SUBFORMER: A PARAMETER REDUCED TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

The advent of the Transformer can arguably be described as a driving force behind many of the recent advances in natural language processing. However, despite their sizeable performance improvements, as recently shown, the model is severely over-parameterized, being parameter inefficient and computationally expensive to train. Inspired by the success of parameter-sharing in pre-trained deep contextualized word representation encoders, we explore parameter-sharing methods in Transformers, with a specific focus on encoder-decoder models for sequence-to-sequence tasks such as Machine Translation. We perform an analysis of different parameter sharing/reduction methods and develop the Subformer, a parameter efficient Transformer-based model which combines the newly proposed Sandwich-style parameter sharing technique and self-attentive embedding factorization (SAFE). Experiments on machine translation, abstractive summarization, and language modeling show that the Subformer can outperform the Transformer even when using significantly fewer parameters. On the WMT'14 English-German test set, we show we can perform equally well, and even sometimes outperform (+0.1 BLEU score) the Transformer-base model while using 40% fewer parameters. We also perform equally well as Transformer-big with 40% fewer parameters, achieve performance within 0.1 BLEU with 70% fewer parameters, and outperform the model by 0.7 BLEU with 12M fewer parameters. We also outperform the standard Transformer-XL model, achieving a significant 3.6 lower perplexity with 37% fewer parameters.[1]

1 INTRODUCTION

Many of the advances in natural language processing over the past few years can be attributed to the self-attention-based Transformer (Vaswani et al., 2017) model. Improving performance on a variety of tasks, the Transformer has led to better deeply contextualized representations (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020) which result in substantial performance improvements on a variety of downstream tasks, including better sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2014).

Despite their success, one main drawback of training these models is their computational cost, being a greatly limiting factor for many, with training times and memory usage ballooning as model sizes increase to attain better performance. With this in mind, there has been recent interest in making the Transformer more parameter-efficient (So et al., 2019; Wu et al., 2020; Lan et al., 2020; Mehta et al., 2020a), with the aim of reaping its performance benefits while making the model more computationally efficient and able to scale better.

Inspired by recent work in model parameter reduction (Lan et al., 2020) while still attaining similar (or better) performance in the context of deeply contextualized word representations, we look to explore whether these ideas and techniques can be applied to sequence-to-sequence models in a simple manner.

Recent work on reducing parameters in Transformer models (Wu et al., 2020; So et al., 2019; Mehta et al., 2020a) has to a great extent focused on automating their design with neural architecture search

[1] We release the code here: `https://u.pcloud.link/publink/show?code=kZSlqJXZ8gzUB1PdOfHmzsda8Bo1HQN8O46k`

approaches that aim at finding more efficient Transformer variations using gradient descent. As such, these techniques are expensive, requiring a significant amount of GPU hours to find good designs. Instead, we look to address these issues by directly designing the Subformer, an intuitively-designed parameter efficient Transformer-based model. The Subformer can be trained with lower memory resources due to its vast parameter reduction. Training speed can also be significantly hampered in distributed training, as the communication overhead is directly proportional to the number of parameters in the model. Moreover, the Subformer can do all of this while still maintaining (or gaining) performance when compared to challenging baselines.

The Subformer incorporates two novel techniques: (1) SAFE (Self-Attentive Factorized Embedding Parameterization), in which we disentangle the embedding dimension from the model dimension, and use a small Transformer-based layer to project the smaller embedding dimension to the model dimension, allowing us to grow the hidden size of the model without significantly impacting the embedding parameter count, and (2) Sandwich-style Cross Layer Parameter Sharing, in which we develop a simple and intuitive technique for cross-layer parameter sharing to be effective in Transformer models (as we demonstrate that naively sharing parameters harms performance significantly), which allows us to exploit the benefits of parameter sharing, i.e. increase of depth without impacting parameter count.

To test our proposals we evaluate the Subformer on three challenging generative tasks: machine translation, abstractive summarization and language modeling. Our experiments show that by incorporating our design choices and techniques, the Subformer is able to achieve similar or better performance compared with a base/big Transformer with a $\sim$40% parameter reduction and minimal modification to the original architecture —further reinforcing the aforementioned claims of the Transformer's over-parameterization (Fan et al., 2020; Mehta et al., 2020a; Lan et al., 2020). Specifically, on WMT'14 EN-DE we achieve a BLEU score of 29.3, compared to Transformer-big's 28.6 with 13M fewer parameters. We also outperform the standard Transformer-XL model, achieving a significant 3.6 perplexity lower, with 37% fewer parameters.

2 Related Work

Improving Transformers Given the effectiveness of the Transformer, improving the architecture has been of much interest to the NLP community. Within this domain, one branch of research concerns the reduction of the quadratic complexity (w.r.t. sequence length) of the Transformer's core self-attention mechanism (Wu et al., 2019; Kitaev et al., 2020), pushing it down to linear or log-linear complexity. The second branch of research regards improving the expressiveness of Transformer models, by using more layers (Dou et al., 2018), or by improving the architecture (Wu et al., 2019; So et al., 2019). A third branch of research regards improving the parameter efficiency of Transformers. Approaches towards this goal include neural architecture search approaches (So et al., 2019; Wu et al., 2020), where new Transformer-based architectures are learned using gradient descent, more manually crafted approaches (Dehghani et al., 2018; Mehta et al., 2020a), as well as weight-sharing approaches (Lan et al., 2020; Wu et al., 2019). The work most similar to ours is ALBERT (Lan et al., 2020) in which complete weight sharing is used to pre-train deep contextualized word representations (Peters et al., 2018; Devlin et al., 2019). Different from this work, we focus on common NLP generative/sequence-to-sequence tasks versus large-scale pre-training and develop an approach to increase model capacity while reducing parameter footprint tailored to this setting.

Compressing Transformers We also find prior work on pruning and/or quantizing Transformer models to reduce their size, either with a focus on sequence-to-sequence settings like machine translation (Prato et al., 2019), on encoder-based methods like BERT (Zafrir et al., 2019; Ganesh et al., 2020) or with a more generic scope in mind (Cheong & Daniel, 2019; Lee et al., 2018b). Our approach is orthogonal to these since we directly aim at reducing the number of parameters of Transformer models by proposing architecture modifications and weight sharing techniques - allowing training from scratch.

Reducing Embedding Dimensionality in Sequence Models As embeddings can substantially increase the parameter count as the vocabulary size increases, especially in sequence modeling scenarios, embedding reduction techniques have been proposed, including using a linear projection to project to a lower dimension (Baevski & Auli, 2019; Dai et al., 2019) or using combinations of block sparse transformations (Mehta et al., 2020b;a). We propose a self-attention based projection layer, which we empirically show to outperform the aforementioned linear projection method with a similar parameter count.

3 THE SUBFORMER

In this section we describe the SUBFORMER. We first briefly review the original Transformer architecture, and then explain in depth the components and reasoning behind the design choices of our model.

Notation We start by defining the notation to be used throughout the paper. We refer to the model dimension as d_m, the vocabulary size as V, and the number of layers as L. Note that, unlike standard Transformer models, in which the embedding dimension is kept the same as d_m, we disentangle to embedding dimension to reduce parameter count (Sec. 3.2). For this reason we denote the embedding dimension to be d_e. Unless specified otherwise, following standard practice (Vaswani et al., 2017), the feed-forward projection dimension $\vec{d_s} = 4d_s$, $\vec{d_m} = 4d_m$, for the Sandwiched layer(s) (Sec. 3.3) and the model layer(s), respectively. By default, we set $L = 6$ and the dimension of a single attention head to be 64.

3.1 THE TRANSFORMER

In an encoder-decoder setting, the Transformer architecture is composed of an encoder and decoder component, both of which are comprised of stacks of identical Transformer layers. Each one of these layers is composed of two sub-layers: a multi-headed self-attention sub-layer and a feed-forward sub-layer, which are defined by the following functions.[2]

$$\text{MultiHeadAttention}(\boldsymbol{x}) = softmax(\boldsymbol{x}^\top \boldsymbol{K}(\boldsymbol{Q}\boldsymbol{x}))\boldsymbol{V}\boldsymbol{x} \quad (1)$$

$$\text{FeedForward}(\boldsymbol{x}) = \boldsymbol{W}_2(ReLU(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1)) + \boldsymbol{b}_2 \quad (2)$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d_m \times d_m}$ are trainable matrices used to compute the queries, keys and values for the self-attention operation, $ReLU$ denotes the ReLU activation (Nair & Hinton, 2010) and $\boldsymbol{W}_1 \in \mathbb{R}^{\vec{d_m} \times d_m}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_m \times \vec{d_m}}$ are trainable weight matrices. This is followed by a residual connection (He et al., 2016) and layer normalization (Ba et al., 2016).

3.2 SAFE: SELF-ATTENTIVE FACTORIZED EMBEDDINGS

We propose to reduce the number of parameters in our embedding layers, which can take up to 25+% of total parameter count in the case of Transformer base, using a small Transformer-based layer. Specifically, we look to reduce the embedding size by disentangling the model dimension from the embedding dimension, reducing the embedding dimension d_e, and then projecting this to the model dimension d_m using a small multi-head attention sub-layer followed by a feed-forward module.

Given a vocabulary size of V, the usage of a standard embedding layer would result in $V \times d_m$ parameters. However, considering that the power of Transformers lies in their ability to learn contextual representations with deep models, using a smaller value of d_e for non-contextual embeddings and then projecting to d_m, is intuitively an effective method for parameter reduction (Lan et al., 2020). When using our self-attentive projection module, our parameter count would result in $V \times d_e + 5d_e^2 + d_e \times d_m$ parameters[3], which results in a significant parameter reduction for values of $d_e \ll d_m$. Current models (Baevski & Auli, 2019; Dai et al., 2019; Lan et al., 2020) often use a single linear projection, i.e. $V \times d_e + d_e \times d_m$. Building on this method, we empirically show that contextualizing this projection with a small Transformer-based layer results in stronger performance with a minimal addition of parameters — especially in the encoder-decoder case, where the decoder input embedding layer and output projection are often tied (Table 1).

[2]Note that we omit bias terms from Equation 1 for clarity.

[3]Note that $V \times d_e$ represents the embedding layer, $5d_e^2$ represents the query, key, and value projections and 2 output feed-forward layers, and $d_e \times d_m$ represents the linear projection from the embedding dimension to the model dimension.

MODEL	Param.	BLEU
$d_e = 128$, Linear	48M	26.0
$d_e = 256$, Linear	53M	27.1
$d_e = 256$, 2-Layer Linear	54M	27.2
$d_e = 128$, SAFE	48M	26.6
$d_e = 256$, SAFE	54M	27.6
Vaswani et al. (2017)	65M	27.3
TRANSFORMER-BASE (reimpl.)	61M	27.7

Table 1: Experiments on the impact on SAFE vs a regular linear projection using TRANSFORMER-BASE on the WMT'14 EN-DE machine translation benchmark

3.3 SANDWICH-STYLE PARAMETER SHARING

Weight sharing techniques, despite being surprisingly effective, have been relatively unexplored so far with regard to Transformer Encoder-Decoder Models. However, this has been shown to be a powerful technique for leveraging models with large capacity and less memory usage/computation (Dehghani et al., 2018; Lan et al., 2020; Wu et al., 2019).

Given that the output of each layer depends directly on its two sub-layers —MultiHeadAttention and FeedForward, when discussing alternatives for parameter sharing across transformer layers there are several options. As we aim to leverage the aforementioned properties of weight sharing, we performed preliminary experiments, investigating the capabilities of weight sharing in the following four settings.

1. Naively sharing all encoder and all decoder layers —that is including both of their sub-layers, following Lan et al. (2020). This is denoted as *All-Shared*.
2. Naively sharing all encoder and all decoder layers, but allowing each layer $l \in [2, \cdot, L]$ to have an independent feed-forward sub-layer. We denote this as *All-Shared (Independent FFN)*.
3. Sharing weights across layers $l \in [1, \dots, L-1]$ such that layer L remains independent —denoted as *All-Shared (except last)*.
4. Sharing every two layers, i.e. $[1, 2], [3, 4], [5, 6]$ in the case of a 6-layer transformer —denoted as *Every 2 layers shared*.
5. Finally, we only share the middle or central layers (i.e. $2 \leq l \leq L-1$), leaving layers 1 and L to have independent sets of parameters —denoted as SANDWICH.

MODEL	Enc. layers	Dec. Layers	Param.	BLEU
All-Shared	6	6	24M	14.3
All-Shared (Independent FFN)	6	6	27M	22.4
All-Shared (except last)	6	6	31M	23.2
Every 2 layers shared	6	6	38M	27.2
SANDWICH	6	6	38M	27.3
SANDWICH	8	8	38M	**27.7**
Vaswani et al. (2017)	6	6	65M	27.3
TRANSFORMER-BASE (Our reimpl.)	6	6	61M	**27.7**

Table 2: Experiments performed on WMT'14 EN-DE using different parameter sharing techniques. For each setting, we report tokenized BLEU scores on the test set.

Table 2 summarizes the results of our exploratory study. As can be seen, naive parameter sharing/tying approaches do not offer any advantages, hurting performance significantly (~50 %) when compared to the regular Transformer. However, our results also show that when combined properly, using Sandwich-style parameter sharing, we can attain a good balance of parameter reduction and performance. In this context, we surmise that the success of Sandwich-style parameter sharing on this sequence-to-sequence task is a consequence of the following property: When compared to tasks such as pre-training deep contextualized word representations, tasks such as machine translation require informative token-level representations for each input token to be accurately translated. Sandwich-style parameter sharing allows the input and output layer (arguably the most important layers) to be trained independently allowing them to learn different operations than the shared sandwich layers, reasonably satisfying the above conditions.

3.4 MODEL ARCHITECTURE: PUTTING IT ALL TOGETHER

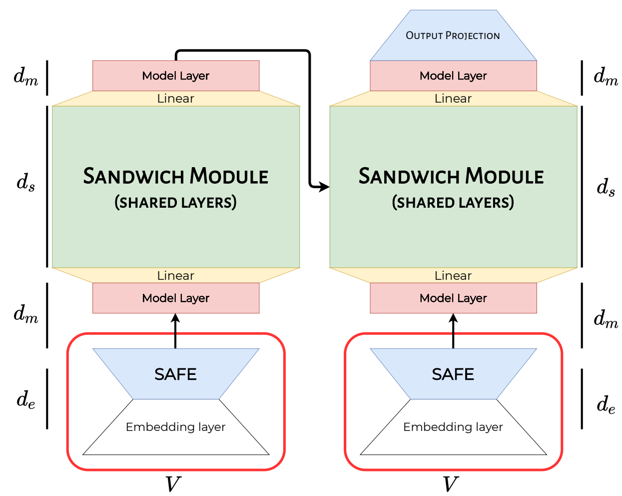


Figure 1: **The SUBFORMER**. In the graphic above there are four main components: (1) the blue portions, denoted by d_e are the ***SAFE (self-attentive factorized embedding)*** and output projection layers. (2) The ***model layers*** which are placed at the top and bottom of the model (colored red, denoted by d_m). (3) The ***Sandwich Module***, in which we use a wider shared layer to compose the central part of our encoder/decoder. (4) ***The projection layers***, which allow for the interaction between the model layers and Sandwich Module despite their different dimensions (colored yellow).

With the aforementioned techniques/components, i.e. SAFE (Section 3.2) and Sandwich-style Parameter Sharing (Section 3.3), we will now explain the SUBFORMER architecture (see Fig. 1).

As we closely follow the Transformer architecture (Vaswani et al., 2017), the SUBFORMER is composed of four main components, for both the encoder and decoder: the embedding layer, the model layers, the sandwich module and the projection layers. Figure 1 offers an overview of how these components are put together. As we want to exploit the parameter reduction effect of sandwich style parameter sharing while increasing model capacity, we increase the width of the sandwich layer. We disentangle the sandwiched layer dimension from that of the model layer, allowing the sandwich layer width to be larger than the rest of the model. For this reason, we denote the dimension of the sandwiched layer to be d_s and its corresponding feed-forward dimension to be $\vec{d_s}$.

Embedding layer When using SAFE, the embedding layer is composed of a regular token $\rightarrow$ vector embedding matrix $\boldsymbol{E} \in \mathbb{R}^{V \times d_e}$. This is followed by projecting the embeddings (summed with the positional encodings (Vaswani et al., 2017), denoted by PE) to the model dimension d_m using SAFE.

$$e = \text{SAFE}\big(\boldsymbol{E}(x) + PE(x)\big) \tag{3}$$

Model layers Once we have our SAFE embeddings, we now feed them through the first model layer - the base of the sandwich. The output of this first layer is then projected to the sandwich dimension d_s, by way of a linear projection parameterized by weight matrix $\boldsymbol{W}_1^p \in \mathbb{R}^{d_m \times d_s}$ and bias vector $\boldsymbol{b}_1^p \in \mathbb{R}^{d_s}$. Once fed through the shared sandwich layers, we then project the output back to the model dimension using a linear projection parameterized by matrix $\boldsymbol{W}_2^p \in \mathbb{R}^{d_s \times d_m}$ and bias vector $\boldsymbol{b}_2^p \in \mathbb{R}^{d_m}$. The output of the projection is then fed through the final model layer to produce the output vectors.

When using SAFE embeddings, as we tie the decoder's output projection layer (returning a distribution over the vocabulary) with the decoder's input embedding matrix, we project the decoder's last hidden state (with dimension d_m) to d_e. We do this using a two layer multi-layer perceptron: $\boldsymbol{W}_o^2(\boldsymbol{W}_o^1 x + \boldsymbol{b}_1^o) + \boldsymbol{b}_1^o$, where $\boldsymbol{W}_o^1 \in \mathbb{R}^{2d_e \times d_m}, \mathbf{b}_o^1 \in \mathbb{R}^{2d_e}$ and $\boldsymbol{W}_o^2 \in \mathbb{R}^{d_e \times 2d_e}, \boldsymbol{b}_o^2 \in \mathbb{R}^{d_e}$. Also, when we perform encoder attention in the decoder's Sandwich Module, we simply linearly project the query from the decoder from d_s to d_m and then project it back to d_s once the attention operation is complete.

4 Experiments

4.1 Evaluation Benchmarks

We apply our method to a variety of sequence modeling tasks: Neural machine translation, Summarization, and language modeling. Our models are implemented in PyTorch (Paszke et al., 2019) using our own modification of `fairseq` (Ott et al., 2019). Additional implementation and training details with hyper-parameter settings are in the Appendix.

Machine Translation We evaluate our model on two standard machine translation benchmarks: (1) WMT'14 English-German (EN-DE) (4.5M train/3K valid/3K test sent. pairs), and (2) WMT'16 English-Romanian (EN-RO) (610K train/3K valid/3K test sent. pairs). We make use of the same preprocessed data used by Ghazvininejad et al. (2019) for WMT'14 EN-DE, with a 32K BPE (Sennrich et al., 2016) vocabulary, as well as the same data as Lee et al. (2018a) for WMT'16 EN-RO, with a 35K BPE vocabulary. Following previous work, we evaluate all models using tokenized BLEU (Papineni et al., 2002) and perform de-hypenation on WMT'14 EN-DE (Vaswani et al., 2017).

For this task, we follow the training setup of Ghazvininejad et al. (2019): we use the same weight initialization scheme as BERT (Devlin et al., 2019), sampling weights from $\mathcal{N}(0, 0.02)$, initializing biases to zero and setting layer normalization parameters β and γ to be 0 and 1, respectively. For regularization we use the best of $[0.1, 0.2, 0.3]$ dropout, weight decay of 0.01, while using label-smoothed cross entropy loss with $\epsilon = 0.1$. We train using an effective batch size of 128K tokens. The models are trained using Adam (Kingma & Ba, 2014), with hyper-parameters $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-6}$. We warm up the learning rate to a peak of 5×10^{-4} within 10K iterations and then decay the learning rate with the inverse square root schedule. When creating the final model, we use the checkpoint with the lowest loss on the development set, and generate using a beam size of 5 (Vaswani et al., 2017), tuning the length penalty of $\alpha \in [0.0, 0.2, \ldots, 2.0]$ in the validation set. We perform early stopping, training for a maximum of 250K iterations.

Abstractive Summarization We test the model's ability to process long documents on the CNN-DailyMail summarization benchmark (Hermann et al., 2015; Nallapati et al., 2016) comprising over 280K news articles paired with multi-sentence summaries. Articles are truncated to 400 tokens (See et al., 2017) and we use a BPE vocabulary of 32K types (Edunov et al., 2019). We follow the training schedule of Edunov et al. (2019). During inference, we tune generation length in the range of {40, 50, 60} and use tri-gram blocking, following standard practice. Evaluation is performed using the ROUGE metric (Lin, 2004), which is the de-facto for the task.

Language Modeling We evaluate on the large-scale Wikitext-103 dataset (Merity et al., 2016), which contains 103M tokens and has a vocabulary of nearly 270K types. Models are evaluated in terms of perplexity on the test portion.

4.2 Baselines

To test how well we are able to increase parameter efficiency while maintaining performance, we compare with current state-of-the-art-methods: namely, the Transformer-base and Transformer-big models from Vaswani et al. (2017), for all tasks. For the machine translation tasks we compare with DeLight (Mehta et al., 2020a) which is contemporaneous work to ours, and with the Evolved Transformer (So et al., 2019), as well as RNMT+ (Chen et al., 2018) and Dou et al.

BASE MODELS	WMT'14 EN-DE		WMT'16 EN-RO	
	Param.	BLEU	Params.	BLEU
EVOLVED TRANSFORMER (So et al., 2019)	48M	27.7	—	—
EVOLVED TRANSFORMER (So et al., 2019)	64M	28.2	—	—
DELIGHT (Mehta et al., 2020a)	37M	27.6	22M	34.3
DELIGHT (Mehta et al., 2020a)	54M	28.0	52M	**34.7**
TRANSFORMER (Vaswani et al., 2017)	65M	27.3	62M	34.2[†]
TRANSFORMER (Our reimpl.)	61M	27.7	62M	34.1
Only SANDWICH	38M	27.3	—	—
Only SAFE, $d_e = 256$	54M	27.6	—	—
SUBFORMER-SMALL	38M	27.7	20M	34.1
SUBFORMER-BASE	52M	**28.1**	48M	**34.7**
SUBFORMER-MID	63M	**28.5**	—	—

Table 3: Results for machine translation on WMT'14 EN-DE and WMT'16 EN-RO task, for our base models. Note that the † superscript indicates results from Kasai et al. (2020).

BIG MODELS	Param.	BLEU
TRANSFORMER-BIG (Vaswani et al., 2017)	213M	28.4
RNMT+ (Chen et al., 2018)	379M	28.5
TRANSFORMER-BIG (Our reimpl.)	210M	28.6
EVOLVED TRANSFORMER (So et al., 2019)	222M	29.0
Dou et al. (2018)	356M	29.2
SANDWICH-BIG	122M	28.6
SUBFORMER-XLARGE	197M	**29.3**

Table 4: Results on the WMT'14 EN-DE for our large models

(2018) who propose using deep representations for NMT. For language modeling, we compare to the base Transformer-XL (Dai et al., 2019) and Deep Equilibrium Model (Bai et al., 2019), which also employs parameter sharing. Lastly, for our summarization task, we compare with specialized architectures such as Pointer-Generator Networks (See et al., 2017), and Convolutional Seq2Seq-based models (Fan et al., 2018), as well as the Transformer model from Edunov et al. (2019).

5 RESULTS AND DISCUSSION

5.1 MACHINE TRANSLATION

We use the following settings for our models: (1) SUBFORMER-SMALL has $d_m = 512$, $d_s = 768$, $d_e = 256$ and $L = 8$, (2) SUBFORMER-BASE has $d_m = 512$, $d_s = 1024$, $\vec{d_s} = 3072$, $d_e = 320$, (3) SUBFORMER-MID has $d_m = 768$, $d_s = 768$, $d_e = 350$ and (4) SUBFORMER-XLARGE has $d_m = 1024$, $d_s = 2048$ and $d_e = 512$. For WMT'16 EN-RO, our small model has $d_m = 320$, $d_s = 512$ and $d_e = 192$ and our base model has $d_m = 512$, $d_s = 640$, and $d_e = 384$.

Table 3 and Table 4 summarize our results on the WMT'14 EN-DE and WMT'16 EN-RO datasets, respectively. Firstly, we take note that our re-implementations of the Transformer baselines outperform Vaswani et al. (2017) (base model: 27.3 → 27.7, big model: 28.4 → 28.6.) We surmise that this is due to training for longer and with a larger batch size.

Table 3 shows that SUBFORMER-BASE outperforms all baselines, with similar or fewer parameters. Specifically, when compared to the baseline Transformer model, we reduce parameters by 40%, outperforming the model by 0.1 BLEU on WMT'14 EN-DE. SUBFORMER-BASE, with 52M parameters outperforms DELIGHT by 0.1 BLEU with less parameters, while also outperforming our Transformer-base baseline by 0.4 BLEU with 7M less parameters. SUBFORMER-MID achieves a BLEU score of 28.5, outperforming the base Transformer and Evolved Transformer (w/64M params)

MODEL	Param.	Context Length	PPL
QRNN (Merity et al., 2018)	151M	—	33.00
DELIGHT (Mehta et al., 2020a)	99M	**480**	24.14
TRANSFORMER-XL (Dai et al., 2019)	151M	640	24.03
Deep Equilibrium Model (DEQ) (Bai et al., 2019)	110M	—	23.20
Transformer (4 Layer)	96M	**480**	26.42
Transformer (8 Layer)	146M	**480**	22.32
SUBFORMER	96M	**480**	**20.39**

Table 5: Results on the WIKITEXT-103 (Merity et al., 2016) language modeling benchmark.

MODEL	Param.	ROUGE-1	ROUGE-2	ROUGE-L
PTR-GEN+COV (See et al., 2017)	—	39.5	17.3	36.4
CNN (Fan et al., 2018)	—	40.4	17.4	37.2
TRANSFORMER (3 Layer)	57M	40.0	17.5	36.7
TRANSFORMER (Edunov et al., 2019)	77M	40.1	17.6	36.8
SUBFORMER-BASE	57M	**40.9**	**18.3**	**37.7**

Table 6: Results on the CNN-Daily Mail Summarization task (Nallapati et al., 2016; See et al., 2017)

by 0.8 and 0.3 BLEU, respectively. The result is within 0.1 BLEU from the Transformer-big model (210M params), despite a 70% parameter reduction. We believe that these results demonstrate the empirical efficacy of the techniques leveraged in the SUBFORMER.

For our big/large set of models, which are evaluated on WMT'14 EN-DE, SANDWICH-BIG achieves the same performance as our Transformer-big re-implementation, but with 40% less parameters — shown in Table 4. We believe that this is an indication towards the larger capability of Sandwich-style parameter sharing as the encoder/decoder layers get wider, while also providing further empirical evidence with respect to the over-parameterized nature of the Transformer architecture. SUBFORMER-XLARGE, with 197M parameters achieves a significant 0.7 BLEU score gain over Transformer-big, despite using 13M less parameters. This again strongly suggests that the current Transformer architecture is over-parameterized, and that training every parameter independently is not necessary to achieve good performance on large translation benchmarks, further validating the effectiveness of our approach.

5.2 LANGUAGE MODELING

When training the SUBFORMER, we follow the schedule of Baevski & Auli (2019) and use adaptive input embeddings (Baevski & Auli, 2019) instead of regular or SAFE embeddings , following common practice. We optimize using Nesterov's accelerated gradient optimizer (Sutskever et al., 2013), warming up the learning rate to 1.0 for 16K iterations, and then annealing for 270K iterations using a cosine annealing schedule. We set $d_m = 768, \vec{d_m} = 4096$ and $d_s = 2048, \vec{d_s} = 6144$ and $L = 12$. We also train two Transformer baselines with the same setup - one with the same amount of parameters and another with a similar parameter count to Transformer-XL - to provide better context for comparison.

Seen in Table 5, the SUBFORMER outperforms all the baselines by a significant margin (between 1.9 and 12.6 perplexity), with a significant reduction in parameters. This demonstrates the surprising effectiveness of the SUBFORMER and the Sandwich-style parameter sharing technique.

5.3 ABSTRACTIVE SUMMARIZATION

For the CNN/Daily Mail summarization task we use a the same configuration as SUBFORMER-BASE, however we set $d_e = 256$. As can be seen in Table 6, the Subformer outperforms two Transformer baselines with both the same parameter count and its respective Transformer-base configuration, demonstrating the Subformer's performance on a variety of tasks and with longer sequences.

Training/Inference Speed As our SUBFORMER largely follows the Transformer model (Vaswani et al., 2017), we are able to reap the benefits of many operations being performed in parallel. However, in the case of DELIGHT, despite is parameter efficiency, we find that due to many operations being performed in a sequential manner with the addition of block sparsity, the model is surprisingly much slower at both training and inference time when compared to both our model and the base Transformer. For comparison, at inference, DELIGHT (38M params) processes 1536 tokens/s, while TRANSFORMER-BASE and SANDWICH-BASE processes 5135 tokens/s (tested on a single Tesla V100 with a batch size of 384 on the test set of WMT'14 EN-DE).

6 CONCLUSION

In this paper we have presented the Subformer, a parameter-efficient Transformer-based model with a larger capacity, despite its very small parameter footprint. The Subformer is composed of two novel techniques, self-attentive embedding factorization and Sandwich-style parameter sharing. Despite their simplicity, these techniques reduce the parameter count of Transformer models heavily, while also improving performance significantly, ultimately offering additional empirical evidence regarding the over-parameterization issue in Transformer models. We also believe the contribution of Sandwich-style parameter sharing to be important as naively sharing parameter sharing doesn't work as one may expect, in these settings. We hope that this work incites interest in using parameter sharing techniques for a wider range of Transformer models, and more parameter sharing techniques for more efficient, highly performant models with larger capacity and expressiveness.

REFERENCES

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

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxZX20qFQ`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models, 2019.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1008. URL `https://www.aclweb.org/anthology/P18-1008`.

Robin Cheong and Robel Daniel. Transformers. zip: Compressing transformers with pruning and quantization. Technical report, Stanford University, Stanford, California, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/p19-1285. URL `http://dx.doi.org/10.18653/v1/P19-1285`.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. Exploiting deep representations for neural machine translation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. doi: 10.18653/v1/d18-1457. URL http://dx.doi.org/10.18653/v1/D18-1457.

Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4052–4059, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1409. URL https://www.aclweb.org/anthology/N19-1409.

Angela Fan, David Grangier, and Michael Auli. Controllable abstractive summarization, 2018.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylO2yStDr.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *arXiv:2002.11985 [cs, stat]*, February 2020.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://www.aclweb.org/anthology/D19-1633.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching Machines to Read and Comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1693–1701. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer, 2020.

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

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL https://www.aclweb.org/anthology/D18-1149.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-shot Network Pruning Based on Connection Sensitivity. In *International Conference on Learning Representations*, September 2018b.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Very deep and light-weight transformer, 2020a.

Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. Define: Deep factorized input token embeddings for neural sequence modeling. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=rJeXS04FPH.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language modeling at multiple scales, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Gabriele Prato, Ella Charlaix, and M. Rezagholizadeh. Fully Quantized Transformer for Improved Translation. *ArXiv*, 2019.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017. doi: 10.18653/v1/p17-1099. URL http://dx.doi.org/10.18653/v1/P17-1099.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 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)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJeYe0NtvH`.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions, 2019.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention, 2020.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8Bit BERT. *arXiv:1910.06188 [cs]*, October 2019.

A APPENDIX

A.1 TRAINING DETAILS

Training done on 8 GPUs was done on a single DGX-1 Machine. Training on 16 GPUs was done using multiple compute nodes of a compute cluster. We train all base/small models on 8 NVIDIA Tesla V100 GPUs. For all big/large models, we train on 16 NVIDIA Tesla V100 GPUs.

A.1.1 MACHINE TRANSLATION

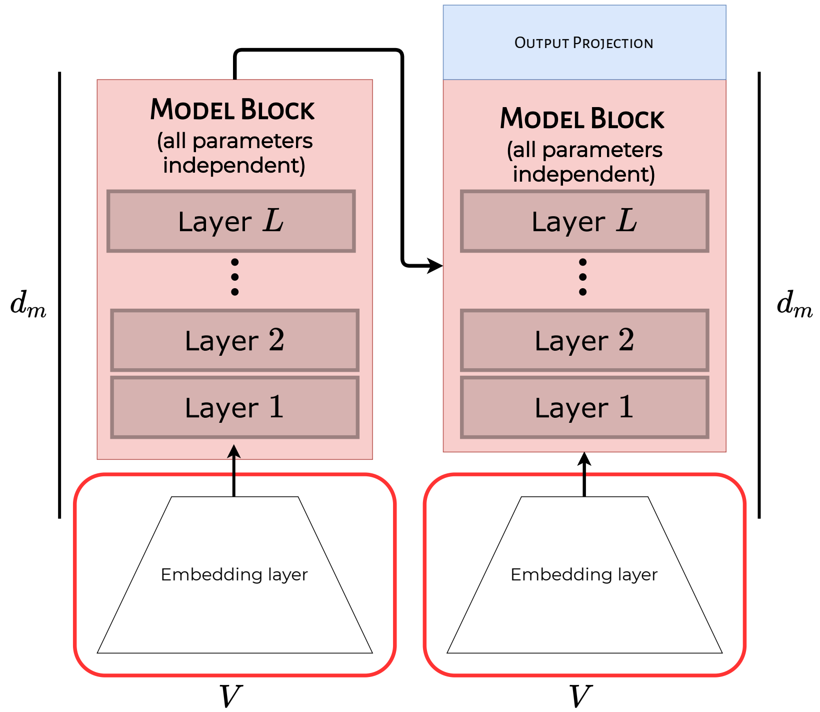
We train using 8192 tokens per GPU on an 8-GPU machine with an update frequency of 2, for small, base models. For large models, we train on 16 GPUs with 4096 tokens per GPU with an update frequency of 2.

A.1.2 ABSTRACTIVE SUMMARIZATION

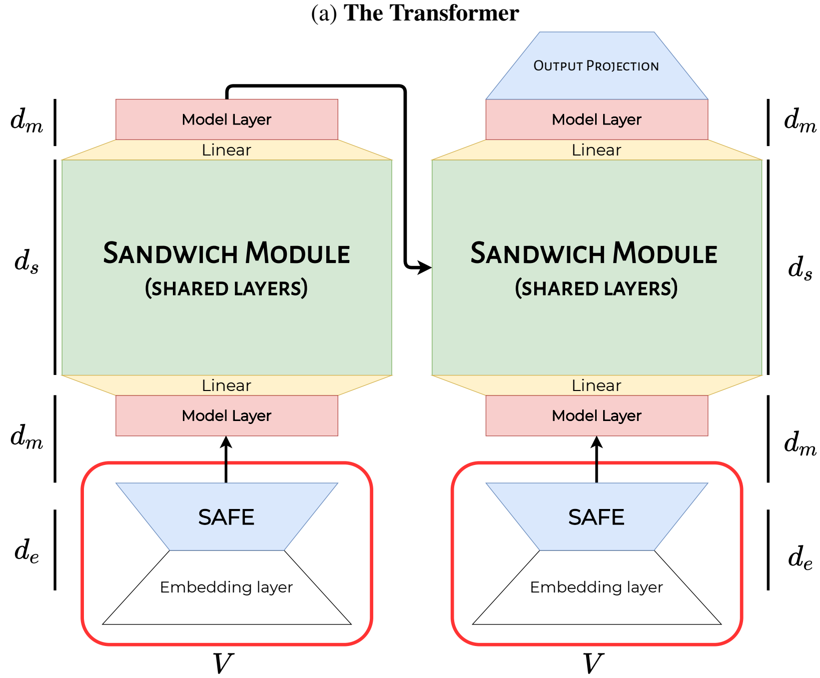
We follow Edunov et al. (2019) and use the offical `ROUGE-1.5.5.pl` script with parameters `-m -a -n 2`.

A.1.3 LANGUAGE MODELING

When training our langauge model, we use 8 GPUs with 1536 tokens per GPU and an update frequency of 3, following Welleck et al. (2020).



(a) **The Transformer**



(b) **The Subformer**

Figure 2: **Comparison between the Subformer and Transformer**.