Efficient Machine Translation with Model Pruning and Quantization
Edinburgh's Submission to WMT22 Efficiency Shared Task

Nikolay Bogoychev[†] Biao Zhang[†], Maximiliana Behnke[‡] Graeme Nail[†]
Jelmer van der Linde[†] Sidharth Kashyap[‡] Kenneth Heafield[†]

[†]University of Edinburgh, [‡]Intel Corporation

Efficiency Strategies Explored

We study various strategies for speed- and size-optimized NMT (student models):

Knowledge distillation
Optimize students on teacher's distilled data

SSRU decoder
Simple RNN-based decoder instead of self-att.

Deep encoder, shallow decoder
Increase encoder depth; decrease decoder depth

Shortlisting
Reduce softmax layer to source-aligned tokens

IBDecoder
Generate left and right words in parallel

Structural pruning with regularisation
Prune out redundant computations

Quantisation (8bit)
Quantise FP32 models into 8-bit integers

Model	Layers		Dims		Quality	Speed
	Enc.	Dec.	Emb.	FFN	COMET	Time
Teacher	6	6	1024	4096	0.591	—
Large	12	1	1024	3072	0.590	170.4
Base	12	1	512	2048	0.584	57.7
Tiny	12	1	256	1536	0.552	23.4
Micro	12	1	256	1024	0.539	20.9
Base	6	2	512	2048	0.588	50.5
Tiny	6	2	256	1536	0.554	19.6
Tied.Tiny	6	2	256	1536	0.547	17.7
Tied.Tiny	8	4	256	1536	0.562	23.0
Base.Wide	12	1	2048	2048	0.577	395.4
Base.Wide	6	2	2048	2048	0.598	374.7

Table: Architectures for the different student models. Quality and speed evaluated and averaged across WMT16–19.

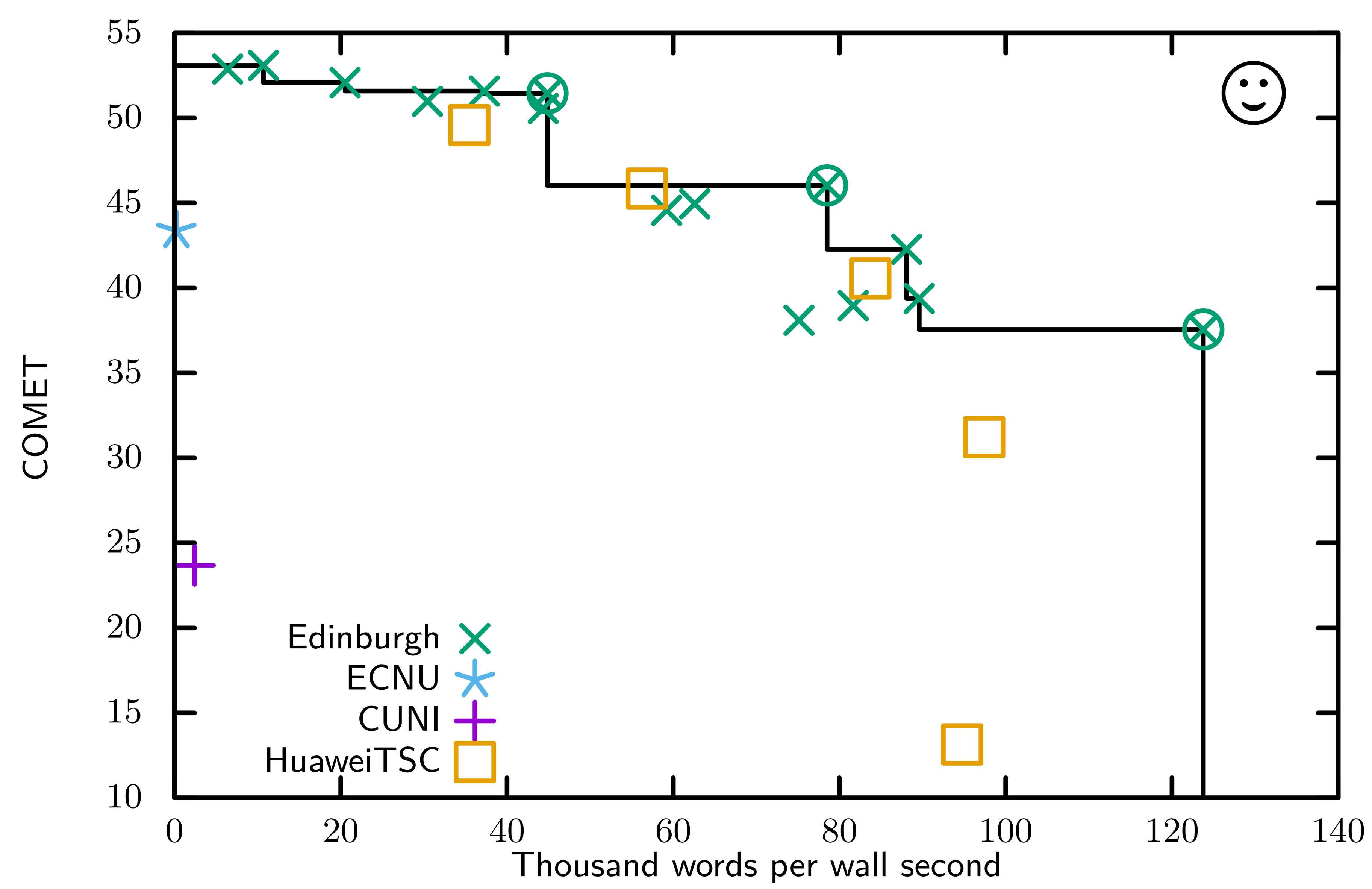

Figure: Pareto trade-off between quality and speed for the CPU throughput task. We highlight pruned models with green circles.

Interleaved Bidirectional Decoder

✓ Semi-autoregressive model by producing multiple tokens per decoding step

✓ Generate tokens from the left and the right directions simultaneously

Model	BLEU	COMET	Speedup
12-1.base	44.06	0.584	1.00
+ IBDecoder	43.84	0.561	1.12
6-2.tiny	42.76	0.554	1.00
+ IBDecoder	41.88	0.507	1.15

Results of IBDecoder compared to the baseline. Quality and speed were evaluated and averaged across WMT16–19.

Structural pruning

Removing entire attention heads and FFN connections makes models smaller and faster with no sparsity support needed.

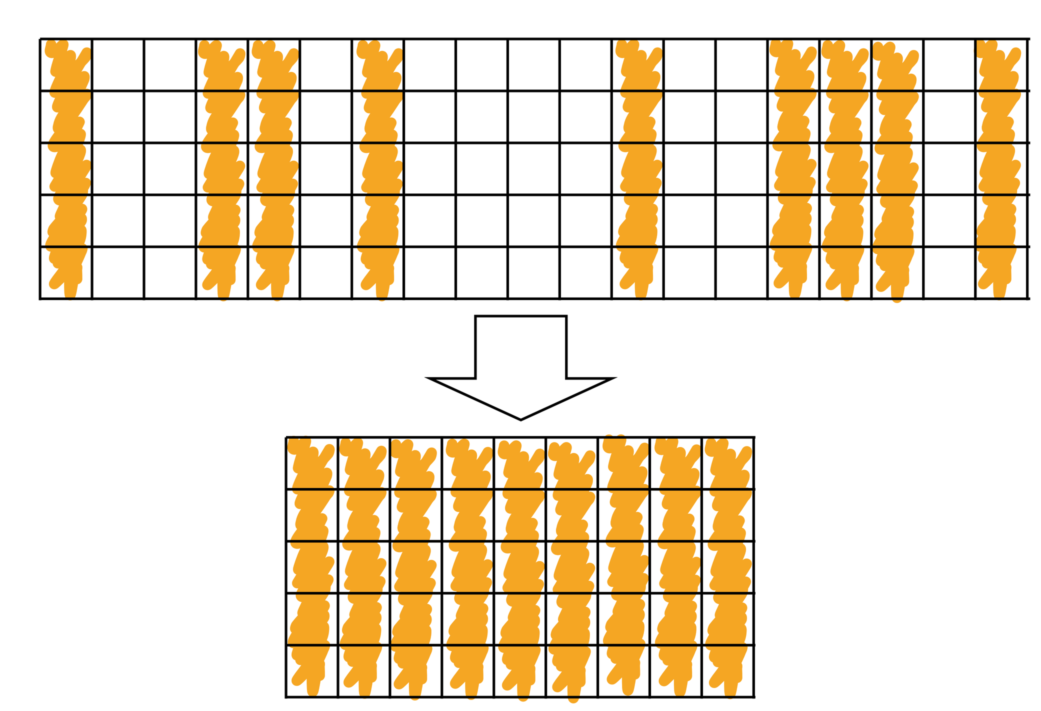

Figure: Structural pruning of nodes in FFN layers.

Aided Regularisation

We structurally pruned our transformer student models using group lasso performed under *gradient-aided regularisation*.
In practice, it means adding a new scalar γ alongside an already existing λ with B being a processed batch:

$$E(B) = \frac{1}{|B|}\left(\sum_{x \in B} CE(x) + \lambda \sum_{l \in layers} \gamma_l^B R(l)\right)$$

γ are exponentially smoothed as training progresses:

$$\gamma^B \leftarrow \alpha\gamma^B + (1-\alpha) * \gamma^{B-1}$$

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

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

Pruning Results

We focused on pruning attention and feed-forward layers in encoder only.

Model	Quality		Sparsity		Time	×
	BLEU	COMET	Att.	FFN		
8-4.tiny.tied	31.9	0.450	0%	0%	318.8	1.00
+ prune	31.9	0.460	46%	20%	254.1	1.25
12-1.base	34.0	0.510	0%	0%	655.5	1.00
+ prune	33.7	0.515	63%	20%	444.7	1.47

Table: A performance of pruned models in comparison to the baselines. Quality evaluated on the WMT22 testset.