Structural Pruning of Transformer with Gradient-Aided Regularisation for Faster Inference

Anonymous ACL submission

Abstract

Structural pruning results in dense networks that make matrix multiplication routines efficient at minimal software change. Group lasso successfully sparsifies transformer architectures on a coarser level. However, regularisation does not distinguish between layers, applying penalties with the same force no matter whether specific parameters are crucial to performance and training flow. Forcing a model to remove such parameters, especially structurally, leads to worse quality. We propose a *gradient-aided* regularisation scheme that scales layer penalties based on their gradient norm during training.

Experiments on neural machine translation with pruning entire attention heads and feedforward connections show that our method pushes the Pareto frontier for the current state-of-the-art. In the Estonian→English task, removing two-thirds of attention and feedforward parameters from *12-1.base* architecture makes inference 1.7× faster at the cost of 0.3 BLEU in quality.

In the WMT2022 Efficiency Shared Task, our pruned models maintain translation quality with 1.2–1.5× faster translation with slightly better quality in COMET compared to the baseline submissions. The pruned models landed on the Pareto frontier, offering the best quality-speed trade-off in the CPU throughput task.

1 Introduction

Transformer-based models (Vaswani et al., 2017) are a core setup in natural language processing tasks, including machine translation. Making those models smaller and faster is a fundamental requirement for successful deployment on CPU servers and mobile. Parameter pruning is one of the key methods for neural efficiency. Much of the prior work on coefficient-wise pruning does not impact speed because sparse matrix operators are difficult to optimise. On the other hand, structural sparsity has been gaining popularity since it is more robust and often does not require any sparse operators to be faster.

Structural regularisation through group lasso (Yuan and Lin, 2006) has proven to be a reliable pruning method, producing state-of-the-art optimal architectures in terms of quality and inference. For example, Behnke and Heafield (2021) gained an additional 51% inference speed-up at no quality loss by pruning feedforward and attention layers in a *12-1.tiny* transformer architecture with no quality loss.

However, a standard regularisation as a pruning approach has several drawbacks.

Firstly, regularising coarser structures forces a model to decide to zero out a large number of parameters simultaneously, leading to a higher quality drop. For example, removing an entire layer at once damages quality more than doing so gradually neuron-wise. On the other hand, removing larger subnetworks makes a transformer even faster. Ideally, entire layers could be skipped for the most optimal outcome. As can be seen in Fig. 1, there is some indication that particular layers get more sparsified than others and vice versa. Layers not eager to be pruned or even proactively refuse to do so may cause a bottleneck with quality worsening as a model is forced to balance perplexity and a regularisation penalty it cannot effectively decrease. The current research on regularisation lacks a more nuanced approach with distinct behaviour towards specific parts of a model.

Rather than pruning whole layers in one go by deciding to keep them in their entirety or not, a regulariser serves as a halfway measure, with a portion of parameters removed from a layer. However, it could also be more strict if it recognises a layer as obsolete during training. Then, it could aim to eradicate this layer from the architecture more firmly. Preferably, the regulariser should also be able to ignore layers deemed important to maintain

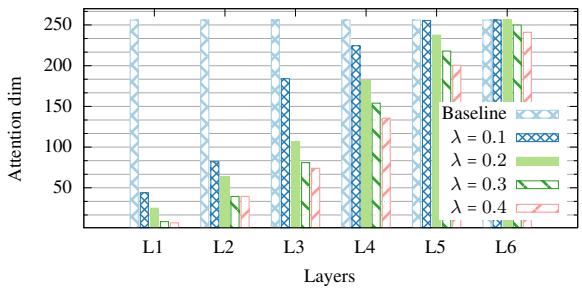

Figure 1: The pruned dimensions of context attention layers for an English→German model regularised with group lasso. In contrast to the first layer, the last one aims to keep the majority of connections active despite large penalties.

the highest translation quality possible.

In this paper, we expand upon the standard group lasso approach and introduce *aided regularisation* that scales penalties for each layer depending on a selected factor. Having a regulariser push and ease off certain layers to a varying degree with additional information plugged into its function is a research direction we explore in our work. Specifically, we propose a *gradient-aided regularisation* which scales penalties based on layer gradient norms.

We investigate this novel approach and its impact on quality and speed by building upon the current state-of-the-art efficient NMT student models trained under sequence-to-sequence knowledge distillation. We experiment with two language pairs: English→German and Estonian→English. We also include our results on the WMT22 Efficiency Shared Task.

Our key findings show that:

1. Layers with the largest gradient flow are important to quality performance. Gradient-aided regularisation reduces their penalties, allowing those layers to keep learning with lessened restrictions.
2. Across different language pairs, pruned layers exhibit similar sparsity patterns. An encoder prefers reducing middle layers, while a pruned decoder results in an ascending parameter distribution.
3. Pruning during training with this method outperforms baseline models trained from scratch, and it matches or outperforms models pruned with a standard group lasso. The resulting models are on the Pareto frontier in terms of quality-speed efficiency.
4. Snipping few parameters left in some layers and rejuvenating others allows for reducing the depth of a model with no impact on quality. For example, reducing encoder depth from 12 to 7 layers makes translation 1.5× faster with no effect on quality in our English→German setup.
5. Our submissions to WMT22 Efficiency Shared Task included models pruned under the regime, which landed them on the Pareto frontier in the task.

2 Related work

A structural approach to pruning is a popular option as it is more hardware friendly than coefficient sparsity. It is especially prevalent in image recognition (Wen et al., 2016; Molchanov et al., 2016; Lemaire et al., 2018) as we can remove entire convolutional kernels, thus reducing computational workload. The attention mechanism in a transformer model is an ideal target for structural pruning. Attention heads have high redundancy, and they can be reused (Xiao et al., 2019), approximated (Kim et al., 2019) or pruned (Michel et al., 2019; Voita et al., 2019; Behnke and Heafield, 2020). Besides attention heads, we also prune feed-forward connections in our experiments.

Pruning natural language processing models is challenging as quality, and human perception of it gets easily damaged with fewer parameters. This is especially true for machine translation as automatic scores for the task are not perfect (Kocmi et al., 2021). For example, the lottery ticket hypothesis method (Brix et al., 2020) sparsified NMT models on a coefficient level by 70 to 90% while losing between 0.6 to 3.3 BLEU points in quality. They used a sparse matrix representation for compression but reported no speed-up whatsoever. Another area for improvement with pruning NMT models is that it is not feasible to prune and retrain since it could take weeks to train a single architecture. In our research, we prune early into training and do not require retraining from scratch to achieve good results.

Gradients carry substantial information on parameters which makes them useful for pruning heuristics. Taylor expansion, utilising first-degree derivatives, can approximate the "importance" of parameters to a model and its cost function (Molchanov et al., 2019). Second-derivative methods such as Optimal Brain Damage (LeCun et al.,

1990), Optimal Brain Surgeon (Hassibi and Stork, 1992) or its group-wise version (Lebedev and Lempitsky, 2015) require a significant amount of calculations. Increasing the gradient norm through pruning for trained models can damage layers irreversibly. Gradient Signal Preservation (Wang et al., 2020) developed the measure of importance that targets parameters with this in mind.

Group lasso (Yuan and Lin, 2006) is a simple but effective structural regularisation method which has been shown to prune transformers early into training, making them smaller and faster (Behnke et al., 2021; Behnke and Heafield, 2021). There have been efforts to expand on regularisation to make it more flexible. In Selective Weight Decay (Tessier et al., 2020), weights that exceed a magnitude threshold are penalised with a standard penalty. Those beneath it get targeted by a sharper decay whose intensity grows during training. In our work, we select no threshold; we utilise gradients to steer regularisation to penalise specific layers more than others. It is a more linear pruning schedule.

Most pruning methods, including those mentioned above, need either tuning or retraining, often multiple times to be quality-efficient. Similarly to Behnke et al. (2021), we opt to prune early into training, allowing a model enough time to converge and recover its quality as much as possible.

3 Methodology

The idea behind *aided regularisation* is to use supplementary information to scale the penalties to steer them towards a specific behaviour. In practice, it means adding a new scalar γ alongside an already existing λ:

$$\frac{1}{|B|}\left(\sum_{x \epsilon B} CE(x) + \lambda \sum_{j \epsilon layers} \gamma_j^B R(j)\right) \quad (1)$$

As shown in Eq. 1, each layer has its individual γ, which gets updated after every backpropagation pass. During training, γ are exponentially smoothed:

$$\gamma_j^B \leftarrow \alpha \gamma_j^B + (1-\alpha) * \gamma_j^{B-1} \quad (2)$$

After every batch B, we calculate a local scalar γ_j for each layer j based on information gathered during this specific update, which then updates a smoothed global scalar. These γ scalars are *not* a part of the computational graph. Instead, they are calculated and updated outside of it and treated as constant nodes in the network. α is a constant used in exponential average that controls the contribution of a new element in a sequence towards the overall average. We use $\alpha = 1\text{e}{-4}$.

We explore *gradient-aided regularisation*, which scales penalties based on layer gradients.

With W_i being a regularised layer and ∇W as accumulated gradients in a model, the gradient-aided γ function is defined as:

$$\gamma_j = -log\left(\frac{\|\frac{\partial W_j}{\partial E}\|_2}{\|\nabla W\|_2}\right)$$

γ scalar increases as gradients stop flowing through a layer since it indicates that this layer does not contribute to training as much. A layer with small gradients is a good candidate to be regularised more aggressively and vice versa.

4 Setup

We experiment with two language pairs: English→German and Estonian→English. We apply knowledge distillation (Kim and Rush, 2016) under the teacher-student regime to build strong student baselines, which are already very small and fast, closely following the latest trends set by the WMT Efficiency Shared Task (Kim et al., 2019; Bogoychev et al., 2020; Behnke et al., 2021).

We trained various architectures specifying how many encoder and decoder layers they have and their dimension sizes. All models have 8 attention heads in each layer except for decoder self-attention, which is replaced by a faster SSRU (Simpler Simple Recurrent Unit) (Kim et al., 2019). The models use a shared vocabulary of 32,000 subword units generated by SentencePiece (Kudo and Richardson, 2018). The shortlisting uses the top 50 words per input token and the 50 most frequent words in the vocabulary to generate translations.

We evaluate the quality and speed on a single CPU core with 32 batch. Besides BLEU, we additionally assess experiments with chrF (Popović, 2017) and COMET[1] (Rei et al., 2020). We use SacreBLEU (Post, 2018) for BLEU and chrF. Training finishes when BLEU stops improving for 20 consecutive validations. Then, we select the checkpoint with the highest BLEU score.

We used Marian NMT toolkit (Junczys-Dowmunt et al., 2018) for training and inference. All training hyperparameters were Marian defaults

[1]We used the default 'wmt20-comet-da' metric model.

for training a base transformer model[2] unless otherwise stated. We used dynamic batching, filling a 10GB workspace on each of 4 GPUs, resulting in about 46,000 words per batch in a *6-6.tiny* student. As is more effective in the teacher-student regime, no dropout or label smoothing is applied. The optimiser is Adam (Kingma and Ba, 2014).

4.1 English→German

As a teacher, we use a WMT2019 system submitted by Microsoft to news translation task (Junczys-Dowmunt, 2019). It is an ensemble of four deep transformer-big models (Vaswani et al., 2017) with 12 layers in encoder and decoder, embedding size of 1024 and feedforward size 4096. The train-set consists of 13M sentences of available parallel data[3]. We concatenated WMT16–18 testsets to use as a development set.

We selected *12-1.tiny* as a student architecture with embeddings and feedforward dimensions set to 256 and 1536.

4.2 Estonian→English

We use a teacher provided by the Bergamot,[4] which is an ensemble of two architectures similar to those for English→German but with 6 layers instead. We trained students on 132M sentences which included about 30M of mixed forward- and backtranslations[3]. WMT18/dev was used as a development set.

We select *12-1.base* and *6-6.tiny* architectures with 512 and 2048 for embeddings and feedforward dimensions for *base*, as well as 256 and 1536 for *tiny*.

4.3 Training regime

The training regime for all of our models has three phases, following the setup of Behnke and Heafield (2021). We apply group lasso on a neuron level to both attention and feedforward layers. We consider a neural connection inactive if it amounts to less than $1e-5$.

1. Pretrain for 25k batches.
2. Train with a regulariser for N batches ($N = 250k$ for en-de, $N = 300k$ for et-en).
3. Remove inactive FFN connections. Remove entire attention heads if at least half of their connections are inactive. Collapse the architecture and then train without regularisation until convergence.

After each step, we copy the latest checkpoint and start a fresh training round with all hyperparameters reset. We checked and found no additional advantage to our baselines by refreshing learning rate scheduling or Adam optimiser. We do so to avoid partially retraining the same settings during our experimental phase, but Brix et al. (2020) found it beneficial in their pruning scheme.

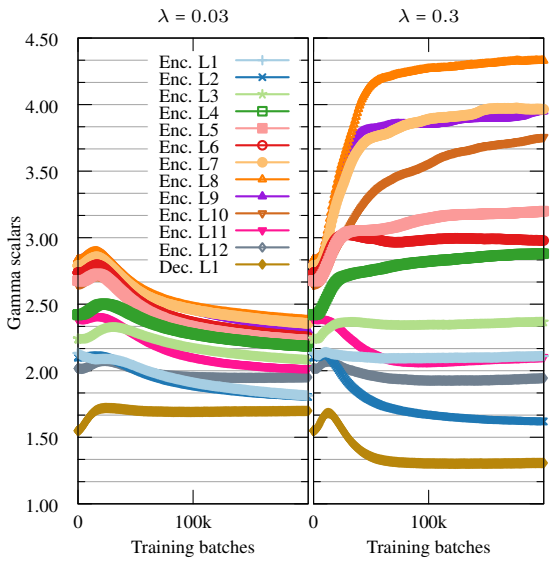

Figure 2: The progress of γ scalars for feedforward layers during training with $\lambda \in \{0.03, 0.3\}$.

5 Analysis of γ scalars

This section analyses how gradient-aided regularisation behaves when applied to a 12-1.tiny English→German model. In Fig. 2, we show how γ scalars update for all feedforward layers during training.

All the layers get immediately sorted from the start based on the gradients. Most importantly, this order is mostly kept as training goes on. Layers get prioritised during pruning based on γ applied to them. The penalties have a decreasing tendency, with some drastically rising with a larger global λ. Most layers plateau in the later stages of training, getting reduced from the start value. This pattern can be attributed to gradients getting progressively larger until they finally stabilise. Those actively pruned layers stop getting updated at some point; thus, their γ scalars rapidly increase. The reverse is also true: the single decoder layer, a known bottleneck, is assigned the lowest γ.

Instead of forcing all layers to be equally pe-

[2] Available via `-task transformer-base`.
[3] CC-BY 4.0, CC0, Public Domain, openUnder-PSI
[4] https://github.com/browsermt/students

	Reg. $\lambda \longrightarrow$	Base	0.03	0.05	0.1	0.3
FFN Dimension / Att. heads	Enc. L1	1536 8	1536 8	1536 7	1501 7	192 4
	Enc. L2	1536 8	1536 8	1536 6	1536 3	494 1
	Enc. L3	1536 8	1536 8	1536 8	649 4	146 0
	Enc. L4	1536 8	1536 8	1536 8	367 7	58 1
	Enc. L5	1536 8	1536 8	1536 8	256 5	29 0
	Enc. L6	1536 8	1536 8	1536 8	161 2	31 0
	Enc. L7	1536 8	1536 8	1535 2	64 2	4 0
	Enc. L8	1536 8	1536 1	1536 0	38 0	1 0
	Enc. L9	1536 8	1536 0	1536 0	44 0	5 0
	Enc. L10	1536 8	1536 1	1536 1	182 0	13 0
	Enc. L11	1536 8	1536 3	1536 2	1532 0	175 0
	Enc. L12	1536 8	1536 8	1536 8	1536 4	277 2
	Dec. L1	1536 8	1536 8	1536 8	1536 8	1536 8
Quality	BLEU	38.2	38.9	38.8	37.8	35.0
	chrF	63.9	64.2	64.1	63.4	61.5
	COMET	49.5	51.5	51.4	48.3	38.1
Efficiency	Att. spar.	0%	26%	37%	60%	85%
	FFN spar.	0%	0%	0%	53%	85%
	WPS	2111	2476	2506	3501	4955
	Speed-up	1.00	1.17	1.19	1.66	2.35

(a) English→German

Base	0.03	0.05	0.075	0.1	0.2	0.3
2048 8	2048 8	938 8	401 7	248 7	109 6	82 3
2048 8	2048 7	1791 6	1405 3	1070 3	417 0	237 0
2048 8	2048 6	1965 2	784 0	478 0	180 0	106 0
2048 8	2048 8	1280 7	629 2	389 2	142 0	82 0
2048 8	2048 8	2011 6	720 2	466 1	192 0	115 0
2048 8	2048 8	2048 8	1555 3	793 2	297 1	181 0
2048 8	2048 7	2048 6	1831 3	862 3	322 1	187 1
2048 8	2048 6	2048 2	1038 2	548 1	161 1	71 0
2048 8	2048 4	2048 3	1313 2	652 2	205 0	121 0
2048 8	2048 3	2048 2	1142 1	573 1	152 0	74 0
2048 8	2048 5	2048 2	1817 1	859 0	287 0	154 0
2048 8	2048 8	2048 7	2048 6	2048 5	990 4	271 2
2048 8	2048 8	2048 8	2048 8	2048 8	2048 8	2048 8
37.8	37.8	37.8	37.6	37.5	36.8	36.0
62.1	62.1	62.2	62.0	61.9	61.4	60.9
62.8	62.8	62.9	62.3	61.7	60.4	57.6
0%	0%	8%	37%	59%	79%	86%
0%	17%	36%	62%	66%	80%	87%
1784	1875	2052	2707	3025	3743	4157
1.00	1.07	1.12	1.52	1.70	2.10	2.33

(b) Estonian→English

Table 1: The results of *12-1.base* and *12-1.tiny* transformer models regularised with gradient-aided approach on a neuron-level in encoder only.

nalised, gradient-aided regularisation scales them, with the previously mentioned middle layers being the main focus of the pruning

6 Experiments

6.1 Pruning deep encoder (English→German, Estonian→English)

We start by pruning models with deep encoders, which have become popular since shifting layers from decoder to encoder reduces translation time at similar quality. We follow the setup described in Sect. 4.3, regularising only encoder layers. We experiment with *12-1.base* for Estonian→English and *12-1.tiny* for English→German. The Estonian models were pretrained for $40k$ updates, regularised for $200k$ and then allowed to converge freely. The learning rate was 0.0002. We ran experiments with $\lambda \in \{0.03, 0.05, 0.1, 0.3\}$. The German models were pretrained for $25k$ updates, regularised for $250k$ batches, and the learning rate set to 0.0003. We experimented with $\lambda \in \{0.03, 0.05, 0.075, 0.1, 0.2, 0.3\}$. The results are presented in Tab. 1a and 1b. Besides the quality and speed analysis, we also show the parameter distribution per layer.

The larger the λ, the more aggressive pruning in general. At some point, pruning tips the quality scale. Our English→German model gets $1.66\times$ faster translation in exchange for 0.4 BLEU and 1.2 COMET points. We achieved this by removing more than half of its encoder parameters. The Estonian→English performs $1.5\times$ faster with only 0.3 BLEU and 0.5 COMET in quality loss. We removed one-third of attention heads and two-thirds of feedforward connections from this specific model. With higher λ chosen, many layers get (almost) completely pruned out, leading to even better speed efficiency. As evident by parameter distribution, layers got pruned based on γ scalars applied to them. As can be seen in Fig. 1 and Tab. 1a, middle encoder layers get favoured during pruning. This pattern is less clean-cut for Estonian→English, but the trend is still noticeable.

7 Pruning both encoder and decoder (Estonian→English)

In this section, we look into pruning both the encoder and decoder layers in a transformer. Specifically, we investigate a *6-6.tiny* architecture for Estonian→English. This time, instead of modifying the λ scalar, we allowed the model to regularise for longer. Aided regularisation puts more effort into sparsifying specific layers. We hypothesise that longer regularisation with less aggressive global λ may reach a good quality-sparsity trade-off, which is especially important when pruning the decoder as it is frail to any changes.

$\lambda = 0.1 \rightarrow$		Base	200k	300k	400k	500k
Dimension / Heads	Enc. L1	1536 8	1536 8	1150 8	1283 8	1048 8
	Enc. L2	1536 8	1523 1	470 0	301 0	213 0
	Enc. L3	1536 8	1536 1	1520 0	1207 0	919 0
	Enc. L4	1536 8	1536 5	1528 4	1488 4	1394 4
	Enc. L5	1536 8	1536 7	1468 6	1405 6	1368 6
	Enc. L6	1536 8	1536 8	1536 6	1532 6	1492 6
	Dec. L1	1536 8	738 1	405 0	309 0	267 0
	Dec. L2	1536 8	1232 3	738 2	622 2	570 2
	Dec. L3	1536 8	1535 7	1083 7	708 7	512 6
	Dec. L4	1536 8	1536 8	1384 8	1247 8	1185 8
	Dec. L5	1536 8	1536 8	1533 8	1529 8	1527 8
	Dec. L6	1536 8	1536 8	1536 8	1536 8	1536 8
Quality	BLEU	36.8	36.9	36.8	36.8	36.9
	chrF	61.5	61.6	61.5	61.5	61.6
	COMET	61.0	61.3	61.2	61.4	61.3
Efficiency	FFN sparsity	0%	6%	22%	29%	35%
	Att. sparsity	0%	32%	41%	41%	42%
	WPS	3055	3635	4050	4446	4598
	Speed-up	1.00	1.19	1.33	1.46	1.51

Table 2: The results of *6-6.tiny* transformer models regularised with gradient-aided approach in both encoder and decoder for Estonian→English.

In this experiment, we allowed regularisation step described in Sect. 4.3 to last $\{200k, 300k, 400k, 500k\}$ updates with $\lambda = 0.1$. There is an upper limit on how many parameters get zeroed out before a model slows down and converges quality-wise, meaning that the choice of global λ is still important. Still, we prune slightly more than one-third of the encoder and decoder parameters under a longer regularisation process. At the same time, translation gets $1.5\times$ faster with quality across BLEU, chrF and COMET maintained. These results show that it is possible to structurally prune transformer models at no quality cost by targeting layers with the lowest gradient flow during training.

8 Pruning vs training from scratch

The question is whether the gradient-aided regularisation leads to a better translation quality or serves more as an architecture search with the same models achieving similar performance when trained from scratch. To test this hypothesis, we take the pruned models, reinitialise their parameters and re-train those architectures from scratch. We have tested this on all our English→German and Estonian→English experiments so far. The direct comparison of pruned and reinitialised experiments for both language models are presented in Tab. 3.

Even though the first two models in Tab. 3a ($\lambda \in \{0.03, 0.05\}$) lose only 0.1–0.3 in BLEU and chrF when trained from scratch relative to the baseline, the quality significantly drops in COMET. Across the experiments, the reinitialised models lose between 3.0 to 4.5 COMET points compared to their pruned counterparts. In Tab. 3b and 3c, this quality gap is also present up to 1.7 COMET loss when models of the same architecture are trained from scratch. Again, this proves that careful structural pruning, this time through aided regularisation, results in superior translation quality, clearly outperforming models trained with the same architecture from the beginning.

Reg. $\lambda \longrightarrow$		Base	0.03	0.05	0.1	0.3
BLEU	Pruned	38.2	38.9	38.8	37.8	35.0
	Reinit	—	38.0	38.1	37.0	33.6
chrF	Pruned	63.9	64.2	64.1	63.4	61.5
	Reinit	—	63.8	63.6	62.7	60.8
COMET	Pruned	49.5	51.5	51.4	48.3	38.1
	Reinit	—	48.5	48.3	44.2	33.6
Speed-up		1.00	1.17	1.19	1.66	2.35

(a) English→German, *12-1.tiny*

Reg. $\lambda \longrightarrow$		Base	0.03	0.05	0.075	0.1	0.2	0.3
BLEU	Pruned	37.8	37.8	37.8	37.6	37.5	36.8	36.0
	Reinit	—	37.8	37.6	37.4	37.0	36.1	35.5
chrF	Pruned	62.1	62.1	62.2	62.0	61.9	61.4	60.9
	Reinit	—	62.1	62.0	61.8	61.6	61.0	60.6
COMET	Pruned	62.8	62.8	62.9	62.3	61.7	60.4	57.6
	Reinit	—	62.6	62.5	61.4	60.8	58.4	56.2
Speed-up		1.00	01.05	1.15	1.52	1.70	2.10	2.33

(b) Estonian→English, *12-1.base*

$\lambda = 0.1$ / Batches →		Base	200k	300k	400k	500k
BLEU	Pruned	36.8	36.9	36.8	36.8	36.9
	Reinit	—	36.6	36.5	36.5	36.4
chrF	Pruned	61.5	61.6	61.5	61.5	61.6
	Reinit	—	61.4	61.3	61.3	61.2
COMET	Pruned	61.0	61.3	61.2	61.4	61.3
	Reinit	—	60.6	60.1	59.9	59.6
Speed-up		1.00	1.19	1.32	1.46	1.51

(c) Estonian→English, *6-6.tiny*

Table 3: The evaluation of student models pruned using gradient-aided regularisation (*Pruned*), compared to the same architecture trained from scratch (*Reinit*).

9 Snip and rejuvenate layers

For our English→German pruned models, across all 12 encoder layers, the regularisers select layers roughly between 7 to 11, meaning that they had the lowest gradient norm during training and were penalised accordingly. The experiments have shown that the layers can be successfully reduced or even entirely removed.

Until now, models kept at least a small portion of connections per feedforward layer, and we avoided

forcefully removing them, believing that maybe these scarce neurons are crucial for a model's performance. A transformer may aim to slim its middle layers while preferring to keep the depth of a model intact, while it may require fewer parameters in the middle layers than those on the edges. On the other hand, structural regularisation pursues depth reduction, and those few leftover parameters are obsolete.

Continuing the line of thought that rejuvenating diminished parameters improves the quality, the hourglass shape of a pruned architecture may serve as a gradual adaptation for *layerwise* pruning. After removing the heavily sparsified layers, the rest of the parameters can reactivate to recover as much quality as possible. This method could be an ideal solution for those who look for more shallow architectures while still preferring quality over speed.

In this section, the goal is to answer the following: after removing selected layers, can the rest of the parameters be rejuvenated to recover quality while the overall depth of a model gets reduced?

Reg. λ ⟶		Base		0.1		0.1 + S		0.1 + S&R	
Layer type ⟶		FFN	Att.	FFN	Att.	FFN	Att.	FFN	Att.
Dimension / Heads	Enc. L1	1536	8	1501	7	1501	7	1536	8
	Enc. L2	1536	8	1536	3	1536	3	1536	8
	Enc. L3	1536	8	649	4	649	4	1536	8
	Enc. L4	1536	8	367	7	367	7	1536	8
	Enc. L5	1536	8	256	5	256	5	1536	8
	Enc. L6	1536	8	161	2	0	2	0	8
	Enc. L7	1536	8	64	2	0	2	0	0
	Enc. L8	1536	8	38	0	0	0	0	0
	Enc. L9	1536	8	44	0	0	0	0	0
	Enc. L10	1536	8	182	0	0	0	0	0
	Enc. L11	1536	8	1532	0	1532	0	1536	0
	Enc. L12	1536	8	1536	4	1536	4	1536	8
	Dec. L1	1536	8	1536	8	1536	8	1536	8
Quality	BLEU	38.2		37.8		37.8		38.3	
	chrF	63.9		63.4		63.4		63.9	
	COMET	49.5		48.3		48.1		50.2	
Efficiency	Att. sparsity	0%		60%		60%		38%	
	FFN sparsity	0%		53%		63%		54%	
	WPS	2111		3501		3605		3052	
	Speed-up	1.00		1.66		1.71		1.45	

Table 4: The results of "12–1" transformer models for English→German pruned with gradient-aided regularisation using $\lambda = 0.1$ compared to it having its layer snipped (*S*) as well as layers snipped and the rest rejuvenated (*S&R*).

As presented in Tab. 4, we select the experiment regularised with $\lambda = 0.1$, which is 1.66× faster at the cost of −0.4 BLEU point in comparison to the baseline. We highlight the already removed attention layers and potential candidates for pruning in feedforward layers with less than 200 neurons, which corresponds to layers 6 to 10.

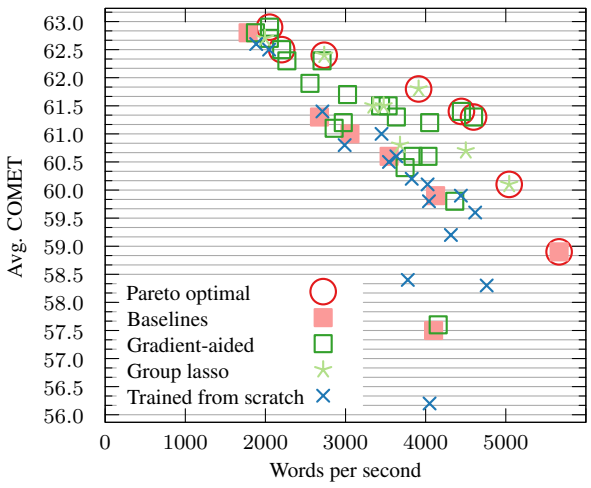

Figure 3: Pareto trade-off for Estonian→English models pruned with the gradient-aided regularisation method compared to the baselines, models trained from scratch and those regularised with a standard group lasso.

To see whether those few neurons are necessary or obsolete to the model, we *snip* the highlighted feedforward parameters out. eNow, there are 5 feedforward and 4 attention layers, with 3 full layers in the model having both types of parameters removed simultaneously. In Tab. 4, this experiment is described as $0.1 + S$ as it is the $\lambda = 0.1$ experiment with additional snipping (S). Quality-wise, the snipped model performs on par with the few removed parameters. It supports the notion that a model regularised structurally strives to reduce its architectural depth and that middle layers become obsolete as training continues.

To confirm it further and see if rejuvenating the remaining parameters helps a model recover its lost quality, we run another experiment with *snipping and rejuvenating* (S&R). We additionally prune the 7th attention layer to have 5 full layers removed in total. This way, the "12–1.tiny" architecture becomes "7–1.tiny", which significantly reduces the depth. The other layers rejuvenate after being penalised towards zero by the regulariser, resulting in 1536 dimensions across the active feedforward layer and 8 heads per attention layer. Reactivating parameters means that this model is slightly slower as it is less sparse. However, reducing the encoder depth from 12 to 7 results in 45% inference speed-up while at the same time maintaining the quality.

10 Pareto analysis

To confirm whether this pruning method is efficient, we perform Pareto analysis on our Estonian→English experiments, comparing pruned models with various baselines. For *12-1.base*, we additionally trained baselines models with encoder feedforward dimensions and attention heads reduced from $\{2048, 8\}$ to $\{1024, 4\}$, $\{512, 2\}$ and $\{256, 1\}$. For $6-6.tiny$, we trained baselines with feedforward dimensions and attention heads across the whole architecture reduced from $\{1536, 8\}$ to $\{1024, 6\}$, $\{768, 4\}$ and $\{384, 2\}$. All pruned models were reinitialised, trained from scratch, and added for consideration. To be fair in our comparison, we also ran multiple experiments with a simple group lasso pruning for both *12-1.base* and *6-6.tiny* trained with $\lambda = \{0.1, 0.2, 0.3, 0.4\}$. We visualise the Pareto analysis in Fig. 3.

Structural pruning, whether with a simple or gradient-aided group lasso, clearly outperforms baselines and identical architectures trained from scratch. Models pruned under gradient-aided regularisation are on the Pareto frontier, showing that they offer the best trade-off between quality in COMET and translation speed.

11 WMT2022 Efficiency Shared Task

To fully put our pruning method to the test, we participated in the WMT2022 Efficiency Shared Task, in which we competed with others for the Pareto optimal NMT models. We pruned three architectures: *12-1.base*, *8-4.tied.tiny* and *12-1.tiny* focusing on encoder only with the following hyperparameters:

Model	λ	Pretrain	Reg.
12-1.base	0.05	40k	300k
8-4.tied.tiny	0.3	50k	200k
12-1.tiny	0.5	50k	200k

For two of those models, we compare the baseline submissions to their pruned counterparts in Tab. 5. In both cases, our pruning made throughput translation on the CPU significantly faster while maintaining quality. For example, *12-1.base* got 1.47× speed-up while outperforming the baseline by 0.5 COMET. As highlighted in Fig. 4, those pruned models are on the Pareto frontier in the CPU throughput task, clearly outperforming other submissions. The performance gap between our pruned models and others strongly suggests that structural sparsity plays a key role in making models more robust and should be used more often for deployment.

	8-4.tiny.tied		12-1.base		12-1.tiny
Model	Base	$\lambda = 0.3$	Base	$\lambda = 0.03$	$\lambda = 0.5$
BLEU	31.9	31.9	34.0	33.7	30.2
chrF	59.8	59.8	61.4	61.4	58.6
COMET	45.0	46.0	51.0	51.5	37.6
Att. sparsity	0%	46%	0%	63%	74%
FFN sparsity	0%	20%	0%	20%	72%
Time	318.8	254.1	655.5	444.7	161.2
Speed-up	1.00	1.25	1.00	1.47	—

Table 5: Throughput CPU comparison between baseline student models and their pruned counterpart submitted to the Efficiency Shared Task. The baseline 12-1.tiny wasn't submitted to the task.

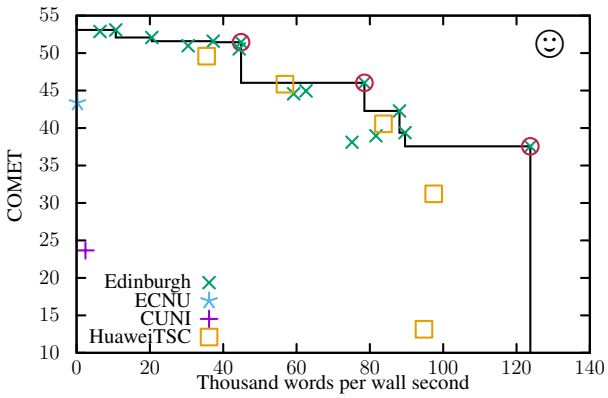

Figure 4: Pareto frontier for the CPU throughput task of the WMT22 Efficiency Shared Task. The pruned models are circled in red.

12 Conclusions

In this paper, we introduced a novel approach to regularisation, in which external information aids its penalties by scaling them accordingly. Specifically, we used layer gradients to amplify the sparsity patterns exhibited by a standard group lasso. When regularised, a transformer aims to reduce its architectural depth by focusing on pruning and collapsing middle layers. Most importantly, the analysis of the aided regularisation methods shows that they result in a significant inference speed-up that pushes the highly optimised state-of-the-art architectures forward. Additional experiments on rejuvenating pruned parameters demonstrate that it is possible to reduce the model's depth through regularisation at no quality loss if the other layers can reactivate without any penalties enforced.

References

Maximiliana Behnke, Nikolay Bogoychev, Alham Fikri Aji, Kenneth Heafield, Graeme Nail, Qianqian Zhu, Svetlana Tchistiakova, Jelmer van der Linde, Pinzhen Chen, Sidharth Kashyap, and Roman Grundkiewicz. 2021. Efficient machine translation with model pruning and quantization. In *Proceedings of the Sixth Conference on Machine Translation*, pages 775–780, Online. Association for Computational Linguistics.

Maximiliana Behnke and Kenneth Heafield. 2020. Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2664–2674, Online. Association for Computational Linguistics.

Maximiliana Behnke and Kenneth Heafield. 2021. Pruning neural machine translation for speed using group lasso. In *Proceedings of the Six Conference on Machine Translation*, Online. Association for Computational Linguistics.

Nikolay Bogoychev, Roman Grundkiewicz, Alham Fikri Aji, Maximiliana Behnke, Kenneth Heafield, Sidharth Kashyap, Emmanouil-Ioannis Farsarakis, and Mateusz Chudyk. 2020. Edinburgh's submissions to the 2020 machine translation efficiency task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 218–224, Online. Association for Computational Linguistics.

Christopher Brix, Parnia Bahar, and Hermann Ney. 2020. Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3909–3915, Online. Association for Computational Linguistics.

Babak Hassibi and David G. Stork. 1992. Second order derivatives for network pruning: Optimal brain surgeon. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'92, page 164–171, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Tom Kocmi, Christian Federmann, Roman Grundkiewicz, Marcin Junczys-Dowmunt, Hitokazu Matsushita, and Arul Menezes. 2021. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. In *Proceedings of the Sixth Conference on Machine Translation*, pages 483–499. Association for Computational Linguistics.

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

Vadim Lebedev and Victor S. Lempitsky. 2015. Fast convnets using group-wise brain damage. *CoRR*, abs/1506.02515.

Yann LeCun, John Denker, and Sara Solla. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. 2018. Structured pruning of neural networks with budget-aware regularization. *CoRR*, abs/1811.09332.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 14014–14024. Curran Associates, Inc.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. *CoRR*, abs/1906.10771.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

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

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Hugo Tessier, Vincent Gripon, Mathieu Léonardon, Matthieu Arzel, Thomas Hannagan, and David Bertrand. 2020. Continuous pruning of deep convolutional networks using selective weight decay. *CoRR*, abs/2011.10520.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. 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*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Chaoqi Wang, Guodong Zhang, and Roger Baker Grosse. 2020. Picking winning tickets before training by preserving gradient flow. *CoRR*, abs/2002.07376.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2082–2090, Red Hook, NY, USA. Curran Associates Inc.

Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. 2019. Sharing attention weights for fast transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5292–5298. International Joint Conferences on Artificial Intelligence Organization.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67.