Bottleneck Low-rank Transformers for Low-resource Spoken Language Understanding

Pu Wang, Hugo Van hamme

Department of Electrical Engineering-ESAT, KU Leuven, Belgium

pu.wang@esat.kuleuven.be, hugo.vanhamme@esat.kuleuven.be

Abstract

End-to-end spoken language understanding (SLU) systems benefit from pretraining on large corpora, followed by fine-tuning on application-specific data. The resulting models are too large for on-edge applications. For instance, BERT-based systems contain over 110M parameters. Observing the model is over-parameterized, we propose lean transformer structure where the dimension of the attention mechanism is automatically reduced using group sparsity. We propose a variant where the learned attention subspace is transferred to an attention bottleneck layer. In a low-resource setting and without pre-training, the resulting compact SLU model achieves accuracies competitive with pre-trained large models.

Index Terms: low-rank transformer, bottleneck attention, low-resource spoken language understanding

1. Introduction

Spoken language understanding (SLU) systems that infer users' intents speech draw increased attention since the adoption of voice assistants such as Apple Siri and Amazon Alexa [1, 2, 3]. A traditional SLU system is the pipeline of an automatic speech recognition (ASR) component which decodes transcriptions from input speech and a natural language understanding (NLU) component which extracts meaning (intents) from the decoded transcriptions. To address several drawbacks of the pipeline structure such as error propagation, more and more research moves to end-to-end (E2E) SLU systems that directly map speech to intents.

An E2E system is usually built with complex neural networks for sequence modeling and requires massive amounts of training data to outperform the traditional cascaded ASR-NLU systems. However, in-domain intent-labeled speech data is scarce compared with ASR and NLU corpora. Therefore, an emerging research interest is joint training or pre-training an E2E SLU system with a variety of related tasks including ASR [4, 5, 6, 7, 8], NLU [4, 6, 7], masked LM prediction [4, 7], and hypothesis prediction from text [4, 5]. The most successful designs are transformer-based large-scale models [8] including the pre-trained BERT model [9, 10] and the default E2E ASR configuration in ESPnet [7]. These models are huge and rely on large computational resources. For instance, the multilingual BERT-base model has 110M parameters and the GPT-2 model has 117M parameters [11]. They are typically too large to store on an edge device thus requiring a cloud server, raising connectivity and privacy concerns. Moreover, large-scale models cause over parameterization issues when fine-tuning for low-resourced SLU tasks. This motivates work on compact SLU systems that run on the edge device itself.

One traditional way to reduce model size involves factorized matrix representations [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]. Chen et al. [21] apply this idea to linear projection layers in transformers. Akin to factorized TDNN models with the bottleneck layers, one linear projection layer with $m \times n$ weight matrix can be replaced with two stacked linear layers: A of shape of $m \times r$ stacked on B of shape $r \times n$, with rank $r \ll m, n$. This way, the number of parameters reduces from $n \times m$ to $r \times (n + m)$. Further research on low-rank approximations in transformer attention weight matrices chooses a fixed value for the rank of the factorization, i.e. the dimension of the bottleneck layer r, e.g. [14, 20], which led to performances degradation. Others utilize matrix factorization (MF) with knowledge distillations (KD). Mao et al. [15] and Saghir et al. [18] firstly pretrain a (Distil)BERT model and then compress all linear layers in the (Distil)BERT model by using singular value decomposition (SVD) and pick the top r singular values as the factorized weights (r therefore is the chosen rank of the bottleneck layer). The weights are further fine-tuned on the target tasks. However, Chen et al. [22] proved that the weight matrices in transformer linear layers are not low rank, i.e. solely pre-training of a (Distil)BERT model will not always achieve a low-rank representation. Moreover, Panahi et al. [11] show that errors from stacking r-rank factorized linear transformations into a deep network add up which constrains the outputs by the smallest of ranks of the decomposed layers.

In this paper, we focus on designing a low-rank transformer structure whose dimensions are inferred by introducing a rank-related penalty term into the training. The well-trained model intrinsically holds a low-rank property without degradation of performance. The low-rank design is applied to the light transformer [23] in a SLU context with very scarce training data. The light transformer is a compact version of the vanilla transformer [24]. [23] shows a successful implementation of the compact low-resource SLU system built with the light transformer combined with the capsule network which has only 1.3M parameters but achieves 98.8% accuracy on FluentSpeech Command data without pre-training.

The light transformer is firstly trained on the task-specific dataset with a rank-related penalty to Key and Query to get the low-rank self-attention representations. After that, a bottleneck transformer is built by inserting a bottleneck linear projection layer for Keys and Queries in the original multi-head attention layer. The bottleneck layer and its dimension are determined from the low-rank self-attention representations of a well-trained low-rank light transformer. The bottleneck transformer is then retrained using the same task specific dataset. The proposed structure is tested on the public SLU tasks comparing with advanced larger-scale pre-training models including the BERT and ESPnet. The contribution of this paper is hence: (1) Presenting a successful approach for low-rank approximation in transformer attention without performance loss in low-resource SLU (2) Proposing a compact low-rank (bottleneck) transformer-based SLU structure trained only on the application-specific data, which has competitive performances

with advanced pre-training models.

In section 2, we explain the light transformer and rank-related penalty firstly. The bottleneck transformer and the overall structure of the designed SLU system are introduced in this section as well. Section 3 discusses the specific experimental setting for evaluation, and the corresponding results will be presented in section 4. In section 5 we will conclude our work.

2. Model

2.1. The baseline light transformer

The light transformer is a light version of the vanilla transformer which includes a low-dimensional (light) relative position encoding (PE) matrix [23]. We will shortly recap vanilla transformers first [24]. The vanilla transformer is a layer-stacked model with each layer containing one multi-head self-attention block and one feed-forward block. The output of the multi-head attention is a linear projection of the concatenated multiple scaled dot-product self-attention operations as shown in Eq. 1

$$MultiHead = W_{out}Concat(Attn^1, Attn^2, ...Attn^N) \quad (1)$$

In each attention head, the feature representation of the sequential data is first linearly transformed into a sequence of Keys, Values and Queries. The feature representation in the next layer is built as a non-linear mapping of a weighted average of the Values. The weight of each Value is determined by the similarity between the Key and the Query, as measured by a dot-product (Eq. 2).

$$Attn^n = \sum_j \text{softmax}(S_{ij})Vx_j \text{ with } S_{ij} = \frac{x_j^T K^T Q x_i}{\sqrt{d_k}} \quad (2)$$

where x is the feature representation or so-called content embedding. $Q, K, V \in \mathbb{R}^{d_k \times d}$ are weight matrices of the Query, the Key, and the Value respectively. d is the dimension of the content embedding x. d_k is the dimension of the transformer model. Softmax sums to unity over j.

To account for order information, the vanilla transformer will add a d-dimensional absolute PE to the content embedding x which requires the network to learn in which subspace relevant data variation occurs and in which subspace position is represented. To avoid the hassle brought by the additive high-dimensional PE, we introduced 6-dimensional relative position encoding in [23]. It is defined by Eq. 3.

$$p_t^T = \left[\cos\tfrac{2\pi t}{L} \ \sin\tfrac{2\pi t}{L} \ \cos\tfrac{2\pi t}{M_1} \ \sin\tfrac{2\pi t}{M_1} \ \cos\tfrac{2\pi t}{M_2} \ \sin\tfrac{2\pi t}{M_2}\right] \quad (3)$$

where t is the position, $L = 100$ to encode global sentence-level position, M_1 and M_2 are small integers which are normally chosen as 4 and 8 to provide sufficient temporal resolution at phone and word level (at a frame shift of 40 ms).

Compared with the absolute PE applied in the vanilla transformer, the 6-d PE is concatenated to the content embedding which pre-defines a content-related and a location-related weight matrices Q and K.

$$Q = \begin{bmatrix} Q_c & 0 \\ 0 & Q_p \end{bmatrix} K = \begin{bmatrix} K_c & 0 \\ 0 & K_p \end{bmatrix} \quad (4)$$

where Q_c and K_c are content-related weight matrices, Q_p and K_p are position-related weights. The attention output is therefore presented by Eq. 5.

$$S_{ij} = \frac{x_j^T K_c^T Q_c x_i}{\sqrt{d_k}} + \frac{p_j^T K_p^T Q_p p_i}{\sqrt{d_p}} \quad (5)$$

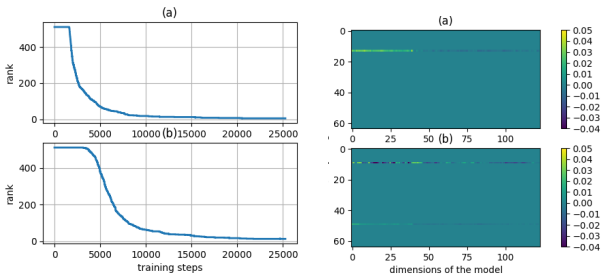


Figure 1: *The dynamic of ranks and values of the Q_c matrix training with the penalty scale (a) 0.0005, (b) 0.0001.*

To consider arbitrary relative relations between Query and Key, we further replace the absolute 6-d p with the relative PE p_{i-j} as shown in Eq. 6. $u \in \mathbb{R}^{d_p}$ is a trainable location-based parameter.

$$S_{ij} = \frac{x_j^T K_c^T Q_c x_i}{\sqrt{d_k}} + \frac{p_{i-j}^T u}{\sqrt{d_p}} \quad (6)$$

2.2. Low-rank light transformer

To achieve a low-rank attention model, group sparse regularization terms for Q_c and K_c are added to the total loss during training:

$$L_{gs} = \lambda_{gs} \sum_{\text{layers heads}} \sum_{k=1}^{d_k} \left(\sqrt{\sum_{d=1}^{d} {Q_c}_{k,d}{}^2} + \sqrt{\sum_{d=1}^{d} {K_c}_{k,d}{}^2} \right) \quad (7)$$

The group sparsity penalty will prefer to steer complete rows in Q_c and K_c to zero, thus reducing the rank of these matrices. Figure 1 shows how the rank decreases during training (left) and the resulting sparsity in Q_c (right) with different penalty scales λ_{gs}.

Normalisation with $\sqrt{d_k}$ in Eq. 2 is introduced in [24] to avoid gradient scale problems. When training with L_{gs}, the effective inner product dimension decreases during training. We therefore replace the scale parameter $\sqrt{d_k}$ of Eq. 6 with the dynamic rank of the weight matrices. The *total* rank of Q_c (K_c) is defined in Eq. 8. Here we only show the expression of Q_c since the ranks of these two matrices are always identical. Indeed, suppose Q_c has a zero row, any non-zero entry in K_c will not affect the transformer behavior, while the regularization term can be reduced by setting its value to zero. The roles of Q_c and K_c can be swapped in this argument.

$$r_{sum} = \sum_{\text{heads}} \sum_{k=1}^{d_k} \begin{cases} 1 \text{ if } \sum_{d=1}^{d} |{Q_c}_{k,d}| \geq 0.001 \\ 0 \text{ if } \sum_{d=1}^{d} |{Q_c}_{k,d}| < 0.001 \end{cases} \quad (8)$$

Notice that the individual ranks of the different heads and layers will be different if it leads to a lower loss. It is hence different from tuning the d_k parameter to a lower value.

2.3. Bottleneck light transformer

The heads in a transformer attend to different data properties. The low-rank light transformer results in very sparse attention matrices with only a few or no dimensions per head left. The idea of this section is to give the transformer a second opportunity to learn combinations of properties it finds useful in attention. We therefore introduce a bottleneck layer in the light trans-

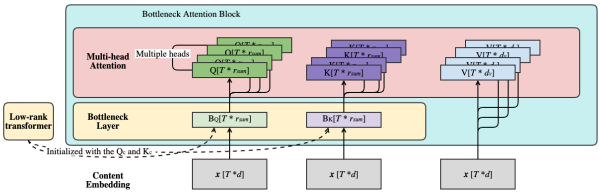


Figure 2: *Bottleneck transformer: bottleneck layer applies to Keys and Values.*

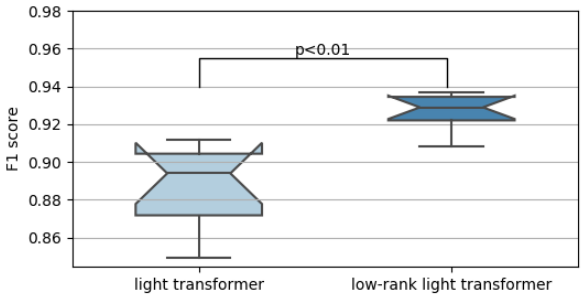


Figure 3: *F1 scores of light transformer and low-rank transformer (low rank with $\lambda_{gs} = 0.0005$, $r \approx 4$)*

former spanning the subspace deemed relevant by the group-sparse heads in the low-rank transformer as shown in the Figure 2. The bottleneck light transformer introduces a different bottleneck layer of dimension r_{sum} for Query and Key before the multi-head attention layer. The bottleneck is however common to all heads. The weight matrices of the bottleneck layer B_Q and B_K are initialized with the low-rank Q_c and K_c transferred from the low-rank light transformer respectively. We do not introduce a bottleneck layer for the Values: the dimension of each head for Values is unchanged from the (low-rank) light transformer. For example, if a low-rank transformer starts with 8 heads with 64-dimensions per head and ends up with 8 heads with on average 2-dimension per head, the dimension of the bottleneck layer will be 16. Query and Key will still have 8 heads but with 16-dimension per head and Value will keep its 64-dimension per head. The bottleneck transformer is re-trained with the same dataset as the low-rank transformer.

2.4. Compact SLU decoder

The whole structure of the compact SLU system is built from [23] with the encoder-decoder concept. The filterbank (Fbank) inputs are first processed by the low-rank (bottleneck) transformer encoder to yield the high-level representations. The high-level representations are fed into a 2-layer capsule network decoder yields the task information. There are 32 hidden capsules with 64 dimensions in the primary capsule layer and one output capsule for each output slot labels with 16-dimensions in the output capsule layer. The detailed description of capsule decoder can be found on [25]. The low-rank (bottleneck) transformer consists of 3 layers. Every layer has an 8-head parallel attention layer and a feed-forward layer with 2048 hidden nodes. The dimension of the Value of each attention head is 64, the dimensions of Key and Query of the low-rank transformer in each attention head are started from 64. The dimensions of Key and Query of the bottleneck low-rank transformer are initialized by the well-trained low-rank transformer. To further speed up the training, self-attentions were restricted to considering only a neighborhood of size 5 centered around the respective output position. To reduce sequence length, we use two 2-dimensional convolution layers with kernel size $(3, 3)$ at the very beginning to implement 4-fold down-sampling in time.

3. Experiments

3.1. Dataset

The low-rank (bottleneck) transformer is applied to two public SLU corpora.

Domotica addresses 27 home automation tasks for Dutch pathological speech. Notice that the pretraining approach is difficult here, for there is a lack of disordered speech data. This database is collected from 17 speakers in three time phases [26]. We use the subsets named Domotica 3 and Domotica 4 which contains 4180 utterances in total.

FluentSpeech Commands (FSC) records 31 home automation tasks from 97 English speakers. This dataset is challenging with varied command phrasings which contains 248 different phrasing with 30043 utterances in total [27]. Pretaining is possible here.

3.2. Experimental setup

To simulate the low-resourced scenarios, we extract small fraction of samples from each dataset to form the training set. All experiments are conducted under a speaker-independent setting using 10-fold cross-validation. Specifically, for the Domotica data, we randomly select 2 samples for each tasks from each speaker as training samples, which totals 735 samples (because not all speakers perform all tasks). The remaining 3440 samples are used for testing. We evaluate the performances by the F1 score of the detected slot values.

The FSC data is divided into train set (23132 utterances), validation set (3118 utterances) and test set (3793) utterances [28]. We randomly select 10%, 30%, 60% and 100% data from the train set, and report intent accuracy [23] on the test set. The accuracy metric is defined as the accuracy of all slots for an utterance taken together [28].

3.3. Hyper parameters

The proposed compact SLU system is trained with the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$, $\epsilon = 10^{-16}$ and warmup [24]. The final model is constructed by averaging the model parameters of the last 10 training steps. To regularize during training, we apply dropout with a rate of $P_{drop=0.1}$ to each sub-layer, including the content and position embeddings.

4. Results

4.1. Domotica dataset

We first compare slot value F1 results of the low-rank light transformer with the light transformer. The low-rank transformer is trained with a penalty scale of 0.0005, the ranks of well-trained self-attention weight matrices are 4. Figure 3 shows the box plot of results of the 10 times experiments with the Wilcoxon significance test of these two models. The p-value is shown in the Figure 3 as well.

With an average rank of 4 in self-attention, the low-rank transformer significantly outperforms the rank-64 light transformer, showing an efficient assignment of learnable parameters

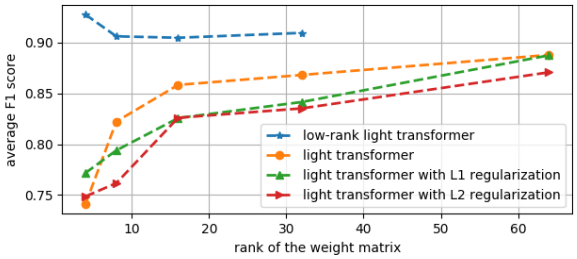


Figure 4: *Average F1 scores of low-rank transformer and light transformer with different ranks*

for scarce training data. One could also argue that the transformer model simply benefits from regularization in an over-parameterized model. We therefore train light transformer models with a smaller common dimension d_k for Query and Key to test if performance can be preserved by using a more compact structure. Secondly, we apply standard L2 regularization as well as (unstructured) sparsity-inducing L1 regularization.

We investigate whether the performance gains come from the group sparsity or regularization in Figure 4, which shows the average F1 scores of various light transformers at different rank choices for the weight matrices. The performance of the transformer dramatically degrades with smaller dimension of Query and Key, which indicates that the benefits brought by the low-rank penalty are not easily obtained by tuning d_k. Introducing L1 or L2 regularization also does not improve performance, which shows the light transformer structure does benefit from group sparsity. Hence, we conclude that group sparsity effectively finds the subspaces that are important for the attention heads.

We compare the average F1 results with other state-of-art encoders including two advanced large-scale pre-trained models [7, 29]. The performance and model sizes are summarized in Table 1. RCCN is a GRU-based SLU system proposed in [25]. ESPnet is a transformer-based model evaluated in [7] with a 12-layer encoder and a 6-layer decoder. To adapt to the Dutch Domotica data, we pre-train this model accompanied with CTC loss using Dutch Copas disordered speech [30]. Kaldi is a TDNN-F-based model proposed in [31], pre-trained on the Dutch Copas data as well. All compared models are combined with the same capsule network decoder for intent classification as described in the Section.2.4.

Table 1: *Average F1 scores and model scales of Domotica.*

Method	F1 score	# of param.
Light transformer [23]	0.888	$1.3M$
+ low-rank	0.930	$1.3M$
+ bottleneck	0.950	$1.1M$
RCCN [25]	0.908	$2.5M$
ESPnet (pre-training) [7]	0.931	$27M + 0.7M$
Kaldi (pre-training) [31]	0.939	$19M + 0.7M$

From Table 1, the light transformer model slightly worse than GRU-based RCCN model. After introducing the low-rank penalty, it outperforms the RCCN model by 3% absolute which indicates the light transformer gives too much freedom to self-attention mechanism that can be constrained with low-rank penalty. In general, low-rank light transformer get comparable results with two pre-training models shows that the low-rank light transformer is capable to extract expressive representations from dysarthric speech inputs. After remedy from inserting bottleneck layer, it outperforms the two pre-training model. One possible reason is speaker-independent dysarthric SLU task is hard to benefit from pre-training while bottleneck transformer is benefits from transfer learning of low-rank transformer.

4.2. FSC dataset

Following the experimental setting in [23], we simulate an insufficient training situation by randomly choosing 10%, 30%, 60%, and 100% data from the train set. We summarize the accuracy results of proposed SLU system as well as mainstream approaches including RNN-based, CNN-based, vanilla transformer-based and advanced pre-training models in the Table 2.

Without pre-training, the low-rank (bottleneck) transformer outperforms most other approaches. Although the bottleneck transformer is slightly worse than the evaluated pre-trained models with 10% training data, with the full training samples, the bottleneck transformer (without pre-training) still shows competitive results with other large-scale pre-trained models.

Table 2: *Average accuracy of FSC.*

Method	10%	30%	60%	100%
Light transformer [23]	91.8	96.7	98.3	98.9
+ low-rank	93.8	97.7	98.7	99.2
+ bottleneck	96.9	98.4	98.9	99.3
RNN [32]	–	–	–	95.3
RNN + capsule network [23]	81.5	94.9	96.7	98.1
Transformer-based [33]	–	–	–	97.6
RNN + SincNet [33]	–	–	–	96.1
+ pre-training [33]	–	–	–	97.2
BERT-based (pre-training)[9]	–	–	–	99.1
ESPnet-based (pre-training) [7]	–	–	–	99.6
CNN-based (pre-training)[29]	–	–	–	99.3
Lugosh et al. [28]	88.9	–	–	96.6
+AM+pre-training [28]	97.9	–	–	98.9
Self-attention + BLSTM [8]	–	–	–	99.1
+ pre-training [8]	–	–	–	99.3

5. Conclusions

Transformer models are over-parameterized for low-resource SLU applications. We therefore proposed using group sparsity to automatically infer a low-rank attention model. The learned attention subspace of the low-rank transformer is then transferred to an attention bottleneck layer to form a compact transformer variant. The compact SLU system built on the (bottleneck) low-rank transformer achieved 95% F1 score on the Domotica disordered speech data and 99.3% accuracy on the FSC data without pre-training, which is comparable to advanced approaches with pre-training (93.9% F1 score on Domotica and the best 99.6% accuracy on FSC).

6. Acknowledgements

The research was supported by the program of China Scholarship Council No.201906090275 and the Flemish Government under “Onderzoeksprogramma AI Vlaanderen”.

7. References

[1] F. Ballati, F. Corno, and L. De Russis, ""hey siri, do you understand me?": Virtual assistants and dysarthria," in *Intelligent Environments 2018*. IOS Press, 2018, pp. 557–566.

[2] 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.

[3] A. Kumar, A. Gupta, J. Chan, S. Tucker, B. Hoffmeister, M. Dreyer, S. Peshterliev, A. Gandhe, D. Filiminov, A. Rastrow *et al.*, "Just ask: building an architecture for extensible self-service spoken language understanding," *arXiv preprint arXiv:1711.00549*, 2017.

[4] S. Rongali, L. B. Liu, K. A. Cai, C. Su, and W. Hamza, "Exploring transfer learning for end-to-end spoken language understanding," in *35th AAAI*, Virtual, Feb. 2021.

[5] Q. Chen, W. Wang, and Q. Zhang, "Pre-training for spoken language understanding with joint textual and phonetic representation learning," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[6] S. Cha, W. Hou, H. Jung, M. Phung, M. Picheny, H. Kuo, and E. M. S. Thomas, "Speak or chat with me: End-to-end spoken language understanding system with flexible inputs," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[7] N. Wang, L. Wang, Y. Sun, H. Kang, and D. Zhang, "Three-module modeling for end-to-end spoken language understanding using pre-trained dnn-hmm-based acoustic-phonetic model," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[8] R. Price, "End-to-end spoken language understanding without matched language speech model pretraining data," in *ICASSP 2020*, Barcelona, Spain, May. 2020.

[9] Y. Jiang, B. Sharma, M. Madhavi, and H. Li, "Knowledge distillation from bert transformer to speech transformer for intent classification," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[10] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *TEMNLP 2020*, Nov. 2020.

[11] A. Panahi, S. Saeedi, and T. Arodz1, "Shapeshifter: a parameter-efficient transformer using factorized reshaped matrices," in *NeurIPS 2021*, Dec. 2021.

[12] S. Bhojanapalli, C. Yun, A. S. Rawat, S. Reddi, and S. Kumar, "Low-rank bottleneck in multi-head attention models," in *PMLR 2020*, 2020.

[13] M. Geng, S. Liu, J. Yu, X. Xie, S. Hu, Z. Ye, Z. Jin, X. Liu, and H. Meng, "Spectro-temporal deep features for disordered speech assessment and recognition," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[14] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *ICLR 2020*, Apr. 2020.

[15] Y. Mao, Y. Wang, C. Wu, C. Zhang, Y. Wang, Y. Yang, Q. Zhang, Y. Tong, and J. Bai, "Ladabert: Lightweight adaptation of bert through hybrid model compression," in *COLING 2020*, Barcelona, Spain, Dec. 2020.

[16] S. Mehta, H. Rangwala, and N. Ramakrishnan, "Low rank factorization for compact multi-head self-attention," *arXiv preprint arXiv:1912.00835*, 2019.

[17] A. Ollerenshaw, M. A. Jalal, and T. Hain, "Insights on neural representations for end-to-end speech recognition," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[18] H. Saghir, S. Choudhary, S. Eghbali, and C. Chung, "Factorization-aware training of transformers for natural language understanding on the edge," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[19] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[20] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. Fung, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *ICASSP 2020*, Barcelona, Spain, May. 2020.

[21] P.-H. Chen, S. Si, Y. Li, C. Chelba, and C. j. Hsieh, "Groupreduce: Block-wise low-rank approximation for neural language model shrinking," in *NeurIPS 2018*, Montréal, Canada, Dec. 2018.

[22] P.-H. Chen, H.-F. Y. I. Dhillon, and C.-J. Hsieh, "Drone: Data-aware low-rank compression for large nlp models," in *NeurIPS 2021*, Dec. 2021.

[23] P. Wang and H. Van hamme, "A light transformer for speech-to-intent applications," in *SLT 2021*, China, Jan. 2021, pp. 997–1003.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS 2017*, Long Beach, CA, USA, Dec. 2017, pp. 997–1003.

[25] V. Renkens and H. Van hamme, "Capsule networks for low resource spoken language understanding," in *INTERSPEECH 2018*, Hyderabad, 2018, pp. 601–605.

[26] "Domotica dataset." [Online]. Available: https://www.esat.kuleuven,be/psi/spraak/downloads/.

[27] "Fluentspeech commands dataset." [Online]. Available: http://www.fluent.ai/reserach/fluent-speech-commands/.

[28] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech model pre-training for end-to-end spoken language understanding," in *INTERSPEECH 2019*, Graz, Austria, Sep. 2019.

[29] Y. Cao, N. Potdar, and A. R. Avila, "Sequential end-to-end intent and slot label classification and localization," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[30] "Copas." [Online]. Available: https://taalmaterialen.ivdnt.org/download/tstc-corpus-pathologische-en-normale-spraak-copas/

[31] P. Wang and H. Van hamme, "A study into pre-training strategies for spoken language understanding on dysarthric speech," in *INTERSPEECH 2021*, Brno, Czech, Aug. 2021.

[32] D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, and Y. Bengio, "Towards end-to-end spoken language understanding," in *ICASSP 2018*, Calgary, Canada, Sep. 2018.

[33] M. Radfar, A. Mouchtaris, and S. Kunzmann, "End-to-end neural transformer based spoken language understanding," in *INTERSPEECH 2020*, Shanghai, China, Oct. 2020.