Direction is what you need: Improving Word Embedding Compression in Large Language Models

Klaudia Bałazy[†*], Mohammadreza Banaei[‡*], Rémi Lebret[‡], Jacek Tabor[†] and Karl Aberer[‡]

[†]Jagiellonian University

klaudia.balazy@doctoral.uj.edu.pl, jacek.tabor@uj.edu.pl

[‡]EPFL

[mohammadreza.banaei, remi.lebret, karl.aberer]@epfl.ch

Abstract

The adoption of Transformer-based models in natural language processing (NLP) has led to great success using a massive number of parameters. However, due to deployment constraints in edge devices, there has been a rising interest in the compression of these models to improve their inference time and memory footprint. This paper presents a novel loss objective to compress token embeddings in the Transformer-based models by leveraging an AutoEncoder architecture. More specifically, we emphasize the importance of the direction of compressed embeddings with respect to original uncompressed embeddings. The proposed method is task-agnostic and does not require further language modeling pre-training. Our method significantly outperforms the commonly used SVD-based matrix-factorization approach in terms of initial language model Perplexity. Moreover, we evaluate our proposed approach over SQuAD v1.1 dataset and several downstream tasks from the GLUE benchmark, where we also outperform the baseline in most scenarios. Our code is public.[1].

1 Introduction

Pretraining deep Transformer models (Vaswani et al., 2017) with language modeling and fine-tuning these models over downstream tasks have led to great success in recent years (Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019), and even enabled researchers to design models that outperform human baselines in the GLUE benchmark (Wang et al., 2018). Although these models are empirically powerful in many natural language understanding (NLU) tasks, they often require a massive number of parameters, making them hard to use for memory-constrained applications (e.g., edge devices). Therefore, there have been efforts to compress BERT-like models while preserving comparable performance with the original model.

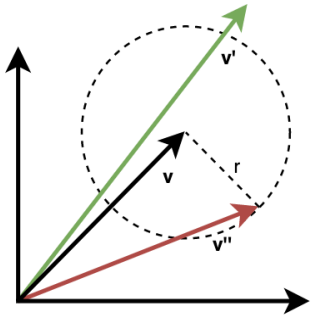

Figure 1: This figure presents a two-dimensional visualization of a token embedding vector v with its two approximations: v' and v''. Vector v' has a larger Euclidean distance error than v'', but its direction is more similar to the reference vector. Our experiments show that v' generally provides a better approximation of the original token compared to v''.

Many of these compression methods are based on knowledge distillation (Hinton et al., 2015) to help the compressed model (student) to perform close to the original model in different NLU tasks. However, these approaches often need high computation resources due to e.g., the necessity of retraining the expensive language modeling on a huge corpus (Sanh et al., 2019) or the use of expensive augmentation techniques to make the distillation effectively work (Jiao et al., 2019). Moreover, compression techniques that rely on training/fine-tuning language models are becoming less feasible due to its ever-increasing cost for current state-of-the-art architectures with hundreds

*Equal contribution

[1]https://github.com/MohammadrezaBanaei/orientation_based_embedding_compression

of millions of parameters (He et al., 2020; Raffel et al., 2019; Brown et al., 2020).

More recently, there have been efforts to compress Transformer-based models for more resource-constrained scenarios (Mao et al., 2020) by using offline methods, such as matrix factorization (Winata et al., 2019; Lan et al., 2019; Wang et al., 2019), weight pruning (Li et al., 2016; Han et al., 2015), and also weight quantization (Zhou et al., 2016; Hubara et al., 2016).

This paper focuses on token embedding matrix compression due to being one of the largest matrices in BERT-based architectures. We specifically question the effectiveness of current low-rank matrix factorization methods in recent literature (Lan et al., 2019; Wang et al., 2019) by comparing them with the performance of a linear AutoEncoder over different compression ratios[2]. We define a new loss objective which is not only dependent on the commonly used Mean Absolute Error (MAE) or Mean Squared Error (MSE) loss between input embeddings and AutoEncoder reconstruction, but is also sensitive to the noise in reconstructed embeddings "direction" (measured by cosine distance). We present the intuition behind the importance of embedding vector direction in the Figure 1. In the following sections we show that cosine distance indeed plays a more critical role than MAE/MSE (Figure 3) as measured by the Perplexity of the entire model in language modeling.

In Section 4, we demonstrate that our compression algorithm is superior or competitive to the Singular Value Decomposition (SVD) baseline over several natural language understanding tasks from GLUE (Wang et al., 2018) benchmark, as well as the SQuAD dataset (Rajpurkar et al., 2016) for question answering. We also compare our performance with the SVD-based compression over different compression ratios, and specifically show that our model performs consistently better in higher compression ratios.

Our contribution can be summarized as follows:

- We demonstrate the importance of direction (measured by cosine distance) in token embeddings compression.
- We leverage the AutoEncoder architecture to explore various multi-objective optimization functions.
- We outperform the SVD-based baseline in terms of Perplexity and over various downstream tasks.

[2]Number of parameters in the original embedding matrix, over the sum of the parameters in factorized matrices.

2 Related work

The current mostly used compression methods can be roughly categorized into four classes, namely knowledge distillation (Hinton et al., 2015), weight pruning (Li et al., 2016; Han et al., 2015), matrix factorization (Lan et al., 2019; Wang et al., 2019; Mao et al., 2020) and weight quantization (Zhou et al., 2016; Hubara et al., 2016). This section focuses on matrix factorization-based methods that are currently used for token embedding compression in the literature.

2.1 Background: Low-rank matrix factorization

This section describes the baseline method that we are comparing our approach with throughout the paper. Let A be $n \times m$ embedding matrix representing m-dimensional embedding for each n different input tokens. The truncated version of the matrix factorization aims to find a low-rank approximation $\widetilde{A}$ of input matrix A (Halko et al., 2011):

$$\widetilde{A} = BC, \tag{1}$$

where B is the size of $n \times k$ and C is the size of $k \times m$. When the inner dimension k is smaller than $min(n, m)$, then the approximation is less expensive for storing it and performing further computations. The objective of this approximation is:

$$L_2(A, \widetilde{A}) = \left\| A - \widetilde{A} \right\|_2, \tag{2}$$

where $\|\cdot\|_2$ denotes the l_2 operator norm. In this paper, we use the SVD method as a low-rank matrix factorization baseline to compare our approach.

2.2 Matrix factorization for token embeddings compression

Lan et al. (2019) proposed to use matrix factorization to limit the number of parameters in the token embedding matrix, which also separates the Transformer hidden layer dimension from the size of vocabulary embedding. It is especially important as token embeddings are supposed to be *context-independent*, but hidden layer representation should be a *context-dependent*

representation and hence needs more parameters. Moreover, reducing the vocabulary embedding dimension reduces the chance of overfitting, as many of the tokens are rarely used in downstream tasks.

There have been more recent efforts that use matrix factorization idea to compress different matrices in the Transformer architecture (Wang et al., 2019; Mao et al., 2020). For instance, Mao et al. (2020) proposed an iterative hybrid approach that uses matrix factorization together with weight pruning (while distilling knowledge from a teacher model) until reaching the final desired compression ratio. Lioutas et al. (2019) also proposed using a non-linear AutoEncoder model with knowledge distillation to compress word embeddings. However, we later demonstrate that only adding non-linearity indeed results in a minor improvement to the resulting compressed language model quality.

In this paper, we specifically focus on the effectiveness of SVD for compression of the token embedding matrix and show that Root Mean Square Error (RMSE) is not an optimal function to minimize the zero-shot Perplexity of the language model, which is the main criterion when language models are trained. We propose a new loss objective for linear matrix factorization using AutoEncoder to achieve a task-agnostic compressed language model with reasonable Perplexity without further fine-tuning the language model. In this work, we mainly investigate the effectiveness of SVD, and other complementary methods such as knowledge distillation can be used later to further boost the performance.

3 Model Description

Although SVD matrix-factorization is one of the most popular methods for matrix compression, we believe it is not an optimal method for compressing token embeddings in BERT-like architectures. The objective of SVD is to minimize the l_2 norm between the original matrix and the reconstructed one; however, focusing on l_2 norm optimization prioritizes the reduction of larger errors, and it may end up ignoring more minor vector differences. It is also sensitive to the influence of outliers. The most crucial reason for the l_2 norm not being the best choice is that it only considers the distance between the original and reconstructed token vector, and it does not necessarily pay attention to the orientation difference between them. In section 4, we demonstrate that vectors representing language tokens are more sensitive to noise in their direction rather than to changes in Euclidean distance from the reference vector. We also discuss the motivation behind it further in this section.

In order to mitigate the problem of focusing only on the largest errors between two vectors, we propose replacing the l_2 norm objective with the l_1 norm raised to the power of α:

$$L_1^{\alpha}(A, \widetilde{A}) = \left\| A - \widetilde{A} \right\|_1^{\alpha}, \tag{3}$$

where A denotes the original embedding matrix, $\widetilde{A}$ denotes the reconstructed embedding matrix, and $\|\cdot\|_1$ denotes the l_1 operator norm. Due to the flexibility in our defined loss objective, by decreasing the α parameter, we can control how much we want to focus on smaller error differences. We may set the α parameter to be a constant value, or linearly decrease it during the training. We denote linearly decreasing strategy for α as:

$$[t_1, t_2], \tag{4}$$

where t_1 is a starting value of α and t_2 is the target value to be reached at the end. The intuition behind using a decreasing α is to sequentially make the reconstruction harder for the model during training (as when the α becomes smaller, small reconstruction errors will also be magnified).

Since we believe that enforcing direction similarity between the original and the reconstructed embedding vectors is crucial for better language model performance, we introduce the second loss objective component, namely, cosine distance. Cosine distance can be interpreted as a measure of the difference in orientation of two vectors. This measure has been widely used in NLP for finding similar words (Mikolov et al., 2013), document clustering (Muflikhah and Baharudin, 2009), detecting plagiarism (Foltýnek et al., 2019), and many more. The goal of introducing cosine distance loss as a part of our objective is to enforce direction similarity of each pair of vectors from the original and reconstructed matrix.

Taking into consideration all points above, we propose to replace the l_2 norm objective with a new multi-objective function consisting of l_1 norm (raised to the power of α, where α is a hyper-parameter that can be changed during

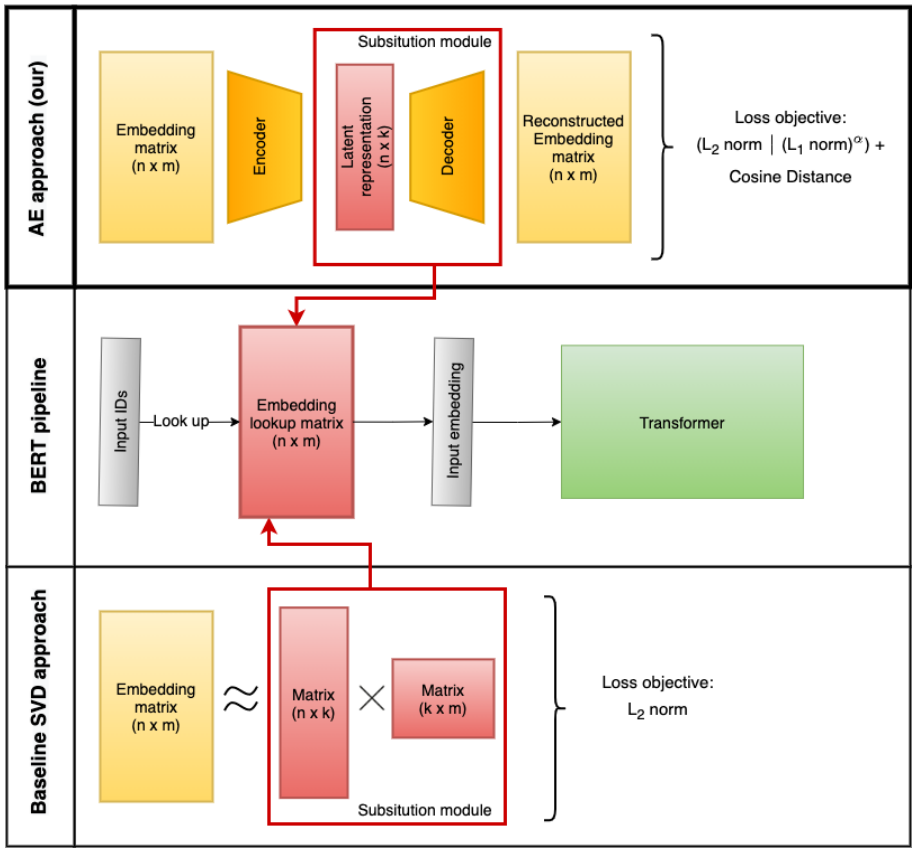

Figure 2: Overview of our AutoEncoder (ours) approach for BERT-like embedding matrix compression.

training) and cosine distance:

$$\Phi_{\alpha,\beta}(A, \widetilde{A}) = L_1^{\alpha}(A, \widetilde{A}) + \beta * CD(A, \widetilde{A}), \quad (5)$$

where A denotes the original embedding matrix, $\widetilde{A}$ denotes the reconstructed embedding matrix, and $CD(A, \widetilde{A})$ represents the mean cosine distance of all embedding vector pairs. It is worth noting that it is the combination of these two functions that gives a powerful tool which allows both to optimize the distance and direction of the reconstructed vectors to the reference. Focusing only on one of these functions may lead to suboptimal results. For comparison, we also define another multi-objective function which is the combination of l_2 norm with cosine distance loss:

$$\Psi_{\beta}(A, \widetilde{A}) = L_2(A, \widetilde{A}) + \beta * CD(A, \widetilde{A}). \quad (6)$$

In addition to the new loss function, we propose leveraging Auto-Encoder architecture for $\Phi_{\alpha,\beta}$ and Ψ_{β} loss optimization (Equation 5 and 6). We use a simple AutoEncoder consisting of a one-layer Encoder/Decoder without any activation function in order to have a fair comparison with the SVD baseline. Using Auto-Encoder enables efficient multi-objective optimization, but it also allows to select the appropriate level of model complexity when needed. At the end of the Auto-Encoder training, we extract an approximation of the original matrix, as shown in Figure 2. We substitute the original embedding matrix with a new module consisting of latent representation of vocabulary tokens along with the Decoder module.

4 Results

In this section, we evaluate our approach, which is based on using AutoEncoder model with a multi-objective loss function that incorporates cosine distance with l_1 or l_2 norm (Equation 5 and Equation 6) on the task of BERT-like token embedding matrix compression. We compare our results versus the commonly used randomized SVD method (Halko et al., 2011) to perform low-rank matrix factorization. We have implemented our token embeddings compression with the PyTorch backend (Paszke et al., 2019) and as an extension of Huggingface's Transformers library (Wolf et al., 2019), enabling researchers to apply our compression method in most of the existing Transformer architectures. It is worth noting that

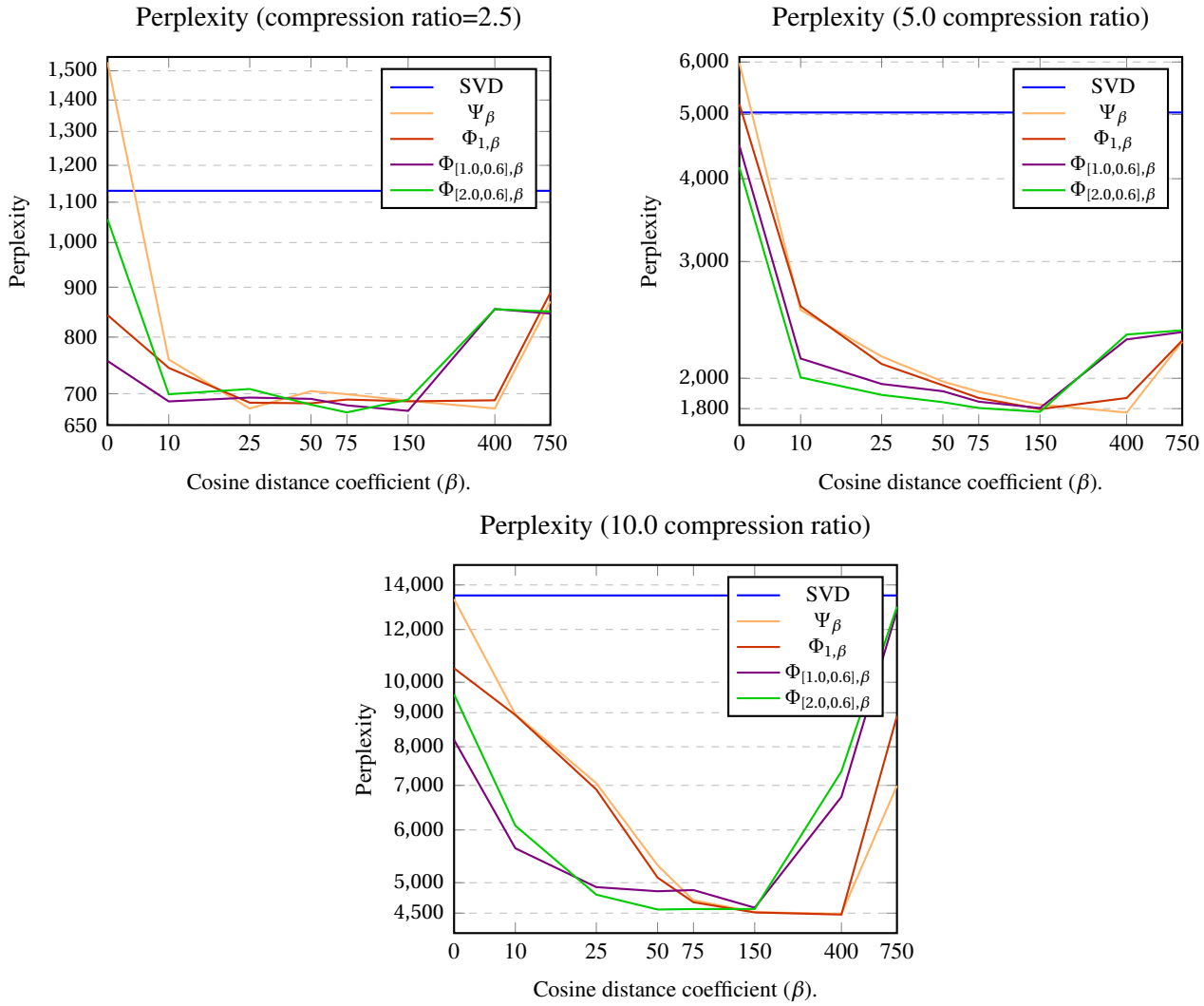

Figure 3: The impact of the β coefficient on Perplexity metric (lower is better) in the linear AutoEncoder loss functions: $\Phi_{\alpha,\beta}$ (Equation 5) and Ψ_β (Equation 6). In all configurations we select a final model based on the best Perplexity achieved during training. The term $[t_1, t_2]$ indicates linearly decreasing α parameter (Equation 4). Setting $\beta = 0$ represents not including cosine distance component in the loss function. We may observe that not including cosine distance in the loss function as well as making it a too dominant component (very big β) is not optimal for achieving good Perplexity. We also present the best Perplexity achieved by the baseline SVD method for three compression ratios: 2.5, 5.0, 10.0. Our approach significantly outperforms the baseline in the studied scenarios.

the offline training of our compression method on BERT-base (Devlin et al., 2018) token embedding matrix takes only few minutes on a single GPU device.

4.1 Experiments

In this paper, we perform our experiments over BERT-base model, but the general idea can be applied to the vocabulary embeddings of any other similar transformer-based architecture. The BERT-Base token embedding matrix consists of more than 23 Million parameters which is around 21% of all parameters in the model. We evaluate the quality of our final compressed embeddings on the masked (Devlin et al., 2018) language modeling task (using WikiText-103 test dataset), GLUE benchmark (Wang et al., 2018) downstream tasks and SQuAD v1.1 dataset (Rajpurkar et al., 2016). We also analyze results on other metrics, namely RMSE, MAE and Cosine Distance.

In Figure 3, we compare the Perplexity score achieved by SVD[3] method versus the results

[3]For SVD training, we select an iteration that minimizes Perplexity over our language modeling dataset.

achieved by a linear AutoEncoder model with different loss configurations, when compressing BERT token embeddings. We specifically examine the importance of cosine distance coefficient (β) in our studied loss functions over three different compression ratios: 2.5, 5, 10. The loss objective $\Phi_{t,\beta}$ (Equation 5) denotes constant (during the entire training) α parameter (equals to t) and $\Phi_{[t_1,t_2],\beta}$ denotes linearly decreasing α parameter (from t_1 to t_2). We present results when $\alpha = 0$, which represents combination of l_1 norm with cosine distance, and also when α linearly decreases from 1.0 or from 2.0 to 0.6 ($[1.0, 0.6]$ and $[2.0, 0.6]$ respectively). These values have been selected experimentally.

Table 1 presents more metrics to compare SVD method with our AutoEncoder-based approach. We show the results of the model with the best performing objective function (in terms of Perplexity) for a given compression ratio. Additionally, we examine the effect of adding non-linear activation function to this selected AutoEncoder model, where it can be seen that the improvements due to addition of non-linearity is marginal.

We further validate the quality of our compressed token embeddings by inserting it into the BERT-base architecture and fine-tuning the model on different downstream tasks from the GLUE benchmark (Wang et al., 2018) and on the SQuAD v1.1 (Rajpurkar et al., 2016) dataset. Table 2 presents an extensive comparison between our best (in terms of perplexity) linear AE and the SVD baseline on eight different downstream tasks and over different compression ratios. More specifically, we can see that our proposed method is superior or competitive to the SVD baseline and performs relatively better (compared to baseline) on higher compression ratios. The original BERT (without compression) performance is also added for a better comparison of studied scenarios.

Figure 4 presents learning curves for three selected NLU downstream tasks: SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005) and SQuAD 1.1 (Rajpurkar et al., 2016). We show results for the compression ratio of 10, as we observed more significant gain for higher compression ratios.

4.2 Discussion

The experiments presented in Figure 3 confirm our claim that the l_2 norm alone is not an optimal measure for evaluating the quality of reconstructed token embeddings in a Transformer-based architecture. We observe that adding cosine distance objective function correlates positively with a better Perplexity metric (Figure 3) and also with higher performance on downstream tasks (Table 2). Figure 3 demonstrates that the best results are achieved when the cosine distance coefficient β is a dominant component of the loss function. However, if the β factor becomes too large, the quality of the solution decreases. Hence, we conclude that taking into account both the commonly used L1/L2 distance and focusing on the direction of the token vectors are indispensable. We show that combining the l_2 or l_1 norm with the cosine distance into one multi-objective loss function and optimizing it by AutoEncoder model outperforms the baseline SVD Perplexity for all tested compression ratios (Figure 3). Our experiments show that depending on the compression ratio l_2 or l_1 norm may be a better choice. However, they are conclusive that adding cosine distance is the key factor.

Moreover, our approach outperforms SVD in terms of accuracy for most GLUE benchmark downstream tasks and on SQuAD v1.1 (Table 2). We also observe that for higher compression ratios, our approach outperforms the SVD approach more significantly. More importantly, Figure 4 demonstrates that using our linear AutoEncoder compressed module in the BERT model generally converges faster than SVD-based compressed module, which is especially important in few-shot learning scenarios.

Looking at the results presented in Table 1, we may also reflect on the importance of preserving the token vector orientation and its effect on Perplexity. More specifically, the mean cosine distance measures for SVD and our approach are pretty close, but its effect on Perplexity metric is significant. Our approach indeed provides a compressed submodule with a much better (lower) Perplexity.

We also show that only adding a non-linear activation function to the studied AutoEncoder model has a little effect on improving Perplexity. Table 1 presents the effect of modifying the original linear AutoEncoder architecture by adding

CR (#Params)	Architecture	Objective	RMSE	Cosine Distance	MAE	Perplexity
2.5 (~9.38M)	SVD	l_2	**0.02233**	0.10300	**0.01734**	1130
	Linear AE (+ ELU)	$\Phi_{[2.0,0.6],75}$	0.02427 (0.02431)	**0.1024 (0.1028)**	0.01896 (0.01902)	**669.8 (664.0)**
5.0 (~4.69M)	SVD	l_2	**0.02848**	0.17490	**0.02216**	5035
	Linear AE (+ ELU)	Ψ_{400}	0.03101 (0.03061)	**0.17390 (0.17410)**	0.02433 (0.02401)	**1776 (1730)**
10.0 (~2.34M)	SVD	l_2	**0.03215**	0.23050	**0.02506**	13501
	Linear AE (+ ELU)	$\Phi_{1,400}$	0.03680 (0.03707)	**0.22900 (0.22910)**	0.02909 (0.02934)	**4478 (4387)**

Table 1: Additional metrics for comparing the performance of SVD baseline and the best performing linear AutoEncoder model (we select the configuration that minimizes Perplexity, as presented in Figure 3) for different compression ratios (CR). For each AutoEncoder model, we also present (in parentheses) the results after adding non-linearity. Bold values indicate best results between SVD and linear AutoEncoder in each compression ratio.

CR	Architecture	SST-2 (Acc)	MRPC (F1/Acc)	STS-B (Pearson/Spearman correlation)	QQP (Acc/F1)	MNLI (Acc)	QNLI (Acc)	RTE (Acc)	SQuAD v1.1 (F1/EM)
-	Original BERT	91.74	88.12/83.58	88.71/88.55	90.67/87.43	84.04	90.96	65.34	81.97/73.42
2.5	SVD	89.22	82.37/75.25	86.27/85.72	89.88/86.39	82.83	**89.46**	**62.92**	80.75/72.34
	Linear AE	**90.83**	**86.64/80.88**	**87.35/86.88**	**90.04/86.72**	**83.13**	89.16	62.58	**81.29/72.85**
5.0	SVD	87.04	83.95/77.70	84.88/84.2	**89.79/86.45**	**81.39**	87.33	59.21	80.37/71.67
	Linear AE	**88.07**	**86.67/81.37**	**85.9/85.43**	89.2/85.66	81.11	**87.53**	**64.26**	**80.53/72.00**
10.0	SVD	82.0	83.95/72.55	80.93/80.67	87.6/83.57	76.59	83.51	54.51	74.15/65.0
	Linear AE	**84.29**	**84.06/77.7**	**84.7/84.16**	**88.32/84.38**	**79.26**	**86.09**	**58.48**	**75.70/66.75**

Table 2: Performance comparison of the best SVD and the best linear AutoEncoder objective configuration on several NLU tasks from GLUE benchmark (Wang et al., 2018) and for SQuAD v1.1 in different compression ratios (CR).

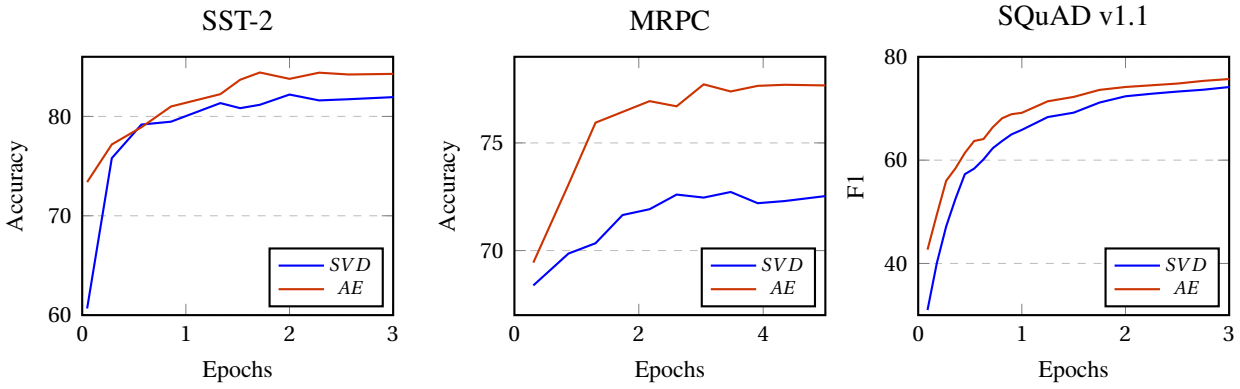

Figure 4: Comparing the learning curves of the best SVD baseline and the best-selected configuration of the AutoEncoder model for SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), and SQuAD v1.1 (Rajpurkar et al., 2016) during fine-tuning for compression ratio=10.0 .

ELU (Clevert et al., 2015) as this activation shows a better impact on Perplexity than other activations in our experiments. It can be seen that the improvements in Perplexity due to the addition of non-linearities are marginal (as previously observed by Lioutas et al. (2019) in a distillation-based approach for token embeddings compression). Hence, we focused only on the linear AutoEncoder in all our downstream tasks experiments.

5 Conclusion

In this work, we propose a simple linear AutoEncoder model with a multi-objective loss function for BERT-like token embeddings compression. We emphasize the importance of the direction

component (measured by the cosine distance between the original and the reconstructed token embeddings) in the compression objective function. We challenge the commonly used SVD-based matrix-factorization method and show that our approach achieves significantly better zero-shot language model Perplexity. Moreover, we show that BERT-like models with our compressed token embeddings submodule converge much faster and outperform the SVD baseline on SQuAD v1.1 and on GLUE benchmark tasks in most scenarios.

6 Acknowledgements

This research was partially funded by the Priority Research Area Digiworld under the program Excellence Initiative – Research University at the Jagiellonian University in Kraków.

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*.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

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*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Tomáš Foltỳnek, Norman Meuschke, and Bela Gipp. 2019. Academic plagiarism detection: a systematic literature review. *ACM Computing Surveys (CSUR)*, 52(6):1–42.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

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

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4114–4122.

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*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Vasileios Lioutas, Ahmad Rashid, Krtin Kumar, Md Akmal Haidar, and Mehdi Rezagholizadeh. 2019. Distilled embedding: non-linear embedding factorization using knowledge distillation. *arXiv preprint arXiv:1910.06720*.

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*.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. *arXiv preprint arXiv:2004.04124*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lailil Muflikhah and Baharum Baharudin. 2009. Document clustering using concept space and cosine similarity measurement. In *2009 International Conference on Computer Technology and Development*, volume 1, pages 58–62. IEEE.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J Barezi, and Pascale Fung. 2019. On the effectiveness of low-rank matrix factorization for lstm model compression. *arXiv preprint arXiv:1908.09982*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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*.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.