INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Factorization-Aware Training of Transformers for Natural Language Understanding On the Edge

Hamidreza Saghir, Samridhi Choudhary, Sepehr Eghbali, Clement Chung

Amazon Alexa AI, Canada
{hssaghi, samridhc, eghbali, chungcle}@amazon.com

Abstract

Fine-tuning transformer-based models have shown to outperform other methods for many Natural Language Understanding (NLU) tasks. Recent studies to reduce the size of transformer models have achieved reductions of $> 80\%$, making on-device inference on powerful devices possible. However, other resource-constrained devices, like those enabling voice assistants (VAs), require much further reductions. In this work, we propose factorization-aware training (FAT), wherein we factorize the linear mappings of an already compressed transformer model (DistilBERT) and train jointly on NLU tasks. We test this method on three different NLU datasets and show our method outperforms naive application of factorization after training by $10\% - 440\%$ across various compression rates. Additionally, We introduce a new metric called factorization gap and use it to analyze the need for FAT across various model components. We also present results for training subsets of factorized components to enable faster training, re-usability and maintainability for multiple on-device models. We further demonstrate the trade-off between memory, inference speed and performance at a given compression-rate for a on-device implementation of a factorized model. Our best performing factorized model, achieves a relative size reduction of 84% with $\approx 10\%$ relative degradation in NLU error rate compared to a non-factorized model on our internal dataset.

Index Terms: NLU, Factorization, Model Compression

1. Introduction

With a growing number of users relying on voice powered digital assistants (VAs) for their day-to-day activities [1, 2], aspects like privacy, faster response time, accurate responses, and availability without internet connection, have become critical to providing a delightful customer experience. 'On device' or 'Edge' processing has been a important enabler, and an increasing number of commercial VA systems are moving to edge [3–6]. The technology powering these VAs is Spoken Language Understanding (SLU), which is the task of extracting meaning from a spoken utterance. Natural Language Understanding (NLU) is a key part of the SLU pipeline that predicts the semantics (domain, intent and slots) from the utterance transcript. In order to fit the stringent resource constraints of on-device systems, candidate models need to be compressed. Moreover, modest compute resources restrict the kind of architectures that can be deployed on edge. This precludes the use of large transformer based models like BERT [7], GPT2 [8] and XLNet [9] for on-device NLU. However, given the high predictive performance of transformers, there have been many efforts to compress these models [10–14]. Studies focusing on on-device inference have investigated knowledge distillation (KD) [15–17]) or matrix factorization (MF) [18–25] as popular hardware agnostic approaches to compress neural models. These studies report a $35\% - 87\%$ size reduction, a 2x-4x speed up and $< 3\%$ accuracy loss. However, for many VA devices like Alexa Echo Dot and FireTV, orders of magnitude smaller and faster models are required.

In this work, we investigate compression of a transformer model for performing NLU, suitable for stringent on-device constraints. We propose a model compression approach based on MF, that can be applied to an already compressed model, leading to further compression with minimal loss in performance. In particular, we use the commonly used DistilBERT model [15] as a starting point and factorize its linear layers. This is done in a task-aware fashion, where the components are learned along with the downstream tasks. Matrix factorization has been studied before as a method for compressing neural networks [19, 23, 26], however, to our knowledge, task-aware factorization for on-device transformer-based NLU models has not been investigated. Even though we showcase our results on DistilBERT, our method is generic and can be applied to any transformer based model. We test our method on 2 tasks and 3 datasets and show consistent improvements across all.

The main contributions of this work are as follows: (1) We propose a 'factorization-aware training' (FAT) method for transformer-based models that helps the model recover lost performance encountered when factorization is done naively post-training; (2) We introduce a new metric called 'factorization-gap' to measure the need for FAT for various model components; (3) We present an approach for improving training speed, re-usability, and maintainability by proposing to fine-tune only subsets of parameters in a factorized model. This allows multiple on-device transformer-based models to share some parameters, leading to further reduction in disk space and training time; and 4) We demonstrate a method to perform a trade-off analysis for various implementation options between model-size, inference time and performance for a given compression rate of the model, providing a blueprint for design decisions when deploying factorized models on device.

2. Methodology

2.1. Model architecture

We design a transformer-based multi-task model to perform NLU, consisting of two categories of tasks - sentence classification, and sequence tagging. The exact tasks can vary for different NLU datasets. Figure 1 shows the basic architecture that we use. The shared layer consists of a DistilBERT encoder that takes in an utterance text as input. Task specific layers (heads) are added on top of the last layer of the shared encoder. The sentence classification task(s) consist of fully-connected (FC) layers where the input is the last layer encoded representation of the [CLS] token. The sequence tagging task(s) use the last encoder layer representation for all tokens that are passed on to an FC head at each token position, to predict an entity tag for each word.

http://dx.doi.org/10.21437/Interspeech.2021-1816

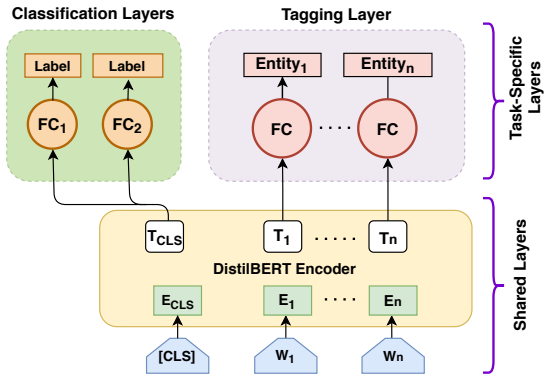

Figure 1: *Multi-Task Transformer Based Architecture for NLU using DistilBERT. For each NLU dataset, we replace the task-specific heads accordingly. Each FC_i head in the Classification Layer can be a distinct sentence classification task.*

The entire network is trained on a joint loss composed of sum of sentence level cross-entropy losses for classifiers, and an average token-level cross-entropy loss for tagging tasks (eq. 1).

$$L = \sum_{i}^{N_{cls}} \sum_{j}^{N_{tag}} (w_i \, l_i^{cls} + w_j \, l_j^{tag}) \tag{1}$$

where N_{cls}, and N_{tag} determine the number of classifier and tagging heads respectively, and w_i, and w_j are corresponding weights for each loss term (l).

2.2. Transformer factorization

A distilled version of a full-sized BERT model, is still too large to be feasible for on-device deployment, especially for devices with stringent resources. For example, for an NLU memory budget of 30MB-50MB, the model needs to have $< 7.5 - 12.5$ million parameters. We propose further compression of the DistilBERT model using SVD decomposition of all linear layers in the network. More specifically, we apply SVD to decompose the weight matrices of all embedding, attention, and FF layers of the underlying transformer model along with the linear layers of task-specific heads. In this formulation, a weight matrix $W_{m \times n}$ is expressed as a product of three matrices as shown below:

$$W = U_{m \times d} S_{d \times d} V^T_{d \times n} \tag{2}$$

where S is a diagonal matrix with d singular values. In order to find the rank $r \leq \min(m, n)$ approximation of W with minimum Frobenius error, we only keep the r largest singular values and their corresponding columns from U and V. In this setup, r specifies the compression rate that we are willing to achieve. In particular, we define **Rank Factor (RF)** $0.0 \leq RF \leq 1.0$ as the percentage of singular values to preserve for each weight matrix (i.e. $d = RF \times \min(m, n)$). SVD automatically assigns importance (the singular values) to the vectors in the decomposed matrices that enable optimal low rank approximation of the original matrix.

If factorization is performed naively after training is finished, the performance of the model will suffer (empirically shown later). In order to solve this problem, we propose **factorization-aware training (FAT)**, where the weight matrices of all linear mappings in a the model are factorized, and represented with lower rank matrices before any training occurs.This setup gives the model a chance to learn to use lower-rank matrices in linear layers and thus recoup some of the lost performance resulting from matrix factorization. Additionally, lower rank components in the network could have a regularizing effect during training due to the reduced number of parameters, that in turn can help with better performance on smaller datasets. Specifically for our architecture (figure 1), we first decompose each weight matrix in the pre-trained model to the three U, S, and V matrices, pick the top RF% of singular values, and eliminate the rest. We then fine-tune the resulting model on our NLU task and update the factorized weights per linear mapping. This results in a model that is inherently factorized with smaller U, S, and V matrices that have been trained to perform the task with minimal performance loss.

2.3. Factorization gap

In order to measure how necessary it is to perform FAT for various components of a model, we introduce a metric called **factorization gap** (ϱ). This metric measures the amount of change that the model parameters undergo in a FAT setup, compared to their values in a post-training factorization setup. The more factorization gap a component has, the more there is a need for FAT. We calculate ϱ as shown in the equation below -

$$\varrho = 1 - \left| \frac{1}{N} \sum_{i}^{N} \frac{\varphi_i \cdot \tilde{\varphi}_i}{|\varphi_i| \cdot |\tilde{\varphi}_i|} \right| \tag{3}$$

where $\tilde{\varphi}_i$ refers to the feature vector weight of the i^{th} component of the model where factorization is applied post training and φ_i is the corresponding weight of the component in a model trained using FAT.

Factorization gap values range between $0 \leq \varrho \leq 1$. The closer the factorization gap is to 1, the more the parameters have to change to produce FAT-equivalent results. For example, at $\varrho = 0$, the factorization gap for a model component is zero, therefore, there is no need to do FAT and performing post-training factorization for that component will provide equivalent results.

3. Experimental Setup

3.1. Data

We experiment with three different NLU datasets to show the efficacy and generalizability of this approach. This includes 2 publicly available datasets - ATIS [27] and MNLI [28], and an internal NLU dataset

Internal NLU Dataset - We use a random snapshot of live traffic of a commercial VA to create this dataset. The data is de-identified to remove any personal information. Each instance consists of an utterance text, along with the domain, intent and the entity tags. The dataset includes $\approx$ 165 hours worth of training data across more than 100 intents and entity types. We similarly curate our validation and test set with $\approx$ 16 hours and $\approx$ 85 hours worth of data.

Both ATIS and MNLI are publicly available datasets. **ATIS** [29] is an NLU dataset with intent and slot annotations for utterances related to queries for a flight reservation system. We used the same version of ATIS as in [30]. **MNLI** [28] is a collection of 433k sentence pairs manually labeled for entailment classification and is used for building Natural Language Inference (NLI) systems. We use the official train/test split of the datasets but split off 10% of training samples for validation purposes. The dataset comes with two test sets, *matched* and *mismatched*, but due to space limitations we only report the results on the matched set.

Table 1: *Model performances (error rates) across varying rank factors and compression for the models in the three experimental setups of post-training factorization, factorization-aware training(FAT), and FAT with frozen-U parameters. Numbers for the internal dataset are reported as relative % change due to privacy concerns, while for public datasets (ATIS , MNLI) are reported as absolute change.*

Rank Factor	Compression Ratio	Post-Training Factorization			FAT			FAT (Frozen-U)		
		IRER % (Internal)	**IRER** % (ATIS)	**CER** % (MNLI)	**IRER** % (Internal)	**IRER** % (ATIS)	**CER** % (MNLI)	**IRER** % (Internal)	**IRER** % (ATIS)	**CER** % (MNLI)
None	1.0	–	4.31	38.78	–	4.31	38.78	–	4.31	38.78
1.00	0.75	0.00	+2.09	+3.58	-9.17 %	+2.59	+1.31	-0.61 %	+3.45	+0.19
0.75	0.93	+34.08%	+5.29	+6.97	-7.44 %	+0.86	+1.88	+8.82 %	+3.45	+0.21
0.50	1.24	+100 %	+16.49	+9.80	-3.81 %	+4.31	+2.16	+15.4 %	+4.31	+0.90
0.25	1.86	+278.98 %	+95.69	+19.63	+7.18 %	+6.90	+2.84	+17.47 %	+5.17	+2.13
0.10	4.65	+291.87 %	+95.69	+26.04	+10.3 %	+12.07	+2.91	+42.73 %	+11.37	+2.39
0.01	46.99	+763.93 %	+95.69	+28.96	+57.8 %	+30.17	+5.09	+99.22 %	+29.62	+4.50

ATIS and the internal dataset consist of both classification (Domain and/or Intent Classification (DC/IC)), and tagging tasks (Named Entity Recognition (NER)), with ATIS containing just IC and NER and the internal set containing IC, DC and NER. MNLI is just a sentence classification task. Equation 1 defines the loss for each dataset based on its prediction tasks.

3.2. Model evaluation

We download the pre-trained DistilBERT model from [31] to use as our shared encoder and fine-tune on our datasets. The fine-tuned models without any factorization act as our baselines. In order to show the efficacy of this approach, we perform our experiments in three different settings:

Post-Training Factorization (Post-TF) - we naively factorize the weights of all the linear layers in the model after it has been fine-tuned on the NLU task.

Factorization-Aware Training (FAT) We use FAT to train our model as explained in section 2.2

FAT with Frozen U (FAT-FU) - Similar to the setup in FAT, we factorize and learn lower rank matrices for each linear layer, but we keep the matrix U frozen in the fine-tuning stage.

We systematically experiment with reducing the RFs across all our datasets and note the model performance. All the models are trained using Adam Optimizer with early stopping on the validation loss with a threshold of 5. We compute and report following metrics to measure the model performance:

Interpretation Error Rate (IRER): Ratio of number of incorrect 'interpretations' to the total number of utterances. An incorrect interpretation is one where either the domain or the intent or any of the slots are wrong. We use this measure for ATIS and our internal NLU dataset.

Classification Error Rate (CER): - Ratio of incorrect class predictions to the total number of utterances. We use this measure for the MNLI dataset.

4. Results

4.1. Empirical evaluations

Table 1 shows the model performance across different RFs on three datasets. As can be seen, in the Post-TF setting, the model catastrophically loses its prediction capability as lower RFs are used, across all datasets. For example, for $RF = 0.01$ on the internal dataset, the IRER jumps a whopping $\approx 763.93\%$ in Post-TF compared to baseline, while for FAT, the increase in IRER is $\approx 57.87\%$. A similar degradation is seen for ATIS with $RF < 0.5$,. MNLI shows a less drastic degradation with a maximum of 74.68% loss in performance at $RF = 0.01$. This is still significantly higher than its corresponding FAT performance at the same RF for both unfrozen FAT ($\approx 13.13\%$) and FAT with Frozen U (11.6%). The performance lost across all datasets in Post-TF is consistently recovered using FAT for both frozen and unfrozen U settings. However, as the compression ratio increases, the model performance, even in the FAT setup declines. Albeit, the decrease in performance is less dramatic when compared to the decrease in the total number of parameters. For $RF = 0.25$, the model has $< 50\%$ of the total parameters of the original model, however there is only $\approx 7\%$ relative decrease in performance when using unfrozen FAT in both the internal and MNLI datasets. Our internal dataset sees the smallest degradation in both frozen and unfrozen setting in this regard. A noteworthy observation here is that for the internal dataset, at $RF = 0.5$, where the model has $\approx 60\%$ of the original parameters, the performance actually improves by 3.81% using FAT. This is probably due to the regularization effects of reduced number of parameters in the network. For a model with just $\approx 2\%$ ($RF = 0.01$) of the original parameters, MNLI sees 13.13% degradation in the FAT setting. The decrease in performance for ATIS and internal set are also not as big as the reduction in model size, especially in the unfrozen FAT setting, indicating that FAT is a promising way to achieve smaller model footprints without significant loss in model performance. In the frozen U setting for FAT, the change in model performance for the MNLI dataset, across all RFs, is $< 12\%$, with better performance compared to the non-frozen counterpart. This is again probably a result of regularization effects of having less learnable parameters in the network due to freezing and reduced rank factors. This indicates that portions of the factorized model can remain fixed and possibly be shared with other tasks, leading to smaller total footprint, shorter training times, and improved reusability.

4.2. Factorization Gap

Figure 2 demonstrates the average factorization gap, ϱ, for various components of transformer along the network depth. Top panel shows that ϱ for the self-attention layers (q, k, v) decreases as we go deeper. However the ϱ for FC layers (attention out, $lin1$, and $lin2$) follows an increasing trend along the depth (bottom panel). This indicates that the FC layers in deeper layers rely more on FAT to preserve the network's performance compared

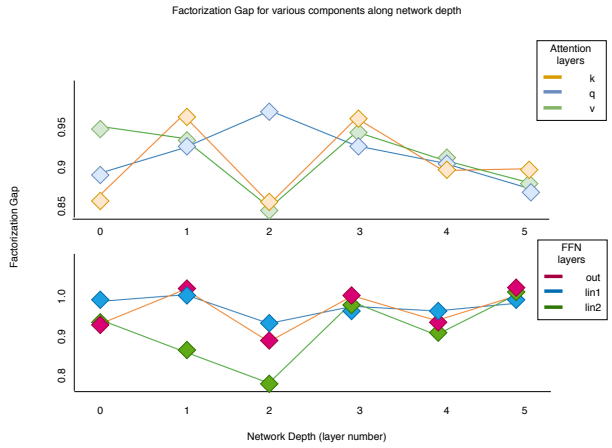

Figure 2: *Factorization gap (ϱ) along network depth- Top panel shows that the ϱ in attention layers get smaller along network depth, while in the FC components ϱ gets larger in deeper layers (bottom panel)*

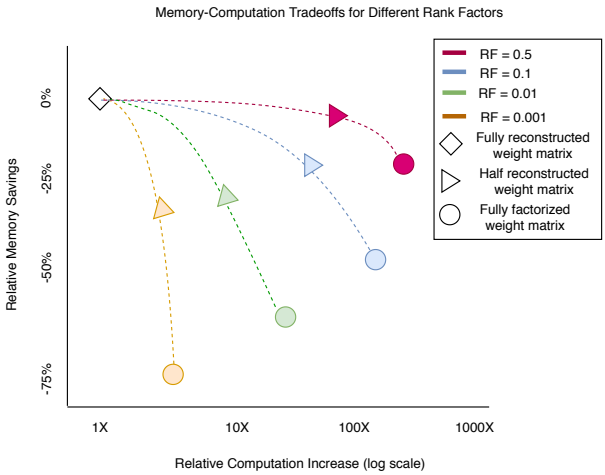

Figure 3: *Relative memory consumption and computation cost at inference with weight matrices at - fully factorized, half reconstructed and fully reconstructed - across various RFs. For a low RF (RF=0.001) the memory saving is 45% at half reconstruction and 80% for full factorization while incurring only 6X and 7X computation cost, respectively. For higher RF (RF=0.5), computational cost increases exponentially for smaller memory saving.*

to the self-attention layers. Therefore, if we were to partially freeze some components of the network, it is better to freeze larger portions of the FC parameters in earlier layers, and do the opposite for the attention layers. Additionally, lower ϱ for attention components in deeper layers can potentially indicate an inherent lower rank for these layers, since less changes are required to recover the lost performance. This metric can therefore help with designing future FAT setups that use adaptive rank factors along the network depth to obtain models with smaller footprint and improved training time.

4.3. Adaptive model loading

After a factorized model is trained, all the weights of the model can be stored on disk using a lower-rank representation of the components saving disk space. However, at inference, the parameters need to be loaded into memory. Depending on the memory and computational constraints of a device, one can choose to either load the factorized components and reconstruct them in real-time or cache the fully reconstructed matrices in memory. The common way to load the model in memory is to cache reconstructed matrices. In contrast, if only the factorized matrices are cached, memory usage will be lower at the cost of higher computation for reconstruction on-the-fly. This insight provides an opportunity adaptively load factorized models into memory and trade-off memory for computation based on available resources. To demonstrate this trade-off, we calculate the memory footprint and FLOPs of the model for various RFs, based on the percentage of the model weights that are loaded into memory as factorized components. As can be seen in Figure 3, different RFs provide different trade-offs. As lower RFs are used, a better trade-off is admitted where not as much FLOPS are needed to achieve lower a memory footprint. For example, for $RF = 0.1$, if all the reconstructed weight matrices are cached, the model will have a memory footprint of $\approx$ 254 MBs, with $\approx$ 10.14 Giga FLOPs required at inference time. Whereas, if only half of the reconstructed weights are cached in memory, the model will consume half as much memory but require $\approx$ 70 times more computational resources at inference time. This trade-off becomes exponentially more favorable with lower RFs. This is primarily because lower rank decomposition requires less number of FLOPs to reconstruct a matrix. For example, with $RF = 0.05$, if half of the reconstructed weights are cached, only $\approx$ 2 times more FLOPS would be needed at inference. This is favorable if the device has idle CPU/GPU time, while available memory is limited. Also, one can observe that for larger RF values, as the memory footprint decreases, the computational overhead becomes prohibitive. Thus, the an adaptive model loading strategy is more applicable when smaller RF values are used. This analysis sheds light on possible design choices for an on-device factorized model deployment and helps in deciding what RF to use in trading off model performance with on-device inference requirements. Note that we have not considered possible hardware optimizations that may change the relationship between memory and computation here.

5. Conclusions

We introduce factorization-aware training (FAT) as a method for compressing transformer-based models for on-device use-cases. FAT learns the factorized weights of linear layers along with downstream NLU tasks, resulting in minimal performance degradation. We apply 3 different factorization techniques and test their performance in a series of experiments. Our results show that FAT can recoup the performance lost in post-training factorization setting, under both unfrozen and frozen-U settings. We study the memory-computation-accuracy trade-off across different RF choices and propose an adaptive model loading strategy for small rank factors. We also analyze the need for FAT across different components of the network, by defining a new metric, called factorization gap and show that in deeper layers, FC layers are more dependent on FAT compared to attention layers. We focused our attention on analyzing model performance using fixed RFs. In future we plan to experiment with adaptive optimization of RFs per layer. Finally, we note that our proposed method (FAT) is a parallel strategy to other model compression techniques and can be used alongside other methods.

6. References

[1] B. F. Rubin. (2020, Jan) Amazon's alexa world just got much bigger. [Online]. Available: https://www.cnet.com/home/smart-home/amazon-sees-alexa-devices-more-than-double-in-just-one-year

[2] J. England. (2020, Feb) Amazon dominates the smart speaker market with a nearly 70% share in 2019. [Online]. Available: https://www.androidcentral.com/amazon-dominates-smart-speaker-market-nearly-70-share-2019

[3] K. Mysore Sathyendra, S. Choudhary, and L. Nicolich-Henkin, "Extreme model compression for on-device natural language understanding," in *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*. Online: International Committee on Computational Linguistics, Dec. 2020, pp. 160–171. [Online]. Available: https://www.aclweb.org/anthology/2020.coling-industry.15

[4] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.

[5] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays *et al.*, "Personalized speech recognition on mobile devices," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 5955–5959.

[6] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[9] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[10] P. Ganesh, Y. Chen, X. Lou, M. A. Khan, Y. Yang, D. Chen, M. Winslett, H. Sajjad, and P. Nakov, "Compressing large-scale transformer-based models: A case study on bert," *ArXiv*, vol. abs/2002.11985, 2020.

[11] F.-M. Guo, S. Liu, F. Mungall, X. Lin, and Y. Wang, "Reweighted proximal pruning for large-scale language representation," *ArXiv*, vol. abs/1909.12486, 2019.

[12] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *ArXiv*, vol. abs/1909.11556, 2020.

[13] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert." *arXiv preprint arXiv:1910.06188*, 2019.

[14] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," *AAAI Conference on Artificial Intelligence*, vol. 34, no. 5, pp. 8815–8821, 2020.

[15] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[16] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[17] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," in *Conference on Empirical Methods in Natural Language Processing*, 2019, pp. 4322–4331.

[18] Z. Wang, J. Wohlwend, and T. Lei, "Structured pruning of large language models," *arXiv preprint arXiv:1910.04732*, 2019.

[19] A. Acharya, R. Goel, A. Metallinou, and I. S. Dhillon, "Online embedding compression for text classification using low rank matrix factorization," in *AAAI*, 2019.

[20] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Interspeech*, 2013.

[21] U. Thakker, J. Beu, D. Gope, G. Dasika, and M. Mattina, "Rank and run-time aware compression of nlp applications," *arXiv preprint arXiv:2010.03193*, 2020.

[22] A. M. Rufai, G. Anbarjafari, and H. Demirel, "Lossy image compression using singular value decomposition and wavelet difference reduction," *Digital Signal Processing*, vol. 24, pp. 117–123, 2014.

[23] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, "Sparse low rank factorization for deep neural network compression," *Neurocomputing*, vol. 398, pp. 185–196, 2020.

[24] H. Kim, M. U. K. Khan, and C.-M. Kyung, "Efficient neural network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5970–5974.

[26] Z. Kaden, T. L. Scao, and R. Olivier, "In-training matrix factorization for parameter-frugal neural machine translation," *ArXiv*, vol. abs/1910.06393, 2019.

[27] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The atis spoken language systems pilot corpus," in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '90. USA: Association for Computational Linguistics, 1990, p. 96–101. [Online]. Available: https://doi.org/10.3115/116580.116613

[28] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018, pp. 1112–1122. [Online]. Available: http://aclweb.org/anthology/N18-1101

[29] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The atis spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

[30] C.-W. Goo, G. Gao, Y.-K. Hsu, C.-L. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, "Slot-gated modeling for joint slot filling and intent prediction," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 753–757.

[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.