Call: **H2020-FETPROACT-2020-01**

Grant Agreement no. **101016776**

Deliverable D5.5 – Co-optimized hardware/ NN architecture for ASR/ MT - V1

Start date of the project: 1st January 2021
Duration: 50 months
Project Coordinator: Cristell MANEUX - University of Bordeaux
Contact: Cristell MANEUX - cristell.maneux@ims-bordeaux.fr

DOCUMENT CLASSIFICATION

Title	Co-optimized hardware/ NN architecture for ASR/ MT - V1
Deliverable	D5.5
Estimated Delivery	30/10/2022 (M20+2)
Date of Delivery Foreseen	30/10/2022 (M20+2)
Actual Date of Delivery	30/10/2022 (M20+2)
Authors	Jean-Luc Rouas – P1 – UBx, Leïla Ben Letaifa – P1 – UBx, Georgeta Bordea – P1 – UBx
Approver	Cristell Maneux (UBx)
Internal reviewers	G. Ansaloni (EPFL)
Work package	WP5
Dissemination	PU
Version	V1.3
Doc ID Code	D5.5_FVLLMONTI_P1-UBX-20221024
Keywords	Neural Network Optimization, Automatic Speech Recognition, Machine Translation

DOCUMENT HISTORY

VERSION	PUBLICATION DATE	CHANGE
1.0	05.10.2022	Initial version from UBx (J.-L. Rouas)
1.1	10.10.2022	Correction by UBx (L. Ben Letaifa, G. Bordea)
1.2	11.10.2022	Internal review by G. Ansaloni (EPFL)
1.3	24.10.2022	Final version (J.-L. Rouas, L. Ben Letaifa)

DOCUMENT ABSTRACT

This document describes the first version of the co-optimization between hardware requirements and neural network transformer architecture for automatic speech recognition (ASR) and machine translation (MT).

Task 5.3 focuses on size reduction of the NN configurations from T.5.1 by using constraints (e.g. NN and kernel size, weight values, types) and quantization (low bit precision for parameter storage enabling low bit hardware operations) based on different hardware configurations from T.5.2 (e.g. computing capabilities, predefined topologies, predefined weight values, Network Resilience). Trade-offs between NN accuracy and N2C2 resource usage will be explored.

From T.5.1 we decided to use Transformer-based models, since they achieve state-of-the-art performance in various areas of machine learning, including automatic speech recognition. However, their cost in terms of computational power, memory or energy consumption can be exorbitant, hence the interest in compression techniques that take into account the hardware constraints. Transformer models are mostly composed of attention and feedforward components. We propose to reduce the size of the transformer model of our end-to-end speech recognition system by decreasing the number and precision of linear layer parameters. Specifically, we investigate the impact of weight pruning and model quantization on system performance.

Experiments carried out on several speech recognition datasets for different languages show that the memory footprint of Transformer Speech Recognition systems can be reduced by up to 84% with an acceptable loss of accuracy using classical pruning and quantization techniques.

After studying the transformer architecture, we proposed a new pruning method, “variable scale pruning” which assigns a decreasing pruning rate according to the depth of the layers. Evaluation on different databases shows that this method outperforms conventional local and global pruning methods and that the pruned model can be compressed up to 57% with a slight decrease in accuracy.

We are currently investigating the usage of similar techniques for our text-to-text translation systems and we will then address the size reduction of the cascade model, composed by a speech-to-text transcription stage followed by a text-to-text translation stage.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101016776.

TABLE OF CONTENT

LIST OF FIGURES AND TABLES

LIST OF ACRONYMS / GLOSSARY

ASR:	Automatic Speech Recognition
CER:	Character Error Rate
CONV:	Convolution
D:	Deliverable
ESTER:	Evaluation of Speech broadcast news Enriched Transcription systems
FF:	Feed-Forward
M:	Month of the project
MHA:	Multi-Head Attention
MT:	Machine Translation
WER:	Word Error Rate
WP:	Work Package

1. Introduction

Neural network compression methods include quantization [1], pruning [2], knowledge distillation [3], matrix decomposition [4] and parameter sharing [5]. Although most of these methods were originally proposed for convolutional neural networks, all of them and others such as the pruning of the attention heads [6], [7] are directly applicable to the transformer model. Compared to basic models such as convolutional or recurrent neural networks, a transformer model [8] has a relatively complex architecture composed of several parts such as embedding layers, self-attention layers and feedforward layers. Thus, the effect of compression methods can vary when applied to different parts of it [5].

Research on transformer model compression in end-to-end speech recognition has mainly focused on quantization [9] and parameter sharing [10]. In this document we review the quantization method and two kinds of pruning schemes: local and global pruning. We then perform baseline experiments to assess the gains obtained using the two techniques and their combination.

After studying in detail the transformer architecture, we proposed a new pruning method, “variable scale pruning”. This method allows to optimize the pruning rate according to the depth of the network which results in further gains in model size while preserving the accuracy.

2. Compression techniques

Model compression methods reduce the inference costs of trained models. In particular, we consider two compression techniques: quantization and pruning.

I. QUANTIZATION

By default, most systems use float 32 types to represent variables and weights. Quantization replaces floating points with integers leading, to less memory consumption and faster calculations [1].

Quantization maps a floating point value x in $[a, b]$ to a $k - bit$ integer x_q in $[-2^{k-1}, 2^{k-1} - 1]$.

$$x_q = round\left(\frac{x - a}{\partial}\right)$$

where $\delta = \frac{b-a}{2^k-1}$. In the case that x in not in the range of $[a, b]$, the clamp operator is applied:

$$clamp(x; a, b) = min(max(x, a), b)$$

The de-quantization function is defined as:

$$D(x_q) = x_q * \delta + a$$

Different quantization approaches have been proposed and can be classified into two categories: post-training quantification and quantification aware training [9]. Post-training quantization trains the model using weights and float32 inputs, then quantizes the weights. Its main advantage is that it is simple to apply. **Quantization-aware training** quantizes the weights during training. Here, even the gradients are calculated for the quantized weights.

II. PRUNING

Deep learning models are often over-parameterized [2], [11], with many insignificant weights that contribute only slightly to the inference of the model. These weights can be set to zero without significantly affecting performance [2], a technique typically referred to as pruning. The importance of the weights is often determined by their magnitude or their gradients [5].

In general, there are two types of pruning:

- unstructured pruning [11], which removes individual weights. Substituting the value of the weight with zero in a weight matrix is equivalent to pruning a connection. This pruning is also called sparse pruning because it results in sparse matrices.
- structured pruning [12] focuses on pruning blocks of weights, for example, by deleting entire channels at a time. In practice, structured pruning sets an entire row or column of a weight matrix to zero, which is the same as deleting a neuron.

Pruning can be incorporated into the training process as an additional step between training epochs (iterative pruning) [9], applied all at once after the model training is complete [7] (one-shot pruning), or applied between fine tuning steps [13]. Based on whether the pruning is applied globally to all model parameters or is calculated independently for each layer, this technique is called global pruning or local pruning. Global pruning groups all the parameters of the layers and selects a global fraction of them to prune. Local pruning removes a fixed percentage of parameters from each layer.

3. Transformer model

I. MODEL DESCRIPTION

The Automatic Speech Recognition (ASR) Transformer model [14] is a sequence-to-sequence model that maps an input sequence of acoustic features $(x_1, x_2, \dots x_T)$ of length T to an output sequence of characters $(y_1, y_2, \dots, y_L)$ of length L. Its architecture can be divided into two parts namely the encoder and the decoder. The encoder converts the input sequence into an intermediate sequence of encoded features $(h_1, h_2, \dots, h_N)$ of length N. The decoder predicts a new character y_l based on the encoded features $(h_1, h_2, \dots, h_N)$ and the previous decoded characters $(y_1, y_2, \dots, y_{l-1})$. Both the encoder and the decoder are composed of a stack of attention and feedforward network blocks. An overview of the transformer architecture is displayed on Figure 1.

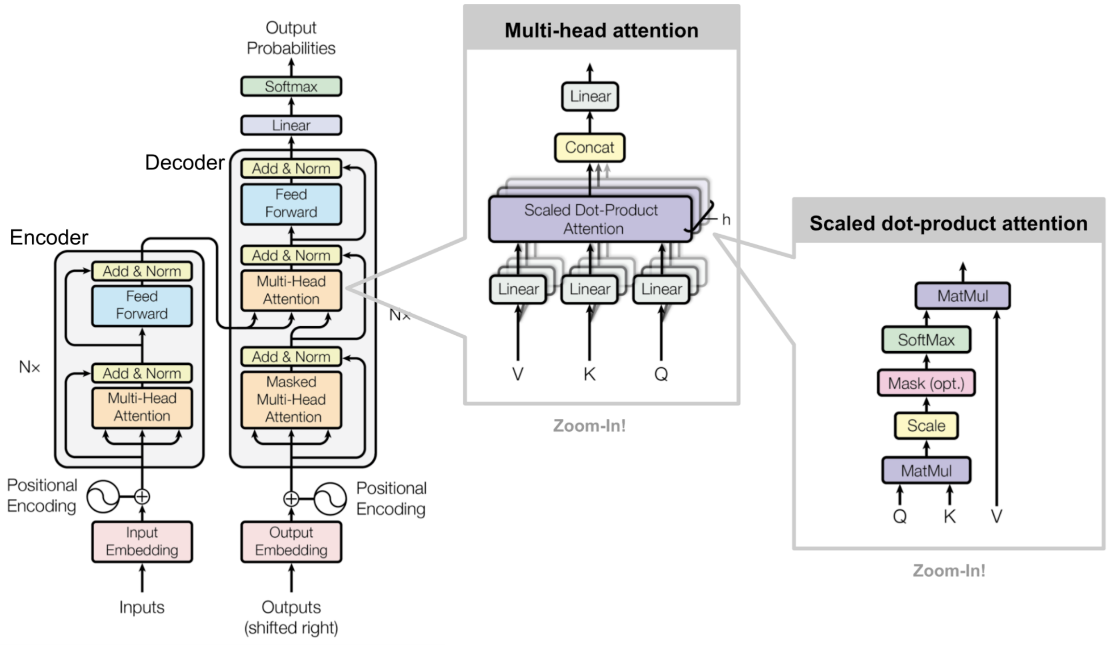


Figure 1: Overview of the transformer architecture (adapted from [8])

Our ASR transformer follows the same architecture as [15]. The input acoustic features are subsampled using two convolution layers (CONV) before being fed into the encoder. Both the encoder and the decoder are composed of multi-head attention (MHA) and feedforward (FF) layers, each followed by a residual connection and normalization. A simplified representation of the transformer model is shown on Figure 2.

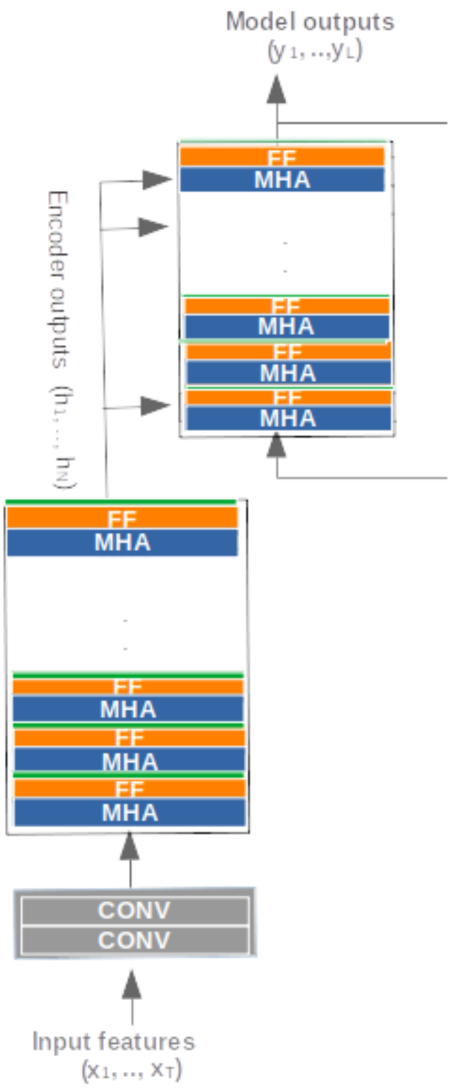


Figure 2: Transformer main components

II. MODEL ARCHITECTURE

The self-attention operation allows frames to gather context from all timesteps and build an informative sequence of high level [15]. Specifically, the inputs of each layer are projected into queries Q, keys K, and values V with $Q \in R^{t_q * d_q}$, $K \in R^{t_k * d_k}$ and $V \in R^{t_v * d_v}$. $t*$ are the elements numbers in different inputs and $d*$ are the corresponding element dimensions. Scaled Dot-Product Attention [8] is then computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The multi-headed attention is obtained by performing this calculation h times. h is the number of heads.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W_0$$

where

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

The projection matrices are $W_i^Q \in R^{d_{model} * d_q}$, $W_i^K \in R^{d_{model} * d_k}$, $W_i^V \in R^{d_{model} * d_v}$ and $W^0 \in R^{h * d_v * d_{model}}$. In this work, $d_k = d_q = d_v = d_{model}/h$.

The outputs of multi-head attention go through a 2-layer position-wise feedforward network (FFN) with hidden size d_{ff}.

$$FFN(x) = W_2 ReLU(W_1 x + b_1) + b_2$$

$b_1 \in R^{d_{ff}}$ and $b_2 \in R^{d_{model}}$ are the biases. The weight matrices are $W_1 \in R^{d_{model} * d_{ff}}$ and $W_2 \in R^{d_{ff} * d_{model}}$.

4. Baseline Models & Preliminary Experiments

We have developed transformer models for three languages including English, French and Italian using Libri-trans [16], Ester [17], and Voxforge [18] databases respectively.

I. DATA DESCRIPTION

The Libri-trans, Ester and Voxforge corpora are produced within the framework of the Librivox project, the French national ESTER project and the Voxforge project. The Libri-trans and Voxforge recordings are extracted from audiobooks, and the Ester recordings are radio broadcasts news. Each dataset is divided into three parts: train, development (dev) and test as described in Table 1. The train data is used for model training. The dev and the test parts are dedicated to evaluation.

	Libritrans	Ester	Voxforge
Train	230	231	18
Dev	2	5.45	1
Test	3.5	6.5	1

Table 1: Duration (in hours) of the three datasets

II. TRAINED MODELS

Baseline ASR transformer models are developed and evaluated with the Espnet toolkit [19]. This toolkit involves Kaldi [20] tools for data processing and parameter extraction and Pytorch[1] modules for model estimation.

First, by using three different speeds (0.9, 1.0, and 1.1) for speed perturbation data augmentation, the train dataset amount tripled. Then, 80 filter bank coefficients are extracted and normalized with respect to the mean and variance. Transcripts are represented by sub-word units, namely characters for the Ester and Voxforge systems and byte-pair coding subwords for the Libritrans system. Finally, several transformer architectures are evaluated.

Table 2 shows the architecture of the best transformer models, their number of parameters (in millions) and the error rates of the ASR systems. We consider the word errors (WER) of the Libri-trans and Ester systems and the character errors (CER) of the Voxforge system.

	Libritrans	Ester	Voxforge
Architecture details			
Enc/Dec	12/6	18/6	12/6
FF Dimension	1024	2048	2048
ATT Dimension	256	512	256
Heads	4	4	4
# Parameters (Millions)	27.92	89.64	35.07
Error rate (%)	6.6	14.1	9.1

Table 2: ASR models specifications - Architecture: number of encoder and decoder blocks (Enc/Dec), dimension of hidden layers (FF Dimension) and attention layers (ATT Dimension) and number of attention heads (Heads) - Number of parameters (Millions) - Error rate (% WER/CER for VoxForge)

III. RESULTS

First, one-shot pruning is applied to the trained end-to-end ASR models for three different languages (French, English, Italian), applying either global or local pruning while varying the pruning rate. For the best trade-off between error and compression rates, we tolerate a relative increase in error rate of 10%. The best trade-offs are achieved with pruning rates of 33% for Libri-trans (English), 37% for Ester (French), 55% for Voxforge (Italian). After applying gzip compression, the disk space occupied by the pruned models is about half of the space occupied by the initial models. Details are given in Table 3.

[1] www.pytorch.org

	LIBRITRANS	ESTER	VOXFORGE
Initial model (MB)	99	318	124
Pruned model (MB)	72	228	72
RATIO (x - %)	1.3x - 72%	1.4x - 71%	1.7x - 58%

Table 3 Size of initial gzip-compressed and pruned gzip-compressed models (in megabytes) and their ratio

We then studied the effect of quantization. Indeed, initially all model parameters are stored and processed in real 32-bit format. An 8-bit integer quantization is performed on the encoder and decoder weight matrices. Activations are quantized on the fly during inference and stored as 32-bit real values. For all systems, the performance decrease is insignificant (less than 0.1%). Regarding the model size, we observe that quantized models are more than 70% smaller (Table 4).

	LIBRITRANS	ESTER	VOXFORGE
Initial model (MB)	99	318	124
Quant. model (MB)	23	72	29
RATIO (x - %)	4.3x - 23%	4.4x - 22%	5.3x - 23%

Table 4 Size of gzip-compressed quantized models (in megabytes) and their ratio to the initial gzip-compressed models

Pruning and quantization are finally applied successively for a better accuracy/size trade-off. This is performed in three steps:
1) set the pruning rate
2) prune the model
3) quantize the model

The initial and final models are then compressed using gzip and their size ratios are computed to assess the joint contribution of pruning and quantization. The results are given inTable 5.

	LIBRITRANS	ESTER	VOXFORGE
Initial model (MB)	99	318	124
Pruned + Quant. model (MB)	20	68	20
RATIO (x - %)	4.9x - 20%	4.6x - 21%	6.2x - 16%
Initial error (%)	6.6	14.1	9.1
Pruned+Quant error(%)	7.2	15.6	10.1

Table 5 Initial and final error rates, size of initial gzip-compressed and pruned+quantized gzip-compressed models (in megabytes) and their ratio

A final compression of about 82% for Voxforge (Italian), 80% for Libri-trans (English) and 79% for Ester (French) is achieved. Experiments carried out on several speech datasets from different languages show that the memory footprint of Transformer Speech Recognition systems can be reduced by up to 82% with an acceptable loss of accuracy using pruning and quantization techniques. These results are published in an international conference paper [21].

To further optimize the networks, we studied in details the different components of the transformer architecture, as discussed in the next section.

5. Further network optimisations

I. MODEL WEIGHTS ANALYSIS

The layers of the transformer model are organized into four groups: Convolution (CONV) layers, Multi-Head Attention (MHA) layers, Feed Forward (FF) layers, and the remaining layers.

1. *Number of parameters*

The proportion in number of weights of each group is calculated and then plotted on Figure 3.

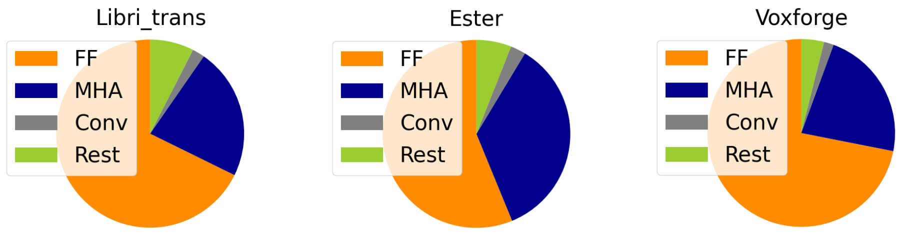


Figure 3: The proportion of weights of convolution layers (Conv), multi-head attention layers (MHA), feedforward layers (FF) and remaining layers (Rest) for the Libri-trans, Ester, and Voxforge models.

In all the models, the parameters of the feedforward layers are the most numerous exceeding 55% of the total number of parameters, those of the attention layers are above 22%, those of the convolution layers are lower than 3% and the rest of the layers represent less than 5%.

2. *Weight distribution (encoder layers)*

Weight pruning sets low-value weights to zero. Here we examine the weight values across the transformer encoder layers. The absolute values of the weights are averaged for each layer class, i.e., the convolution layers ($cv0$ and $cv1$), the feedforward layers $FF1$ and $FF2$, and the multi-head attention layers W_q, W_k, W_v, and W_0 and plotted on Figure 4, Figure 5 and Figure 6.

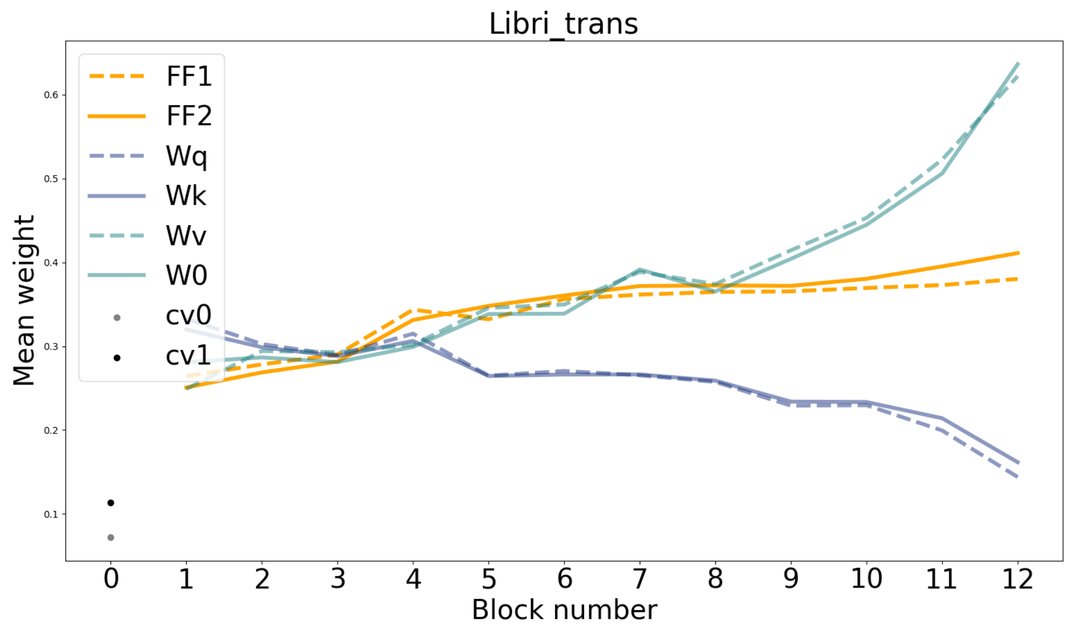


Figure 4: Class weight distribution across the encoder blocks for Libritrans

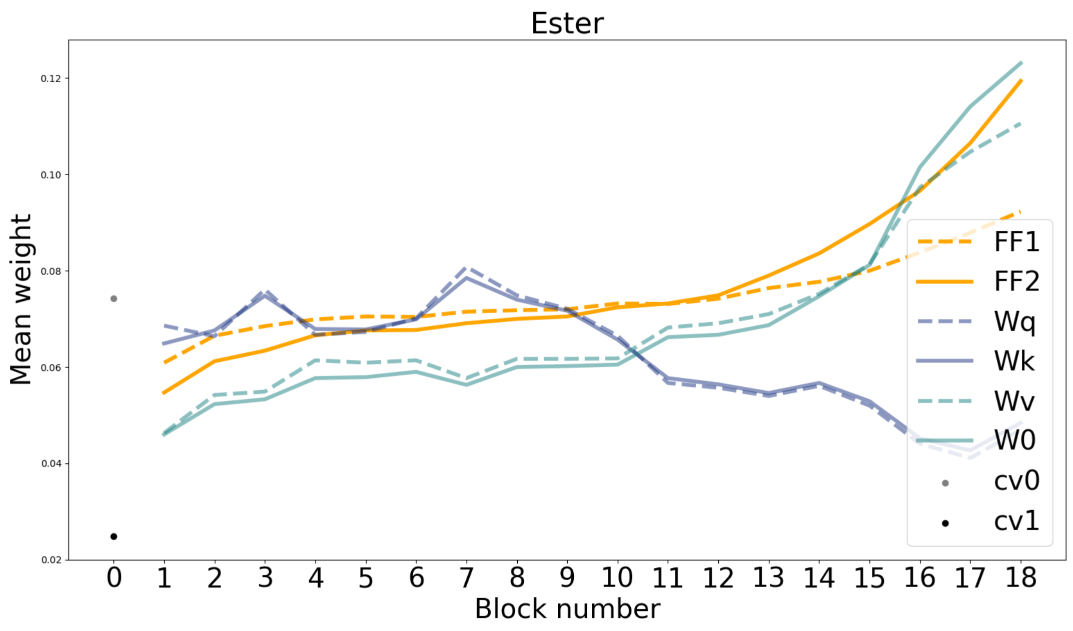


Figure 5: Class weight distribution across the encoder blocks for Ester

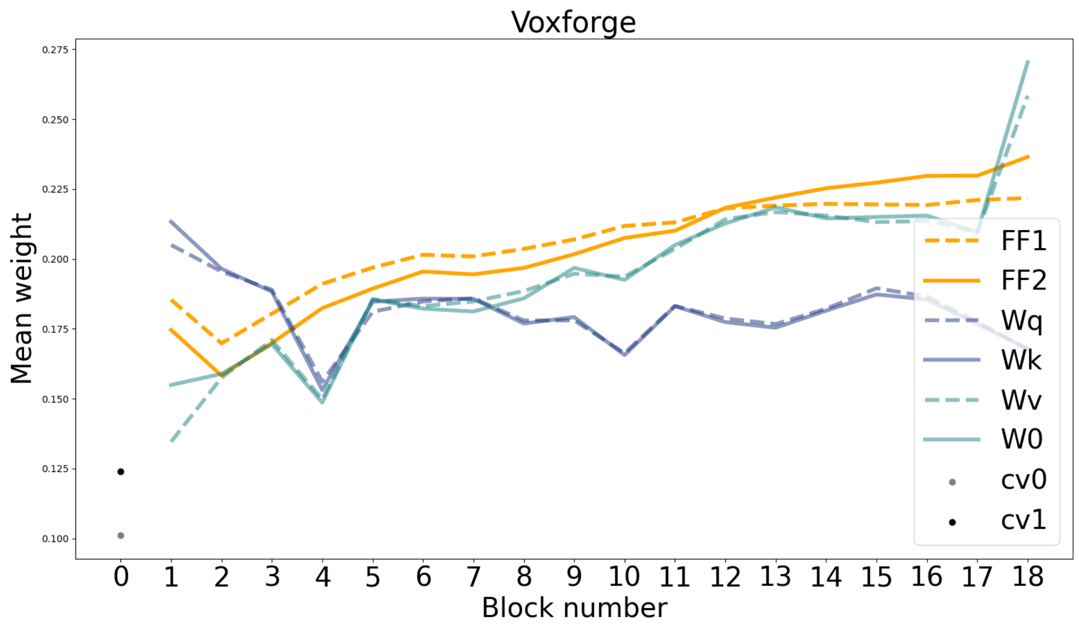


Figure 6: Class weight distribution across the encoder blocks for VoxForge

Briefly, we can make a few global observations, regardless of the model:

- The mean weights of $FF1$ and $FF2$ layers are close and their evolution is similar: the deeper the layer is, the more important the weights are.
- The same observation can be made for W_k and W_q, but this time the mean magnitude of their weights decrease with the depth of the block, except for Voxforge.
- Similarly, W_v and W_0 are also very close and increasing with the depth of the blocks.

The observations made in the previous section shed the light on the fact that the most important layers in terms on number of weights are the feed-forward (FF) layers. We have also seen that the mean weights of these layers increase with the depth of the network. This suggests that the early feedforward layers of encoder and decoder are less important than the deeper layers.

We thus propose in the next section an original pruning method based on this observation.

II. FEEDFORWARD LAYERS ADAPTIVE PRUNING

1. *Method*

We define "Variable scale pruning" as a pruning approach where the early feedforward layers are pruned more than the deeper layers, proposing an arithmetically decreasing pruning rate. Let the two sequences of parameter , and of parameter represent respectively the pruning rates of the encoder and decoder layers. For two successive layers and , if is the pruning rate of the current encoder layer and is the pruning rate of next layer so:

$$U_{n+1} = U_n - \alpha$$

Similarly, for the decoder layers:

$$V_{n+1} = V_n - \beta$$

where α and $\beta$are positive numbers.

For the first layers of the encoder and decoder, the initial pruning rates U_0 and V_0 are set empirically.

2. *Experiments and results*

In this section, experiments are performed with the Libri-trans and Voxforge datasets, using the development set to fix the adaptive pruning parameters and the test data for evaluation.

Baseline systems

We first carry out global and local pruning experiments on all layers for comparison purposes. Graphical results from those experiments are shown on Figure 7 and Figure 8. When the pruning rate is less than 30%, the error rate increases only slightly (about 4.7% relative for Libri-trans and 2.1% for Voxforge).

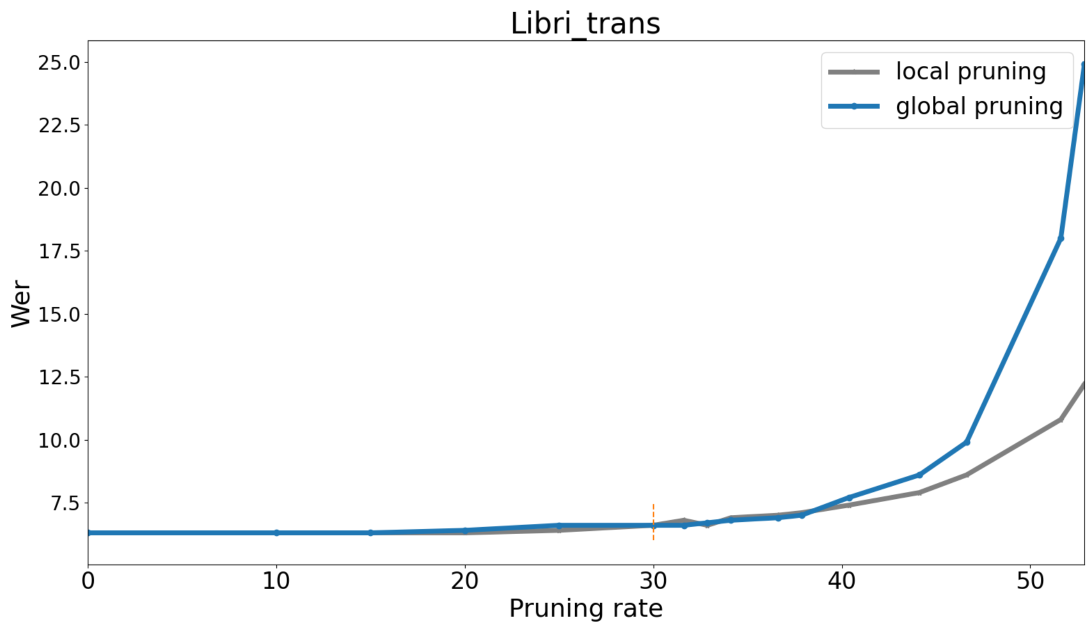


Figure 7: Error rate as a function of global and local pruning rate for Libritrans

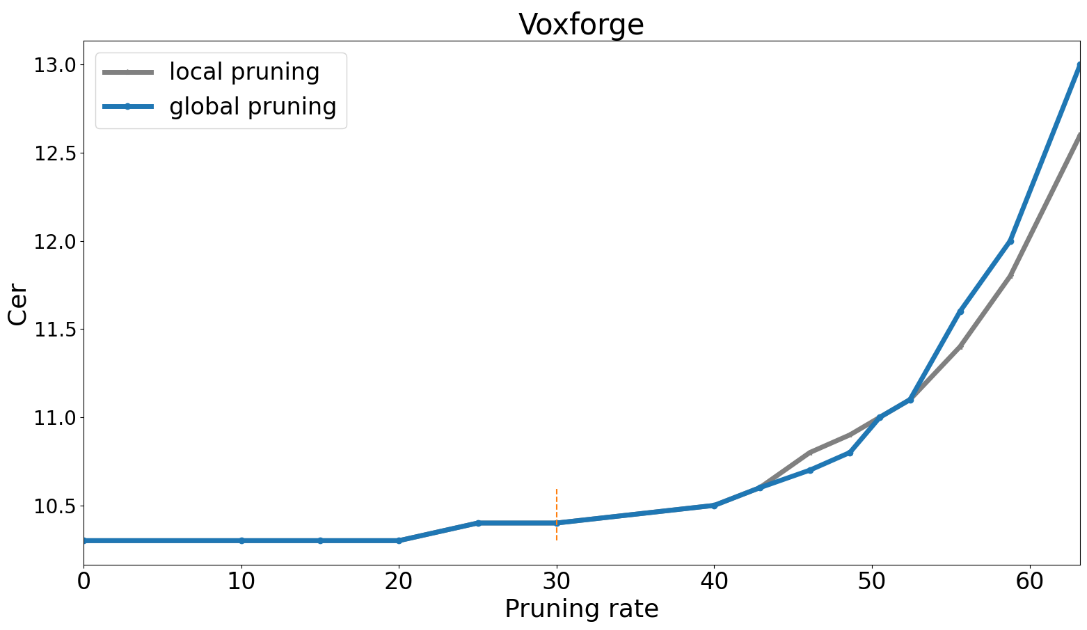


Figure 8: Error rate as a function of global and local pruning rate for VoxForge

Adaptative pruning

Adaptive pruning is applied in one shot on the trained models. The encoder and decoder pruning rates are represented by the arithmetic sequences U and V. To fix the initial pruning amounts U_0 and V_0 and the differences α and β, adaptive pruning is applied to the development data as follows:

1) fix α and β
2) for each pair (U_0, V_0):
    a. use the formulas for U_{n+1} and V_{n+1} to prune the feedforward layers
    b. decode the system using the pruned model and the development dataset
    c. note the error rate and the sparsity

The algorithm is executed, on each dataset, three times using different pruning rates of the attention layers. For Libri-trans, the pruning rates are 30%, 35%, and 40%. As for Voxforge, they are 35%, 40% and 45%. The coefficients α and β are chosen empirically and set to 0.01. For example, for a starting pruning rate at 30% and $\alpha = 0.01$, the first layer will be pruned with the coefficient $U_0 = 0.3$, the second with $U_1 = U_0 - \alpha = 0.3 - 0.01 = 0.29$. Thus for a 12 block encoder, the pruning rate of the last layer will be $U_{12} = 0.19$ i.e. with a pruning rate of 19%.

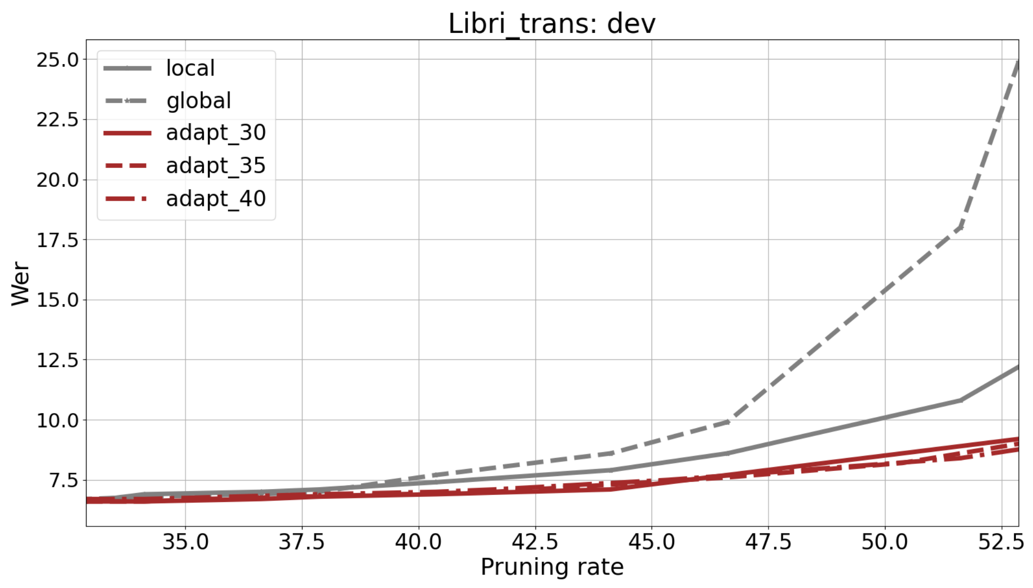
The Libri-trans models adaptive pruning results when setting the attention layer pruning to 40%, and the parameters α and β to 0.01 are shown on

Table 6. The best tradeoffs between WER and sparsity (which are highlighted in bold) are selected.

		U0(%)						
	V0(%)	35	40	45	50	55	60	65
WER	35	6.8	6.8	**6.9**	7.1	7.4	8.0	8.6
Sparsity		32.87	35.37	**37.87**	40.37	42.87	45.37	47.87
WER	40	6.8	6.9	7.0	**7.1**	7.5	7.9	8.8
Sparsity		34.12	36.62	39.12	**41.62**	44.12	46.62	49.12
WER	45	**6.8**	7.0	7.0	7.2	7.5	8.0	8.9
Sparsity		**35.37**	37.87	40.37	42.87	45.37	47.87	50.37
WER	50	6.9	6.9	7.2	7.4	7.6	8.0	8.9
Sparsity		36.62	39.12	41.62	44.12	46.62	49.12	51.62
WER	55	7.0	**7.0**	7.2	**7.5**	7.8	8.4	9.0
Sparsity		37.87	**40.37**	42.87	**45.37**	47.37	50.37	52.87
WER	60	6.9	7.1	7.4	7.6	7.9	**8.4**	9.3
Sparsity		39.12	41.62	44.12	46.62	49.12	**51.12**	54.12
WER	65	7.1	7.3	7.4	7.8	8.1	8.9	**9.5**
Sparsity		40.37	42.87	45.37	47.87	50.37	52.87	**55.37**

Table 6: Adaptive pruning of the Libri-trans model: WER and Sparsity for several pairs of (U_0, V_0) The fixed parameters are: attention layer pruning rate = 40% and $\alpha = \beta = 0.01$.

Global and local pruning is performed using selected sparsities. The results on development sets of the different pruning (global, local and adaptive pruning with three pruning rates of the attention layers) are shown on Figure 9. For both Libri-trans and Voxforge, adaptive pruning outperforms local and global pruning. This is expected since the variable scale pruning parameters are estimated on the development data.



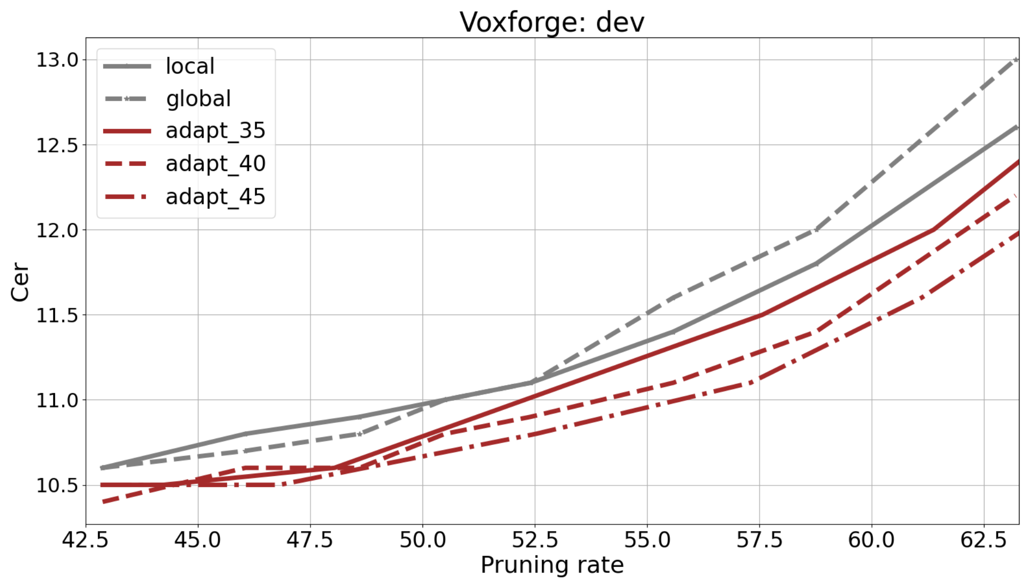
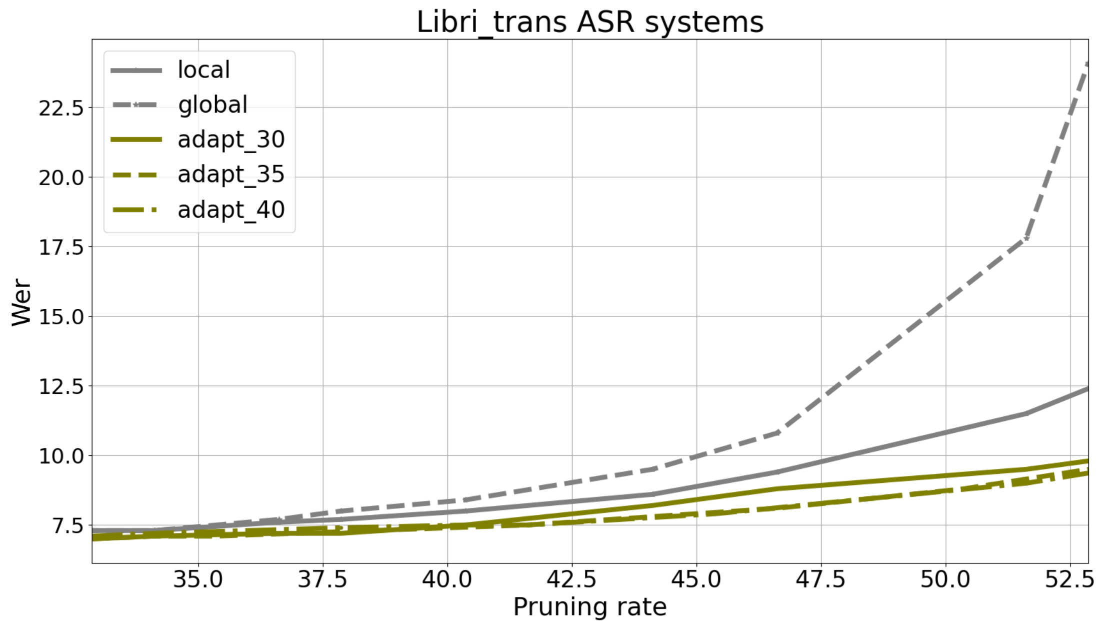


Figure 9: Development error rate as a function of sparsity for the two ASR systems Libri-trans (top) and Voxforge (bottom) after applying local, global and variable scale pruning.

Evaluation

The three types of local, global, and variable scale pruning are applied on the test data without parameter settings. U_0, V_0 and α and β keep the same values that are set on the development data. Figure 10 reports all error rates for a sparsity up to 52.5% for Libri-trans and 62.5% for Voxforge. In fact, beyond that the error rate resulting from the application of the global pruning increases greatly exceeding 24% for Libri-trans and 13.9% for Voxforge.

Variable scale pruning outperforms global and local pruning in all cases. For Libri-trans, when we fix the target WER at7.3%, the variable scale pruning provides a gain in sparsity of 7%. The model can be further compressed to 52.5% with a relative WER increase of about 13.5% (final WER= 8.9%).
Regarding Voxforge, for a target CER at 9.5%, the variable scale pruning models are 6% sparser than the models that are locally or globally pruned. Allowing a 7% relative increase in error rate, the model can be further compressed to 57%.



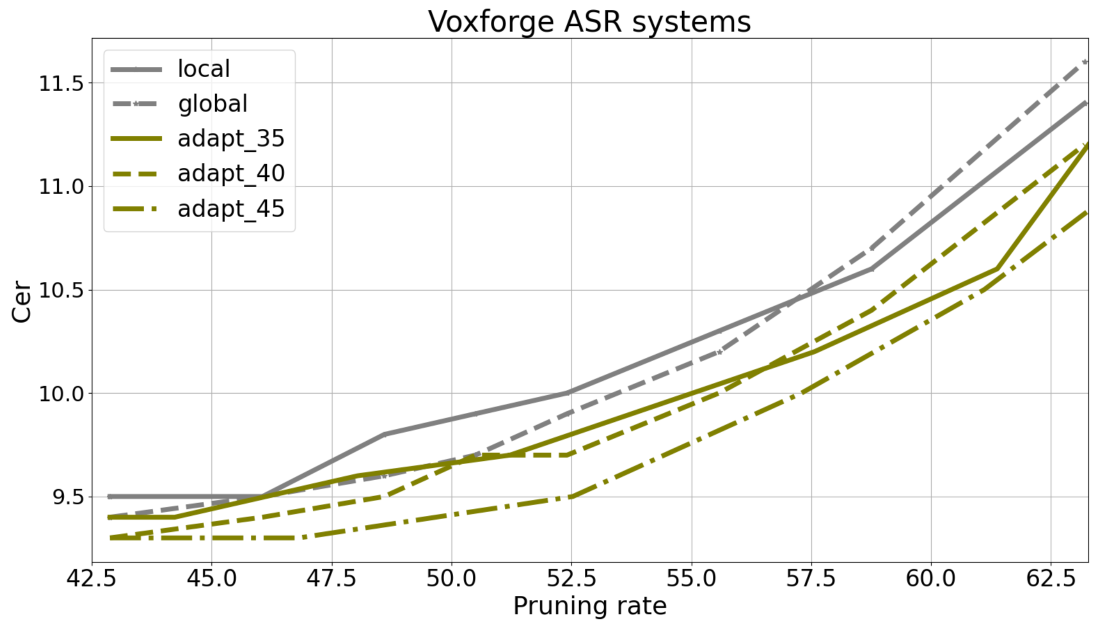


Figure 10: Test error rate as a function of sparsity for the two ASR systems Libri-trans (top) and Voxforge (bottom) after applying local, global and variable scale pruning

6. CONCLUSION

This deliverable describes the first version of an optimised architecture for speech recognition that takes into consideration hardware limitations. First experiments were carried out on several speech datasets from different languages, showing that the memory footprint of Transformer Speech Recognition systems can be reduced by up to 84% with an acceptable loss of accuracy using pruning and quantization techniques.

After studying the transformer architecture, we proposed a new pruning method, "variable scale pruning". It is based on an analysis of the progression of feedforward layer weights through the encoder and decoder blocks. Since deep layers have more important weights than early layers, variable scale pruning assigns a variable pruning rate that decreases with the layer depth. Evaluation on different databases shows that this method outperforms conventional local and global pruning methods and that the pruned model can be compressed up to 57% with a slight decrease in accuracy.

Future work includes exploring variable scale pruning on attention head layers and fine tuning of models. We are currently investigating the usage of similar techniques for our text-to-text translation systems and we will then address the size reduction of the cascade model (speech-to-text transcription followed by text-to-text translation).

7. REFERENCES

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, 'Quantized neural networks: training neural networks with low precision weights and activations', *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, Jan. 2017.

[2] Y. LeCun, J. Denker, and S. Solla, 'Optimal Brain Damage', in *Advances in Neural Information Processing Systems*, 1989, vol. 2 [Online]. Available: https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html. [Accessed: Sep. 30, 2022]

[3] H.-G. Kim *et al.*, 'Knowledge Distillation Using Output Errors for Self-attention End-to-end Models', in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, p. 6185, doi: 10.1109/ICASSP.2019.8682775.

[4] M. Ben Noach and Y. Goldberg, 'Compressing Pre-trained Language Models by Matrix Decomposition', in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, Suzhou, China, Dec. 2020, pp. 884–889 [Online]. Available: https://aclanthology.org/2020.aacl-main.88. [Accessed: Sep. 30, 2022]

[5] P. Ganesh *et al.*, 'Compressing Large-Scale Transformer-Based Models: A Case Study on BERT', *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1061–1080, 2021, doi: 10.1162/tacl_a_00413.

[6] P. Michel, O. Levy, and G. Neubig, 'Are Sixteen Heads Really Better than One?', in *Advances in Neural Information Processing Systems*, 2019, vol. 32 [Online]. Available: https://papers.nips.cc/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html. [Accessed: Oct. 24, 2022]

[7] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, 'Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned', in *Proceedings of the 57th annual meeting of*

the association for computational linguistics, Florence, Italy, Jul. 2019, pp. 5797–5808, doi: 10.18653/v1/P19-1580 [Online]. Available: https://aclanthology.org/P19-1580

[8] A. Vaswani *et al.*, ‘Attention is All you Need’, *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[9] A. Bie, B. Venkitesh, J. Monteiro, M. A. Haidar, and M. Rezagholizadeh, ‘A Simplified Fully Quantized Transformer for End-to-end Speech Recognition’. arXiv, Nov. 08, 2019 [Online]. Available: http://arxiv.org/abs/1911.03604. [Accessed: Sep. 30, 2022]

[10] S. Li, D. Raj, X. Lu, P. Shen, T. Kawahara, and H. Kawai, ‘Improving Transformer-Based Speech Recognition Systems with Compressed Structure and Speech Attributes Augmentation’, in *Interspeech 2019*, Sep. 2019, pp. 4400–4404, doi: 10.21437/Interspeech.2019-2112 [Online]. Available: https://www.isca-speech.org/archive/interspeech_2019/li19u_interspeech.html. [Accessed: Sep. 30, 2022]

[11] S. Han, J. Pool, J. Tran, and W. J. Dally, ‘Learning both weights and connections for efficient neural networks’, in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, Cambridge, MA, USA, Dec. 2015, pp. 1135–1143.

[12] S. Anwar, K. Hwang, and W. Sung, ‘Structured Pruning of Deep Convolutional Neural Networks’, *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, p. 32:1-32:18, Feb. 2017, doi: 10.1145/3005348.

[13] V. Sanh, T. Wolf, and A. Rush, ‘Movement Pruning: Adaptive Sparsity by Fine-Tuning’, in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 20378–20389 [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/eae15aabaa768ae4a5993a8a4f4fa6e4-Abstract.html. [Accessed: Oct. 24, 2022]

[14] S. Karita *et al.*, ‘A Comparative Study on Transformer vs RNN in Speech Applications’, in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2019, pp. 449–456, doi: 10.1109/ASRU46091.2019.9003750.

[15] L. Dong, S. Xu, and B. Xu, ‘Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition’, in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2018, pp. 5884–5888, doi: 10.1109/ICASSP.2018.8462506.

[16] A. C. Kocabiyikoglu, L. Besacier, and O. Kraif, ‘Augmenting Librispeech with French Translations: A Multimodal Corpus for Direct Speech Translation Evaluation’, in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018 [Online]. Available: https://aclanthology.org/L18-1001. [Accessed: Sep. 08, 2022]

[17] S. Galliano, G. Gravier, and L. Chaubard, ‘The ester 2 evaluation campaign for the rich transcription of french radio broadcasts’, in *In In: Proceddings of Interspeech, Brighton (United Kingdom*, 2009.

[18] Voxforge.org, ‘VoxForge’. [Online]. Available: http://www.voxforge.org/

[19] S. Watanabe *et al.*, ‘ESPNet: End-to-end speech processing toolkit’, in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, vol. 2018-September, pp. 2207–2211, doi: 10.21437/Interspeech.2018-1456 [Online]. Available: https://waseda.pure.elsevier.com/en/publications/espnet-end-to-end-speech-processing-toolkit. [Accessed: Jan. 29, 2021]

[20] D. Povey *et al.*, ‘The kaldi speech recognition toolkit’, in *IEEE 2011 workshop on automatic speech recognition and understanding*, Hilton Waikoloa Village, Big Island, Hawaii, US, Dec. 2011.

[21] L. Ben Letaifa and J.-L. Rouas, ‘Transformer Model Compression for End-to-End Speech Recognition on Mobile Devices’, in *EUSIPCO 2022*, 2022.