Parameter Sharing Methods for Multilingual Self-Attentional Translation Models

Devendra Singh Sachan
Data Solutions Team
Petuum Inc.
Pittsburgh, USA
devendra.singh@petuum.com

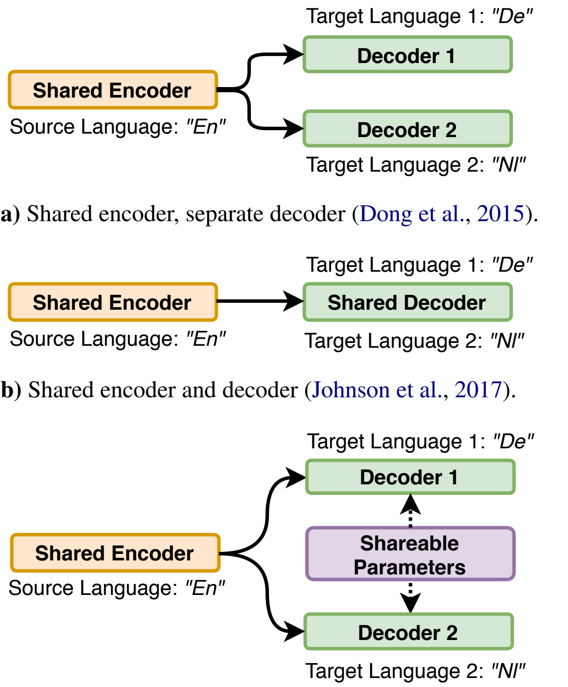
Graham Neubig
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA
gneubig@cs.cmu.edu

Abstract

In multilingual neural machine translation, it has been shown that sharing a single translation model between multiple languages can achieve competitive performance, sometimes even leading to performance gains over bilingually trained models. However, these improvements are not uniform; often multilingual parameter sharing results in a decrease in accuracy due to translation models not being able to accommodate different languages in their limited parameter space. In this work, we examine parameter sharing techniques that strike a happy medium between full sharing and individual training, specifically focusing on the self-attentional *Transformer* model. We find that the full parameter sharing approach leads to increases in BLEU scores mainly when the target languages are from a similar language family. However, even in the case where target languages are from different families where full parameter sharing leads to a noticeable drop in BLEU scores, our proposed methods for partial sharing of parameters can lead to substantial improvements in translation accuracy.[1]

1 Introduction

Neural machine translation (NMT; Sutskever et al. (2014); Cho et al. (2014)) is now the de-facto standard in MT research due to its relative simplicity of implementation, ability to perform end-to-end training, and high translation accuracy. Early approaches to NMT used recurrent neural networks (RNNs), usually LSTMs (Hochreiter and Schmidhuber, 1997), in their encoder and decoder layers, with the addition of an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) to focus more on specific encoded source words when deciding the next translation target output. Recently, the NMT research community has been transitioning from RNNs to an alternative method for encoding sentences using self-attention (Vaswani et al., 2017), represented by the so-called "*Transformer*" model, which both improves the speed of processing sentences on computational hardware such as GPUs due to its lack of recurrence, and achieves impressive results.



(a) Shared encoder, separate decoder (Dong et al., 2015).

(b) Shared encoder and decoder (Johnson et al., 2017).

(c) Proposed shared decoder with partial parameter sharing.

Figure 1: Examples of MTL frameworks for the translation of one source language (for example *"En"*) to two target languages (for example *"De"*, *"Nl"*). The principle remains the same with more than two target languages. Best viewed in color.

In parallel to this transition to self-attentional models, there has also been an active interest in the multilingual training of NMT systems (Firat et al., 2016; Johnson et al., 2017; Ha et al.,

[1] Data and code of this paper is available at: https://github.com/DevSinghSachan/multilingual_nmt

Proceedings of the Third Conference on Machine Translation (WMT), Volume 1: Research Papers, pages 261–271
Belgium, Brussels, October 31 - Novermber 1, 2018. ©2018 Association for Computational Linguistics
https://doi.org/10.18653/v1/W18-64027

2016). In contrast to the standard bilingual models, multilingual models follow the multi-task training paradigm (Caruana, 1997) where models are *jointly trained* on training data from several language pairs, with some degree of *parameter sharing*. The objective of this is two-fold: First, compared to individually training separate models for each language pair of interest, this maintains competitive translation accuracy while reducing the total number of models that need to be stored, a considerable advantage when deploying practical systems. Second, by utilizing data from multiple language pairs simultaneously, it becomes possible to improve the translation accuracy for each language pair.

In multilingual translation, *one-to-many* translation —translation from a common source language (for example English) to multiple target languages (for example German and Dutch) — is considered particularly difficult. Previous multi-task learning (MTL) models for this task broadly consist of two approaches as shown in Figure 1: (a) a model with a shared encoder and one decoder per target language (Dong et al. (2015), shown in Figure 1a). This approach has the advantage of being able to model each target separately but comes with the cost of slower training and increased memory requirements. (b) a single *unified* model consisting of a shared encoder and a shared decoder for all the language pairs (Johnson et al. (2017), shown in Figure 1b). This simple approach is trivially implementable using a standard bilingual translation model and has the advantage of having a constant number of trainable parameters regardless of the number of languages, but has the caveat that the decoder's ability to model multiple languages can be significantly reduced.

In this paper, we propose a third alternative: (c) a model with a shared encoder and multiple decoders such that some decoder parameters are shared (shown in Figure 1c). This hybrid approach combines the advantages from both the approaches mentioned above. It carefully moderates the types of parameters that are shared between the multiple languages to provide the flexibility necessary to decode two different languages, but still shares as many parameters as possible to take advantage of information sharing across multiple languages. Specifically, we focus on the aforementioned self-attentional Transformer models, with the set of shareable parameters consisting of the various attention weights, linear layer weights, or embedding weights contained therein. The *full sharing* and *no sharing* of decoder parameters used in previous work are special cases (refer to Section 2.2 for a detailed description).

To empirically examine the utility of this approach, we examine the case of translation from a common source language to multiple target languages, where the target languages can be either related or unrelated. Our work reveals that while full parameter sharing works reasonably well when using target languages from the same family, partial parameter sharing is essential to achieve the best accuracy when translating into multiple distant languages.

2 Method

In this section, we will first briefly describe the key elements of the Transformer model followed by our proposed approach of parameter sharing.

2.1 Transformer Architecture

As is common in sequence-to-sequence (*seq2seq*) models for NMT, the self-attentional Transformer model (Figure 2; Vaswani et al. (2017)) consists of an embedding layer, multiple encoder-decoder layers, and an output generation layer. Each encoder layer consists of two sublayers in sequence: self-attentional and feed-forward networks. Each decoder layer consists of three sublayers: masked self-attention, encoder-decoder attention, and feed-forward networks. The core building blocks in all these layers consist of different sets of weight matrices that compute affine transforms.

First, an embedding layer obtains the source and target word vectors from the input words: $\boldsymbol{W}_E \in \mathbb{R}^{d_m \times V}$, where d_m is model size, and V is vocabulary size. After the embedding lookup step, word vectors are multiplied by a scaling factor of $\sqrt{d_m}$. To capture the relative position of a word in the input sequence, *position encodings* defined in terms of sinusoids of different frequencies are added to the scaled word vectors of the source and target.

The encoder layer maps the input word vectors to continuous hidden state representations. As mentioned earlier, it consists of two sublayers. The first sublayer performs *multi-head dot-product self-attention*. In the single-head case, defining the input to the sublayer as $x = (x_1, \ldots, x_T)$ and the output as $z = (z_1, \ldots, z_T)$, where $x_i, z_i \in \mathbb{R}^{d_m}$,

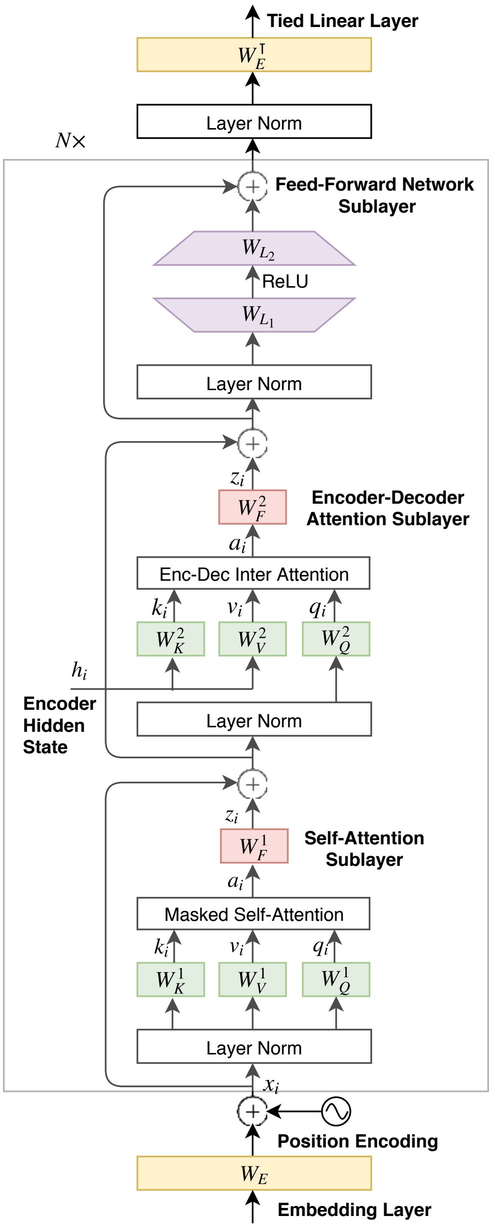


Figure 2: Block diagram illustrating the Transformer decoder's shareable parameters (in color) that includes embedding layer weights ($\boldsymbol{W_E}$), tied linear layer weights ($\boldsymbol{W_E^T}$), transformation weights as a part of self-attention ($\boldsymbol{W_K^1}, \boldsymbol{W_V^1}, \boldsymbol{W_Q^1}, \boldsymbol{W_F^1}$), encoder-decoder attention ($\boldsymbol{W_K^2}, \boldsymbol{W_V^2}, \boldsymbol{W_Q^2}, \boldsymbol{W_F^2}$), and feed-forward network ($\boldsymbol{W_{L_1}}, \boldsymbol{W_{L_2}}$) sublayers. Best viewed in color.

the input is linearly transformed to obtain key (k_i), value (v_i), and query (q_i) vectors

$$k_i = x_i \boldsymbol{W}_K, v_i = x_i \boldsymbol{W}_V, q_i = x_i \boldsymbol{W}_Q.$$

Next, similarity scores (e_{ij}) between query and key vectors are computed by performing a scaled dot-product

$$e_{ij} = \frac{1}{\sqrt{d_m}} q_i k_j^T.$$

Next, attention coefficients (α_{ij}) are computed by applying softmax function over these similarity values.

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{l=1}^{T} \exp e_{il}}$$

Self-attention output (z_i) is computed by the convex combination of attention weights with value vectors followed by a linear transformation

$$z_i = (\sum_{j=1}^{T} \alpha_{ij} v_j) \boldsymbol{W}_F.$$

In the above equations, $\boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_Q, \boldsymbol{W}_F$ are learnable transformation matrices of shape $\mathbb{R}^{d_m \times d_m}$. To extend to multi-head attention (ℓ), one can split the key, value, and query vectors into ℓ vectors, perform the attention computation in parallel for each of the ℓ vectors followed by concatenating before the final linear transformation by $\boldsymbol{W}_F$. The second sublayer consists of a two-layer deep *position-wise feed-forward network* (FFN) with ReLU activation (Glorot et al., 2011).

$$\mathrm{FFN}(z_i) = \max(0,\ z_i \boldsymbol{W_{L_1}} + b_1) \boldsymbol{W_{L_2}} + b_2$$

where $\boldsymbol{W_{L_1}} \in \mathbb{R}^{d_m \times d_h}$, $\boldsymbol{W_{L_2}} \in \mathbb{R}^{d_h \times d_m}$, b_1 and b_2 are biases, and d_h is hidden size. The FFN sublayer outputs are subsequently given as input to the next encoder layer.

The decoder layer consists of three sublayers. The first sublayer, similar to the encoder, performs masked self-attention where masks are used to prevent positions from attending to subsequent positions. The second sublayer performs *encoder-decoder inter-attention* where the input to the query vector comes from the decoder layer while the input to the key and value vectors comes from the encoder's last layer. To denote parameters in these two sublayers, the transformation weights of the masked self-attention sublayer are referenced as $\boldsymbol{W_K^1}, \boldsymbol{W_V^1}, \boldsymbol{W_Q^1}, \boldsymbol{W_F^1}$ and encoder-decoder attention sublayer as $\boldsymbol{W_K^2}, \boldsymbol{W_V^2}, \boldsymbol{W_Q^2}, \boldsymbol{W_F^2}$, which is also indicated in Figure 2. The third sublayer consists of an FFN. To generate predictions for the next word, there is a linear layer on top of the decoder layer. The weight of this linear layer is shared with the weight of the embedding layer (Inan et al., 2016).

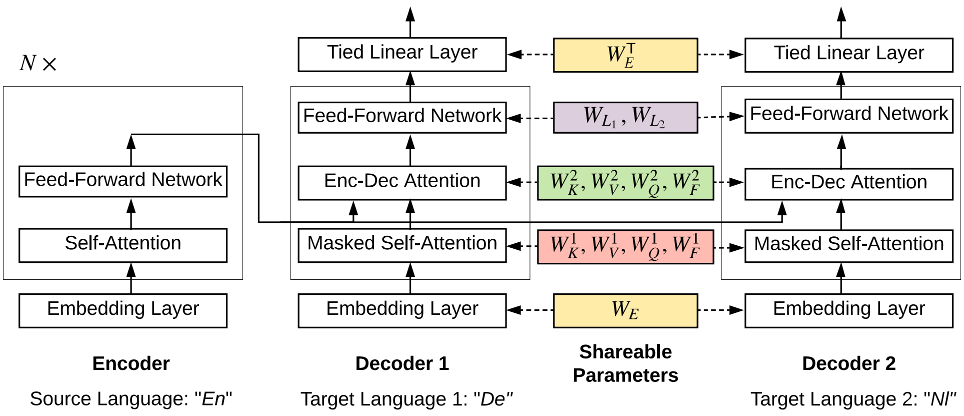


Figure 3: Block diagram illustrating our MTL approach for *one-to-many* multilingual translation task that is based on the partial sharing of parameters between the multiple decoders. Best viewed in color.

Residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) are applied on each sublayer and to the output vector from the final encoder and decoder layers.

2.2 Parameter Sharing Strategies

In this paper, our objective is to investigate effective parameter sharing strategies for the Transformer model using MTL, mainly for *one-to-many* multilingual translation. Here, we will use the symbol $\boldsymbol{\Theta}$ to denote the set of shared parameters in our model. These parameter sharing strategies are described below:

- The base case consists of separate bilingual translation models for each language pair $\left(\boldsymbol{\Theta} = \emptyset\right)$.
- Use of a common embedding layer for all the bilingual models $\left(\boldsymbol{\Theta} = \{\boldsymbol{W}_E\}\right)$. This will result in a significant reduction of the total parameters by sharing parameters across common words present in the source and target sentences (Wu et al., 2016).
- Use of a common encoder for the source language and a separate decoder for each target language $\left(\boldsymbol{\Theta} = \{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}\}\right)$. This has the advantage that the encoder will now see more source language training data (Dong et al., 2015).

Next, we also include the decoder parameters among the set of shared parameters. While doing so, we will assume that the embedding and the encoder parameters are always shared between the bilingual models. Because there can be exponentially many combinations considering all the different feasible sets of shared parameters between the multiple decoders, we only select a subset of these combinations based on our preliminary results. These selected weights are shared in all the layers of the decoder unless stated otherwise. A schematic diagram illustrating the various possible parameter matrices that can be shared in each sublayer of our MTL model is shown in Figure 3.

- We share only the FFN sublayer parameters $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}, \boldsymbol{W}_{L_1}, \boldsymbol{W}_{L_2}\right\}\right)$.
- Sharing the weights of the self-attention sublayer $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}, \boldsymbol{W}_K^1, \boldsymbol{W}_Q^1, \boldsymbol{W}_V^1, \boldsymbol{W}_F^1\right\}\right)$.
- Sharing the weights of the encoder-decoder attention sublayer $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}, \boldsymbol{W}_K^2, \boldsymbol{W}_Q^2, \boldsymbol{W}_V^2, \boldsymbol{W}_F^2\right\}\right)$.
- We limit the attention parameters that are shared to only include either the key and query weights $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}, \boldsymbol{W}_K^1, \boldsymbol{W}_Q^1, \boldsymbol{W}_K^2, \boldsymbol{W}_Q^2\right\}\right)$ or the key and value weights $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC}, \boldsymbol{W}_K^1, \boldsymbol{W}_V^1, \boldsymbol{W}_K^2, \boldsymbol{W}_V^2\right\}\right)$. The motivation for doing so is so that the shared attention sublayer weights can model the common aspects of the target languages while the individual FFN sublayer weights can model the distinctive or unique aspects of each language.
- We share all the parameters of the decoder to have a single unified model $\left(\boldsymbol{\Theta} = \left\{\boldsymbol{W}_E, \boldsymbol{\theta}_{ENC},\right.\right.$

Language Pair	Training	Dev	Test
EN–RO	180,484	3,904	4,631
EN–FR	192,304	4,320	4,866
EN–NL	183,767	4,459	5,006
EN–DE	167,888	4,148	4,491
EN–JA	204,090	4,429	5,565
EN–TR	182,470	4,045	5,029

Table 1: Number of sentences in the training, dev, and test splits for each language pair used in our experiments. The languages are represented by their ISO 639-1 codes *En:English*, *Fr:French*, *Nl:Dutch*, *De:German*, *Ja:Japanese*, *Tr:Turkish*.

$\boldsymbol{\theta}_{DEC}\})$. Fewer parameters in the decoder indicates limited modeling ability, and we expect this method to obtain good translation accuracy mainly when the target languages are related (Johnson et al., 2017).

3 Experimental Setup

In this section, first, we describe the datasets used in this work and the evaluation criteria. Then, we describe the training regimen followed in all our experiments. All of our models were implemented in PyTorch framework (Paszke et al., 2017) and were trained on a single GPU.

3.1 Datasets and Evaluation Metric

To perform multilingual translation experiments, we select six language pairs from the openly available TED talks dataset (Qi et al., 2018) whose statistics are mentioned in Table 1. This dataset already contains predefined splits for training, development, and test sets. Among these languages, Romanian (RO) and French (FR) are *Romance* languages, German (DE) and Dutch (NL) are *Germanic* languages while Turkish (TR) and Japanese (JA) are unrelated languages that come from distant language families. For all language pairs, tokenization was carried out using the `Moses` tokenizer,[2] except for Japanese, where word segmentation was performed using the `KyTea` tokenizer (Neubig et al., 2011). To select training examples, we filter sentences with a maximum length of 70 tokens. For evaluation, we report the model's performance using the standard BLEU score metric (Papineni et al., 2002). We use the `mtevalv14.pl` script from the `Moses` toolkit to compute the tokenized BLEU scores.

3.2 Training Protocols

In this work, we follow the same training process for all the experiments. We jointly encode the source and target language words with subword units by applying *byte pair encoding* (Gage, 1994) with 32,000 merge operations (Sennrich et al., 2016). These subword units restrict the vocabulary size and prevent the need for explicitly handling out-of-vocabulary symbols as the vocabulary can be used to represent any word. We use *LeCun uniform initialization* (LeCun et al., 1998) for all the trainable model parameters. Embedding layer weights are randomly initialized according to truncated Gaussian distribution $\boldsymbol{W}_E \sim \mathcal{N}(0, d_m{}^{-1/2})$.

In all the experiments, we use *Transformer base model* configuration (Vaswani et al., 2017) that consists of six encoder-decoder layers, $d_m = 512$, $d_h = 2,048$, and $\ell = 8$. For optimization, we use SGD with Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.997$, and $\epsilon = 1e^{-9}$.[3] The learning rate (*lr*) schedule is varied at every optimization step (*step*) according to:

$$lr = 2d_m^{-0.5}\min\left(step^{-0.5}, step \cdot 16000^{-1.5}\right)$$

Each mini-batch consists of approximately $3,000$ source and $3,000$ target tokens such that similar length sentences are bucketed together. We train the models until convergence and save the best checkpoint using development set performance. For model regularization, we use label smoothing $(\epsilon = 0.1)$ (Pereyra et al., 2017) and apply dropout (with $p_{drop} = 0.1$) (Srivastava et al., 2014) to the word embeddings, attention coefficients, ReLU activation, and to the output of each sublayer before the residual connection. During decoding, we use beam search with beam width 5 and length normalization with $\alpha = 1$ (Wu et al., 2016).

3.3 Multilingual Training

During the multilingual model's training and inference, we include an additional token representing the desired target language at the start of each source sentence (Johnson et al., 2017). The presence of this additional token will help the model learn the target language to translate to during decoding. For preprocessing, we apply byte pair en-

[2] https://github.com/moses-smt/mosesdecoder/tree/master/scripts/tokenizer

[3] These hyperparameter values are based on the single GPU transformer model from the open-source *tensor2tensor* toolkit.

coding over the combined dataset of all the language pairs. We perform model training using balanced mini-batches *i.e.* it contains roughly an equal number of sentences for every target language. While training, we compute weighted average cross-entropy loss where the weighting term is proportional to the total word count observed in each of the target language sentences.

4 Results

In this section, we will describe the results of our proposed parameter sharing techniques and later present the broader context by comparing them with bilingual translation models and previous benchmark methods.

4.1 Parameter Sharing

Here, we first analyze the results of *one-to-many* multilingual translation experiments when there are two target languages and both of them belong to the same language family. The first set of experiments are on *Romance* languages (EN→RO+FR) and the second set of experiments are on *Germanic* languages (EN→DE+NL). We report the BLEU scores in Table 2a when different sets of parameters are shared in these experiments. We observe that sharing only the embedding layer weight between the multiple models leads to the lowest scores. Sharing the encoder weights results in significant improvement for EN→RO+FR but leads to a small decrease in EN→DE+NL scores.

We then gradually include both the decoder's weights to the set of shareable parameters. Specifically, we include the parameters of FFN, self-attention, encoder-decoder attention, both the attention sublayers, key, query, value weights from both the attention sublayers, and finally all the parameters of the decoder layer. From the results, we note that the sharing of the encoder-decoder attention weights leads to substantial gains. Finally, sharing the entirety of the parameters (*i.e.* having one model) leads to the best BLEU scores for EN→RO+FR and sharing only the key and query matrices from both the attention layers leads to the best BLEU scores for EN→DE+NL. One of the reasons for such large increase in BLEU is that encoder has access to more English language training data and for the decoder, as the target languages belong to the same family, they may contain common vocabulary, thus improving the generalization error for both the target languages.

Next, we analyze the results of *one-to-many* translation experiments when both the target languages belong to distant language families and are unrelated. The first set of experiments are on *Germanic, Turkic* languages (EN→DE+TR) and the second set of experiments are on *Germanic, Japonic* languages (EN→DE+JA). We present the results in Table 2b when different sets of parameters are shared. Here, we observe that the approach of sharing all the parameters leads to a noticeable drop in the BLEU scores for both the considered language pairs. Similar to the above discussion, sharing the key and query matrices results in a large increase in the BLEU scores. We hypothesize that in this partial parameter sharing strategy, the sharing of key and query attention weights effectively models the common linguistic properties while the separate FFN sublayer weights model the unique characteristics of each target language, thus overall leading to a large improvement in the BLEU scores. The results of other decoder parameter sharing approaches lie close to the key and query parameter sharing method. As the target languages are from different families, their vocabularies may have some overlap but will be significantly different from each other. In this scenario, a useful alternative is to consider a separate embedding layer for every source-target language pair while sharing all the encoder and decoder parameters. However, we did not experiment with this approach, as the inclusion of separate embedding layers will lead to a large increase in the model parameters and as a result model training will become more memory intensive. We leave the investigation of such parameter sharing strategy to future work.

4.2 Overall Comparison

In Table 3, we show an overall performance comparison of no parameter sharing, full parameter sharing for both GNMT (Wu et al., 2016) and Transformer models, and the best approaches according to maximum BLEU score from our partial parameter sharing strategies. For training the GNMT models, we use its open-source implementation[4] (Luong et al., 2017) with four layers[5] and default parameter settings. First, we note that the BLEU scores of the Transformer model are always better than the GNMT model by a significant margin for both bilingual (no sharing) and multilingual

[4] https://github.com/tensorflow/nmt

[5] We found that the four layer model for GNMT didn't overfit and obtained the best BLEU scores.

Set of shared parameters (Θ)	En→Ro+Fr		En→De+Nl		*params*
	→Ro	→Fr	→De	→Nl	$\times 10^6$
W_E	27.21	43.36	30.32	33.51	105
W_E, θ_{ENC}	27.82	43.83	29.97	33.33	86
W_E, θ_{ENC}, W_1, W_2	27.78	43.87	29.95	33.12	74
W_E, θ_{ENC}, W_K^1, W_Q^1, W_V^1, W_F^1	27.80	43.76	30.68	33.99	80
W_E, θ_{ENC}, W_K^2, W_Q^2, W_V^2, W_F^2	28.36	44.19	30.50	33.75	80
W_E, θ_{ENC}, W_K^1, W_V^1, W_K^2, W_V^2	27.77	43.83	30.54	34.00	80
W_E, θ_{ENC}, W_K^1, W_Q^1, W_K^2, W_Q^2	27.58	43.84	**30.70**	**34.05**	80
W_E, θ_{ENC}, W_K^1, W_Q^1, W_V^1, W_F^1, W_K^2, W_Q^2, W_V^2, W_F^2	28.14	44.12	30.64	33.92	74
W_E, θ_{ENC}, θ_{DEC}	**28.52**	**44.28**	30.45	33.69	61

(a) The target languages in this *one-to-many* translation task belong to the same language family. Ro and Fr are *Romance* languages while De and Nl are *Germanic* languages.

Set of shared parameters (Θ)	En→De+Tr		En→De+Ja		*params*
	→De	→Tr	→De	→Ja	$\times 10^6$
W_E	30.35	19.66	30.10	18.62	105
W_E, θ_{ENC}	30.55	19.29	30.21	18.70	86
W_E, θ_{ENC}, W_{L_1}, W_{L_2}	30.21	19.17	30.36	18.92	74
W_E, θ_{ENC}, W_K^1, W_Q^1, W_V^1, W_F^1	30.35	19.24	30.05	18.78	80
W_E, θ_{ENC}, W_K^2, W_Q^2, W_V^2, W_F^2	30.49	19.40	30.16	18.73	80
W_E, θ_{ENC}, W_K^1, W_V^1, W_K^2, W_V^2	30.66	19.34	30.36	18.92	80
W_E, θ_{ENC}, W_K^1, W_Q^1, W_K^2, W_Q^2	**30.71**	**19.67**	**30.48**	**19.00**	80
W_E, θ_{ENC}, W_K^1, W_Q^1, W_V^1, W_F^1, W_K^2, W_Q^2, W_V^2, W_F^2	30.40	19.35	30.35	18.80	74
W_E, θ_{ENC}, θ_{DEC}	28.74	18.69	29.68	18.50	61

(b) The target languages in this *one-to-many* translation task belong to distant language families. De, Tr, and Ja are unrelated as they belong to *Germanic*, *Turkic*, and *Japonic* language families respectively.

Table 2: BLEU scores for various parameter sharing strategies when the target languages either belong to the same family ({Ro, Fr}, {De, Nl}) or to distant families (De, Tr, Ja). θ_{ENC} denotes that all the encoder parameters are shared between the models; θ_{DEC} denotes that all the decoder parameters are shared between the models.

Method	En→De+Tr		En→De+Ja		En→Ro+Fr		En→De+Nl		*params*
	→De	→Tr	→De	→Ja	→Ro	→Fr	→De	→Nl	$\times 10^6$
GNMT NS	27.01	16.07	27.01	16.62	24.38	40.50	27.01	30.64	–
GNMT FS	29.07	18.09	28.24	17.33	26.41	42.46	28.52	31.72	–
Transformer NS	29.31	18.62	29.31	17.92	26.81	42.95	29.31	32.43	122
Transformer FS	28.74	18.69	29.68	18.50	**28.52**	**44.28**	30.45	33.69	61
Transformer PS	**30.71**	**19.67**	**30.48**	**19.00**	27.58	43.84	**30.70**	**34.05**	80

Table 3: BLEU scores for different models for *one-to-many* translation task. **NS**: *No Sharing* corresponds to the bilingual models when the two language pairs are trained independently; **FS**: *Full Sharing* means one model is used for the translation of all the language pairs; **PS**: *Partial Sharing* means that the embedding, encoder, decoder's key, and value weights are shared between the two models.

(full sharing) translation tasks. This reflects that the Transformer model is well-suited for both multilingual and bilingual translation tasks compared with the GNMT model. We also surprisingly note that the GNMT fully shared model is able to consistently obtain higher BLEU scores compared with its bilingual version irrespective of which families the target languages belong to.

However, for the *one-to-many* translation task when the target languages are from distant families, we observe that fully shared Transformer model leads to a substantial drop or small gains in the BLEU score compared with the bilingual models. Specifically, for the En→De+Tr setting, BLEU drops by 0.6 for En→De, while staying even for En→Tr. In contrast, our method of sharing embedding, encoder, decoder's key, and query parameters leads to substantial increases in BLEU scores (1.4↑ for En→De and 1.1↑ for En→Tr). Similarly, for En→De+Ja, using the fully shared Transformer model, we observe small gains of 0.3 and 0.5 BLEU points for En→De and En→Ja respectively while our partial parameter sharing method again leads to significant improvements (1.5↑ for En→De and 1.1↑ for En→Ja). This demonstrates the utility of our proposed partial parameter sharing method.

We also note that fully shared Transformer models can be an effective strategy only when both the target languages are from the same family. For the task of En→Ro+Fr, the fully shared model performs surprisingly well and yields significant improvements of 1.7 and 1.3 BLEU points compared with bilingual models for En→Ro and En→Fr respectively. A similar increase in performance can also be observed for the En→De+Nl task, although for this task, our partial parameter sharing method (encoder, embedding, decoder's key, and query weights) obtains even higher BLEU scores. (1.4↑ for En→De and 1.6↑ En→Nl).

4.3 Analysis

Here, we analyze the generated translations of the partial sharing and full sharing approaches for En→De when *one-to-many* multilingual model was trained on unrelated target language pairs En→De+Tr. These translations were obtained using the test set of En→De task. Here partial sharing refers to the specific approach of sharing the embedding, encoder, and decoder's key and query parameters in the model.

We show example translations in Table 4 where partial sharing method gets a high BLEU score (shown in parentheses) but the full sharing method does not. We see that sentences generated by partial sharing method are both semantically and grammatically correct while the full sharing method generates shorter sentences compared with reference translations. As highlighted in table cells, the partial sharing method is able to correctly translate a mention of relative time "*half a year*" and a co-reference expression "*mich*". In contrast, the fully shared model generates incorrect expressions of time mentions "*eineinhalb Jahren*" (one and half years) and different verb forms ("*schlägt*" is generated vs "*schlagen*" in the reference).

We also perform a comparison of the F-measure of the target words for En→De, bucketed by frequency in the training set. As displayed in Figure 4, this shows that the partial parameter sharing approach improves the translation accuracy for the entire vocabulary, but in particular for words that have low-frequency in the dataset.

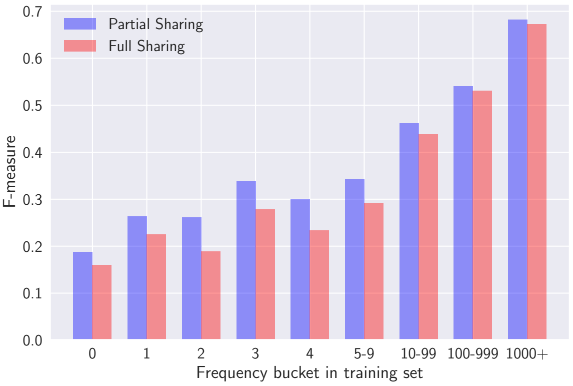


Figure 4: The F-measure for the target language (De) words in *one-to-many* multilingual translation task (En→De+Tr). Best viewed in color.

5 Related Work

In this section, we will review the prior work related to MTL and multilingual translation.

5.1 Multi-task learning

Ando and Zhang (2005) obtained excellent results by adopting an MTL framework to jointly train linear models for NER, POS tagging, and language modeling tasks involving some degree of parameter sharing. Later, Collobert et al. (2011) applied MTL strategies to neural networks for tasks such as POS tagging, NER, and chunking by sharing the

source	So half a year ago , I decided to go to Pakistan myself .
reference	Vor einem halben Jahr entschied ich mich , selbst nach Pakistan zu gehen .
partial sharing	Vor einem halben Jahr entschied ich mich , selbst nach Pakistan zu gehen . (1.0)
full sharing	Vor eineinhalb Jahren beschloss ich , nach Pakistan zu gehen . (0.35)
source	Your heart starts beating faster .
reference	Ihr Herz beginnt schneller zu schlagen .
partial sharing	Ihr Herz beginnt schneller zu schlagen . (1.0)
full sharing	Ihr Herz schlägt schneller . (0.27)

Table 4: Sample translations from En→De when *one-to-many* multilingual model was trained on unrelated target language pairs En→De+Tr. In these examples, the method of partial sharing of decoder parameters obtains a very high BLEU score (mentioned in parentheses).

sequence encoder and reported moderate improvements in results. Recently, Luong et al. (2016) investigated MTL for a tasks such as parsing, image captioning, and translation and observed large gains in the translation task. Similarly, for MT tasks, Niehues and Cho (2017) also leverage MTL by using additional linguistic information to improve the translation accuracy of NMT models. They share the encoder representations to perform joint training on translation, POS, and NER tasks. MTL has also been widely applied to multilingual translation that will be discussed next.

5.2 Multilingual Translation

On the multilingual translation task, Dong et al. (2015) obtained significant performance gains by sharing the encoder parameters of the source language while having a separate decoder for each target language. Later, Firat et al. (2016) attempted the more challenging task of *many-to-many* translation by training a model that consisted of one shared encoder and decoder per language and a shared attention layer that was common to all languages. This approach obtained competitive BLEU scores on ten European language pairs while substantially reducing the total parameters. Recently, Johnson et al. (2017) proposed a unified model with full parameter sharing and obtained comparable or better performance compared with bilingual translation scores. During model training and decoding, target language was specified by an additional token at the beginning of the source sentence. Coming to low-resource language translation, Zoph et al. (2016) used a transfer learning approach of fine-tuning the model parameters learned on a high-resource language pair of French→English and were able to significantly increase the translation performance on Turkish and Urdu languages. Recently, Gu et al. (2018) addresses the *many-to-one* translation problem for extremely low-resource languages by using a transfer learning approach such that all language pairs share the lexical and sentence-level representations. By performing joint training of the model with high-resource languages, large gains in the BLEU scores were reported for low-resource languages.

In this paper, we first experiment with the Transformer model for *one-to-many* multilingual translation on a variety of language pairs and demonstrate that the approach of Johnson et al. (2017) and Dong et al. (2015) is not optimal for all kinds of target-side languages. Motivated by this, we introduce various parameter sharing strategies that strike a happy medium between full sharing and partial sharing and show that it achieves the best translation accuracy.

6 Conclusion

In this work, we explore parameter sharing strategies for the task of multilingual machine translation using self-attentional MT models. Specifically, we examine the case when the target languages come from the same or distant language families. We show that the popular approach of full parameter sharing may perform well only when the target languages belong to the same family while a partial parameter sharing approach consisting of shared embedding, encoder, decoder's key and query weights is generally applicable to all kinds of language pairs and achieves the best BLEU scores when the languages are from distant families.

For future work, we plan to extend our parameter sharing approach in two directions. First, we aim to increase the number of target languages to more than two such that they contain a mix of both similar and distant languages and analyze the performance of our proposed parameter sharing strategies on them. Second, we aim to experiment with additional parameter sharing strategies such as sharing the weights of some specific layers (*e.g.* the first or last layer) as different layers can encode different morphological information (Belinkov et al., 2017) which can be helpful in better multilingual translation.

Acknowledgments

The authors would like to thank Mrinmaya Sachan, Emmanouil Antonios Platanios, and Soumya Wadhwa for useful comments about this work. We would also like to thank the anonymous reviewers for giving us their valuable feedback that helped to improve the paper.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal Machine Learning Research*, 6:1817–1853.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computing Research Repository*, arXiv:1409.0473.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354.

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *Computing Research Repository*, arXiv:1611.01462.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*, pages 80–89.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *Computing Research Repository*, arXiv:1701.06548.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535.

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository*, arXiv:1609.08144.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575. Association for Computational Linguistics.