ϕ is an activation function (e.g., GELU), and the $+$ operator is applied in row-wise manner.

3.2 OUR APPROACH

We now propose a simple and efficient method for the weight initialization of transformer models. The main idea here is to take $\boldsymbol{E}_S$ as the PCA compression of $\boldsymbol{E}_T$. By projecting $\boldsymbol{E}_T$ onto its PCA projection matrix $\boldsymbol{M} \in \mathbb{R}^{d_T \times d_S}$, we retain the most important patterns in d_S dimensions. Unlike previous pruning approaches that "approximate" the teacher's embedding table by picking some subset of the features, here we aim to extract as much information from it as possible.

More importantly, $\boldsymbol{M}$ can be used as a "bridge" connecting the student embedding space and the teacher embedding space. Note that multiplying any student's embedding by $\boldsymbol{M}^T$ would reconstruct the original teacher's embedding. Therefore, by projecting the input sequence $\boldsymbol{X}$ onto $\boldsymbol{M}^T$ before pushing it into the first block, the teacher model now works harmonically with the new smaller embedding table $\boldsymbol{E}_S$ as illustrated in Figure 1. The query, key and value matrices in this case are $\boldsymbol{Q}_0 = (\boldsymbol{X}\boldsymbol{M}^T)\boldsymbol{W}_0^Q, \boldsymbol{K}_0 = (\boldsymbol{X}\boldsymbol{M}^T)\boldsymbol{W}_0^K$, and $\boldsymbol{V}_0 = (\boldsymbol{X}\boldsymbol{M}^T)\boldsymbol{W}_0^V$. By the associativity of matrix multiplication, we can also absorb $\boldsymbol{M}^T$ into $\boldsymbol{W}_0^Q$, and $\boldsymbol{W}_0^K, \boldsymbol{W}_0^V$, transforming their shape

Algorithm 1 GUIDE(S, T) *# S, T represent the student and teacher models*

$\boldsymbol{E} \leftarrow \text{Concat}(\boldsymbol{E}_S, \boldsymbol{P}_S)$ *# concatenate in row-wise manner*
$\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V} \leftarrow \text{SVD}(\boldsymbol{E})$
$\boldsymbol{M} \leftarrow \boldsymbol{V}[:, : d_S]$ *# the PCA projection matrix, taking the most d_S important directions*
Initialize $\boldsymbol{E}_S \leftarrow \boldsymbol{E}_T \boldsymbol{M}$ and $\boldsymbol{P}_S \leftarrow \boldsymbol{P}_T \boldsymbol{M}$
Initialize Q, K, V projection matrices of the first block: $\boldsymbol{W}^Z_{S,0} \leftarrow \boldsymbol{M}^T \boldsymbol{W}^Z_{T,0}$ for each $Z \in \{Q, K, V\}$
Let D, D_h, H, and F be the outputs of GETEVENLYSPACEDINDICES when applying to pairs $(d_S, d_T), (\ell_S, \ell_T), (h_S, h_T)$, and (f_S, f_T) respectively.
$\boldsymbol{W}^Z_{S,0} \leftarrow \boldsymbol{W}^Z_{S,0}[:, D_h]$ for each $Z \in \{Q, K, V\}$
Reshape $\boldsymbol{W}^O_{T,0}$ to $h_T \times \ell_T \times d_T$
Initialize $\boldsymbol{W}^O_S \leftarrow \boldsymbol{W}^O_{T,0}[H, D_h, D]$
Reshape $\boldsymbol{W}^O_{S,0}$ back to $d_S \times d_S$
$\boldsymbol{W}^{(1)}_{S,0} \leftarrow \boldsymbol{W}^{(1)}_{T,0}[D, F], \boldsymbol{b}^{(1)}_{S,0} \leftarrow \boldsymbol{b}^{(1)}_{T,0}[F]$
$\boldsymbol{W}^{(2)}_{S,0} \leftarrow \boldsymbol{W}^{(2)}_{T,0}[F, D], \boldsymbol{b}^{(2)}_{S,0} \leftarrow \boldsymbol{b}^{(2)}_{T,0}[D]$

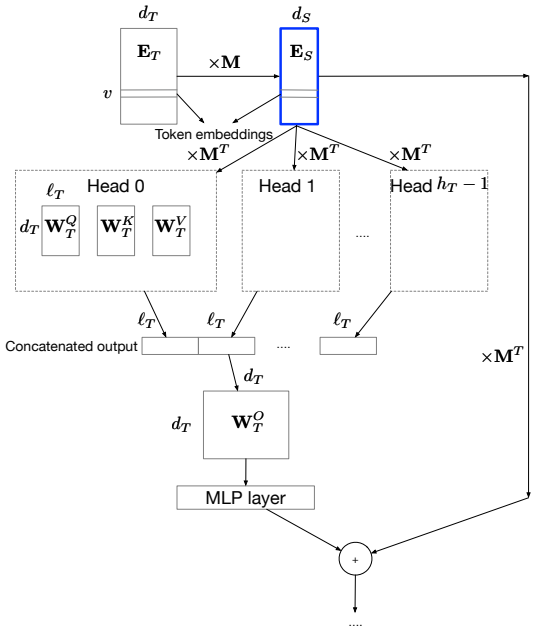

Figure 1: The teacher transformer can be "approximated" by using a PCA-compressed embedding table along with reconstructed embeddings.

to $d_S \times \ell_T$. For the rest of the block, we can now use Uniform Selection to reduce the remaining dimensions of ℓ_T, d_T, and f_T to ℓ_S, d_S, and f_S, respectively.

The details of GUIDE are shown in Algorithm 1.

Figure 2 illustrates how M^T is used to transform Q, K, V weight matrices, and evenly-spaced weights are selected from the remaining of the first transformer block by GUIDE.

4 Experimental Results

In this section, we experiment with GUIDE using traditional transformers of various sizes. In Section 4.1, we describe the setting of our experiments. The main experimental results of will be

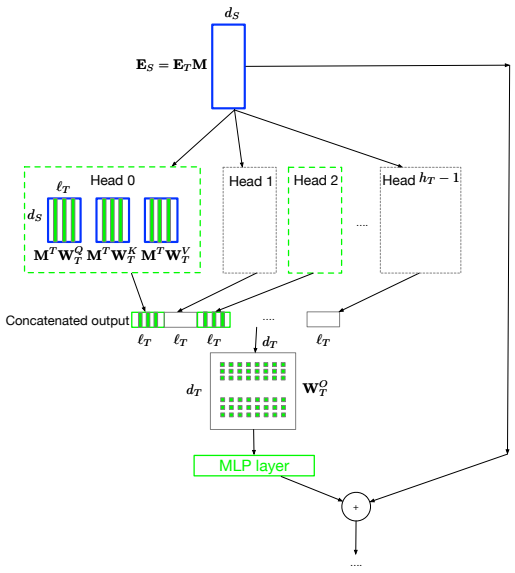

Figure 2: GUIDE uses M^T to transform Q, K, V matrices. Then Uniform Selection is used to take evenly-spaced green rows and columns to initialize the student's first block.

presented in Section 4.2. In Section A.2, we show that GUIDE can be combined with standard distillation, giving improved performance. Finally, we present the ablation study and the results when initializing intermediate transformer blocks in Section A.3.

4.1 Experimental Setup

In all of our experiments, we pre-train a decoder-only transformer on the public Colossal Cleaned Common Crawl (C4) dataset Raffel et al. (2020) using the NanoDO training framework by Liu et al. (2024). The configurations of the models presented in this section are shown in Table 1.

Table 1: Model configurations for the teacher and the student models in our experiments.

Type	**Model Size**	**Model Dims**	**Num Layers**	**Num Heads**	**Head Dims**	**FFN Dims**
Student	400M	960	23	30	32	5,760
Student	1B	1,728	25	36	48	6,912
Teacher	4.2B	3,072	36	48	64	12,288

All models have the same vocabulary size of 32,000, and a context length of 2048 tokens. It takes about 8.5 hours, 14 hours, and 3.5 days to train the 400M student, 1B student, and 4.2B teacher models, respectively, on the Google TPU Trillium (v6e) chips. Table 2 shows the training configurations of the above models. For training the students, we use the Adam optimizer with an initial learning rate that increases (linearly) from 0.0 to 0.0011 in 100 warmup steps, then decays (linearly) to 0.0 in the remaining steps.

Table 2: Training configurations for the student and teacher models.

Type	Model Size	Batch Size	Num TPU Chips	Training Steps
Student	400M	192	64	50,000
Student	1B	192	64	50,000
Teacher	4.2B	1,024	128	50,000

4.2 Main Results

We compare GUIDE with a student trained from scratch with random initialization and variants of Uniform Selection by Xu et al. (2023). Interestingly, using the "first-N selection" option for layer selection as suggested by Xu et al. (2023) performs quite poorly, causing the student to perform worse than using random initialization. Taking a few evenly-spaced layers (1 or 2) works better and is used as the main baseline in our study. The final perplexities of the student model are reported in Tables 3 and 4.

Table 3: Comparison between GUIDE, Uniform Select, and Random Initialization for the 400M parameter student model when using the 4.2B parameter teacher model.

Initialization Algorithm	Perplexity	Teacher-Student Gap Reduction (%)
Random Initialization	15.915±0.015	N/A
Uniform Select. (first-N)	15.967±0.015	-0.82
Uniform Select. (1 layer)	14.458±0.013	23.15
Uniform Select. (2 layers)	14.498±0.013	22.51
GUIDE	**14.245±0.013**	**26.53**
Teacher	9.621±0.004	N/A

Table 4: Comparison between GUIDE, Uniform Select, and Random Initialization for the 1B parameter student model when using the 4.2B parameter teacher model.

Initialization Algorithm	Perplexity	Teacher-Student Gap Reduction (%)
Random Initialization	13.382±0.012	N/A
Uniform Select. (first-N)	14.088±0.013	-18.77
Uniform Select. (1 layer)	12.459±0.011	24.54
Uniform Select. (2 layers)	12.491±0.011	23.70
GUIDE	**12.438±0.011**	**25.11**
Teacher	9.621±0.004	N/A

We plot the perplexity of the student models on the training trajectory in Figures 3 and 4. In both cases, GUIDE consistently outperforms existing approaches throughout the training process. In summary, GUIDE can reduce the teacher-student perplexity gap by 26.52% and 25.11% for the 400M and 1B models, respectively.

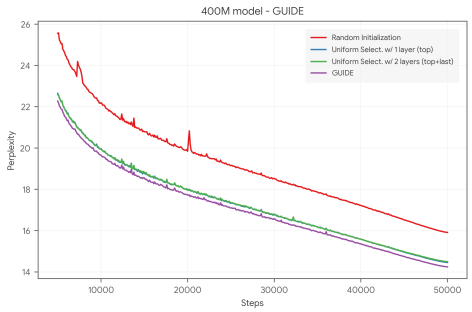

Figure 3: Training trajectory of the 400M parameter student model with GUIDE and other initialization methods. x-axis is the step number, and y-axis is the perplexity on the evaluation dataset.

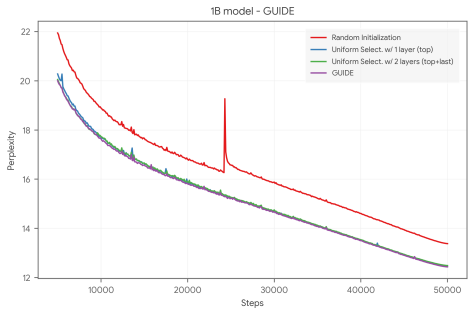

Figure 4: Training trajectory of the 1B parameter student model with GUIDE and other initialization methods. x-axis is the step number, and y-axis is the perplexity on the evaluation dataset.

Observe that there are several "loss spikes" during the training trajectory of the non-GUIDE models in Figures 3, 4, 5, and 6. In fact, this phenomenon has been reported to negatively affect the training of large models Chowdhery et al. (2023). Notably, GUIDE does not suffer from any big loss spike, demonstrating the stability of our approach.

For analysis of GUIDE, we also compute the explained variance ratio of the newly initialized embedding table (i.e., the ratio between the variance of columns of the new embedding table and the total variance of the original teacher's table) when applying GUIDE and Uniform Select. For the 400M student, the explained variance ratios of Uniform Select and GUIDE are ≈ 0.29 and ≈ 0.64, respectively. For the 1B student, the explained variance ratios of Uniform Select and GUIDE are ≈ 0.58 and ≈ 0.82, respectively. The relatively large gap indicates that the columns of the teacher's embedding table are somewhat correlated, and PCA projection was able to successfully combine them into efficient "super-features" while Uniform Select is just throwing away valuable unique data.

5 Discussion

In this section, we will summarize our results and observations from the previous section. We started with describing the model configuration and the training setup. Our choice of student and teacher model sizes is two orders of magnitude larger than the closest related work of Xu et al. (2023). The student models were trained for $\approx$ 20B tokens and the teacher model was trained with $\approx$ 100B tokens.

As mentioned in Section 4.2, the baseline approach of 'Uniform Selection' and matching the 'first-N' layers of the teacher model with the student model as described Xu et al. (2023) performs poorly

(where N is the number of student layers). We suspect that since the authors report the results of their method on student and teacher models of sizes 5M and 22M parameters, it is possible that their initialization scheme does not transfer well to the much larger models such as LLMs of today. In fact, we find that choosing a smaller number of layers to match by evenly spacing them in the teacher model works better, and we use that as an 'enhanced' baseline in our experiments.

In Tables 3 and 4 we see that GUIDE gives a 26.53% and 25.11% reduction teacher-student quality gap for the 400M and 1B param baselines, respectively. It significantly outperforms our 'enhanced' baselines of Uniform Select with 1 and 2 layers.

In Section A.2, we explored combining GUIDE with standard knowledge distillation. Interestingly, GUIDE leads to additive gains on top of knowledge distillation, while also getting better model quality by itself than when knowledge distillation. The unique gains provide an interesting opportunity for those training models from scratch to leverage the teacher's knowledge at initialization time as well using GUIDE, apart from learning from it during training with standard KD.

Finally, we note that we get the best results when initializing the embedding table and the first / top layer from the teacher, as seen in Tables 7 and 8. Matching more than 1 layer seems to lead to degradation in performance (except when matching 3 layers), suggesting that there is room for improvement in the method.

To summarize, given that GUIDE has no training and inference overhead, when a larger pre-trained model is present, applying GUIDE on a student model's embedding table and first layer is likely to improve model performance substantially. Combining GUIDE with knowledge distillation would help the student learn from the teacher both in the parametric space and the output space.

6 Conclusion

In this paper, we introduce GUIDE, which is a novel algorithmic efficiency technique that helps improve model quality without any impact to the model size or latency.

We ran a host of experiments to demonstrate the efficacy of our method, and show that not only does GUIDE beat the baselines from existing literature, it also beats our 'enhanced' versions of those baselines. GUIDE also combines very well with vanilla knowledge distillation method where we see that the quality gains from GUIDE are nearly additive on top of those from distillation. This strong result implies that GUIDE brings model quality gains that are distinct from the standard knowledge distillation approach. Furthermore, applying GUIDE alone led to better model quality than applying knowledge distillation alone.

Therefore, when a large pre-trained models is available, using GUIDE to initialize smaller models before training is likely to provide virtually free improvements to model quality. In terms of future work, we aim to further improve GUIDE by developing novel variants that help us extract even more information from the teacher models during initialization and training.

REFERENCES

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2BERT: Towards reusable pretrained language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2134–2148, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.151. URL https://aclanthology.org/2022.acl-long.151/.

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, Sashank 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. *J. Mach. Learn. Res.*, 24(1), January 2023. ISSN 1532-4435.

Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. The efficiency misnomer. In *ICLR*, 2022.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference, 2024. URL https://arxiv.org/abs/2310.07707.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv*, 1902.09574, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Deep Learning Workshop (NeurIPS)*, 2014.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pp. 2704–2713, 2018.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372. URL https://aclanthology.org/2020.findings-emnlp.372/.

Seungyeon Kim, Ankit Singh Rawat, Manzil Zaheer, Sadeep Jayasumana, Veeranjaneyulu Sadhanala, Wittawat Jitkrittum, Aditya Krishna Menon, Rob Fergus, and Sanjiv Kumar. Embeddistill: A geometric knowledge distillation for information retrieval, 2023. URL https://arxiv.org/abs/2301.12005.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv*, 1806.08342, 2018.

Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. distillation: Transferring the knowledge in neural network parameters. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural*

Language Processing (Volume 1: Long Papers), pp. 2076–2088, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.162. URL https://aclanthology.org/2021.acl-long.162/.

Peter J. Liu, Roman Novak, Jaehoon Lee, Mitchell Wortsman, Lechao Xiao, Katie Everett, Alexander A. Alemi, Mark Kurzeja, Pierre Marcenac, Izzeddin Gur, Simon Kornblith, Kelvin Xu, Gamaleldin Elsayed, Ian Fischer, Jeffrey Pennington, Ben Adlam, and Jascha-Sohl Dickstein. Nanodo: A minimal transformer decoder-only language model implementation in JAX., 2024. URL http://github.com/google-deepmind/nanodo.

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 2023.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. Laurel: Learned augmented residual layer, 2025. URL https://arxiv.org/abs/2411.07501.

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. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. arxiv 2014. *arXiv preprint arXiv:1412.6550*, 2014.

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. Scaling smart: Accelerating large language model pre-training with small model initialization. In *ENLSP*, 2024. URL https://api.semanticscholar.org/CorpusID:272753343.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. URL https://arxiv.org/abs/1910.01108.

Sam Shleifer and Alexander M. Rush. Pre-trained summarization distillation. *CoRR*, abs/2010.13002, 2020. URL https://arxiv.org/abs/2010.13002.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE*, 105(12):2295–2329, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL https://arxiv.org/abs/2203.05482.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=09iOdaeOzp.

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones, 2023. URL https://arxiv.org/abs/2311.18823.

Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. Seeking neural nuggets: Knowledge transfer in large language models from a parametric perspective. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mIEHIcHGOo.

A APPENDIX

A.1 INITIAL ATTEMPTS

In this study, we explore new ways to improve standard knowledge distillation. Can we extract even more useful information from the teacher to help train a better student? A popular approach, in the same spirit as the original distillation work of Romero et al. (2014), would be to exploit the teacher's final token representations (that is, the output of the last block used to compute the logits) $\boldsymbol{B}_{T,h_T}$. In fact, matching the student's and teacher's token embeddings has yielded positive results in Jiao et al. (2020); Kim et al. (2023). Here we encounter the first challenge: the dimensional mismatch between the teacher and student. A common approach to this problem is to introduce another learnable projection matrix $\boldsymbol{M} \in \mathbb{R}^{d_T \times d_S}$ and optimize the following additional MSE loss:

$$\text{loss}_{\text{embedding}} := \text{MSE}(\boldsymbol{B}_{S,n_S}, \boldsymbol{B}_{T,n_T} \times \boldsymbol{M}),$$

where the parameters in $\boldsymbol{B}_{T,n_T}$ are frozen Romero et al. (2014). Unfortunately, this approach not only incurs the teacher's inference cost in the training proccess, but also does not seem to work consistently when pretraining decoder-only LLMs. In our experiments, adding this embedding loss actually hurts the student's performance.

We want to directly transfer knowledge from the teacher to the student via an "almost-free" weight initialization method. Basically, the weights of the pretrained teacher can be used as a good starting point for the student. As mentioned in Section 2, notable works on this subject include Xu et al. (2023); Shleifer & Rush (2020); Sanh et al. (2019). The weight initialization methods consist of two main components: layer selection and weight selection. The former is a strategy to select teacher's layers to be matched to student's layers. For a given pair of teacher's and student's layers, the latter helps select a subset of teacher's weights to be used as initial weights for the student's layer.

A.1.1 UNIFORM SELECTION BY XU ET AL. (2023)

For isotropic architectures (e.g., transformer models), Xu et al. (2023) suggests the so-called "first-N selection" for layer selection. Basically, the first N layers of the teacher will be used as the initialization source. The authors also mention taking evenly-spaced layers in teacher as an alternative. Note that this approach has also been tried in Shleifer & Rush (2020). For weight selection, they propose the "Uniform Selection" strategy as follows. Suppose that we want to initialize the student's weight tensor $\boldsymbol{W}_S$ with shape $s_1 \times s_2 \times \ldots s_n$ from $\boldsymbol{W}_T$ with shape $t_1 \times t_2 \times \ldots \times t_n$ where the indices are 0-indexed.

Algorithm 2 GETEVENLYSPACEDINDICES(m, n) *Note: m, n are integers and $n \geq m > 1$* and the function returns m evenly-spaced integers from $[0, n-1]$.

1: $I \leftarrow \left(0, \frac{m-1}{n-1}, \frac{2(m-1)}{n-1}, \ldots, \frac{(m-1)^2}{n-1}\right)$
2: Round numbers in I to the closest integer (half-integers are rounded down)
3: Return I

To construct $\boldsymbol{W}_S$, for each dimension $i \in [1, n]$, we select indices returned by GETEVENLYSPACEDINDICES(s_i, t_i) along the i-th dimension of $\boldsymbol{W}_T$. While this procedure is simple and can be applied to different model architectures, it is agnostic to how transformers actually work.

To the best of our knowledge, none of the current works take advantage of the specific structure of transformers in weight initialization. Here we shall try to extract even more information from the teacher's embedding table and the first transformer layer, which directly takes the token embeddings from the table as input.

A.1.2 LOW-RANK APPROXIMATION TO PRESERVE PAIRWISE DOT-PRODUCTS

Let us start with the embedding table of the teacher $\boldsymbol{E}_T$. One natural idea would be to replace the student's embedding table by $\boldsymbol{E}_T\boldsymbol{M}$ where $\boldsymbol{E}_T$ is frozen and $\boldsymbol{M} \in \mathbb{R}^{d_T \times d_S}$ is learnable as above. The drawback of this approach is the extra parameters in the projection $\boldsymbol{M}$. Moreover, learning this projection does not seem to yield significant improvements in practice. Thus, we want to directly

initialize $\boldsymbol{E}_S$ by using $\boldsymbol{E}_T$. One idea would be to find $\boldsymbol{E}_S$ such that the pairwise dot products between tokens in the dictionary are preserved:

$$\boldsymbol{E}_S \boldsymbol{E}_S^T \approx \boldsymbol{E}_T \boldsymbol{E}_T^T,$$

which is a low-rank approximation problem. Let $\boldsymbol{D} := \boldsymbol{E}_T \boldsymbol{E}_T^T$. Observe that $\boldsymbol{D}$ is symmetric positive semidefinite. So we can find the spectral eigendecomposition of $\boldsymbol{D}$:

$$\boldsymbol{D} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T,$$

where $\boldsymbol{U}$ is orthogonal and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_L)$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L \geq 0$. Let $\boldsymbol{U}_{d_S} := \boldsymbol{U}_{:,:d_S}$ and $\boldsymbol{\Lambda}_{d_S} := \text{diag}(\lambda_1, \ldots, \lambda_{d_S})$. Then according to the Eckart–Young–Mirsky theorem, taking

$$\boldsymbol{E}_S := \boldsymbol{U}_{d_S} \boldsymbol{\Lambda}_{d_S}^{1/2}$$

gives the best solution among all matrices of rank at most d_S to minimize $\|\boldsymbol{E}_S \boldsymbol{E}_S^T - \boldsymbol{D}\|_{\text{F}}$.

Note that the above low-rank approximation method is a parametric knowledge transfer that does not require the teacher to participate in the student training. However, it is not trivial to apply the same approach to the transformer blocks and/or to make these blocks work in sync with the embedding table constructed this way. For example, suppose that we want to extract knowledge from some attention head i of a teacher's block. One straightforward idea is to let the student learn the teacher's attention scores $\boldsymbol{A}_{T,i}$. However, these scores depend not only on parametric projections $\boldsymbol{W}_{T,i}^Q$ and $\boldsymbol{W}_{T,i}^K$, but also on the teacher's input embeddings $\boldsymbol{X}$. Traditionally, this would require the use of an additional loss, for example, $\|\boldsymbol{A}_{S,i} - \boldsymbol{A}_{T,i}\|_2$. Then we still have to deal with $\boldsymbol{W}_{T,i}^V$, $\boldsymbol{W}_{T,i}^O$, and the teacher's MLP layer.

A.2 Combining with Knowledge Distillation

Here we show that applying GUIDE before using Knowledge Distillation (KD) will help boost the quality of the student significantly. In our distillation experiments, we use the same 4B parameter teacher model for both, GUIDE and knowledge distillation. For KD, we load the teacher into memory for inference and freeze its parameters. The teacher is then used to generate its per-token distributions that the student model is forced to match. More specifically, we minimize the following loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \alpha \mathcal{L}_{\text{distill}},$$

where $\mathcal{L}_{\text{pred}}$ is the usual next-token prediction loss and $\mathcal{L}_{\text{distill}}$ is the distillation loss, which is defined as the cross entropy between the per-token distributions returned by the student and the teacher: $\mathcal{L}_{\text{distill}} := \mathbf{CE}(y_T, y_S)$. Note that this distillation loss should not be confused with the embedding loss, which is the MSE between the student's and teacher's final token representations, as mentioned in Section A.1. We set the distillation weight $\alpha := 0.5$ in our experiments.

The performance of the students when using GUIDE + KD is reported in Tables 5 and 6. GUIDE itself is capable of transferring more information from the teacher to the student than KD alone. Moreover, the gap reduction of the combined GUIDE + KD setup is almost equal to the sum gap reductions of using the two methods separately. This suggests that GUIDE and KD work synergistically, with each bringing distinct improvements.

Table 5: Combining GUIDE and Knowledge Distillation for the 400M parameter student model.

Model	**Perplexity**	**Teacher-Student Gap Reduction (%)**
KD	15.154±0.014	12.10
Uni. Select. (1 layer)	14.458±0.013	23.15
GUIDE	14.246±0.013	26.53
Uni. Select. (1 layer) + KD	13.804±0.013	33.55
GUIDE + KD	**13.662±0.012**	**35.80**

Table 6: Combining GUIDE and Knowledge Distillation for the 1B parameter student model.

Model	**Perplexity**	**Teacher-Student Gap Reduction (%)**
KD	12.636±0.011	19.86
Uni. Select. (1 layer)	12.459±0.011	24.54
GUIDE	12.438±0.011	25.11
Uni. Select. (1 layer) + KD	11.825±0.010	41.42
GUIDE + KD	**11.813±0.010**	**41.73**

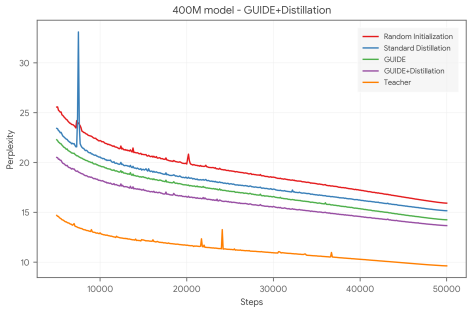

Figure 5: Combining Knowledge Distillation with GUIDE when training 400M model.

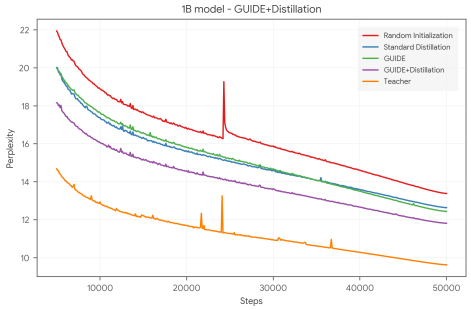

Figure 6: Combining Knowledge Distillation with GUIDE when training 1B model.

A.3 Ablation Study and Initializing Intermediate Layers

Our most interesting observation here is that initializing the first layer is crucial. This will significantly improve the student's performance compared to initializing the student's embedding table alone. However, initializing more intermediate layers after this point would not help much in general. Furthermore, our results show that the "first-N" strategy suggested by Xu et al. (2023) for layer selection performs poorly on standard transformers. The complete results for the 400M and 1B models are presented in Tables 7 and 8.

Table 7: Performance of GUIDE with different strategies for layer selection on the 400M parameter student model. We find that applying GUIDE on the embedding table and the top most layer gives the best results.

GUIDE	**Perplexity**	**Teacher-Student Gap Reduction (%)**
Embed. Table only	14.453±0.013	23.24
Embed. Table +		
1 layer (top)	**14.246±0.013**	**26.53**
2 layers (top+last)	14.317±0.013	25.40
3 layers	14.268±0.013	26.18
4 layers	14.356±0.013	24.79
8 layers	14.509±0.013	22.35
first-N layers	15.501±0.014	6.58

Table 8: Performance of GUIDE with different strategies for layer selection on the 1B parameter student model. Consistent with Table 7 we find that applying GUIDE on the embedding table and the top most layer gives the best results.

GUIDE	**Perplexity**	**Teacher-Student Gap Reduction (%)**
Embed. Table only	12.866±0.012	13.73
Embed. Table +		
1 layer (top)	**12.438±0.011**	**25.12**
2 layers (top+last)	12.612±0.011	20.50
3 layers	12.505±0.011	23.34
4 layers	12.609±0.011	20.58
8 layers	12.630±0.011	20.01
first-N layers	14.037±0.013	-17.40