Kronecker Decomposition for GPT Compression

Ali Edalati
McGill University
ali.edalati@mail.mcgill.ca

Marzieh Tahaei
Huawei Noah Ark Lab
marzieh.tahaei@huawei.com

Ahmad Rashid
Huawei Noah Ark Lab
ahmad.rashid@huawei.com

Vahid Partovi Nia
Huawei Noah Ark Lab
vahid.partovinia@huawei.com

James J. Clark
McGill University
james.clark1@mcgill.ca

Mehdi Rezagholizadeh
Huawei Noah Ark Lab
mehdi.rezagholizadeh@huawei.com

Abstract

GPT is an auto-regressive Transformer-based pre-trained language model which has attracted a lot of attention in the natural language processing (NLP) domain. The success of GPT is mostly attributed to its pre-training on huge amount of data and its large number of parameters. Despite the superior performance of GPT, this overparameterized nature of GPT can be very prohibitive for deploying this model on devices with limited computational power or memory. This problem can be mitigated using model compression techniques; however, compressing GPT models has not been investigated much in the literature. In this work, we use Kronecker decomposition to compress the linear mappings of the GPT-2 model. Our Kronecker GPT-2 model (KnGPT2) is initialized based on the Kronecker decomposed version of the GPT-2 model and then is undergone a very light pre-training on only a small portion of the training data with intermediate layer knowledge distillation (ILKD). Finally, our KnGPT2 is fine-tuned on downstream tasks using ILKD as well. We evaluate our model on both language modeling and General Language Understanding Evaluation benchmark tasks and show that with more efficient pre-training and similar number of parameters, our KnGPT2 outperforms the existing DistilGPT2 model significantly.

1 Introduction

Recently, development and deployment of pre-trained language models (PLMs) has improved the performance of NLP models significantly (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; Shoeybi et al., 2019; Radford et al., 2019). PLMs are mostly Transformer-based models, which are pre-trained on enormous unlabeled data. Although Transformer-based PLMs are powerful in performance, their huge size is a barrier for efficient training or inference of these models on lower capacity devices with memory, computation and energy constraints. Therefore, there has been a growing volume of literature focused on developing frameworks for compressing these large PLMs.

Like other deep learning models, the main directions of model compression for PLMs are using following methods in isolation or combination: low-bit quantization (Gong et al., 2014; Prato et al., 2019), pruning (Han et al., 2015), knowledge distillation (KD) (Hinton et al., 2015) and matrix decomposition (Yu et al., 2017; Lioutas et al., 2020).

PLMs can be divided into encoder-based and auto-regressive models such as the BERT (Devlin et al., 2018; Liu et al., 2019) and GPT (Brown et al., 2020) family respectively. Although the size of BERT family models is usually smaller than the GPT family, compressing the BERT family has been investigated much more in the literature (e.g. DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2019), MobileBERT (Sun et al., 2020), ALP-KD (Passban et al., 2021), MATE-KD (Rashid et al., 2021), Annealing-KD (Jafari et al., 2021) and BERTQuant (Zhang et al., 2020)). On the other hand, to the best of our knowledge, the GPT family has barely a handful of compressed models (Li et al., 2021), among them the DistilGPT2[1] model has attracted wide attention in the literature. The DistilGPT2 model is heavily pre-trained for 3 epochs on the large OpenWebText dataset[2]. Moreover, it is evident in the literature that the GPT model cannot compete with BERT on natural language understanding (NLU) tasks (Liu et al., 2021). Therefore, developing an efficient compressed GPT model with comparable NLU performance is still an open problem.

In this paper, we use Kronecker decomposition, which has been recently used for BERT compression (Tahaei et al., 2021), for compression of the GPT-2 model (we refer to our model as KnGPT2 in this paper). We use Kronecker decomposition to represent the weight matrices of linear layers in

[1] https://transformer.huggingface.co/model/distil-gpt2
[2] https://huggingface.co/datasets/openwebtext

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 219 - 226

GPT-2 by smaller matrices which can reduce the size and computation overhead. We use Kronecker decomposition to compress the embedding and Transformer layers of GPT-2. For Transformer layers, the linear layers of multi-head attention (MHA) and the feed-forward network (FFN) blocks of Transformer layers are replaced with their Kronecker decomposition.

Kronecker decomposition leads to reduction in expressiveness of the model. We use a very light pre-training with intermediate layer knowledge distillation (ILKD) to address this issue, which improves the performance of the compressed model significantly. It is worth mentioning that for our pre-training, we use $1/10^{\text{th}}$ of the DistilGPT2's pre-training data (i.e. OpenWebText) only for 1 epoch (instead of 3 epochs in DistilGPT2). Furthermore, in this paper, our framework is applied to GPT-2 but it can be easily exploited to compress other models as well. To summarize contributions of this paper, we mention the following points:

- To the best of our knowledge, we are the first work which uses Kronecker decomposition for compression of the GPT model.

- Our KnGPT2 model, which is evaluated on both language modeling and GLUE benchmark tasks, improves training efficiency and outperforms DistilGPT2 significantly.

2 Related Works

(Zhou and Wu, 2015) is the first work that used summation of multiple Kronecker products to compress the weight matrices in fully-connected networks and small convolutional neural networks. (Thakker et al., 2019) proposed a hybrid method which separates the weight matrices into an upper and a lower part, upper part remains untouched but the lower part decomposes to Kronecker products. They used this approach for small language models to be utilized on internet of things (IoT) applications. Recently, (Thakker et al., 2020) extended the mentioned hybrid method to non-IoT applications by adding a sparse matrix to the Kronecker products. (Tahaei et al., 2021) has deployed a similar approach to ours to compress BERT which achieved promising results but to the best of our knowledge, this work is the first attempt for GPT compression using Kronecker decomposition.

DistilGPT2 [3] is one of the most successful and well-known compressed versions of GPT-2 which is considered as a baseline in this paper. DistilGPT2 has 82M parameters compared to 124M parameters for GPT-2_{Small} and is trained using KD on OpenWebTextCorpus which is a reproduction of OpenAI's WebText dataset.

3 Methodology

3.1 Kronecker Product

The Kronecker product is a matrix operation (denoted by $\otimes$) which takes two matrices as input and generates a block matrix as output. Assume that $\mathbf{A}$ is a matrix $\in \mathbf{R}^{m_1 \times n_1}$ and $\mathbf{B}$ is a matrix $\in \mathbf{R}^{m_2 \times n_2}$, $\mathbf{A} \otimes \mathbf{B}$ is equal to a block matrix $\in \mathbf{R}^{m \times n}$, where $m = m_1 m_2$, $n = n_1 n_2$ and each block (i, j) is obtained by multiplying element a_{ij} by matrix .

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B}, \end{bmatrix} \tag{1}$$

3.2 GPT-2 Compression using Kronecker Factorization

We can represent a weight matrix, $\mathbf{W} \in \mathbf{R}^{m \times n}$, by two smaller matrices, $\mathbf{A} \in \mathbf{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbf{R}^{m_2 \times n_2}$ such that $\mathbf{W} = \mathbf{A} \otimes \mathbf{B}$ and $m = m_1 m_2$, $n = n_1 n_2$. This leads to reduction in the number of parameters from mn for the original matrix to $m_1 n_1 + m_2 n_2$ for the Kronecker factorized version. In large language models, embedding layer usually takes a large portion of the memory. Let $\mathbf{W}^E \in \mathbf{R}^{v \times d}$ be the lookup table for the input embedding where v is the vocabulary size and d is the embedding dimension. To compress the embedding layer using Kronecker decomposition we use the same method as in (Tahaei et al., 2021). We define $\mathbf{A}^E \in \mathbf{R}^{v \times d/f}$ and $\mathbf{B}^E \in \mathbf{R}^{1 \times f}$, where f is a factor of d. There are two reasons for this decision: first, similar to $\mathbf{W}^E$, in the $\mathbf{A}^E$ matrix every row will indicate embedding of a single word. Second, the embedding of each word, E_i, can be obtained by $\mathbf{A}_i^E \otimes \mathbf{B}$, therefore the computation complexity of this operation is $\mathcal{O}(d)$ which is very efficient.

The transformer architecture is composed of N identical layers each having MHA followed by FFN. In the MHA module, there are linear layers which calculate the Query, Key and Value by

[3]For further details, see https://huggingface.co/distilgpt2

multiplying the input vector by $\mathbf{W}^{Q}, ^{K}, ^{V}$, respectively. Also, in the FFN module, there are two fully connected layers that can be represented as $\mathbf{W}^{c_{\text{fc}}}$ and $\mathbf{W}^{c_{\text{proj}}}$. In this work, all of the mentioned weight matrices at different heads and layers of the transformer are decomposed into Kronecker factors.

For initialization, similar to (Tahaei et al., 2021), the Kronecker factors $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are estimated from the corresponding weight matrix $\mathbf{W}$ in the original uncompressed pre-trained model using the solution to the nearest Kronecker problem

$$(\hat{\mathbf{A}}, \hat{\mathbf{B}}) = \arg \min_{(\mathbf{A}, \mathbf{B})} \|\mathbf{W} - \mathbf{A} \otimes \mathbf{B}\|^2$$

The solution to this optimization can be found by rank-1 singular value decomposition (SVD) approximation of the reshaped , see (Van Loan, 2000) for details.

3.3 Knowledge Distillation

In this section, the KD method used for both pre-training and fine-tuning the KnGPT2 is explained.

Let T and S represent the teacher model, GPT-2, and the student model, KnGPT2, respectively. For a batch of data $(\mathbf{x}, \mathbf{y})$, Att^{S}_{last} and Att^{T}_{last} are the attention distributions of the last transformer layer, obtained by applying softmax on the scaled dot product between query and key (For more details, see (Wang et al., 2020). H^{S}_{l} and H^{T}_{l} are the normalized hidden state outputs of the layer l. In our experiments, the output of the embedding layer is considered as a hidden state. Therefore, l represents both transformer layers and embedding layer. Note that by using the Kronecker factorization, like other decomposition methods, the number of layers and dimensions of the output matrices in the student model remain intact so we can directly obtain the difference of output of a specific layer in student an teacher model without the need for projection.

For the MHA modules, similar to (Wang et al., 2020), we use Kullback–Leibler divergence (KL) between the attention distributions of the last transformer layers of the student and the teacher.

$$L_{\text{Attention}}(x) = \text{KL}\{Att^{S}_{last}(x), Att^{T}_{last}(x)\} \quad (2)$$

For the FFN modules and embedding layer, we simply use the MSE between the output hidden states of embedding and transformer layers in the student and teacher:

$$L_{\text{Hidden States}}(x) = \frac{1}{L} \sum_{l} \text{MSE}\{H^{S}_{l}(x), H^{T}_{l}(x)\} \quad (3)$$

Where L is number of hidden states (number of transformer layers plus one for embedding).

The final loss is calculated by a linear combination of the above losses as well as the cross entropy loss.

$$\text{Loss}(x, y) = \sum_{(x,y)} \alpha_1 L_{\text{Attention}}(x) + \alpha_2 L_{\text{Hidden States}}(x) + \alpha_3 L_{\text{Cross Entropy}}(x, y) \quad (4)$$

After decomposing the teacher model, GPT-2, into KnGPT2, the performance of the model drops significantly. This drop is mainly because of the approximation of linear weight matrices using the corresponding Kronecker factors. Therefore, pre-training of the compressed model on a small corpus for a few epochs is necessary to retrieve the information which are lost during decomposition. Inspired by (Jiao et al., 2019), we pre-trained the model on a small portion, 10%, of the OpenWebText dataset (Gokaslan and Cohen, 2019) for one epoch and we used the KD method which is discussed in Section 3.3 to improve the performance of the compressed model.

4 Experiments

We evaluated our proposed algorithm, KnGPT2, on language modeling and text classification. For language modeling we use the Wikitext-103 (Merity et al.) dataset.For classification we use seven of the classification tasks of the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). These datasets can be broadly divided into 3 families of problems. Single set tasks which include linguistic acceptability (CoLA) and sentiment analysis (SST-2), similarity and paraphrasing tasks (MRPC and QQP), and inference tasks which include Natural Language Inference (MNLI and RTE) and Question Answering (QNLI).

4.1 Experimental Setup

The KnGPT2 model is compressed from the GPT-2$_{\text{Small}}$ (Radford et al., 2019) model. GPT-2$_{\text{Small}}$ has 124 million parameters. Our baseline is DistilGPT2 which has about 82 million parameters so our KnGPT2 model is compressed to the same size (83 million parameters) for a fair

	GPT-2$_{\text{Small}}$	DistilGPT2	KnGPT2
Parameters*	124	82	83
Training time (hrs)	-	>90[4]	6.5
Dataset size (GB)	40	38	3.2

Table 1: Training details for GPT-2 compression. Note that number of parameters of the models are reported excluding the output embedding layer in language modelling which is not compressed, is equal to row Parameters*

Model	CoLA	RTE	MRPC	SST-2	MNLI	QNLI	QQP	Average
GPT-2$_{\text{Small}}$	44.0	63.2	84.5	92.8	81.75	88.7	88.0	77.56
DistilGPT2	32.4	61.9	84.3	90.8	79.55	85.4	87.3	74.52
DistilGPT2 + KD	33	61.5	84.4	90.7	79.85	85.7	87.6	74.67
KnGPT2	36.7	**64.4**	84.5	89.0	78.45	85.6	86.5	75.02
KnGPT2 + ILKD	**41.8**	63.7	**86.5**	**91.5**	**81.6**	**88.4**	**88.5**	**77.42**

Table 2: This table shows performance of the models on test set of GLUE tasks. Note that GPT-2$_{\text{Small}}$ is used as teacher for KD.

comparison. To achieve this, we compress half the layers of transformer block (odd numbered ones) in addition to the embedding layer by a factor of 2.

4.2 Pre-training

After the base model is compressed, performance of the compressed model drops significantly since the weight matrices with the Kronecker factors are approximate. Pre-training on a relatively small dataset for one epoch helps in retrieving the accuracy. Therefore, KnGPT2 is pre-trained (using the ILKD method discussed in Section 3.3) on 10% of OpenWebText which is 10 times less the DistiGPT2 model. As shown on Table 1 the training time for KnGPT2 is much faster as well.

	GPT-2$_{\text{Small}}$	DistilGPT2	KnGPT2
PPL	18.8	23.7	20.5

Table 3: Test Perplexity on WikiText-103.

4.3 Results

First we evaluate the models on language modeling using the Wikitext-103 dataset. The results are shown on Table 3. Although the DistilGPT2 is pre-trained longer and on a larger dataset the KnGPT2 achieves a lower perplexity.

[4]This number is presented in (Sanh et al., 2019) for training DistilBERT by the same authors. That uses the same KD algorithm and dataset for pre-training but is applied to BERT rather than GPT. Using a similar hardware we expect this number to be larger for DistilGPT

Next, performance of the models is evaluated on the test (Table 2) sets of seven datasets of the GLUE benchmark. Similar to the pre-training, we used the ILKD method discussed in Section 3.3 to fine-tune KnGPT2. For DistilGPT, we apply the basic KD algorithm also referred to in the literature as Vanilla KD (Jafari et al., 2021). For DistilGPT since the number of layers between the teacher and the student are different, it is not clear which teacher layer should be distilled to which student layer. Although there has been work on intermediate distillation for mismatched layers for BERT (Passban et al., 2021), extensive experimentation is required to conclude the best practice for GPT.

On the test set results (Table 2), we observe that KnGPT2 outperforms DistilGPT2 for all datasets. Applying ILKD, even improves performance of KnGPT2. Another interesting result is that Vanilla KD does not improve DistilGPT2 fine-tuning. Interestingly KnGPT2 with KD reaches close to the GPT-2$_{\text{Small}}$ performance on average.

5 Conclusion

In this paper, we compressed GPT-2 by compressing linear layers of a GPT model using Kronecker decomposition. Our model is pre-trained on a relatively small (10 times smaller than the dataset used for baseline) dataset which makes the pre-training much faster. Our proposed model significantly outperformed the baseline on the GLUE benchmark. Using KD can help to further reduce the performance drop of the compressed model. Using Kronecker decomposition on larger GPT models and for higher compression factors are two interesting future directions.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Harvey Goldstein. 2011. *Multilevel statistical models*, volume 922. John Wiley & Sons.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Harold V Henderson, Friedrich Pukelsheim, and Shayle R Searle. 1983. On the history of the kronecker product. *Linear and Multilinear Algebra*, 14(2):113–120.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. 2021. Annealing knowledge distillation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2493–2504, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11(2).

Tianda Li, Yassir El Mesbahi, Ivan Kobyzev, Ahmad Rashid, Atif Mahmud, Nithin Anchuri, Habib Hajimolahoseini, Yang Liu, and Mehdi Rezagholizadeh. 2021. A short study on compressing decoder-based language models. *arXiv preprint arXiv:2110.08460*.

Vasileios Lioutas, Ahmad Rashid, Krtin Kumar, Md Akmal Haidar, and Mehdi Rezagholizadeh. 2020. Improving word embedding factorization for compression using distilled nonlinear neural decomposition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2774–2784.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2021. ALP-KD: attention-based layer projection for knowledge distillation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13657–13665. AAAI Press.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2019. Fully quantized transformer for machine translation. *arXiv preprint arXiv:1910.10485*.

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

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. Mate-kd: Masked adversarial text, a companion to knowledge distillation. *arXiv preprint arXiv:2105.05912*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Marzieh S. Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. 2021. Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation.

Urmish Thakker, Jesse Beu, Dibakar Gope, Chu Zhou, Igor Fedorov, Ganesh Dasika, and Matthew Mattina. 2019. Compressing rnns for iot devices by 15-38x using kronecker products. *arXiv preprint arXiv:1906.02876*.

Urmish Thakker, Paul Whatamough, Matthew Mattina, and Jesse Beu. 2020. Compressing language models using doped kronecker products. *arXiv preprint arXiv:2001.08896*.

Charles F Van Loan. 2000. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. 2017. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*.

Shuchang Zhou and Jia-Nan Wu. 2015. Compression of fully-connected layer in neural network by kronecker product. *arXiv preprint arXiv:1507.05775*.

A Configurations

Table 4 shows sizes of matrices in GPT-2$_{\text{Small}}$, DistilGPT2 and KnGPT2.

B Kronecker product further explanation

Kronecker product has attractive abstract algebraic properties such as

$$\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$

$$(\mathbf{A} \otimes \mathbf{B})^{\top} = \mathbf{A}^{\top} \otimes \mathbf{B}^{\top}$$

for more details see (Henderson et al., 1983). The interesting properties of the Kronecker product makes it an attractive tool for decomposition of large matrices. The Kronecker product is also a flexible method to simplify the notation of large block matrices, both in linear mixed effect models and multilevel models (Goldstein, 2011). It is also a well-known technique to represent large repetitive structured graphs using the Kronecker product (Leskovec et al., 2010). One of the most important characteristics of a matrix is its determinant and it is well-known that for two square matrices $\mathbf{A}$ and $\mathbf{B}$ of size n, and m, $|\mathbf{A} \otimes \mathbf{B}| = |\mathbf{A}|^n |\mathbf{B}|^m$. This property explains the superiority of Kronecker compared to the other decomposition methods for large matrices. By choosing the right n and m, a large matrix $\mathbf{W} = \mathbf{A} \otimes \mathbf{B}$ can be decomposed to much smaller matrices such that the above determinant equation holds.

C Hyperparameters

Table 5 shows tuned hyperparameters for pre-training on OpenWebText and fine-tuning on GLUE. Also, to fine-tune models on Wikitext-103, we used the same hyperparamters as pre-training.

D GLUE dev results

Table 6 show performance of the models on dev set of GLUE.

E Ablation Study

(Tahaei et al., 2021) uses MSE as the distance metric between output of attention layers from all of the transformer layers of teacher and student. One of differences of our work from (Tahaei et al., 2021) is that inspired from (Wang et al., 2020), we used Kullback–Leibler (KL) divergence of the output of last attention layer. Using KL divergence over MSE improved average performance of the model, for 0.92% on the following GLUE tasks: COLA, MNLI, MRPC, QNLI, QQP, RTE and SST2 (Table 7).

(Tahaei et al., 2021) only minimizes the KD at the pre-training stage. we performed two experiments to study the effect of KD on the pre-training of KnGPT2 to improve performance of our model. In the first experiment, KnGPT2 is pre-trained by KD loss, with and without cross entropy (CE) loss

Model	Embedding	Q,K,V	FFN*
GPT-2_{Small}	50527×768	768×768	3072×768
DistilGPT2	50527×768	768×768	3072×768
KnGPT2	$A : 50527 \times 384$, $B : 1 \times 2$	$A : 384 \times 768$, B:2×1	$A : 1536 \times 768$, $B : 2 \times 1$

Table 4: This table shows configuration of the models. Note that FFN block has two projections that shape of one is the transpose of the other one and here, only shape of one of them is mentioned. Also, for KnGPT2, mentioned shapes for transformer layer belong to half of the layers that are decomposed -layers with odd numbers- and shape of the other half are the same with the GPT-2 model.

Phase	Epoch	Sequence Length	Seed	Batch size	Learning rate	α_1	α_2	α_3
Pre-training	1	1024	42	1	0.00025	0.5	0.5	0.1
Fine-tuning	20	128	42	16	2e-5	0.5	0.5	0.02

Table 5: hyper-parameters that are used for pre-training and fine-tuning.

Model	CoLA	RTE	MRPC	SST-2	MNLI	QNLI	QQP	Average
GPT-2_{Small}	47.6	69.31	87.47	92.08	83.12	88.87	90.25	79.81
DistilGPT2	38.7	65.0	87.7	91.3	79.9	85.7	89.3	76.8
DistilGPT2 + KD	38.64	64.98	87.31	89.80	80.42	86.36	89.61	76.73
KnGPT2	37.51	**70.4**	**88.55**	88.64	78.93	86.10	88.87	77
KnGPT2 + ILKD	**45.36**	69.67	87.41	**91.28**	**82.15**	**88.58**	**90.34**	**79.25**

Table 6: This table shows performance of the models on dev set of GLUE tasks. Note that GPT-2_{Small} is used as teacher for KD.

Model	CoLA	RTE	MRPC	SST-2	MNLI	QNLI	QQP	Average
KnGPT2 + ILKD$_{MSE}$	41.65	68.95	**88.89**	90.48	80.69	87.66	90.00	78.33
KnGPT2 + ILKD$_{KL}$	**45.36**	**69.67**	87.41	**91.28**	**82.15**	**88.58**	**90.34**	**79.25**

Table 7: This table shows performance of the Kronecker models (on dev set of GLUE tasks) that are fine-tuned using MSE and KL divergence as the distance metric in Equation 2.

Model	α_3	CoLA	RTE	MRPC	SST-2	MNLI	QNLI	QQP	Average
KnGPT2 + ILKD	0	40.80	**70.04**	**88.25**	90.71	80.12	87.64	89.64	78.17
KnGPT2 + ILKD	0.1	**45.36**	**69.67**	87.41	**91.28**	**82.15**	**88.58**	**90.34**	**79.25**

Table 8: This table shows performance of the Kronecker models (on dev set of GLUE tasks) that are pre-trianed with and without CE loss. α_3 indicates coefficient of CE loss during pre-training.

Model	Wikitext-103(PPL)	MNLI (f1)
KnGPT2	28608	69.33
KnGPT2 + LM	**21.94**	77.87
KnGPT2 + KD	144.1	77.50
KnGPT2 + LM + KD	23.04	**77.97**

Table 9: Ablation on the effect of pre-training with KD on language model and MNLI classification

then fine-tuned on GLUE with mentioned ILKD method discussed in Section 3.3. Empirical results (Table 8) show that adding cross entropy loss improves performance of Kronecker model on downstream tasks so we used KD + CE loss for both pre-training and fine-tuning. In the second experiment we used Wikitext-103 as our pre-training dataset. We compare four models and evaluate on LM as well as on classification using the MNLI dataset from GLUE. As shown on Table 9 we compare KnGPT2 without pre-training, with language modeling pre-training only, with KD pre-training only and both language modeling and KD pre-training. Note that we apply ILKD, discussed before for fine-tuning, as our KD algorithm. We observe that pre-training is important for good performance on the downstream task but lower perplexity on LM is not always a good indicator of better downstream performance.