One Wide Feedforward is All You Need

Telmo Pessoa Pires[*†] **António V. Lopes** **Yannick Assogba** **Hendra Setiawan**[*]
Equall
telmo@equall.ai
Apple
{antoniovilarinholopes, yassogba, hendra}@apple.com

Abstract

The Transformer architecture has two main non-embedding components: Attention and the Feed Forward Network (FFN). Attention captures interdependencies between words regardless of their position, while the FFN non-linearly transforms each input token independently. In this work we explore the role of the FFN, and find that despite taking up a significant fraction of the model's parameters, it is highly redundant. Concretely, we are able to substantially reduce the number of parameters with only a modest drop in accuracy by removing the FFN on the decoder layers and sharing a single FFN across the encoder. Finally we scale this architecture back to its original size by increasing the hidden dimension of the shared FFN, achieving substantial gains in both accuracy and latency with respect to the original Transformer Big.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has become the de facto paradigm in many Natural Language Processing (NLP) tasks, including Machine Translation (MT). Several studies have shown that Transformers exhibit impressive scaling-law properties (Gordon et al., 2021; Bansal et al., 2022; Ghorbani et al., 2022), wherein increasing the number of model parameters leads to further accuracy gains. In parallel with this architecture's impressive scaling of the numbers of parameters (Chowdhery et al., 2022), there is a growing trend towards reducing model footprints for real-world deployment, to satisfy practical constraints like latency requirements as well as memory and disk space limitations. In turn, researchers are actively exploring parameter sharing (Ge et al., 2022; Takase and Kiyono, 2023; Lou et al., 2022), reducing the dimensionality of Transformer components, and pruning components like attention heads (Voita et al., 2019; Michel et al., 2019).

Although the role of attention in learning pairwise dependencies between tokens is relatively well understood (Voita et al., 2019; Clark et al., 2019; Vig and Belinkov, 2019), the role of the Feed Forward Network (FFN) remains under-explored. Recently, Geva et al. (2021) established a connection between the FFN and attention by positing that the FFN corresponds to learnable *key-value* pairs where the weights of the first layer of the FFN corresponds to the *keys* and those of the second to the *values*. They find that the keys are able to capture salient textual patterns at each layer, and they notice that the classes of patterns tend to overlap between neighboring layers, indicating redundancy in the representation.

This observation motivates our work, where we revisit the conventional practice of allocating an individual FFN per layer. We investigate the effect of sharing and dropping the FFN across different layers on MT models. We conduct thorough experiments with different configurations of the Transformer, across different language pairs, including a low resource language pair and multilingual. In addition, we investigate the effect of the FFN in a decoder-only Transformer-based model. We find that a considerable level of redundancy exists between the encoder and decoder FFNs. As a result, we are able to eliminate the decoder FFN and share a single FFN across the encoder without significantly compromising the model's accuracy. This step leads not only to significant parameter savings but also opens up opportunities for further improvements. We also suggest using wider FFNs in the encoder while dropping the decoder's FFN, which results in a model with a similar size, but improved accuracy and reduced latency.

Finally we conduct a fine-grained analysis of the representational similarity between the original model, using one independent FFN per layer,

[*]Equal contribution.
[†]Work conducted while at Apple.

Proceedings of the Eighth Conference on Machine Translation (WMT), pages 1031–1044
December 6–7, 2023.

and various models with shared FFNs. Our results reveal that both model accuracy and the internal representation of Transformer blocks remain stable when sharing the FFN.

2 Background and Methodology

2.1 Transformer

The Transformer architecture has two main components: attention and the FFN, which are connected via a residual connection (He et al., 2016) and layer normalization (Ba et al., 2016). In an encoder-decoder model, there are two types of attention: self-attention and cross-attention. Self-attention is used in both the encoder and the decoder, allowing the model to focus on relevant information within the same sequence. Cross-attention is exclusive to the decoder and allows it to attend to the encoder's output. Attention takes as input a set of *queries*, *keys* and *values*, projected using four $\mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ matrices (one for the queries, keys, values, and final output) where d_{model} is the model's hidden dimension. It then applies the SOFTMAX function to allow it to focus on the most relevant values.

The FFN is applied after attention on both the encoder and the decoder and consists of the following 2-layer linear transformation:

$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2, \quad (1)$$

where a RELU non-linearity is applied to the transformation of the input sequence ($\boldsymbol{x}$). At each layer, the FFN is parameterized with two matrices, $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ where d_{ff} is the *FFN dimension* and is usually set to $4 \times d_{\text{model}}$ (Vaswani et al., 2017).

Recent work has drawn a significant link between attention and the FFN (Geva et al., 2021), wherein $\mathbf{W}_1$ and $\mathbf{W}_2$ assume roles akin to the *keys* and *values* to an unnormalized attention where the input ($\boldsymbol{x}$) acts as the *query*. Unlike regular attention, the FFN employs a RELU, which allows *multiple* keys to significantly contribute to the final output (Geva et al., 2021). Additionally, these keys correspond to an inventory of salient patterns that are learned from the training data. Geva et al. (2021) suggest that at the lower layers the FFN learns shallow syntactic patterns and progressively learns deep semantic patterns on the deeper layers. Moreover, the authors find that there's a substantial overlap between patterns captured by adjacent layers, indicating that there are redundancies in the FFNs and suggesting a better allocation of these parameters might be beneficial for performance.

2.2 Sharing and Widening the FFN

The vanilla Transformer allocates one FFN for each layer of the encoder and decoder, i.e. FFN_i^{enc} or FFN_i^{dec}, respectively. Excluding embedding parameters, these FFNs occupy around two thirds of the parameter budget, while attention occupies the remaining third[1]. Earlier work found that constraining the parameterization of the decoder FFNs causes no degradation in accuracy (Ge et al., 2022). In this work, we share the parameters of the FFN across layers and/or across the encoder and decoder to minimize redundancy between FFNs.

Let N_{enc}, N_{dec} be the numbers of encoder and decoder layers, respectively. We consider multiple configurations for parameter sharing as follows:

- One $\textbf{FFN}_{all}^{enc}$ for the whole encoder:

  $$\text{FFN}_i^{enc}(\cdot) \overset{\text{tied}}{=} \text{FFN}_{all}^{enc}(\cdot), \forall i : 1 \leq i \leq N_{enc}$$

- One $\textbf{FFN}_{all}^{dec}$ for the whole decoder:

  $$\text{FFN}_j^{dec}(\cdot) \overset{\text{tied}}{=} \text{FFN}_{all}^{dec}(\cdot), \forall j : 1 \leq j \leq N_{dec}$$

- One $\textbf{FFN}_{all}^{encdec}$ for both the encoder and the decoder:

  $$\text{FFN}_i^{enc}(\cdot) \overset{\text{tied}}{=} \text{FFN}_j^{dec}(\cdot) \overset{\text{tied}}{=} \text{FFN}_{all}^{encdec}(\cdot),$$
  $$\forall i, j : 1 \leq i \leq N_{enc}, 1 \leq j \leq N_{dec}$$

Additionally, we explore modifying the dimension of the shared FFN, which we denote as $d_{\text{ff}'}$. Setting $d_{\text{ff}'} > d_{\text{ff}}$ widens the shared FFN while $d_{\text{ff}'} < d_{\text{ff}}$ narrows it. We also consider the extreme cases of setting $d_{\text{ff}'}$ to 0 or to $(N_{enc} + N_{dec}) \times d_{\text{ff}}$ (and beyond). Setting $d_{\text{ff}'} = 0$ is equivalent to dropping the FFN[2] while setting $d_{\text{ff}'} = (N_{enc} + N_{dec}) \times d_{\text{ff}}$ is akin to sharing the concatenation of all individual FFNs.

Sharing the FFNs directly affects the number of parameters and, to a certain extent, latency. For instance, sharing FFN_{all}^{enc} for the whole encoder reduces the number of parameters by $(N_{enc} - 1) \times 2 \times d_{\text{model}} \times d'_{\text{ff}}$[3]; whereas removing the FFN on the

[1] Ignoring layer normalization, there are $4 \times d_{\text{model}} \times d_{\text{model}}$ parameters for attention vs $2 \times d_{\text{model}} \times d_{\text{ff}} = 8 \times d_{\text{model}} \times d_{\text{model}}$ parameters for the FFN, assuming $d_{\text{ff}} = 4 \times d_{\text{model}}$.

[2] In our experiments without the FFN (i.e., $d_{\text{ff}'} = 0$) we remove the residual connection and layer normalization associated with it, as they become redundant.

[3] Plus the layer normalization parameters, which we are ignoring for simplicity.

decoder, i.e., setting $d_{\text{ff}'} = 0$ for FFN^{dec}_{all}, reduces the parameters by $(N_{dec}) \times 2 \times d_{\text{model}} \times d'_{\text{ff}}$ and reduces the amount of computation to be done. This is particularly important during inference since the forward pass of the decoder is autoregressive, and changing the decoder's FFN dimension has a higher latency impact than on the encoder.

Since different configurations have different impacts, we analyse the trade-off between model size, latency, and accuracy: (i) How many parameters can be shared/pruned with negligible (if any) accuracy degradation? (ii) Are the encoder and decoder FFNs affected similarly? (iii) Keeping the same model size, can the FFN parameters be allocated more efficiently?

We propose a novel configuration, which we call the *One Wide FFN* model, consisting of a single shared wide FFN on the encoder and no FFN on the decoder. To keep the number of parameters the same as in the baseline, we increase the shared FFN dimension accordingly: FFN^{enc}_{all} with $d_{\text{ff}'} = (N_{enc} + N_{dec}) \times d_{\text{ff}}$.

For completeness, we include similar experiments on the attention mechanism in Appendix B. These experiments show that, contrary to the FFN, individual layer-specific attention weights are more important and not as redundant, as sharing the attention leads to significant accuracy drops.

2.3 Representational Similarity

Besides investigating the impact on accuracy, we study the similarity between different models in terms of their *internal representations* and the *semantic space* they produce.

We use Linear Centered Kernel Alignment (CKA, Kornblith et al., 2019) to measure the similarity between the internal representations of different models. CKA uses inner products to estimate how similar the kernel matrices of two different representations are, and is based on the Hilbert-Schmidt Independence Criterion (HSIC, Gretton et al., 2005), a statistical measure of independence of two random variables. Linear CKA uses the dot product as a kernel and can be written as:

$$\text{CKA}(\mathbf{A}, \mathbf{B}) = \frac{||\mathbf{A}\mathbf{B}^{\text{T}}||^2_{\text{F}}}{||\mathbf{A}^{\text{T}}\mathbf{A}||_{\text{F}}||\mathbf{B}^{\text{T}}\mathbf{B}||_{\text{F}}},$$

where $||\cdot||_{\text{F}}$ is the Frobenius norm while $\mathbf{A}$ and $\mathbf{B}$ are mean-centered (i.e., we subtract the mean) feature matrices of the layers under comparison, computed on the same dataset. Both matrices are $n \times d$, where n is the number of sentences in the dataset and d is the output dimension of the component, and are obtained by averaging the activation of all tokens in each sentence[4]. The linear kernel is straightforward to compute and Kornblith et al., 2019 report strong empirical performance of linear CKA compared to other kernels and methods.

To measure the similarity between the semantic spaces of different models, we use Local Neighborhood Similarity (LNS, Boggust et al., 2022). Local neighborhood similarities have been previously been used in analyzing semantic shifts in word embeddings (Hamilton et al., 2016). The premise of LNS is that two semantic spaces are similar if a sentence has similar neighbors in the two spaces. The LNS of a sentence s between models 1 and 2 is defined as:

$$\text{LNS}(s) = Sim(k\text{-NN}_1(s),\ k\text{-NN}_2(s)),$$

where k-NN(s) is the set of k nearest neighbors of sentence s for a model and Sim is the intersection-over-union (Jaccard similarity) of the two sets of neighbors. For each pair of components (attention and FFN) in models 1 and 2 we compute the LNS of all sentences in the evaluation dataset and take the mean LNS as our layer similarity measure. The smaller the value of k the more local the neighborhoods we are comparing, and the more specific the retrieval task. We pick k to be small enough to visually inspect sentence neighborhoods if necessary. In our analysis, we use cosine distance as the distance metric between activations and set k to 5% of the dataset size (~ 100 sentences).

3 Experimental Setup

Data In our experiments, we show results on WMT22 English (EN) → German (DE) (296M pairs), which we obtained using the provided `mt-data` scripts[5], WMT16 EN → Romanian (RO) (610K pairs), and for the multilingual setup of Pires et al. (2023), consisting of 10 languages: German, English, Spanish, French, Italian, Japanese, Korean, Portuguese, Swahili, and Chinese. In our analysis, we mostly focus on WMT22 EN →DE.

Following Schmidt et al. (2022), we use WMT'16 provided scripts to normalize the RO side. EN →RO keeps diacritics for producing accurate translations. For more details refer to Schmidt et al.

[4]We use the source sentence and force decode the first reference to compute the encoder and decoder representations, respectively.

[5]https://www.statmt.org/wmt22/mtdata/

(2022). For the multilingual experiments, we replicated the setup of Pires et al. (2023), which includes all details, including data preprocessing and dataset sizes.

Metrics We compute BLEU[6] using `sacreBLEU`[7] version 2.3.1, with evaluation signatures `nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp` for BLEU, and `nrefs:1 | case:mixed | eff:no | tok:flores101 | smooth:exp` for SPBLEU. For our main results, we also report COMET using the `wmt20-comet-da` model and CHRF using the signature `nrefs:1 | case:mixed | eff:yes | nc:6 | nw:0 | space:no`.

Latency We report inference time in tokens/second (the higher, the better), averaged over 5 runs. For the multilingual models, we use the DE →EN test set. Our measurements were collected using a single NVIDIA V100 GPU on a single-threaded Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with batch size of 1 and beam size of 5, in order to realistically mimic the inference of a deployed model. For experiments with larger batch sizes, see Appendix D.

Tokenization For WMT22 EN →DE, we use SENTENCEPIECE (Kudo and Richardson, 2018), with a vocabulary size of 32K and a character coverage of 1.0, while for the multilingual experiments we use a vocabulary size of 250k and a character coverage of 0.9995. For WMT16 EN →RO we use byte-pair encoding (BPE, Sennrich et al., 2016) with $40,000$ merge operations.

Model Architectures We focus our analysis on the Transformer `Big` where $N_{enc} = N_{dec} = 6$, $d_{model} = 1024$, $d_{\text{ff}} = 4096$, and it has 16 attention heads. We also report results on Transformer `Base` ($N_{enc} = N_{dec} = 6$, $d_{model} = 512$, $d_{\text{ff}} = 2048$, and 8 attention heads), and a deep encoder shallow decoder (Kasai et al., 2021) Transformer Big with 12 encoder layers, and 2 decoder layers. For our decoder-only experiments, the model is identical to the Transformer Big, except that all 12 layers are on the decoder. Our decoder-only model is similar to a Transformer-based language model, particularly Prefix-LM (Raffel et al., 2020), where we apply a non-autoregressive mask on the source side and an autoregressive mask on the target. The source and target embeddings and the output projection matrix are shared in all models (Press and Wolf, 2017).

Hyperparameters All experiments are implemented using FAIRSEQ (Ott et al., 2019). Our optimizer is ADAM (Kingma and Ba, 2015) with a learning rate of 0.0007. We train for 80k, 80k, 150k steps on WMT22, WMT16, and multilingual, respectively, at which point the models had converged. We use 4000 warm-up steps, and an inverse square root learning rate scheduler (Vaswani et al., 2017). We use a dropout rate of 0.1 for WMT22, 0.3 for WMT16, and 0 for the multilingual experiments due to the abundance of data, following Pires et al. (2023). All models are trained using `fp16` (Ott et al., 2018).

Nomenclature In our experiments, we run a number of different configurations per model architecture that differ in the way the FFN is used, shared, or dropped, as well the size of the shared FFN ($d_{\text{ff}'}$). To facilitate our discussion, we introduce in Table 1 the nomenclature that will serve as reference for the rest of the text. Unless otherwise stated, the dimension of the shared FNN^{*}_{all}, i.e. $d_{\text{ff}'}$ is equal to the d_{ff} of the original model.

For decoder-only models, only `SharedDec` and `NoDec` configurations are defined. For conciseness, we drop the mention of FFN from the text when possible, i.e. `SharedEnc` instead of `SharedEncFFN`.

FFN Description	Encoder	Decoder
`SharedEnc`	FNN^{enc}_{all}	FNN^{dec}_{i}
`SharedDec`	FNN^{enc}_{i}	FNN^{dec}_{all}
`SharedEncSharedDec`	FNN^{enc}_{all}	FNN^{dec}_{all}
`SharedEncDec`	$\text{FNN}^{encdec}_{all}$	
`NoDec`	FNN^{enc}_{i}	`No-op`
`SharedEncNoDec`	FNN^{enc}_{all}	`No-op`

Table 1: Nomenclature used in our experiments. `No-op` indicates an identity function, which is equivalent to dropping the FFN.

Representational Similarity We use the WMT22 EN →DE evaluation set for both CKA and LNS analysis. We analyze encoder and decoder representations independently and present these metrics in a matrix heatmap plot showing pairwise similarity between layers. The diagonal of this matrix is the similarity of corresponding layers, i.e., layer i on both architectures. In order to

[6]For the multilingual experiments, we select the Flores101 tokenizer in `sacreBLEU`, so technically we report SPBLEU.

[7]https://github.com/mjpost/sacrebleu

facilitate an "apples-to-apples" comparison across models, we extract decoder representations by force decoding the (first) reference. We establish 2 crucial similarity scores: a *benchmark* on similarity for each of these metrics, where we train two additional models using the same architecture but with different random seeds; a similarity lower bound, where we compare the baseline Transformer Big with a randomly initialized (i.e., untrained) model with the same architecture. We present these bounds in Appendix C.

4 Experimental Results

4.1 Sharing FFNs

The results of various FFN sharing configurations are summarized in Table 2, including their impact on accuracy and model size (in millions of parameters and percentage). Sharing either the encoder (SharedEnc) or the decoder FFN (SharedDec) results in just a 0.2 to 0.3 BLEU point decrease, while reducing the parameter count by nearly 20%. Sharing the FFN on each side (ShareEncShareDec) leads to a more substantial degradation of 0.9 BLEU points, albeit reducing the parameter count by 37%, while sharing a single FFN on the encoder and decoder (ShareEncDec) results in a slightly higher degradation of 1.1 BLEU points. Nonetheless, these findings support the hypothesis that the FFN contains some degree of redundancy, as we expected a greater accuracy degradation given the substantial $(20-40\%)$ reduction in model size.

Architecture	BLEU	$\lvert\theta\rvert$	(%)
Transformer Big	35.6	228M	(100)
+ SharedEnc	35.4	186M	(82)
+ SharedDec	35.3	186M	(82)
+ SharedEncSharedDec	34.7	144M	(63)
+ SharedEncDec	34.5	136M	(59)

Table 2: sacreBLEU results on WMT 22 EN →DE for different FFN sharing configurations. $\lvert\theta\rvert$ is the number of parameters.

While we focus on sharing *one* FFN for all layers within a module, we compare with sharing multiple FFNs following Takase and Kiyono (2023) in Appendix A. We find that sharing one FFN is as accurate as sharing multiple FFNs within a module, while being more parameter-efficient.

4.2 Dropping FFNs

Table 3 summarizes the performance of models with no FFNs. Besides BLEU and number of parameters, we report the inference speed for each architecture. Dropping the FFN on the encoder (NoEnc) leads to a 0.9 BLEU point drop while reducing the parameter count by 22% and with minimal effect on inference speed. Dropping the FFN on the decoder (NoDec), on the other hand, causes a degradation of only 0.4 BLEU points while increasing the inference speed by 20%[8]. The highest latency reduction is obtained by removing the FFNs on both the encoder and the decoder (NoEncNoDec), but it comes with a significantly larger degradation of over 2 BLEU points.

Architecture	BLEU	Speed	$\lvert\theta\rvert$	(%)
Transformer Big	35.6	$111^{\pm 1.2}$	228M	(100)
+ NoEnc	34.7	$112^{\pm 1.0}$	178M	(78)
+ NoDec	35.2	$133^{\pm 0.9}$	178M	(78)
+ NoEncNoDec	33.5	$138^{\pm 1.9}$	127M	(56)
+ SharedEncNoDec	35.3	$136^{\pm 1.1}$	136M	(60)
+ NoEncSharedDec	33.9	$127^{\pm 1.0}$	136M	(60)

Table 3: sacreBLEU results on WMT 22 EN →DE for different FFN dropping configurations.

Combining sharing and dropping These results, together with those from Table 2, suggest that the encoder and decoder FFNs have different contributions: the decoder's are more redundant, corroborating previous work on FFNs parametrization (Ge et al., 2022). With this in mind, we experiment with one shared FFN on the encoder and dropping it on the decoder, reported as SharedEncNoDec in Table 3. As shown, with just approximately 60% of Transformer Big parameters we observe a 22% improvement in inference speed, at the cost of 0.3 BLEU point.

4.3 One Wide FFN Model

Previous sections describe models that share and/or drop FFNs, effectively reducing model size at some modest accuracy cost. In this section, we investigate whether we can regain the accuracy lost while preserving the parameter efficiency and the latency reduction. We focus on ShareEncNoDec model as

[8]The reason for this difference between NoEnc and NoDec is that the encoder output is computed in parallel, while the decoder operates in a step-by-step fashion.

	BLEU	CHRF	COMET	Speed	$\mid \theta \mid$ (%)
Transformer Big EN →DE	35.6	62.6	57.2	$110.8^{\pm 1.2}$	228M (100)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 4,096$	35.3	62.1	56.1	$135.7^{\pm 1.1}$	135M (60)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 24,576$	35.7	62.7	57.9	$138.2^{\pm 0.9}$	177M (80)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 49,152$	**36.5**†	**63.2**†	**59.6**	$137.5^{\pm 1.6}$	228M (100)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 98,304$	36.4†	**63.2**†	59.0	$134.5^{\pm 1.6}$	328M (145)

Table 4: Accuracy of One Wide FFN for Transformer Big EN →DE on WMT22. † implies the system is statistical significantly different at $p < 0.05$.

it provides a strong baseline with significant parameter savings and inference speedups.

We propose increasing the dimension of the shared FFN to match the number of parameters of the original (fully-parameterized) model, so as to avoid increasing the overhead of model storage. In particular, ShareEncNoDec saves around $(N_{enc} + N_{dec} - 1) \times 2 \times d_{model} \times d_{\text{ff}}$ parameters as there's one single shared FFN in the encoder. On the other hand, the Transformer Big has $(N_{enc} + N_{dec})$ FFNs. Thus, we match the size of the original model by setting the dimension of the shared FFN, $d_{\text{ff}'}$, to $(N_{enc} + N_{dec}) \times d_{\text{ff}}$.

Table 4 summarizes our results. It includes our proposed model, the *One Wide FFN* model ($d_{\text{ff}'} = 49,152$), as well as the baseline Transformer Big, and the corresponding ShareEncNoDec ($d_{\text{ff}'} = 4,096$). It also includes a wide model with $d_{\text{ff}'} = 24,576$, which uses the same number of parameters as NoDec, with $d_{\text{ff}'} = N_{enc} \times d_{\text{ff}}$. This model achieves an accuracy on par (or slightly above) the baseline Transformer Big with 20% fewer parameters and a significant inference speed-up.

Our proposed model with $d_{\text{ff}'} = 49,152$ goes beyond that, achieving a gain of 1.2 BLEU points over the vanilla ShareEncNoDec and 0.9 BLEU points over the Transformer Big. These gains remain consistent across CHRF and COMET. Furthermore, it has a similar inference speed as the ShareEncNoDec model. For completeness, we include a wider model with $d_{\text{ff}'} = 98,304$. Despite the extra capacity, this model does not provide any additional accuracy gains, which we suspect is due to the lack of data to train such a large model.

4.4 Analyzing Internal Representations

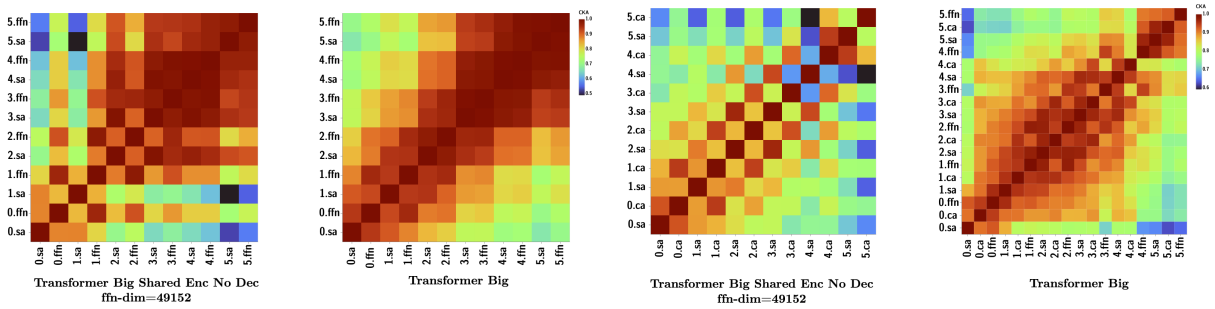
We now report a *post-hoc* analysis of the internal representations of the models introduced in preceding sections. Our objectives are twofold: 1) to ascertain whether the proposed models' internal representations exhibit a significant degree of similarity to those of the original base model; 2) to delve into the impact of the proposed methods on redundancy. We adopt the definition of redundancy of Dalvi et al. (2020), who *visually* inspect the similarity between adjacent modules within a model (high similarity entails high redundancy).

Architecture	Encoder		Decoder	
	CKA	LNS	CKA	LNS
Benchmark	100.0	100.0	100.0	100.0
SharedEnc	98.0	96.2	100.8	100.6
SharedDec	100.2	101.4	98.3	94.6
SharedEncSharedDec	98.9	97.2	99.5	95.4
SharedEncDec	97.6	94.4	98.4	93.5
NoEnc	90.0	70.5	101.0	96.8
NoDec	100.0	98.6	96.0	87.4
SharedEncNoDec	97.6	98.9	97.5	89.0
SharedEncNoDec$^{d'_{\text{ff}}=49152}$	97.0	83.2	94.0	82.9

Table 5: Similarity of the representations (%) of corresponding modules of different architectures vs. the Transformer Big for WMT22 EN →DE. These scores are normalized by comparing them to the CKA and LNS benchmark scores. For NoDec configurations we compare the final output of the Transformer layer as a whole as they have different modules than the baseline. The columns for shared and for dropped FFNs are highlighted in gray and blue respectively.

4.4.1 Similarity to Baseline

We ground the pairwise similarity metrics, by normalizing them against a benchmark. As mentioned in Section 3, we establish the *benchmark scores* by training two additional Transformer Big models, but using different random seeds. These models achieve similar accuracy as the baseline model (see Appendix C.1 for more details). The benchmark score is the similarity between the baseline and these models Because the benchmark is calculated by averaging similarity scores from different train-



(a) Encoder self similarity.

(b) Decoder self similarity.

Figure 1: CKA self similarity of encoder and decoder layers of the *One Wide Encoder* model vs. the Transformer Big baseline. We identify each component with a label: `index.name`. For example, `0.sa` refers to the self-attention on layer 0, while `4.ca` refers to the cross-attention on layer 4.

ing runs of our baseline, individual runs can have a normalized score above 100%.

Table 5 shows normalized similarity scores for several models. Under the Encoder columns we compare the encoder representations, and under the Decoder columns we compare decoder representations. Sharing FFNs leads to consistenly lower (normalized) similarity scores than models that do not share, both in terms of internal representation (CKA) and semantic spaces (LNS). As shown, although models that share FFNs have lower similarity scores compared to those that do not, the scores are still very close to 100%. Moreover, these decreases align with the drops in BLEU seen in Table 2, where the model with the lowest similarity score (`ShareEncDec`) is also the least accurate model. We observe a similar trend for models that drop the FFNs in the encoder or decoder, these models exhibit lower similarity scores with the respective component than models sharing them, as shown by `NoEnc` and `NoDec`. In addition, the former result again suggests the FFNs in the encoder are more important than in the decoder as the similarity shifts drastically compared to all other settings.

For completeness, we report on the last row the similarity scores for the One Wide FFN model, which is more accurate than the base model. The internal representations generated by that model diverge from those of the base model. Interestingly, we observe a larger drop in LNS scores than in CKA scores, indicating that the shift occurs mostly in semantic space, rather than the Euclidean space captured by CKA. For a detailed layer-wise similarity analysis that breaks out the aggregate analysis in Table 5 see Appendix C.2.

4.4.2 A Qualitative View of Redundancy

We now study into the impact of our One Wide FFN model on the redundancy of the internal representations. In addition to adopting their definition of redundancy, we also adopt Dalvi et al. (2020)'s method of computing self-similarity, namely looking at how the representations change as they go through each module (self-attention, FFN, or cross-attention) of the model. In particular, we use CKA to compute similarity between the output of different modules within the same model.

In Figure 1a, we show the CKA self-similarity matrices for the encoders of the One Wide FFN model and the `Transformer Big`. We do the same for the decoders in Figure 1b. These matrices show how similar each module of the network is to all other modules *within that network*. The diagonal of the matrix is the similarity between a module and itself and is always 1.

As shown, there is high similarity between adjacent modules of the `Transformer Big`, both on the encoder and decoder, indicated by areas with darker red around the diagonal. The prevalence of high similarity patterns among adjacent modules suggests a substantial degree of redundancy, and eliminating a module has a negligible impact on the final representations. On the other hand, we observe a distinct checkerboard pattern on the self-similarity matrices of the One Wide FFN model, where individual modules tend to exhibit lower similarity with their immediate neighbors than with their second neighbors (i.e., the neighbors of the neighbors). On the encoder, the checkerboard pattern emerges especially in the earlier modules while on the decoder, that pattern appears more consistently throughout the layers. This pattern gives an indication that our model is learning non-trivial

transformations of the input, leading to decreased redundancy within the network.

4.5 Other architectures and Languages

So far, all our experiments focused on the Transformer Big and on WMT22 EN →DE. In this section, we apply what we learned to other architectures and language pairs. We run experiments on the low resource language direction EN →RO and a large scale multilingual model.

For EN →DE, we apply our proposal to a Transformer Base model, a Deep Encoder Shallow Decoder model (Kasai et al., 2021), and a Decoder-Only model. For the Transformer Base, we observe an accuracy gain of 0.5 BLEU (2.2 BLEU over the vanilla `SharedEncNoDec` model) and an inference speedup of around 25%. In the Deep Encoder Shallow Decoder model, we observe a more modest accuracy gain of 0.2 BLEU points (0.9 BLEU over the vanilla `SharedEncNoDec` model). However, the inference speedup from dropping the decoder FFNs is minimal ($< 1\%$), which is expected because of the small depth of the decoder in this architecture.

Decoder-only models With the advent of Large Language Models (LLMs) like GPT (Brown et al., 2020), and PaLM (Chowdhery et al., 2022), a lot of effort has been put on decoder-only Transformer models. We train a decoder-only model on WMT22 EN →DE, as shown on Table 6. Due to the absence of an encoder, we are limited to applying a wide FFN on the decoder side. As in the other setups, we get an accuracy gain of +0.3 BLEU over the baseline decoder-only model (+1.7 BLEU over `ShareDec`), but the latency degrades by 12%. This is not surprising: due to the autoregressive nature of the decoder, increasing the size of its FFN has a bigger impact on speed.

Low-resource languages In EN →RO the accuracy of the One Wide FFN Model is only on par compared to the base model, even though it is a higher than the vanilla `SharedEncNoDec` model. We hypothesize that due to the low resource condition, our proposed model already reaches saturation as there are not that many salient textual patterns to be learned by the FFN.

Multilingual Finally, we observe the similar trend on the multilingual setup, where the One Wide FFN Model is +1.2 SPBLEU points more accurate than the baseline Transformer Big and +2.5 SPBLEU points more accurate than the vanilla `SharedEncNoDec`, this gain is significant in 79 out of 90 directions and when all tests sets are concatenated. Additionally, this large accuracy gain also comes with around 18% inference speed-up, consistent with our previous results.

5 Related Work

Weight pruning and parameter sharing are well-known techniques to reduce a model's footprint. Given the scale of the latest models (Chowdhery et al., 2022), there have been multiple efforts to prune neurons based on different automatic methods (Dalvi et al., 2020; Michel et al., 2019; Voita et al., 2019), sharing parameters efficiently (Ge et al., 2022; Reid et al., 2021), and factorizing certain components (Lan et al., 2020; Hu et al., 2022).

Neuron pruning methods often focus on finding and pruning redundant neurons through correlation methods (Dalvi et al., 2020), but also on how Transformer components like the multi-head attention can be pruned significantly due to model redundancy in the encoder or decoder either by checking the gradients salience (Michel et al., 2019) or a differentiable relaxation of the l_0 regularization at training time (Voita et al., 2019).

For parameter sharing, the Universal Transformer (Dehghani et al., 2019) proposed a model where all layers are shared (i.e., in effect it reduced the model to a single shared layer). Takase and Kiyono (2023) proposes finding an optimal configuration of shared layers in the encoder or decoder through different methods of sharing (in sequence, in cycle, or in reversed cycle) always keeping a specified number of final layers[9]. Similarly, Reid et al. (2021) proposes an approach where just the middle layers are shared, while the bottom and top layers are independent, and using a lower dimensionality for the embedding layer. Analogously, Ge et al. (2022) focus on minimizing the number of parameters and the number of calls to each parameters' group in order to optimise on-device models. They achieve this by sharing the encoder and decoder in a similar way to both previous methods, particularly by sharing all layer parameters in cycle like Takase and Kiyono (2023).

Previous works also focus on reducing the dimensionality of certain parameters, mostly through low rank factorization. Lan et al. (2020) decomposes the embedding layer into a lower rank em-

[9]See Appendix A for a detailed description and comparison.

	BLEU	CHRF	COMET	Speed	$\lvert\theta\rvert$ (%)
Transformer Base EN →DE	34.2	61.6	54.1	$116.3^{\pm 0.9}$	70M (100)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 2,048$	$32.5^{\dagger}$	$60.1^{\dagger}$	50.0	$146.0^{\pm 1.6}$	47M (67)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 24,576$	**34.7**	**61.8**	**55.6**	$146.8^{\pm 1.3}$	70M (100)
Transformer Decoder-Only EN →DE	35.8	62.8	57.7	$79.8^{\pm 1.9}$	202M (100)
+ ShareDec FFN $d_{\text{ff}'} = 4,096$	$34.4^{\dagger}$	$61.7^{\dagger}$	54.1	$79.7^{\pm 1.3}$	110M (48)
+ ShareDec FFN $d_{\text{ff}'} = 49,152$	**36.1**	**62.9**	**59.4**	$69.3^{\pm 0.2}$	202M (100)
Transformer Deep Enc. Shallow Dec. EN →DE	35.5	**62.4**	58.0	$230.1^{\pm 0.8}$	236M (100)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 4,096$	$34.8^{\dagger}$	$61.6^{\dagger}$	55.4	$235.0^{\pm 0.5}$	127M (54)
+ ShareEncNoDec FFN $d_{\text{ff}'} = 57,344$	**35.7**	**62.4**	**58.9**	$233.5^{\pm 0.7}$	236M (100)
Transformer Base EN →RO	**22.9**	**52.9**	**50.9**	$119.3^{\pm 1.1}$	64M (100)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 2,048$	$22.2^{\dagger}$	$52.5^{\dagger}$	45.8	$152.8^{\pm 1.4}$	41M (64)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 24,576$	**22.9**	52.8	46.7	$150.6^{\pm 0.5}$	64M (100)
Transformer Big Multilingual	26.8	46.3	47.7	$94.6^{\pm 1.6}$	422M (100)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 4,096$	$25.5^{\dagger}$	$45.1^{\dagger}$	40.8	$107.1^{\pm 1.4}$	330M (78)
+ SharedEncNoDec FFN $d_{\text{ff}'} = 49,152$	$\mathbf{28.0}^{\dagger}$	$\mathbf{47.3}^{\dagger}$	**50.7**	$111.5^{\pm 1.1}$	422M (100)

Table 6: Accuracy of One Wide FFN for EN →DE with Transformer Base, Decoder Only, and Deep Encoder Shallow Decoder on WMT22; for low resource EN →RO with Base version on WMT16, and multilingual with Transformer big on Flores. † implies the system is statistical significantly different at $p < 0.05$.

bedding matrix and a projection to the actual hidden size while also sharing all parameters across all layers. In addition to sharing parameters efficiently, Ge et al. (2022) proposes a lightweight decomposition of the FFN where instead of a single component there are 2 projections with a smaller dimensionality than vanilla Transformers. Our work is close to Ge et al. (2022) but instead of factorizing we explore sharing and full pruning of the FFN. In contrast with previous works, we also explore increasing the encoder FFN size while dropping the decoder's completely.

6 Conclusion

In this work, we studied the importance of the FFN in Transformer models. We analyzed the impact of removing and/or sharing the FFN across layers and found that, due to this component's redundancy, the model sizes can be substantially reduced with little impact on accuracy for Machine Translation. In particular, we found that sharing the FFN across all encoder layers while making it larger and removing it from the decoder layers leads to models that are more accurate and faster at inference.

Our findings are applicable across multiple settings, including decoder-only and multilingual models. In a low-resource setting the results are modest but our approach can still recover the baseline's performance with a faster inference.

Finally, we conducted a thorough similarity analysis between the vanilla Transformer and our proposed architectures, and found that the latter's internal representations do not differ significantly from the former's, except in that they are less redundant.

Limitations

In this work, our focus was Machine Translation. Although we expect the results to generalize to other sequence-to-sequence tasks, further experiments are needed, which we leave for future work.

Ethics Statement

One important consideration is the energy consumption for model training, which results in greenhouse emissions (Strubell et al., 2019). Our work uses existing datasets, and inherits some of the risks associated with them, such as privacy leakage (Carlini et al., 2021) and gender bias (Cho et al., 2019). Mitigation strategies such as those from Vanmassenhove et al. (2018) may be necessary.

Acknowledgements

We would like to thank Robin Schmidt, Matthias Sperber, and Stephan Peitz for their feedback and support in reviewing this work.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. 2022. Data scaling laws in NMT: The effect of noise and architecture. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1466–1482. PMLR.

Angie Boggust, Brandon Carter, and Arvind Satyanarayan. 2022. Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples. In *27th International Conference on Intelligent User Interfaces*, pages 746–766, Helsinki Finland. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Won Ik Cho, Ji Won Kim, Seok Min Kim, and Nam Soo Kim. 2019. On measuring gender bias in translation of gender-neutral pronouns. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 173–181, Florence, Italy. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online. Association for Computational Linguistics.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Tao Ge, Si-Qing Chen, and Furu Wei. 2022. EdgeFormer: A parameter-efficient transformer for on-device seq2seq generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10786–10798, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. 2022. Scaling laws for neural machine translation. In *International Conference on Learning Representations*.

Mitchell A Gordon, Kevin Duh, and Jared Kaplan. 2021. Data and parameter scaling laws for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5915–5922, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. 2005. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? comparing two computational measures of semantic change.

In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2116–2121, Austin, Texas. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2021. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *9th International Conference on Learning Representations, ICLR*, virtual. OpenReview.net.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

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, Brussels, Belgium. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Qian Lou, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2022. Dictformer: Tiny transformer with shared dictionary. In *International Conference on Learning Representations*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Telmo Pires, Robin Schmidt, Yi-Hsiu Liao, and Stephan Peitz. 2023. Learning language-specific layers for multilingual machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14767–14783, Toronto, Canada. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robin Schmidt, Telmo Pires, Stephan Peitz, and Jonas Lööf. 2022. Non-autoregressive neural machine translation: A call for clarity. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2785–2799, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Sho Takase and Shun Kiyono. 2023. Lessons on parameter sharing across layers in transformers. In *Proceedings of The Fourth Workshop on Simple and*

Efficient Natural Language Processing (SustaiNLP), pages 78–90, Toronto, Canada (Hybrid). Association for Computational Linguistics.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting gender right in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3003–3008, Brussels, Belgium. Association for Computational Linguistics.

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*, volume 30. Curran Associates, Inc.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

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.

A Custom Sharing of Multiple FFNs

There is a combinatorial number of ways of sharing $M < N$ FFNs within a module of N layers. Since this is prohibitive, we investigate the following strategies from Takase and Kiyono (2023):

- `Sequence`: assign one FFN for every M/N consecutive layers, forming a block pattern.

  $\text{FFN}_i(\cdot) \overset{\text{tied}}{=} \text{FFN}_{seq_m}(\cdot), \forall i : 1 \leq i \leq N, m = \lfloor (i-1)/(N/M) \rfloor$

- `Cycle`: stack M FFNs in an identical order, forming a repetitive checkerboard pattern.

  $\text{FFN}_i(\cdot) \overset{\text{tied}}{=} \text{FFN}_{cyc_m}(\cdot), \forall i : 1 \leq i \leq N, m = (i-1) \textbf{ modulo } M$

- `Cycle (Rev)`: stack M FFNs in a reverse order, forming a repetitive palindrome series.

  $\text{FFN}_i(\cdot) \overset{\text{tied}}{=} \text{FFN}_{cycrev_m}(\cdot), \forall i : 1 \leq i \leq N, m = N/M - i$

Note that we assume that N is an even number and divisible by N. `Cycle (Rev)` is only valid for $M = N/2$. The `EdgeFormer` (Ge et al., 2022) adopts `Cycle` with $M = 2$ for the encoder FFNs.

Table 7 shows the results of these strategies applied on the encoder. As references, we copy the results of the `Transformer Big` and `ShareEnc` from Table 2. Not only is the accuracy of `ShareEnc` similar to Takase and Kiyono (2023)'s strategies, but it also uses fewer parameters and is easier to extend.

Architecture	BLEU	$\lvert\theta\rvert$	(%)
`Transformer Big`	35.6	228M	(100)
+ `SharedEnc (M=1)`	35.4	186M	(82)
+ `Sequence M=2`	35.2	194M	(85)
+ `Sequence M=3`	35.3	202M	(88)
+ `Cycle M=2`	35.2	194M	(85)
+ `Cycle M=3`	35.5	202M	(88)
+ `Cycle Rev M=2`	35.2	194M	(85)
+ `Cycle Rev M=3`	35.5	202M	(88)

Table 7: Accuracy of different FFN sharing strategies on WMT22 EN → DE.

B Sharing or Dropping Attention

We report the results of sharing attention modules (either self, cross or both) across layers in Table 8. In contrast with the FFN, attention seems to play a more crucial role in the model's performance, as sharing the different attention mechanisms in both encoder and decoder causes a large accuracy drop across all settings, with the exception of sharing the decoder's cross attention and the encoder's self attention.

Encoder Self-Att	Decoder Self-Att	Decoder Cross-Att	BLEU	$\lvert\theta\rvert$	(%)
`Transformer Big`			35.6	228M	(100)
Shared	Shared	Shared	27.5	165M	(72)
Shared	Shared	Indiv.	27.6	186M	(82)
Shared	Indiv.	Indiv.	35.5	207M	(91)
Indiv.	Shared	Indiv.	26.5	207M	(91)
Indiv.	Shared	Shared	25.7	186M	(82)
Indiv.	Indiv.	Shared	35.5	207M	(91)

Table 8: BLEU scores on WMT 22 EN →DE when sharing the attention of both encoder and decoder (self and cross). Nomenclature follows Section 3 but with Self Attn an Cross Attn as the encoder/decoder's self attention and cross-attention (decoder), respectively.

C Details on Internal Representations Analysis

C.1 Raw Similarity Scores for Benchmarking

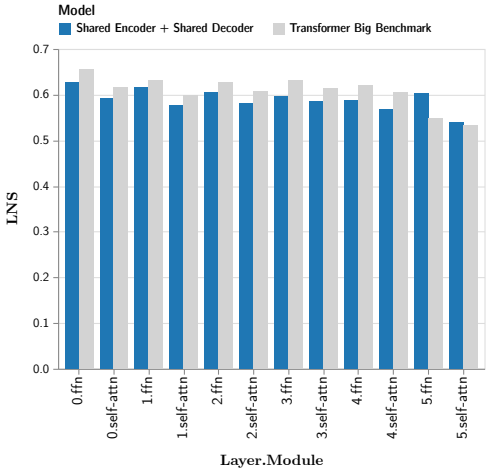
We establish a *benchmark* score for the expected similarity of our two metrics by comparing the baseline Transformer Big with identical models trained from different random seeds. Table 9 presents the raw similarity scores from which we compute the normalized scores presented in Table 5. As shown, the similarity between

Architecture	Encoder		Decoder	
	CKA	LNS	CKA	LNS
TransformerBig Seed 2	.96	.61	.94	.62
TransformerBig Seed 3	.96	.62	.95	.62
SharedEnc	.94	.58	.95	.62
SharedDec	.97	.62	.93	.59
SharedEncSharedDec	.95	.59	.94	.59
SharedEncDec	.94	.57	.93	.58
NoEnc	.87	.43	.95	.60
NoDec	.96	.60	.90	.54
ShareEncNoDec	.94	.59	.92	.55
ShareEncNoDec$^{d'_{\text{ff}}=41952}$	.94	.51	.89	.51

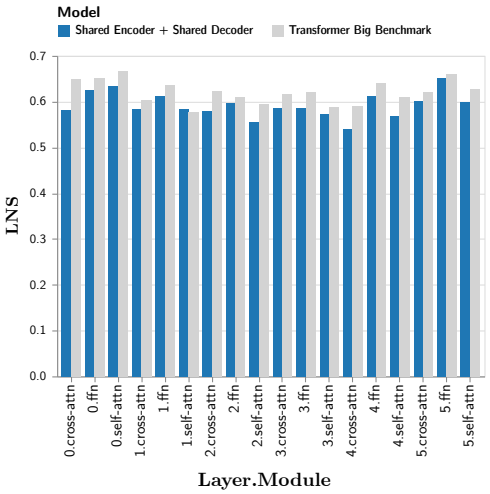
Table 9: Raw similarity of the representations of corresponding layer-modules of different architectures vs. the Transformer Big for WMT22 EN →DE. For *NoDec* configurations we compare the final output of the transformer layer as a whole as they have different sub-modules. The columns for shared and for dropped FFNs are highlighted in gray and blue respectively.

C.2 Layer-wise Analysis

In Table 5, we report the aggregated similarity scores across all layers of Transformer encoder and decoder. Here, we report a more fine-grained layer-wise similarity score mostly to showcase the reliability of the aggregated scores. In Figure 2, we plot layerwise LNS to study how similar the semantic information captured at each layer is to that of the baseline model at *every layer*. When LNS scores are high, the network is producing similar local neighborhoods for each sentence in our evaluation set. In particular, we are interested in comparing the benchmark LNS scores and those of SharedEncSharedDec at each layer. As shown, the layer-wise LNS scores of SharedEncSharedDec track the baseline scores at almost every layer, confirming the reliability of the aggregated score. We observe similar pattern for all the models that we evaluate in this paper.



(a) Encoder



(b) decoder

Figure 2: Layerwise LNS between SharedEncSharedDec and Transformer Big (blue bars). LNS between two versions of Transformer Big trained from different random initializations are shown by the grey bars to ground the comparison. FFN sharing does not dramatically change activations produced at each layer.

D Effect of batch size on decoding speed

In Section 4.3, we compared the decoding speeds of the One Wide FFN model and the Transformer Big, with a batch size of 1. In Table 10, we delve into how the decoding speed evolves as the batch size increases. As shown, the One Wide FFN model is faster for smaller batch sizes, but its advantage diminishes as the batch size increases, being slower than the Transformer Big for large batch sizes. We suspect this slowdown is due to the fact that the

Batch	Transformer Big	*One Wide FFN*	Speed-up (%)	*# batches*
1	$110.8^{\pm 1.2}$	$137.5^{\pm 1.1}$	24	2,047
2	$221.7^{\pm 14.3}$	$260.9^{\pm 6.5}$	18	1,024
4	$397.4^{\pm 8.0}$	$448.9^{\pm 2.0}$	13	512
8	$718.3^{\pm 8.0}$	$748.7^{\pm 10.6}$	4	256
16	$1,220.7^{\pm 56.2}$	$1,226.9^{\pm 17.2}$	1	128
32	$1,958.5^{\pm 112.4}$	$1,837.6^{\pm 15.3}$	−6	64
64	$1,319.1^{\pm 36.7}$	$1,259.0^{\pm 70.0}$	−5	32
128	$1,925.1^{\pm 64.8}$	$1,705.0^{\pm 62.3}$	−11	16
256	$2,312.1^{\pm 67.4}$	$1,976.5^{\pm 123.2}$	−15	8
512	$2,512.0^{\pm 50.1}$	$1,957.9^{\pm 32.6}$	−22	4

Table 10: Effect of batch size on decoding speed (in tokens/s) for the Transformer Big and *One Wide FFN* ($d_{\text{ff}'} = 49,152$). Δ is the percentage change in inference speed, and # batches is the number of batches used to evaluate. For large batch sizes, there are fewer batches (since the dataset size is fixed), which leads to higher variance in the measurements.

large FFN size requires higher peak memory, making the larger sizes non-optimal for this model.