Improving Word Embedding Factorization for Compression Using Distilled Nonlinear Neural Decomposition

Vasileios Lioutas[1]*, Ahmad Rashid[2], Krtin Kumar[2],
Md. Akmal Haidar[2], Mehdi Rezagholizadeh[2]
[1]University of British Columbia, [2]Huawei Noah's Ark Lab
contact@vlioutas.com, ahmad.rashid@huawei.com, krtin.kumar@huawei.com,
md.akmal.haidar@huawei.com, mehdi.rezagholizadeh@huawei.com

Abstract

Word-embeddings are vital components of Natural Language Processing (NLP) models and have been extensively explored. However, they consume a lot of memory which poses a challenge for edge deployment. Embedding matrices, typically, contain most of the parameters for language models and about a third for machine translation systems. In this paper, we propose Distilled Embedding, an (input/output) embedding compression method based on low-rank matrix decomposition and knowledge distillation. First, we initialize the weights of our decomposed matrices by learning to reconstruct the full pre-trained word-embedding and then fine-tune end-to-end, employing knowledge distillation on the factorized embedding. We conduct extensive experiments with various compression rates on machine translation and language modeling, using different data-sets with a shared word-embedding matrix for both embedding and vocabulary projection matrices. We show that the proposed technique is simple to replicate, with one fixed parameter controlling compression size, has higher BLEU score on translation and lower perplexity on language modeling compared to complex, difficult to tune state-of-the-art methods.

1 Introduction

Deep Learning models are the state-of-the-art in NLP, Computer Vision, Speech Recognition and many other fields in Computer Science and Engineering. The remarkable deep learning revolution has been built on top of massive amounts of data (both labeled and unlabeled), and faster computation. In NLP, large pre-trained language models like BERT (Devlin et al., 2019) are state-of-the-art on a large number of downstream NLP problems. The largest publicly available language models are trained with hundred of billions of parameters (Brown et al., 2020). In machine translation the state-of-the-art models have parameters in the order of billions. Data privacy and server cost are some major issues, driving research towards deploying these models on edge-devices. However, running these models on edge-devices, faces memory and latency issues due to limitations of the hardware. Thus, there has been considerable interest towards research in reducing the memory footprint and faster inference speed for these models (Sainath et al., 2013; Acharya et al., 2019; Shi and Yu, 2018; Jegou et al., 2010; Chen et al., 2018; Winata et al., 2019).

The architecture of deep-learning-based language generation models can be broken down into three components. The first component, represents the embedding, which maps words in the vocabulary to continuous dense vector representations of the words. In language modeling we typically have one dictionary but machine translation has at least two dictionaries corresponding to a translation pair. We model these as a single dictionary with a common embedding matrix. The second component, consists of a function f, typically a deep neural-network (Schmidhuber, 2015; Krizhevsky et al., 2012; Mikolov et al., 2010) which maps the embedding representation for different NLP problems (machine-translation, summarization, question-answering and others), to the output-space of function f. The third component, is the output layer which maps the output of function f to the vocabulary-space, followed by a softmax function. Since, the first and third components depend upon a large vocabulary-size, they require large number of parameters which results in higher latency and larger memory requirements. For instance, the Transformer Base model (Vaswani et al., 2017) uses 37% of the parameters in the first and

[*] Work done during an internship at Huawei Noah's Ark Lab.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2774–2784
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

third components using a vocabulary size of 50k, and with parameter-tying between the components. The percentage of parameters increases to 54%, when parameters are not shared between the first and third components. Thus, an obvious step towards model compression is to reduce the parameters used by the embedding matrices.

Recently, there has been considerable work on compressing word-embedding matrices (Sainath et al., 2013; Acharya et al., 2019; Shi and Yu, 2018; Jegou et al., 2010; Chen et al., 2018; Winata et al., 2019). These techniques have proven to perform at-par with the uncompressed models, but still suffer from a number of issues.

First, state-of-the-art embedding compression methods such as GroupReduce, Structured Emebedding and Tensor Train Decomposition (Shi and Yu, 2018; Chen et al., 2018; Khrulkov et al., 2019; Shu and Nakayama, 2018), require multiple hyper-parameters to be fine-tuned to optimize performance on each dataset. These hyper-parameters influence the number of parameters in the model, and thus the compression rate. This leads to an additional layer of complexity for optimizing the model for different NLP problems. Additionally, Chen et al. (2018) requires an additional optimization step for grouping words, and lacks end-to-end training through back-propagation. Shi and Yu (2018) also requires an additional step for performing k-means clustering for generating the quantization matrix. Thus, most of the current state-of-the-art systems are much more complicated to fine-tune for different NLP problems and data-sets.

Second, all the state-of-the-art embedding compression models compress the input and output embedding separately. In practice, state-of-the-art NLP models (Vaswani et al., 2017; Lioutas and Guo, 2020) have shown better performance with parameter sharing between the two (Press and Wolf, 2017). Thus, there is a need for an exhaustive analysis of various embedding compression techniques, with parameter sharing.

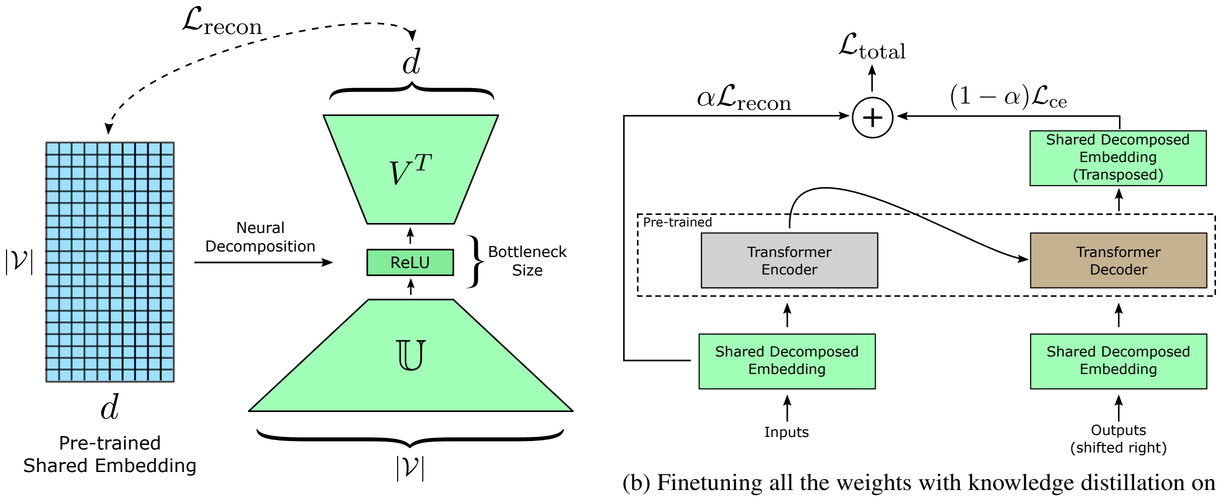
Lastly, embedding compression models not based on linear SVD (Khrulkov et al., 2019; Shi and Yu, 2018) require the reconstruction of the entire embedding matrix or additional computations, when used at the output-layer. Thus during runtime, the model either uses the same amount of memory as the uncompressed model or pays a higher computation cost. This makes linear SVD based techniques more desirable for running models on edge-devices.

In this paper, we introduce Distilled Embedding, a matrix factorization method, based on Singular Value Decomposition (SVD) with two key changes a) a neural network decomposition instead of an eigenvalue decomposition and b) a distillation loss on the word embedding while fine-tuning. Our method, first compresses the vocabulary-space to the desired size, then applies a non-linear activation function, before recovering the original embedding-dimension. Additionally, we also introduce an embedding distillation method, which is similar to Knowledge Distillation (Hinton et al., 2015) but we apply it to distill knowledge from a pre-trained embedding matrix and use an $L2$ loss instead of cross-entropy loss. To summarize, our contributions are as follows:

- We demonstrate that SVD, when fine-tuned till convergence, is comparable to recently proposed, difficult to tune methods.
- We demonstrate that at the same compression rate Distilled Embedding outperforms existing state-of-the-art methods on machine translation and SVD on language modeling.
- Our proposed method is much simpler than the current state-of-the-methods, with only a single parameter controlling the compression rate.
- Unlike the current state-of-the-art systems, we compress the embedding matrix with parameter sharing between input and output embeddings. We perform an exhaustive comparison of different models in this setting.
- Our method is faster at inference speed than competing matrix factorization methods and only slightly slower than SVD.

2 Related Work

We can model the problem of compressing the embedding matrix as a matrix factorization problem. There is a considerable amount of work done in this field and some of the popular methods include Singular Value Decomposition (SVD) (Srebro and Jaakkola, 2003; Mnih and Salakhutdinov, 2008), product quantization (Jegou et al., 2010) and tensor decomposition (De Lathauwer et al., 2000). A number of prior works in embedding compression


$\mathcal{L}_{\text{recon}}$
d
V^T
$|\mathcal{V}|$
Neural Decomposition
ReLU
Bottleneck Size
$\mathbb{U}$
d
Pre-trained Shared Embedding
$|\mathcal{V}|$
$\mathcal{L}_{\text{total}}$
$\alpha\mathcal{L}_{\text{recon}}$
$(1-\alpha)\mathcal{L}_{\text{ce}}$
Shared Decomposed Embedding (Transposed)
Pre-trained
Transformer Encoder
Transformer Decoder
Shared Decomposed Embedding
Shared Decomposed Embedding
Inputs
Outputs (shifted right)

(a) Funneling Decomposition

(b) Finetuning all the weights with knowledge distillation on the embedding

Figure 1: Distilled Embedding method to compress the shared embedding matrix of a transformer based sequence to sequence model.

are influenced by these fields and have been applied to various NLP problems. In this Section, we will discuss some of the significant works across different NLP problems.

Low-rank Factorization Low-rank approximation of weight matrices, using SVD, is a natural way to compress deep learning based NLP models. Sainath et al. (2013) apply this to a convolutional neural network for language modeling and acoustic modeling. Winata et al. (2019) use SVD on all the weight matrices of an LSTM and demonstrate competitive results on question-answering, language modeling and text-entailment. Acharya et al. (2019) use low-rank matrix factorization for word-embedding layer during training to compress a classification model. However, they do not study the effects of applying a non-linear function before reconstructing the original dimension.

GroupReduce Chen et al. (2018) apply weighted low-rank approximation to the embedding matrix of an LSTM. They first create a many-to-one mapping of all the words in the vocabulary into g groups based upon word frequency. For each group g they apply weighted SVD to obtain a lower rank estimation, the rank is determined by setting a minimum rank and linearly increasing it based upon average frequency. Finally, they update the groups by minimizing the reconstruction error from the weighted SVD approximation. They demonstrate strong results on language modeling and machine translation compared to simple SVD. In their models they use different embedding matrices for input and softmax layers and apply different compression ratios to each.

Product Quantization Jegou et al. (2010) introduced product quantization for compressing high dimensional vectors, by uniformly partitioning them into subvectors and quantizing each subvector using K-means clustering technique. Basically, product quantization assumes that the subvectors share some underlying properties which can be used to group similar ones together and unify their representation. That being said, this approach breaks the original matrix into a set of codebooks coming from the center of the clusters in different partitions together with a separate index matrix which refers to the index of the clusters for each subvector. Shi and Yu (2018) applied product quantization to a language model and were able to show better perplexity scores. Shu and Nakayama (2018) extended this technique by first representing the product quantization as a matrix factorization problem, and then learning the quantization matrix in an end-to-end trainable neural network. Li et al. (2018) implement product quantization through randomly sharing parameters in the embedding matrix, and show good results on perplexity for an LSTM based language model.

Tensor Decomposition De Lathauwer et al. (2000) introduced multilinear SVD, which is a generalization of SVD for higher order tensors. Oseledets (2011) introduced an efficient algorithm Tensor Train (TT) for multilinear SVD Tensor.

Novikov et al. (2015) applied the Tensor Train decomposition on fully connected layers of deep neural networks. Khrulkov et al. (2019) applied Tensor Train algorithm to the input embedding layer on different NLP problems like language modeling, machine translation and sentiment analysis. They demonstrate high compression rate with little loss of performance. However, they compress only the input embedding and not the softmax layer for language modeling and machine translation.

Knowledge Distillation Knowledge distillation (Buciluă et al., 2006; Hinton et al., 2015). has been studied in model compression where knowledge of a large cumbersome model is transferred to a small model for easy deployment. In this paper, we propose an embedding factorization of word-embedding matrix using knowledge distillation to mimic the pre-trained word-embedding representation.

3 Methodology: Distilled Embedding

3.1 Funneling Decomposition and Embedding Distillation

We present an overview of our proposed method in Figure 1. Given an embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$, we can decompose it into three matrices (Equation 1), using the SVD algorithm

$$E = U_{|\mathcal{V}| \times |\mathcal{V}|} \Sigma_{|\mathcal{V}| \times d} V^T_{d \times d} \tag{1}$$

where $|\mathcal{V}|$ is the vocabulary size and d is the embedding dimension. Σ is a diagonal matrix containing the singular values, and matrices U and V represent the left and right singular vectors of the embedding matrix respectively. We can obtain the reduced form of the embedding matrix, $\tilde{E}$, by only keeping r $(< d)$ largest singular values out of d.

$$\tilde{E} = U_{|\mathcal{V}| \times r} \Sigma_{r \times r} V^T_{r \times d} = \mathbb{U}_{|\mathcal{V}| \times r} V^T_{r \times d} \tag{2}$$

where the matrix $\mathbb{U} = U\Sigma$. The reduced form of the embedding matrix will need $r \times (|\mathcal{V}| + d)$ parameters compared to $|\mathcal{V}| \times d$.

Our proposed approach in this work, is to apply a non-linear transformation on the matrix $\mathbb{U}$, before reconstructing the original embedding dimension using V (see Figure 1a), as shown in Equation 3,

$$\tilde{E} = f(\mathbb{U}_{|\mathcal{V}| \times r}) V^T_{r \times d} \tag{3}$$

We use the ReLU as our non-linear function $f(.)$ throughout this paper. We postulate that this neural decomposition helps in end-to-end training during the fine-tuning stage, although, we can only demonstrate empirical evidence for that. We train a sequence to sequence model (Sutskever et al., 2014; Vaswani et al., 2017) with tied input and output embedding (i.e. the output embedding is the transpose of the input embedding matrix $\tilde{E}_{\text{out}} = \tilde{E}^T = V_{d \times r}[f(\mathbb{U}_{|\mathcal{V}| \times r})]^T$. We train our model end-to-end by replacing the embedding function with Equation 3. The matrix $\mathbb{U}$ and V are trainable parameters, and for the output layer we use $\tilde{E}^T$, with the parameter sharing. We train on two losses. The standard cross entropy loss defined as:

$$\mathcal{L}_{\text{ce}} = -\sum_{i=1}^{M} \mathrm{y}_i \log(p_i) \tag{4}$$

where M is the sequence length, y_i is the one-hot representation for the i^{th} label and p_i is the softmax probability of the i^{th} term generated by the decoder.

In addition to the cross-entropy loss, we introduce a novel embedding reconstruction loss (Equation 5), which we refer to as embedding distillation as we distill information from the pre-trained embedding into our model,

$$\begin{aligned}\mathcal{L}_{\text{recon}} &= \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \|e_i - \tilde{e}_i\|_2 \\ &= \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \|e_i - f(u_i) V^T_{r \times d}\|_2\end{aligned} \tag{5}$$

where e_i and $\tilde{e}_i$ are the embedding vectors corresponding to the i^{th} word in the original embedding matrix E and the reconstructed embedding matrix $\tilde{E}$ respectively and u_i refers to the i^{th} row of the matrix $\mathbb{U}$. This helps in better generalization since during fine-tuning the words seen in the training corpus are given higher weight at the expense of low-frequency word. This loss helps maintain a balance between the two.

We use Equation 6 as our final loss function

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{recon}} + (1 - \alpha) \mathcal{L}_{\text{ce}} \tag{6}$$

where $\alpha \in [0, 1]$ is a hyper-parameter, which controls the trade-off between reconstruction and cross-entropy loss. $\mathcal{L}_{\text{recon}}$ acts as the knowledge distillation loss by which we try to distill information from the original pre-trained embedding layer as a teacher to the funneling decomposed embedding layer as a student. The training process of our Distilled Embedding method is summarized in **Algorithm 1**.

Algorithm 1 Distilled Embedding

Step 1) Pre-training the Embedding Matrix Pre-train the sequence to sequence model with the full embedding matrix for better initialization.

Step 2) Initializing the Weights of Funneling Decomposition Layer We extract the trained embedding matrix E from Step 1 and train our decomposed matrices $\mathbb{U}$ and V on reconstruction loss defined in Equation 5, as shown in Figure 1a.

Step 3) Embedding Distillation The pre-trained funneling decomposition layer is plugged into the model (replacing the original embedding matrix E) and the entire model is trained based on Equation 6.

4 Experimental Setup

4.1 Datasets and Evaluation

We test our proposed method on machine translation and language modeling which are fundamental problems in NLP and challenging for embedding compression since we typically have an input and output embedding.

On machine translation, we present results on three language pairs: WMT English to French (En-Fr), WMT English to German (En-De) and IWSLT Portuguese to English (Pt-En). We decided that these pairs are good representatives of high-resource, medium-resource and low-resource language pairs.

WMT En-Fr is based on WMT'14 training data which contains 36M sentence pairs. We used SentencePiece (Kudo and Richardson, 2018) to extract a shared vocabulary of 32k subwords. We validate on newstest2013 and test on newstest2014. For WMT English to German (En-De), we use the same setup as Vaswani et al. (2017). The dataset is based on WMT'16 training data and contains about 4.5M pairs. We use a shared vocabulary of 37k subwords extracted using SentencePiece.

For the IWSLT Portuguese to English (Pt-En) dataset, we replicate the setup of Tan et al. (2019) for training individual models. Specifically, the dataset contains about 167k training pairs. We used a shared vocabulary of 32k subwords extracted with SentencePiece.

For all language pairs, we measure case-sensitive BLEU score (Papineni et al., 2002) using SacreBLEU[1] (Post, 2018). In addition, we save a checkpoint every hour for the WMT En-Fr and WMT En-De language pairs and every 5 minutes for the IWSLT Pt-En due to the smaller size of the dataset. We use the last checkpoint which resulted in the highest validation BLEU and average the last five checkpoints based on this. We use beam search with a beam width of 4 for all language pairs.

For language modeling, we decided to use the WikiText-103 dataset (Merity et al., 2017) which contains 103M training tokens from 28K articles, with an average length of 3.6K tokens per article. We replicate the setup of Dai et al. (2019) for training the base and the compressed models.

4.2 Experiment Details

Hyper-Parameters For WMT En-Fr and WMT En-De, we use the same configuration as Transformer Base which was proposed by Vaswani et al. (2017). Specifically, the model hidden size d_{model} is set to 512, the feed-forward hidden size d_{ff} is set to 2048 and the number of layers for the encoder and the decoder was set to 6. For the IWSLT Pt-En, we use Transformer Small configuration. Specifically, the model hidden-size d_{model} is set to 256, the feed-forward hidden size d_{ff} is set to 1024 and the number of layers for the encoder and the decoder was set to 2. For Transformer Small, the dropout configuration was set the same as Transformer Base. All models are optimized using Adam (Kingma and Ba, 2015) and the same learning rate schedule as proposed by Vaswani et al. (2017). We use label smoothing with 0.1 weight for the uniform prior distribution over the vocabulary (Szegedy et al., 2016; Pereyra et al., 2017). Additionally, we set the value α of Equation 6 to 0.01.

For the WikiText-103 we use the same configuration as Transformer-XL Standard which was proposed by Dai et al. (2019). Specifically, the model hidden size d_{model} is set to 410, the feed-forward hidden size d_{ff} is set to 2100 and the number of layers for was set to 16.

Hardware Details We train the WMT models on 8 NVIDIA V100 GPUs and the IWSLT models on a single NVIDIA V100 GPU. Each training batch contained a set of sentence pairs containing approximately 6000 source tokens and 6000 target tokens for each GPU worker. All experiments were run using the TensorFlow framework[2].

[1] https://github.com/mjpost/sacreBLEU

[2] https://www.tensorflow.org/

Model	WMT En-Fr		WMT En-De		IWSLT Pt-En	
	Emb. CR	BLEU	Emb. CR	BLEU	Emb. CR	BLEU
Transformer Base	1.0x	38.12	1.0x	27.08	1.0x	41.43
Smaller Transformer Network (416)	1.23x	37.26	1.28x	26.72	1.88x	40.71
End-to-End NN compression with non-linearity	7.87x	37.23	7.89x	26.14	3.96x	42.27
SVD with rank 64	7.87x	37.44	7.89x	26.32	3.96x	42.37
GroupReduce (Chen et al., 2018)	7.79x	37.63	7.88x	26.75	3.96x	42.13
Structured Embedding (Shi and Yu, 2018)	7.90x	**37.78**	7.89x	26.34	3.97x	41.27
Tensor Train (Khrulkov et al., 2019)	7.72x	37.27	7.75x	26.19	3.96x	42.34
Distilled Embedding (Ours)	7.87x	**37.78**	7.89x	**26.97**	3.96x	**42.62**

Table 1: Machine translation BLEU score for the three language pairs. CR refers to the compression rate.

5 Results

5.1 Machine Translation

We present BLEU score for our method and compare it with SVD, GroupReduce (Chen et al., 2018), Structured Emedding (Shi and Yu, 2018), Tensor Train (Khrulkov et al., 2019) and a smaller transformer network with the same number of parameters. We learn a decomposition for all the methods except Tensor Train since it was pointed out in Khrulkov et al. (2019) that there is no difference in performance between random initialization and tensor train learnt initialization. Once initialized we plug the decomposed embedding and fine-tune till convergence. None of the weights are frozen during fine-tuning.

Table 1 presents the results on translation. We see that on the English-French language pair our method along with Structured Embedding performs the best. Group Reduce is next, and SVD performs better than Tensor Train, showing that SVD is a strong baseline, when fine-tuned till convergence. We also compare against end-to-end compression using a 2 layer neural network (NN) with the same parameterization as distilled embedding which has not been initialized offline. The results show that initializing the neural decomposition with the embedding weights is important.

On English-German translation, our method outperforms all other methods. The smaller transformer network does well and is only surpassed by GroupReduce amongst the competing methods. SVD again performs better than Tensor Train.

The Portuguese-English task presents a problem where the embedding matrix constitutes the majority of the parameters of the neural network. The embedding dimension is smaller (256) compared to the other two tasks but embedding compression yields a BLEU score increase in all methods except Structured Embedding. This is due to a regularization effect from the compression. Our model again achieves the highest BLEU score.

On these three experiments we demonstrate that our funneling decomposition method with embedding distillation consistently yields higher BLEU scores compared to existing methods.

Model	Emb. CR	Val. PPL	Test PPL
Transformer-XL std (Dai et al., 2019)	1.0x	23.23	24.16
SVD (rank 64)	3.23x	25.34	26.51
Distilled Emb (rank 64)	3.23x	**24.88**	**25.75**
SVD (rank 32)	6.47x	27.06	27.91
Distilled Emb (rank 32)	6.47x	**26.15**	**27.46**

Table 2: Language Modeling perplexity for WikiText-103 on validation and test sets. We compressed the embedding matrix from 151M parameters to 34M (3.23x) and 17M (6.47x) parameters. Std is an abbreviation of the word Standard.

5.2 Language Modeling

As a second task we consider language modeling on the WikiText-103 dataset. We compare our method against SVD with two compression rates. The results are presented in Table 2. We demonstrate that our distilled embedding method consistently yields

Model	Emb. CR	Init.	No Distill.	Emb. Distill.
En-Fr	7.87x	Random	37.04	37.21
En-Fr	7.87x	Model	37.54	**37.78**
En-De	7.89x	Random	26.07	26.35
En-De	7.89x	Model	26.7	**26.97**
Pt-En	3.96x	Random	42.29	42.36
Pt-En	3.96x	Model	42.5	**42.62**

Table 3: Comparison of different methods for Funneling (64).

lower perplexity (PPL) compared to SVD.

5.3 Ablation Study

We present different experiments on machine translation to demonstrate the effect of 1) Model Initialization, 2) Embedding Distillation, 3) Fine-tuning strategies, 4) Compression capability, 5) Alpha Value Sensitivity and 6) Extension and generality of our method.

Initialization We do an ablation study on all the three language pairs defined in Section 4.1, to conclude, if random initialization is better than model-based initialization. We conclude that model-based initialization, consistently performs better (Table 3).

Embedding Distillation Table 4 presents different compression rates on the Pt-En task, and embedding distillation performs better across all of them. In Table 3, we see that across all language pairs when we initialize our model using weights from the funneling decomposition, we improve when using Embedding Distillation during finetuning. We performed embedding distillation with random initialization only on the smaller Pt-En dataset and observed that Embedding Distillation improves BLEU score even with random initialization.

Compression Rate We demonstrate in Table 4 that it is possible to compress the embedding up to 15.86x with only a 2% drop in BLEU score for Pt-En.

Re-training Fine-tuning is an important component in our method and we demonstrate through our experiments that at convergence most of the techniques are close in performance. Table 5 shows that freezing embedding weights and re-training just the network weights or vice versa leads to a sharp drop in BLEU score, thus, we need to re-train all the weights. The use of a non-linearity and adding embedding distillation also improves BLEU score after finetuning.

Params	Emb. Params	Emb. CR	No Distill.	Emb. Distill.
11M	8M	1.0x	**41.43**	-
5M	2M	3.96x	42.50	**42.62**
4M	1M	7.93x	42.44	**42.60**
4M	516k	15.86x	40.42	**40.60**

Table 4: BLEU scores for different compression rates with bottleneck sizes of 64, 32 and 16 accordingly for IWSLT Pt-En.

Model	BLEU
Proposal	**42.60**
- embedding distillation	42.44
- non-linearity	42.34
Proposal (Freeze non-emb. weights)	33.34
Proposal (Freeze emb. weights)	20.49

Table 5: BLEU score for IWSLT Pt-En with compression rate 7.93x.

Alpha (α) Value Sensitivity Analysis We performed a sensitivity analysis on the α hyper-parameter introduced by our method. Table 6 presents our findings. We can see that the method is not very sensitive to the change in α value. We did not tune the alpha for our different experiments but chose the value which gave us good validation results on the WMT En-De translation task. The results of this analysis suggest that we can gain a little performance if we tune alpha for every dataset.

Extension We experimented with applying two key lessons from our method, namely, using a non-linear function and embedding distillation, to a model initialized with group partitions of the GroupReduce method (Chen et al., 2018), we refer to this method as *GroupFunneling*. Table 7 shows that, *GroupFunneling* achieves a higher BLEU score on Pt-En compared to GroupReduce.

6 Discussion

Importance of Non-linearity We postulate that only a subset of word vector dimensions, explains most of the variance, for most word vectors in the

Alpha	BLEU
0	42.50
0.01	42.62
0.1	42.65
0.3	42.66
0.5	42.72
0.7	42.57
0.9	42.03

Table 6: Alpha value sensitivity analysis on IWSLT Pt-En.

Model	BLEU
GroupFunneling (Rand. Initialized + Emb. Distil.)	**42.52**
GroupFunneling (Rand. Initialized)	42.49
GroupReduce	42.13

Table 7: GroupFunneling (i.e. GroupReduce + Funneling) on IWSLT Pt-En.

embedding matrix. Thus, using ReLU activation might help in regularizing the less important dimensions for a given word vector.

Importance of Reconstruction Loss We propose that the embedding reconstruction might suffer from adding the ReLU activation function. The consequence would be loss of information on words not seen during training and loss of generalization performance. Thus, adding a loss for embedding reconstruction helps in grounding the embedding and not lose a lot of information. The amount of regularization is controlled by the hyperparameter α. Our intuition is partly justified by results shown in Table 5, as reconstruction loss performs worse without the ReLU activation function.

Comparison of Inference Speed We compare the number of floating-point operations used by different models. Table 8 presents these results. As it is expected, our method is slightly slower than plain SVD method due to the use of the non-linear activation function and the bias additions but notably faster than other more complex methods. Structured embedding does not use any additional floating-point operations, though it requires $groups - 1$ additional embedding lookup and concatenate operations. Also, structured embedding requires the reconstruction of the entire embedding matrix at the output projection layer, making it ineffective for model compression.

Model	Approx. GFLOPs
SVD	1.21
Distilled Embedding	1.22
Tensor Train	2.18
GroupReduce	3.41

Table 8: Approximate GFLOPs on reconstructing the WMT En-De embedding matrix with size $[37000 \times 512]$ and compression rate 7.89x.

Model	Inference Time (Sec)
Base Model	27.92
SVD	29.63
Structured Embedding	31.18
Distilled Embedding	29.23

Table 9: Average inference speed on the IWSLT PT-En model with compression rate 3.96x.

In addition, we demonstrate on Table 9 the average inference time needed for each method to do a forward pass on the IWSLT Pt-En validation dataset which has a size of 7590 examples. We used a single NVIDIA P100 GPU (12GB) with a batch size of 1024. We averaged the time for 30 runs. We did not perform experiments on GroupReduce and Tensor Train, but according to the Table 8 we are expecting these methods to be even slower.

7 Conclusion and future work

In this paper we proposed Distilled Embedding, a low-rank matrix decomposition with non-linearity in the bottleneck layer for a shared word-embedding and vocabulary projection matrix. We also introduce knowledge distillation of the embedding during fine-tuning using the full embedding matrix as the teacher and the decomposed embedding as the student. We compared our proposed approach with state-of-the-art methods for compressing word-embedding matrix. We did extensive experiments using three different sizes of datasets and showed that our approach outperforms the state-of-the art methods on the challenging task of machine translation. Our method also generalized well to the task of language modeling. For future work, we will apply our approach to compress feed-forward and multi-head attention layers of the transformer network.

References

Anish Acharya, Rahul Goel, Angeliki Metallinou, and S. Inderjit Dhillon. 2019. Online embedding compression for text classification using low rank matrix factorization. In *AAAI*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv: 2005.14165*.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Patrick H. Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 11011–11021.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

Valentin Khrulkov, Oleksii Hrinchuk, Leyla Mirvakhabova, and Ivan Oseledets. 2019. Tensorized embedding layers for efficient model compression. *arXiv preprint arXiv:1901.10787*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao, and Shuang Wu. 2018. Slim embedding layers for recurrent neural language models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5220–5228. AAAI Press.

Vasileios Lioutas and Yuhong Guo. 2020. Time-aware large kernel convolutions. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. 2015. Tensorizing neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 442–450.

Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Kaiyu Shi and Kai Yu. 2018. Structured word embedding for low memory neural network language model. In *Interspeech*, pages 1254–1258.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representations*.

Nathan Srebro and Tommi Jaakkola. 2003. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 720–727.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

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*, abs/1908.09982.

A Appendices

A.1 Additional Hyper-parameters

WMT En-Fr Smaller Transformer Network denotes a network with the same configuration as Transformer Base but with hidden size d_{model} of 416. For GroupReduce, to match the same compression rate we used number of clusters c being equal to 10 and minimum rank r_{min} to be 22. For SVD, we decided to set the rank to 64. For Tensor Train, we set the embedding shape to be $[25, 32, 40] \times [8, 8, 8]$ and the Tensor Train Rank to be 90. For structured embedding we use group size as 32 and number of clusters as 2048, we then use the quantization matrix and learn the clusters from scratch.

WMT En-De Smaller Transformer Network denotes a network with the same configuration as Transformer Base but with hidden size d_{model} of 400. For GroupReduce, to match the same compression rate we used number of clusters c being equal to 10 and minimum rank r_{min} to be 23. For SVD, we decided to set the rank to 64. For Tensor Train, we set the embedding shape to be $[25, 37, 40] \times [8, 8, 8]$ and the Tensor Train Rank to be 90. For structured embedding we use group size as 32 and number of clusters as 2376, we then use the quantization matrix and learn the clusters from scratch.

IWSLT Pt-En Smaller Transformer Network denotes a network with the same configuration as Transformer Small but with hidden size d_{model} of

Parameters	Embedding	FFN	Multi-head attention	Linear
Number	26M	25M	14M	5M
Percentage	37%	36%	20%	7%

Table 10: Parameters in the Transformer Base model (Vaswani et al., 2017) based on a 50k dictionary size and tied input and output embedding.

136. For GroupReduce, to match the same compression rate we used number of clusters c being equal to 15 and minimum rank r_{min} to be 30. For SVD, we decided to set the rank to 64. For Tensor Train, we set the embedding shape to be $[25, 32, 40] \times [8, 4, 8]$ and the Tensor Train Rank to be 125. For structured embedding we use group size as 32 and number of clusters as 4048, we then use the quantization matrix and learn the clusters from scratch.

A.2 Parameter count

Table 10 presents the the number of parameters in the different transfomer layers for the transformer base architecture.