FVLLMONTI

Call: **FETPROACT-2020-01**

Grant Agreement no. **101016776**

Deliverable D5.2 – Pre-trained speech ASR/MT model and use cases - V2

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	Pre-trained speech ASR/MT models
Deliverable	D5.2
Estimated Delivery	31/12/2023 (M36)
Date of Delivery Foreseen	31/12/2023 (M36)
Actual Date of Delivery	31/12/2023 (M36)
Authors	Jean-Luc Rouas – P1 – UBx, Leïla Ben Letaifa – P1 – UBx
Approver	Giovanni Ansaloni – P5 – EPFL
Work package	WP5
Dissemination	PU
Version	V1.3
Doc ID Code	D5.2_FVLLMONTI_P1-UBX-20231231
Keywords	Speech Recognition, Machine Translation, Speech Translation

DOCUMENT HISTORY

VERSION	PUBLICATION DATE	CHANGE
1.0	05.12.2023	Initial version from UBx (J.L. Rouas, L. Ben Letaifa)
1.1	11.12.2023	First content complete version (J.L. Rouas)
1.2	12.12.2023	Feedback from EPFL (J.L. Rouas)
1.3	12.12.2023	Minor corrections (G. Ansaloni, J.L. Rouas)

DOCUMENT ABSTRACT

This deliverable 5.2 describes the work carried out in the framework of the FVLLMONTI project WP5. The topics covered in this deliverable are as follows:

1. Training and evaluation of conformer models for ASR
2. Optimisation of the ASR models using pruning and quantization (from **T.5.3: Hardware-specific neural network optimizations**)
3. Comparison of the performances of the optimized models based on transformer and conformer models
4. Towards "smart" pruning to optimize performances and energy consumption (link with **T.5.2: Architecture design** and **T.5.4: Run-time optimization strategies**)

All our experiments are carried out without using any re-training method. This enables the optimisation of any model without the important cost (in money and time) linked to training large deep neural network models on graphic processors (GPU).

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
BLEU:	BiLingual Evaluation Understudy
BPE:	Byte Pair Encoding
CER:	Character Error Rate
D:	Deliverable
ESTER:	Evaluation of Speech broadcast news Enriched Transcription systems
Europarl-ST:	European parliament Speech Translation
M:	Month of the project
MT:	Machine Translation
MuST-C:	Multilingual Speech Translation Corpus
PU:	Public
SE:	Speech Enhancement
ST:	Speech Translation
TTS:	Text To Speech
VC:	Voice Conversation
WER:	Word Error Rate
WP:	Work Package

1. Framework and Databases

I. COMMON FRAMEWORK FOR SPEECH RECOGNITION / MACHINE TRANSLATION

The project is proceeding towards the creation of a unified deep learning framework for both automatic speech recognition (or speech to text) and machine translation. We described our choice to adopt the ESPNET toolkit[1] in deliverable 5.1 because of its abilities to cover all the envisioned applications in an end-to-end manner with state-of-the-art performance in various benchmarks by incorporating transformer, advanced data augmentation, and conformer models.

ESPNET is Licensed under the Apache License, Version 2.0.

II. AUTOMATIC SPEECH RECOGNITION DATASETS

For this deliverable, we only consider the English language, using the LIBRISPEECH[2] and LIBRITRANS databases. Details on these databases can be found in the deliverable 5.1.

Briefly, the LIBRISPEECH database [1] is a standard database for evaluating Automatic Speech Recognition systems, as it is publicly available and widely used to the community. It contains approximatively 1000 hours of read speech in English.

LIBRITRANS[3] [2] is a subset of the LIBRISPEECH corpus augmented with French translations[4].This dataset offers ~236h of speech aligned to translated text.

2. Baseline Models

Transformer-type ML models (see Figure 1) have seen significant interest recently, due to their good performance in various sequence-to-sequence applications, such as MT [3], language modelling and ASR [4]. Hence the choice of these models for the development of our baseline speech recognition and translation systems. The characteristics and parameters of these systems were studied in deliverable 5.1. Thus, we only recall here below the description of two transformer-based ASR systems used in this study.

[1] https://espnet.github.io/espnet/index.html
[2] https://www.openslr.org/12
[3] https://github.com/alicank/Translation-Augmented-LibriSpeech-Corpus
[4] https://perscido.univ-grenoble-alpes.fr/datasets/DS91.

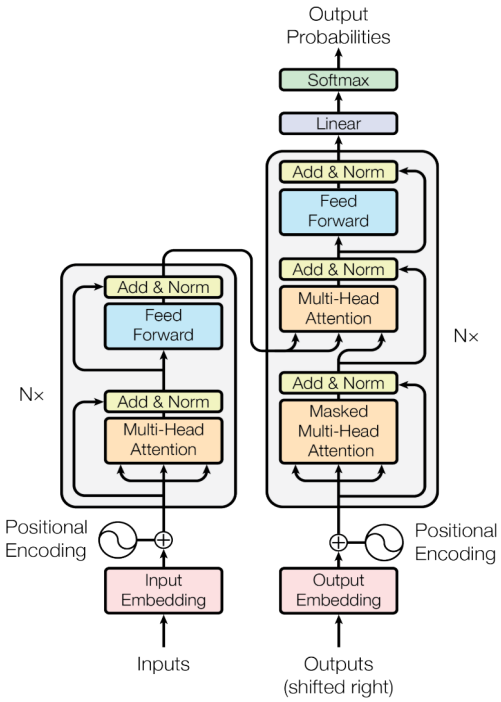

Figure 1. Block scheme of Transformer models

Additionally, after the writing of the FVLLMONTI project proposal, a new ML model type has been successfully used for speech recognition: the conformer model (see Figure 2) obtains state-of-the-art performances on several speech recognition databases [5], including the LIBRISPEECH database we use in this study. The convolution layer that differentiates conformers from transformers however increases their size.

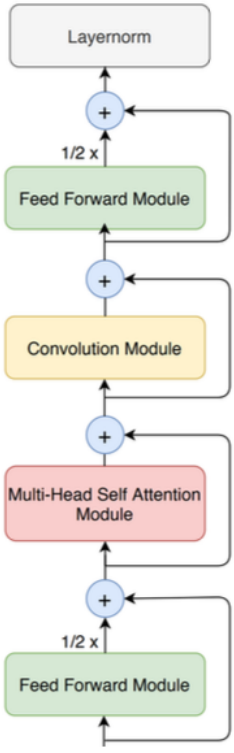

Figure 2. Block scheme of conformer models

Our aim here is thus to evaluate the performances of conformer models and compare them to transformers. To this end, we describe in the next sections the details of the models and evaluate their performances on the LIBRISPEECH database.

I. SPEECH RECOGNITION : TRANSFORMER MODELS

For this study, we built 2 different transformer models on the LIBRISPEECH corpus. We name them “Transformer-Large” and “Transformer”, respectively, in the rest of this document.

The “Transformer-Large” comprises a higher count of encoder blocks (18 as opposed to 12). The models' architectural details are shown on Table 1, where Enc/Dec is the number of encoder and decoder blocks, FFdim is the dimension of hidden layers, ATTdim the dimension of attention layers, Heads is the number of attention heads. Mem and Mem.zip are respectively the Initial and zipped memory footprint. Parameters is the number of parameters in Millions.

	Transformer Large	Transformer
Enc/Dec	18/6	12/6
FFdim	2048	2048
Att dim	512	256
Heads	4	4
Mem/Mem.zip	372/345	119/110
Parameters (M)	97.27	31.01

Table 1. Characteristics of considered transformer models

II. CONFORMER MODELS

In the same manner, we selected two conformer-based models for our experiments: a sizable conformer model (“Conformer-Large”) and a more compact conformer variant (“Conformer”) featuring reduced attention heads and smaller dimensions within the attention layers. Details are provided on Table 2.

	Conformer Large	Conformer
Enc/Dec	12/6	12/6
Att dim	512	256
Heads	8	4
Mem/Mem.zip	445/412	180/166
Parameters (M)	116.41	46.90

Table 2. Characteristics of considered transformer models

III. TRAINING SETTINGS

Both models are trained on the LIBRISPEECH database. Data augmentation is performed using 3-fold speed perturbation [6] with ratios of 0.9, 1.0, and 1.1.

During the training process, we utilize stochastic gradient descent (SGD) with the Adam update rule [7], incorporating square root learning rate scheduling [3], using 25000 warmup steps and a minibatch size of 64. Each transformer model undergoes 100 epochs of training, while each conformer model is trained for 120 epochs. We then average the results from the last 5 best checkpoints for each model type to obtain the final models.

IV. PERFORMANCE EVALUATION

The standard metric of ASR evaluation is the Word Error Rate (WER). It is defined as the proportion of word errors to words processed:

$$WER = Sub + Del + Ins / N$$

where Sub is the number of substitutions, Del is the number of deletions, Ins is the number of insertions and N is the number of words in the reference transcription.

Using this metric, the best performing systems have the lowest WERs. The expected WER varies according to the task, depending on the nature of the speech content (read speech, dictation, broadcast or conversational speech) and the noise background. It is generally considered that an "acceptable" WER should be below 10%[5].

As seen on Table 3, the leading model is "Conformer-Large", achieving a 2.8% error rate, on par with state-of-the-art conformer models with LM on the LIBRISPEECH database [5]. However, it is the largest model, with 116M parameters and a 445MB disk space footprint. The second-best

[5] https://learn.microsoft.com/en-us/azure/ai-services/speech-service/how-to-custom-speech-evaluate-data?pivots=speech-studio

performing model is “Conformer”, closely trailing with a 3.1% WER while being more space-efficient, requiring only 180MB of disk space and 47M parameters.

Regarding performances of the transformer models, “Transformer-Large” achieves a 3.4% error rate with slightly fewer parameters (97M). Finally, “Transformer” attains a 4.0% WER with significantly fewer parameters (31M) and a corresponding disk space usage of 119MB.

	Conformer Large	Conformer	Transformer L	Transformer
WER(%)	2.8	3.1	3.4	4.0

Table 3. WER (%) results obtained with conformer and transformer models.

3. Models Optimisation

We continue here the comparison of the Conformer and Transformer models by considering two compression techniques: quantization and pruning. These techniques have the advantage of being simply and directly applicable to already trained models which make them suitable for rapid deployment on mobile devices.

We studied these techniques in **T.5.3: Hardware-specific neural network optimizations** and briefly recall here the basics of these techniques before applying them to our model.

I. PRUNING

Pruning removes unimportant weights/components, by setting to zero values close to zero.

More formally, a neural network model can be defined as a function family $f(x, W)$ where x denotes the network model an W its parameters. Pruning a neural network involves taking an existing model $f(x, W)$ and generating a new model $f(x, W')$ such that $W' = M \odot W$ where $M \in \{0,1\}^{|W|}$ is a binary mask that sets some parameters to zero and $\odot$ is the elementwise product operator [8].

Pruning methods are broadly categorized into two groups:

-- Unstructured pruning [9], which involves removing individual weights. This is essentially like cutting connections by replacing weight values with zeros, resulting in sparse matrices.

-- Structured pruning [10], which targets pruning entire blocks. For instance, zeroing out entire rows or columns in a weight matrix is akin to removing neurons [11]. In the context of a transformer model, attention heads can also be removed, and sometimes entire layers or combinations of structures can be pruned.

Our focus is on pruning connections because they are the smallest, independent elements of the model. This makes them abundant enough to be pruned in significant quantities without adversely affecting performance.

Pruning can be integrated during training in various ways: Iterative pruning involves pruning between training epochs, one-shot pruning applies pruning after model training is finished [12] and pruning between fine-tuning [10].

There are two primary pruning approaches: Global pruning selects a global fraction of all layer parameters for pruning and local pruning removes a fixed percentage of parameters from each layer [8].

II. QUANTIZATION

Quantization reduces the precision of these parameters leading to less memory consumption, faster calculations and reduced hardware resource requirements [13].

Here we use quantization to map floating point values x in $[a, b]$ to $k-bit$ integers x_q in $[-2^{k-1}, 2^{k-1}-1]$.

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

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:

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

The quantization approaches can be classified into two categories: post-training quantification (PTQ) and quantification aware training (QAT) [10]

Quantization of a tensor can be done with dynamic quantization or static quantization [14].

III. NUMBER OF PARAMETERS

Table 4 displays weight proportions in the proposed models. Across all cases, the weight distribution is consistent: FF layers house the majority of weights (always above 50%), with MHA layers dominating in "Large" models or "other" layers in smaller models. When combining FF and MHA layers, they consistently account for over 75% of the weights, irrespective of the model. Consequently, we have chosen to prioritize these layers and focus on them exclusively when employing compression techniques.

	Conformer Large	Conformer	Transformer Large	Transformer
FF (%)	54.31	67.06	51.74	61.29
Conv 2d. (%)	1.98	1.26	2.42	1.93
Conv 1d. (%)	8.29	5.23	0	0
MHA (%)	19.0	11.73	25.86	16.12
Other (%)	16.42	14.72	19.98	20.66

Table 4. The proportion of weights of convolution layers (Conv), multi-head attention layers (MHA), feedforward layers (FF) and remaining layers (Other) for the conformer and the transformer models.

4. Experiments and Results

Quantization and pruning were conducted under similar conditions, allowing for a direct comparison of their outcomes, especially since both were applied in one-shot after model training. The metrics used are Memory footprint and WER.

In quantization, compression rate measures storage gain, while in pruning, it focuses on sparsity. To compare these methods, we suggest using a compression rate based on doubly compressed models. All experiments are run on a machine equipped with two 48 GB RTX8000 graphics processing units (GPUs).

I. PRUNING

We apply global unstructured pruning to the trained models in a single step using the weight magnitude and the L1 norm criteria. After pruning, we compress the models and report the WER and memory footprint data for each pruning rate in Table 5 and Table 6.

	Conformer Large		Conformer	
Pruning rate	WER	Mem.zip	WER	Mem.zip
0%	2.8	412	3.1	166
5%	2.8	399	3.1	162
10%	2.8	387	3.1	157
15%	2.8	374	6.3	151
20%	3.0	360		
25%	4.6	346	-	-

Table 5. Conformer models: WER (%) and memory footprint of the pruned then zipped models (in Mb).

	Transformer Large		Transformer	
Pruning rate	WER	Mem.zip	WER	Mem.zip
0%	3.4	345	3.7	110
5%	3.4	334	3.9	107
10%	3.4	322	3.9	103
15%	3.5	309	4.1	100
20%	3.7	297	4.3	96
25%	3.8	284	4.6	91
30%	4.0	271	5.8	87
35%	4.7	256		
40%	5.9	243	-	-

Table 6. Transformer models: WER (%) and memory footprint of the pruned then zipped models (in Mb).

Smaller models are more sensitive to pruning. For example, at a 15% pruning rate, the WER of the "Conformer" model increases to 6.3%, while that of the "Conformer-Large" model remains unchanged (2.8%). This trend is consistent in the case of transformer models. When comparing models of similar size, we find that transformer models tend to exhibit greater robustness to pruning compared to conformer models. For instance, at a 25% pruning rate, "Conformer-Large" has a WER of 4.6%, while "Transformer-Large" maintains a lower 3.8%. This same conclusion applies to "Transformer" and "Conformer" models.

II. QUANTIZATION

By default, model parameters are represented by 32-bit floats. We employ the PTQ method to reduce this precision, using 8-bit integers.

We chose the dynamic quantization because it does not require any dataset calibration and is more suitable for transformers [14]. In dynamic PTQ the weights are quantized ahead of time but the activations are dynamically quantized during inference. The quantization results, in terms of memory footprint of the quantized (and quantized then zipped) models and WER, are reported in Table 7.

	Conformer-Large	Conformer	Transfprmer-Large	Transformer
Mem.ini	445	180	372	119
Mem.ini.zip	412	166	345	110
WER.ini	2.8	3.1	3.4	4.0
Mem.quant	154	58	108	36
Mem.quant.zip	122	44	77	27
WER.quant	3.9	6.1	3.7	4.3

Table 7. Memory footprint and WER before and after quantization for conformer and transformer models

Quantization notably reduces the memory footprint of all four models by over a third. However, this memory gain comes at a cost for the conformer models, as they experience a significant increase in WER. In contrast, the transformer models handle quantization better, with their WER not surpassing 0.3% increase. This indicates that transformer models exhibit greater resilience to quantization compared to conformer models.

Our results for the transformer models align with [15], showing that larger compressed transformer models perform better than smaller, uncompressed ones. For example, the WER of large compressed models (WER=3.7% in Table 7) is lower than that of small, uncompressed transformer models (WER=4% in Table 3). It is worth noting, however, that this trend does not apply to the conformer models.

III. COMBINING PRUNING AND QUANTIZATION

We prune then quantize all models except the “Conformer” model due to its high post-quantization WER. We again measure the memory size and WER for each pruning rate of the compressed models (pruned then quantized then zipped). See Table 8 for the results.

	Conformer Large		Transformer Large		Transformer	
Pruning Rate	WER	Mem.zip	WER	Mem.gz	WER	Mem.gz
0	4	122	3.7	77	**4.3**	27
5	3.9	122	3.7	77	4.2	27
10	4	122	3.7	77	4.2	27
15	**4.3**	121	3.8	76	**4.5**	27
20	5.9	119	3.9	75	4.6	26
25	16.7	118	4.1	74	8.2	26
30			**4.5**	72	7	25
35			5	71		
40			6.7	68		

Table 8. Conformer-Large, Transformer-Large and Transformer models: WER (%) and memory footprint of the pruned, quantized and zipped models (in Megabytes).

The "Transformer-Large" model obtains the lowest WER, while the "Transformer" model presents the smallest memory footprint. This underscores the transformer models' superior compression robustness compared to conformer models. For instance, achieving a 4.3% WER is possible with either a 121 MB compressed "Conformer-Large" model or a much smaller "Transformer" model at 27 MB. In the context of transformer models, even with a 4.5% WER, the "Transformer" model maintains a smaller size (27 MB compared to "Transformer-Large"'s 72 MB).

5. Intermediate Conclusion

Results in the previous sections indicate that the Conformer models are superior to Transformer ones when using standard non-compressed models, the drawback being an increased number of parameters and therefore larger models.

While keeping in mind that FVLLMONTI aims at finding the best compromise between size and performances, we experimented what are the outcomes of compression on both Conformer and Transformer models. The results we obtained show that the Conformers are quite impacted by compression and that a better trade-off can be reached using the Transformers.

These observations support our choice of Transformer models for the FVLLMONTI project. Focusing on this type of ML algorithms, we describe in the next section further optimisation opportunities.

6. Towards "smart" optimisation for N2C2

This section describes our most recent efforts on optimisation of Transformer Neural Networks. All the following experiments are carried out using the LIBRITRANS dataset which enables us to process more experiments due to its reduced size compared to the LIBRISPEECH database.

The transformer model we use in these experiments have exactly the same configuration as the "Transformer" model trained on the LIBRISPEECH dataset described on Table 1. This model is trained only on the training subset of the LIBRITRANS dataset and have the characteristics described on Table 9

Size (MB)	107
Parameters (M)	25.4
WER	7.5

Table 9. LIBRITRANS Transformer model specifications.

First, we investigate how different settings for pruning MHA and FF layer impacts the sparsity and the WER. Then, preliminary experiments on "smart" pruning for N2C2 cells are carried out, where the pruning is done using square blocks that should allow to skip entirely some operations in the systolic array matrix-to-matrix multiplication.

I. DIFFERENT PRUNING SETTINGS FOR MHA AND FF

Results on the LIBRITRANS dataset for different settings of MHA and FF pruning rates are summarized in Table 10. Sparsity (%) and WER (%) for different pruning rate for Attention (MHA) and Feed Forward (FF) layers.

MHA	FF	SPARSITY (%)	WER (%)
0	0	0	**7.5**
0.3	0.3	29.6985	**7.9**
0.3	0.4	37.1232	**8.5**
0.3	0.5	44.5478	9.8
0.3	0.6	51.9725	14.0
0.4	0.3	32.1732	**8.2**
0.4	0.4	39.5978	**8.9**
0.4	0.5	47.0225	10.2
0.4	0.6	54.4471	15.2
0.5	0.3	34.6482	**8.9**
0.5	0.4	42.0728	9.9
0.5	0.5	49.4975	11.9
0.5	0.6	56.9221	21.4
0.6	0.3	37.1232	17.0
0.6	0.4	44.5479	21.1
0.6	0.5	51.9725	30.9
0.6	0.6	59.3972	60.0

Table 10. Sparsity (%) and WER (%) for different pruning rate for Attention (MHA) and Feed Forward (FF) layers.

Figure 3 shows the sparsity measured for different settings in MHA and FF layers. Since, the FF layers have much more parameters than MHA layers, sparsity increase more when increasing the FF pruning rate (with MFA pruning fixed at 0.3, the measured sparsity for FF pruning 0.3 is 29.9%, while

it is 37.1 when the FF pruning rate is 0.4) than when increasing the MHA pruning rate (with FF pruning rate at 0.3, if MHA pruning rate is 0.4, the sparsity is 32.2)

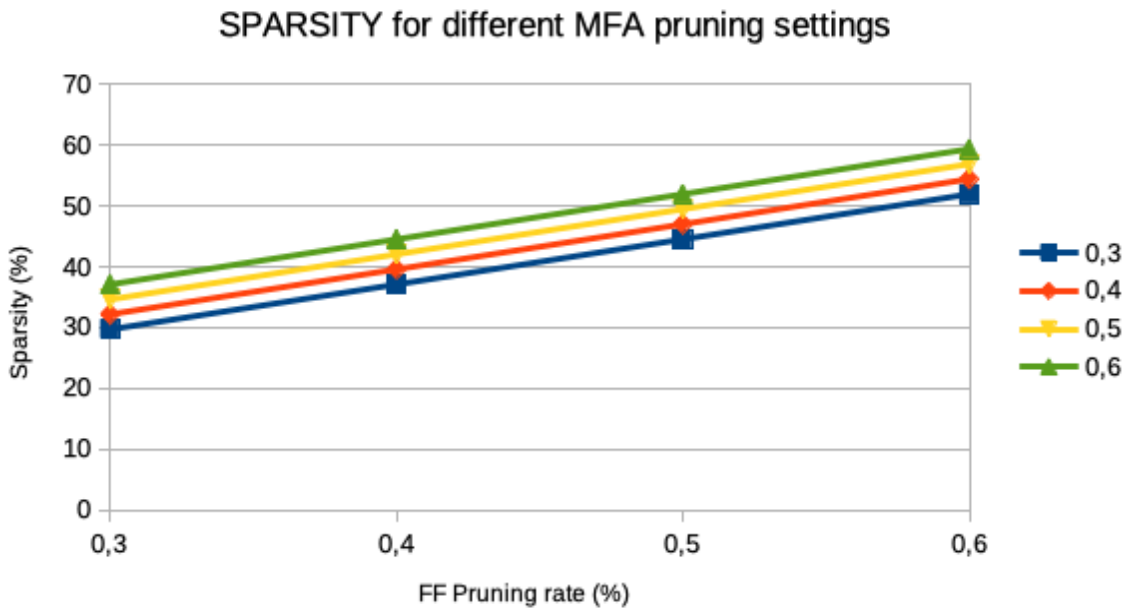

Figure 3. Sparsity as function of pruning rate for MHA layers (different colors) and FF layers (X-axis)

However, as shown in Figure 4, increasing the pruning rate for MHA layers up to 50% or 60% results in a greater increase of WER than when only increasing the pruning rate of FF layers.

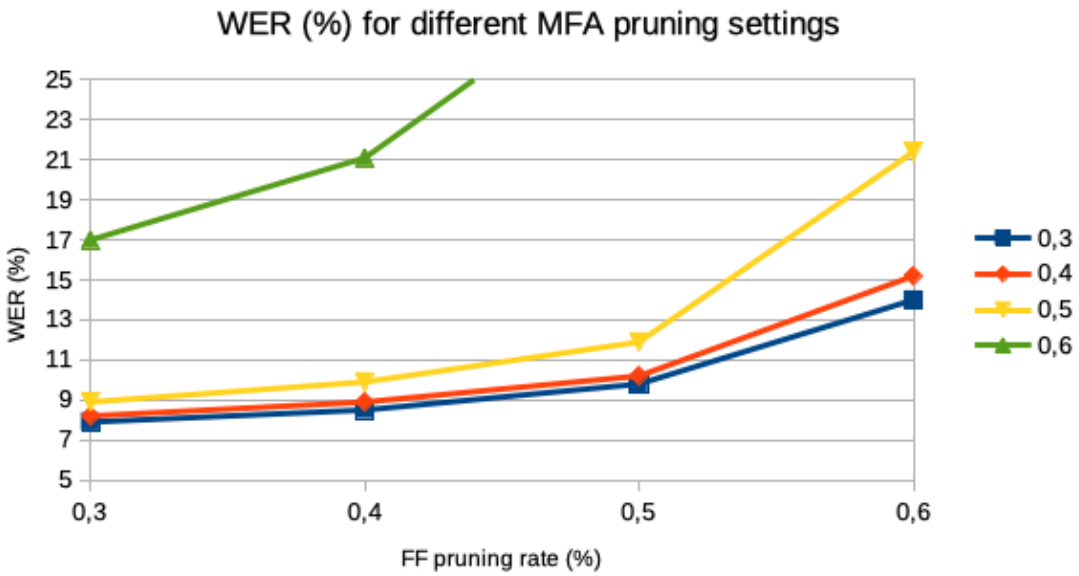

Figure 4. WER as function of different pruning settings for MHA layers (different colors) and FF layers (X-axis)

To summarize, we plot on Figure 5 the WER (%) versus the measured sparsity (%) for all settings excluding MHA pruning rate of 60%. The five best performing systems (up to 8.9% WER) are obtained using MHA pruning for 30% to 50% and FF pruning rates of 30% and 40% (in old on Table 10). In terms of overall sparsity, the best choice would be a system using equal pruning rates for MHA and FF layers at 40%, resulting in a sparsity slightly under 40% (due to the presence of other unpruned layers) and a WER of 8.9%.

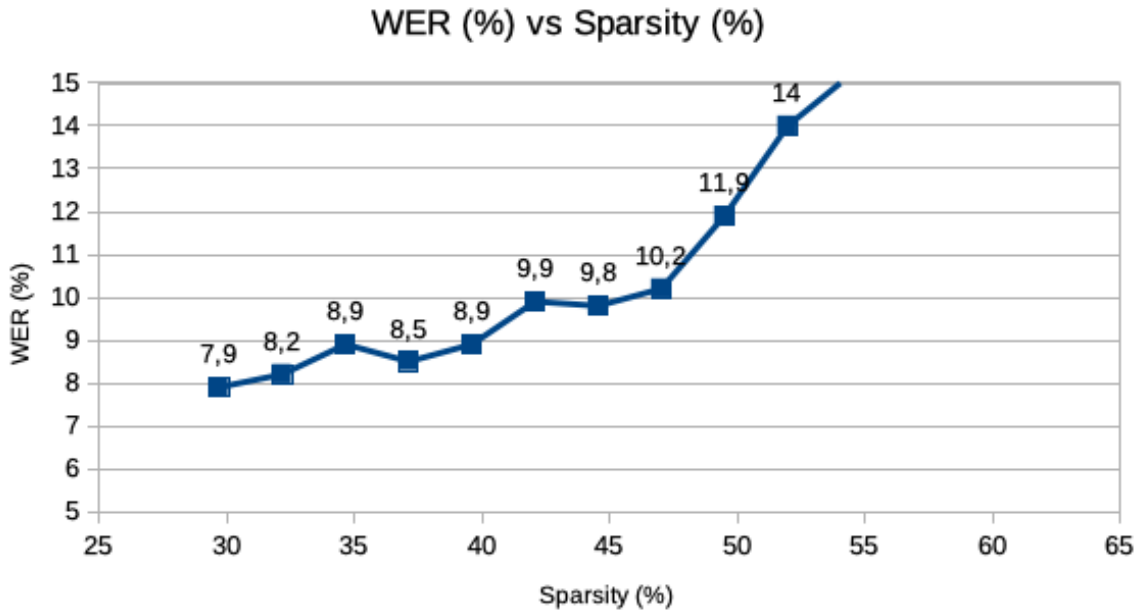

Figure 5. WER (%) vs. Sparsity (%) for different pruning rates for MHA and FF layers.

II. STRUCTURED PRUNING

Finally, we address here the idea of pruning entire blocks of weights. This idea is based on the fact that matrix to matrix multiplication can be done efficiently using a systolic array, which implies to decompose large matrices into smaller ones to compute efficiently the multiplication as devised by **T.5.2: Architecture design**. In this framework, if a small sub-matrix (hence, block) can be set entirely to zero, this will allow to completely skip the multiplication operation, resulting in gains both in terms of compute time and energy efficiency.

To this end, we studied possible settings for blocks varying in size from 2x2, 4x4 and 8x8. We first apply global pruning with different rates for MHA and FF layers as discussed in the previous section, and apply block pruning only on the FF layers. We decided to prune blocks according to their relative importance for each considered FF layer:

- First, we compute the mean of the weights of the entire layer: μ_{layer}
- Then, we compute the mean of the weights for a specified block: μ_{block}
- If $\mu_{block} < (threshold * \mu_{layer})$, with threshold a set parameter, we set all the weights of this block to zero.
- Finally, we measure the percentage of blocks pruned, the overall sparsity and the WER.

The trade-off between sparsity and WER for different block sizes and different thresholds for pruning is represented on Figure 6. Overall, the performances of the transformer model are similar in terms of WER and sparsity to what was obtained without using the block pruning (see Figure 5).

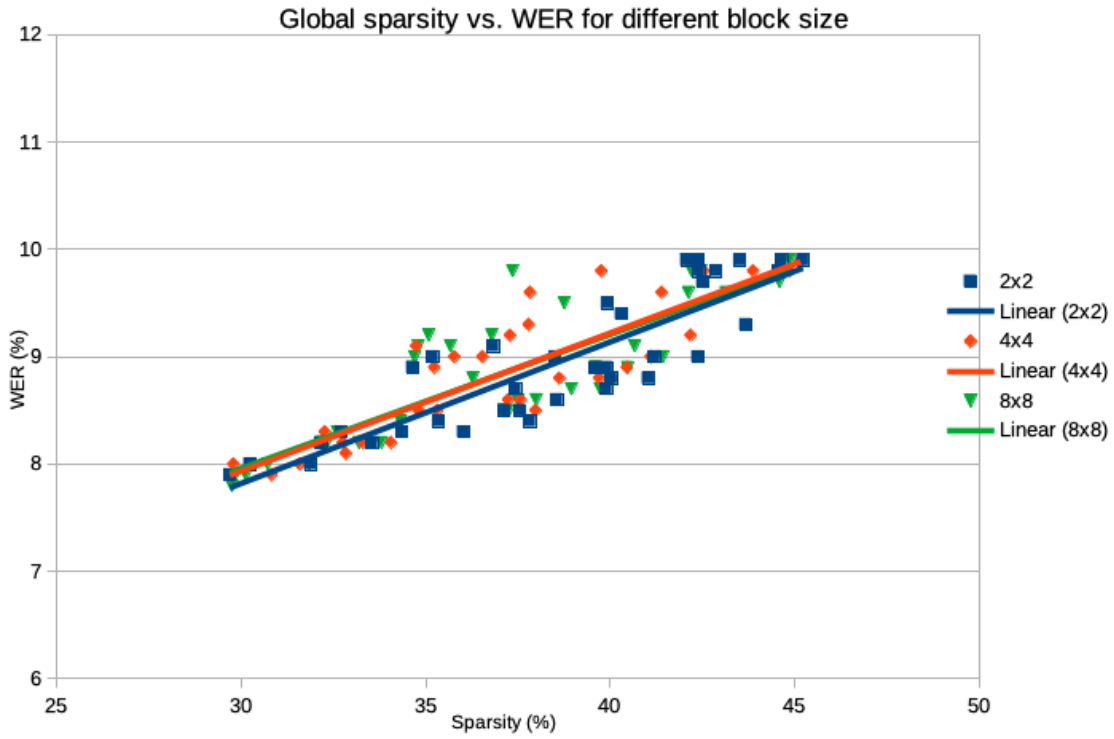

Figure 6. Block pruning: trade off between WER and sparsity for different block sizes

The best trade-off for models achieving reasonable performances (under 10% WER is considered "good quality") for the different block sizes is displayed on Table 11. Considering 2x2 blocks, we can prune more than 20% of them, resulting in an overall sparsity of about 40% while keeping the WER at 9.5%. Similar results can be obtained using 4x4 or 8x8 blocks, the percentage of pruned blocks being in these cases respectively 20% and 10%.

MHA pruning rate	FF pruning rate	block size	threshold	Sparsity (%)	#blocks (FF layers)	% pruned blocks (FF layers)	WER (%)
30,00 %	30,00 %	2x2	0,6	39,9237	4718592	23.26	9,5
30,00 %	30,00 %	4x4	0,7	37,8276	1179648	19.86	9,6
40,00 %	40,00 %	8x8	0,7	43,1395	294912	10.43	9,6

Table 11. Best trade-offs between WER and % blocks pruned.

7. Conclusion and future works

We presented in this document our most recent experiments on speech recognition in the framework of the FVLLMONTI project.

We first questioned our choice to specifically work on the Transformer models instead of Conformers. Indeed, our Conformer-based ASR system show slightly better performances, to the cost of and increased size. However, we carried out comparative compression experiments using both Conformer and Transformer models that showed that Conformer models are more impacted by compression.

Next, we analysed the pruning settings for different layers to look out if pruning attention layers may have more impact than pruning feed-forward layers. We have demonstrated that it is indeed the case, particularly when using relatively high pruning rates (from 40% up).

Finally, we described some preliminary experiments using structured pruning, that is pruning that enables further hardware optimisations. Indeed, pruning blocks inside matrices enables to completely skip operations resulting in benefits both in terms of compute time and energy consumption, which are both goals of the FVLLMONTI project. Our experiments have shown that more than 20% of the 2x2 sub-matrices of the feed-forward layers of the Transformer model can be pruned while keeping a reasonable performance in terms of WER.

It is worth noting that all the proposed optimisations are carried out without the need to retrain the model, making it possible to use any freely available model and optimize it for deployment on low-resources devices.

The future work will address the use of block pruning for a variety of models, including larger models and several languages. We will also carry out experiments on Machine Translation models and on the entire Speech-to-Text Translation chain. Moreover, in close cooperation with the other partners of WP5, in the framework of **T.5.4: Run-time optimization strategies**, we will carry out experiments to measure precisely the gains in compute time and energy consumption, particularly for the block pruning method.

REFERENCES

[1] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, 'Librispeech: An ASR corpus based on public domain audio books', in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia: IEEE, Apr. 2015, pp. 5206–5210. doi: 10.1109/ICASSP.2015.7178964.

[2] 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: European Language Resources Association (ELRA), May 2018. Accessed: Sep. 08, 2022. [Online]. Available: https://aclanthology.org/L18-1001

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

[4] 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.

[5] P. Guo *et al.*, 'Recent Developments on Espnet Toolkit Boosted By Conformer', in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 5874–5878. doi: 10.1109/ICASSP39728.2021.9414858.

[6] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, 'Audio augmentation for speech recognition', in *Interspeech 2015*, ISCA, Sep. 2015, pp. 3586–3589. doi: 10.21437/Interspeech.2015-711.

[7] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization', in *International Conference on Learning Representations*, Dec. 2014. Accessed: Dec. 08, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87e8

[8] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, 'What is the State of Neural Network Pruning?', in *machine learning and systems*, 2020. doi: 10.48550/arXiv.2003.03033.

[9] 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*, in NIPS'15. Cambridge, MA, USA: MIT Press, Dec. 2015, pp. 1135–1143.

[10] M. Gupta and P. Agrawal, 'Compression of Deep Learning Models for Text: A Survey', *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 4, p. 61:1-61:55, Jan. 2022, doi: 10.1145/3487045.

[11] 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.

[12] L. Ben Letaifa and J.-L. Rouas, 'Fine-grained analysis of the transformer model for efficient pruning', in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2022, pp. 897–902. doi: 10.1109/ICMLA55696.2022.00149.

[13] 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.

[14] O. Rybakov *et al.*, '2-bit Conformer quantization for automatic speech recognition', in *INTERSPEECH 2023*, ISCA, Aug. 2023, pp. 4908–4912. doi: 10.21437/Interspeech.2023-1012.

[15] Z. Li *et al.*, 'Train large, then compress: rethinking model size for efficient training and inference of transformers', in *Proceedings of the 37th International Conference on Machine Learning*, in ICML'20, vol. 119. JMLR.org, Jul. 2020, pp. 5958–5968.