Improving Language Transfer Capability of Decoder-only Architecture in Multilingual Neural Machine Translation

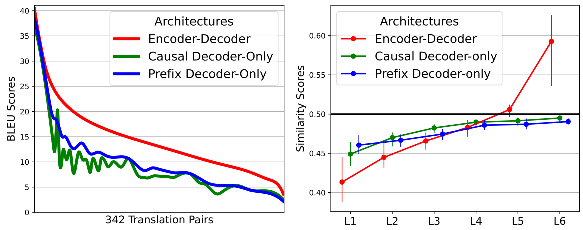
Anonymous ACL submission

Abstract

Decoder-only architecture performs poorly in multilingual neural machine translation, despite its potential benefits in zero-shot translation, i.e., translation of unseen language pairs during training. In this work, we identify the main issue of the decoder-only architecture as its lack of language transfer capability. Specifically, representations from different source languages are not aligned in the representational subspace of the target language. We propose dividing the decoding process into two stages so that target tokens are explicitly excluded in the first stage to implicitly boost the transfer capability across languages. Additionally, we impose contrastive learning on translation instructions, resulting in improved performance in zero-shot translation. We conduct experiments on TED-19 and OPUS-100 datasets, considering both training from scratch and fine-tuning scenarios. Experimental results show that, compared to the encoder-decoder architecture, our methods not only perform competitively in supervised translations but also achieve improvements of up to 3.39 BLEU, 6.99 chrF++, 3.22 BERTScore, and 4.81 COMET in zero-shot translations.[1]

1 Introduction

Multilingual neural machine translation (MNMT) task (Firat et al., 2016; Johnson et al., 2017), which aims to integrate multiple language translation directions into a single model, can achieve performance comparable to large language models with fewer parameters (Zhu et al., 2023; Xu et al., 2024). Decoder-only architecture has been shown to excel at zero-shot generalization (Brown et al., 2020; Wang et al., 2022), which potentially benefits the zero-shot translation, i.e., translation of unseen language pairs during training. However, state-of-the-art MNMT models are still based on encoder-decoder architecture (Fan et al., 2020; Team et al., 2022), because decoder-only architectures (Dong et al., 2019), including the casual manner (Radford et al., 2018) and the prefixed manner (Dong et al., 2019), perform weaker in MNMT (Gao et al., 2022; Raffel et al., 2023) in practice (Figure 1a).



(a) Performance (b) Preference

Figure 1: Comparison between different architectures in preliminary experiments on TED-19. 1a shows the performance. 1b shows the linguistic preference of layer-wise representation, and the x-axis indicates the layer number. Specifically, a similarity score higher than 0.5 means the representation prefers the target language, while a score lower than 0.5 indicates a preference for the source language. Additionally, the vertical line indicates the value range. Appendix A provides a detailed explanation of 1b.

MNMT models typically add a language tag, indicating the target language, at the beginning of the source tokens as a translation instruction (Johnson et al., 2017; Wu et al., 2021; Team et al., 2022). Recently, Qu et al. (2024) state that the success of the encoder-decoder architecture in MNMT is attributed to the language transfer capability of the encoder. Specifically, as shown in Figure 1b, the encoder-decoder model aligns representations from different source languages in the representational subspace of the target language, making the decoding process rely on the representation with target language features. However, this process is absent in the decoder-only architecture because the generation of target tokens solely relies on source tokens from the beginning.

In this work, we propose dividing the decoder-only architecture into two stages, termed Two-stage Decoder-only (TDO). Specifically, the representa-

[1] We will release all codes on GitHub for reproduction if our paper is accepted.

tions of target tokens are not used in the first stage to allow language transfer, and the target representations are recovered in the second stage, which follows the normal decoder-only manner. Additionally, a potential degradation occurs in the second stage due to the lack of an explicit optimization objective for the source tokens. Therefore, we further introduce Instruction-level Contrastive Learning (InstruCL), which enhances the significance of translation instruction to prevent degradation.

We evaluate the proposed methods on two datasets, TED-19 (Ye et al., 2018), and OPUS-100 (Zhang et al., 2020a; Yang et al., 2021), using four automatic evaluation metrics for a comprehensive understanding of the improvement: BLEU (Papineni et al., 2002; Post, 2018), chrF++ (Popović, 2015, 2017), BERTScore (Zhang et al., 2020b) and COMET (Rei et al., 2020). Experimental results show that, compared to models with the encoder-decoder architecture, our models perform competitively in supervised translations and achieve improvements of up to 3.39 BLEU, 6.99 chrF++, 3.22 BERTScore, and 4.81 COMET in zero-shot translations. Furthermore, we analyze the variation of layer-wise representation to demonstrate the effects of our proposed methods. Results prove that the gains of our proposed methods in the decoder-only architecture derive from the improvement of language transfer.

2 Backgrounds

2.1 Multilingual Neural Machine Translation

Multilingual Neural Machine Translation (MNMT) task aims to train a single model capable of supporting translations between multiple languages. Given a parallel multilingual corpus, denoted by $\mathbb{C}$, the raw data within $\mathbb{C}$ consists of translation pairs in the form of $(\boldsymbol{x}, \boldsymbol{y})$. Here, $\boldsymbol{x} = x_1, \ldots, x_I$ is the source sentence composed of I tokens, and $\boldsymbol{y} = y_1, \ldots, y_J$ is the target sentence composed of J tokens. We also denote language tags by $\boldsymbol{l} = l_1, \ldots, l_K$, where each tag is an artificial token uniquely corresponding to one of the K languages in $\mathbb{C}$. To serve as a translation instruction[2], we add the language tag specifying the target language at the beginning of the source tokens (Johnson et al., 2017; Wu et al., 2021), denoted by $l_{\boldsymbol{y}}$. Thus, the training data comprises instances in the form of $(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y})$. The model is trained over all instances in $\mathbb{C}$ by the standard training objective:

[2]Appendix B shows the comparison between different strategies of translation instructions in MNMT.

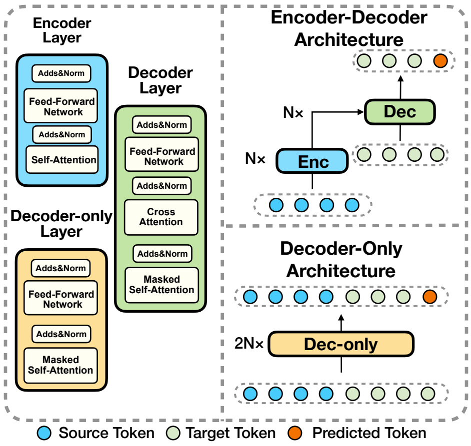


Figure 2: Illustration of the encoder-decoder architecture and the decoder-only architecture.

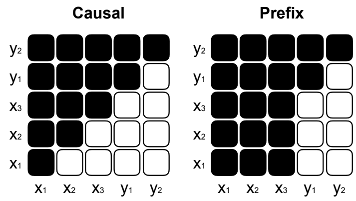


Figure 3: Different manners of the masked self-attention mechanism in the decoder-only architectures. Black blocks mean visible and white blocks mean masked. Thus, source tokens are masked in the causal decoder-only while are visible in the prefix decoder-only.

$$\mathcal{L}_{\text{ce}} = -\sum_{l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}} \sum_{j=1}^{J} \log\ p(y_j \mid l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y}_{<j}), \quad (1)$$

where $p(y_j \mid l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y}_{<j})$ is a probability distribution generated by MNMT model.

2.2 Architectures

All architectures discussed in this work follow the Transformer architecture (Vaswani et al., 2017), which is the de facto standard of MNMT.

Almost all MNMT models are based on the encoder-decoder architecture (Johnson et al., 2017; Fan et al., 2020; Team et al., 2022; Raffel et al., 2023), as illustrated in Figure 2, which comprises two components, an encoder and a decoder. Both the encoder and decoder are composed of N layers with each encoder layer comprising a self-attention mechanism and a feed-forward network (FFN), and with each decoder layer comprising a masked self-attention mechanism, a cross-attention mechanism, and an FFN. The encoder receives the input of $(l_{\boldsymbol{y}}, \boldsymbol{x})$, and output the representations $\mathbf{H} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_{I+1}\}, \boldsymbol{h} \in \mathbb{R}^d$, d is the model dimension. Then, the decoder relies on $\mathbf{H}$ and $\boldsymbol{y}_{<j}$

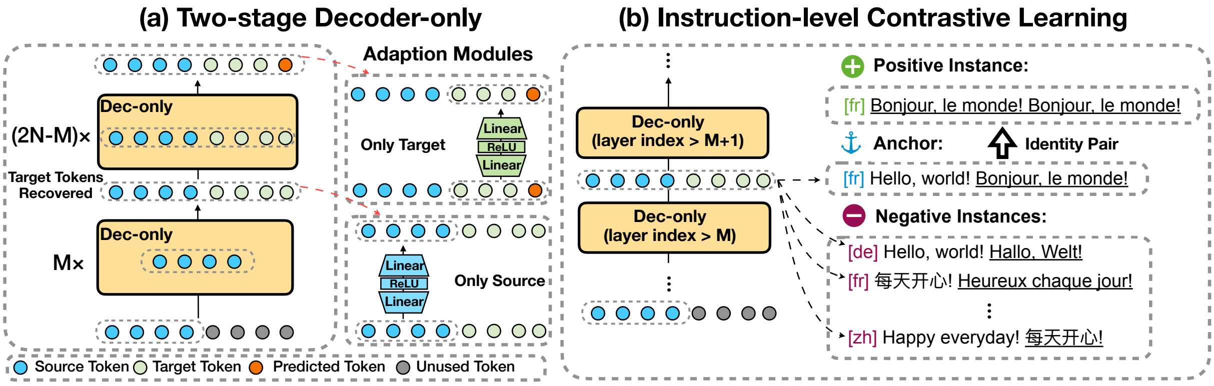


Figure 4: Illustration of proposed methods. Notably, the term, Token, not only means the real token before and after the processing of model, but also refers to the representation in the corresponding position. (a) shows the Two-stage Decoder-only and shows the Adaption, i.e., using an additional FFN to narrow the gap between source representations and target representations by non-linear transformation. (b) shows the Instruction-level Contrastive Learning. Underline marks target tokens, and [*] means the instruction of this instance. For the anchor, negative instances in this figure meet at least one of two features: 1) different target language and 2) unparallel semantics.

to generate the next token:

$$\mathbf{H}^N = \text{encoder}(l_{\boldsymbol{y}}, \boldsymbol{x}), \tag{2}$$

$$y_j = \text{decoder}(\mathbf{H}^N, \boldsymbol{y}_{<j}), \tag{3}$$

where N is the layer number of the encoder, and $\mathbf{H}^N$ is an intermediate state used in the cross-attention mechanism in each decoder layer without further transformation. Thus, Equation 1 implicitly aligns the output of the encoder in the representational subspace of the target language, i.e., the language transfer as shown in the red line of Figure 1b, because the ideal decoder should translate two sentences $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$, which have the same target language, parallel semantics, and different source languages, to the same target sentence $\boldsymbol{y}$. Formally, an ideal encoder meets the following:

$$\text{encoder}(l_{\boldsymbol{y}}, \boldsymbol{x}^a) = \text{encoder}(l_{\boldsymbol{y}}, \boldsymbol{x}^b). \tag{4}$$

A decoder-only architecture refers to a model that consists solely of a decoder (Figure 2). Each decoder-only layer consists of a masked self-attention mechanism and an FFN (Radford et al., 2018), and each model has $2N$ layers to approximately match the parameter size of an encoder-decoder architecture. We define the decoder-only process as follows:

$$y_j = \text{decoder-only}(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y}_{<j}). \tag{5}$$

Notably, the difference between $\text{decoder-only}(\cdot)$ and $\text{decoder}(\cdot)$ is that $\text{decoder-only}(\cdot)$ fuses the source and target information by a concatenated input[3], namely, $l_{\boldsymbol{y}}$, $\boldsymbol{x}$, and $\boldsymbol{y}$ are equally treated, instead of using a cross-attention mechanism. Thus, there is not an intermediate state to align different source languages as Equation 4, resulting in the blue and green lines of Figure 1b. Moreover, we follow Gao et al. (2022); Raffel et al. (2023) to distinguish the decoder-only by the manner of masked self-attention mechanism as causal decoder-only and prefix decoder-only (Figure 3). Finally, compared to the encoder-decoder architecture, the decoder-only architecture requires around 10% fewer parameters.[4]

3 Methodologies

3.1 Two-stage Decoder-only Architecture

The limitations of the decoder-only architecture in MNMT likely arise from inadequate language transfer capabilities, i.e., the absence of Equation 4. To address this issue, we propose the Two-stage Decoder-only (TDO) architecture, which divides the decoder-only process into two stages to align source representations in the subspace of the target language. Specifically, as illustrated in Figure 4, target representations are not used in the first stage, i.e., the first M layers, and these target representations are recovered in the second stage, i.e., the subsequent $2N - M$ layers. The process of TDO is formally expressed as follows:

$$\mathbf{H}^M = \text{decoder-only}_1(l_{\boldsymbol{y}}, \boldsymbol{x}), \tag{6}$$

$$y_j = \text{decoder-only}_2(\mathbf{H}^M, \boldsymbol{y}_{<j}), \tag{7}$$

where $\text{decoder-only}_1(\cdot)$ enables the implicit alignment as done in Equation 4. Notably, the first stage logically acts as an encoder when prefixed masking

[3] Appendix C introduces the input forms in this work.

[4] Appendix D introduces the estimation process.

is applied to the self-attention mechanism. However, the first and second stages remain unified structures, and the fusing of source and target information follows the manner of $\mathrm{decoder\text{-}only}(\cdot)$ rather than $\mathrm{decoder}(\cdot)$. Therefore, TDO architecture is a revision of the decoder-only architecture.

We also introduce two optional Adaptation modules in the information fusing. Specifically, a representational gap arises at the $M+1$ layer because the source representation has been passed through prior M layers while the target representation has not. As shown in Figure 4, we employ an FFN, which includes an up-projection linear layer, a ReLU activation function, and a down-projection linear layer (Vaswani et al., 2017), to nonlinearly transform the source representation to bridge the gap (Geva et al., 2021). Similarly, since the two types of information share the same representational space in the second stage, we use an FFN to nonlinearly transform the target representation to ensure that it remains unaffected by the source information in the representational subspace of the target language.

3.2 Instruction-level Contrastive Learning

Although the first stage aligns the representation with the target language, the source representation potentially tends to degrade towards the source language in the second stage because Equation 1 does not supervise source tokens[5]; and the second stage naturally focuses on source features.

Contrastive learning, which is a technique to softly encourage the representation towards the target states (Jaiswal et al., 2021), is helpful to mitigate this degradation. However, there are two challenges in this optimization process. The first challenge is the lack of optimization targets for representation transfer. For instance, a translation from German to English cannot be considered an anchor for a translation from French to English because neither adequately represents the optimal state of English. The second challenge is the alignment, because of the lack of token correspondence between different translations. Although using averaged pooling of sentences to obtain rough sentence representations (Pan et al., 2021) can act as proxies for alignment, this potentially leads to suboptimal results.

[5]Although the language modeling task (Radford et al., 2018) does provide supervision for source tokens, supervising source tokens does not substantially benefit MNMT (Gao et al., 2022), which may be attributed to insufficient parameters and insufficient training data in the MNMT task.

In this work, we propose Instruction-level Contrastive Learning (InstruCL), which only aligns the instruction of each instance, for effective constraints because MNMT remains sensitive to the instruction (Wu et al., 2021). Moreover, as shown in Figure 4, we suggest using the identity pair, which is established by translating the target sentence to itself and belongs exclusively to the target language, as the positive instance in InstruCL because the identity pair serves as a proxy for the optimal state of the target language (Qu et al., 2024). Specifically, we collect the representation of $l_{\boldsymbol{y}}$, i.e., $\boldsymbol{h}_1$, from $\mathbf{H}$. In a training batch, we then have a set of representations $\mathbb{B} = \{\boldsymbol{h}_1^1, \boldsymbol{h}_1^2, \dots\}$. As illustrated in Figure 4, first, we designate one instance of $\mathbb{B}$ as the anchor, denoted by $\boldsymbol{h}^{\mathrm{anc}}$. Other instances are treated as negative instances, which meet one or both of the following features compared to the anchor: different target languages or unparallel semantics. Subsequently, we use the target sentence of the anchor to establish the identity pair and pass it into the model to obtain its representation at the same layer, denoted by $\boldsymbol{h}^{\mathrm{pos}}$. The objective of InstruCL is formulated as:

$$\begin{aligned} \mathcal{L}_{\mathrm{ctr}} &= -\sum_{\boldsymbol{h}\in\mathbb{B}} \log \frac{\exp(s^{+})}{\exp(s^{+}) + \sum_{i=1}^{|\mathbb{B}|-1}\exp(s_i^{-})}, \\ s^{+} &= \mathrm{sim}(\boldsymbol{h}^{\mathrm{anc}}, \boldsymbol{h}^{\mathrm{pos}}), \\ s_i^{-} &= \mathrm{sim}(\boldsymbol{h}^{\mathrm{anc}}, \boldsymbol{h}_1^{i}), \boldsymbol{h}_1^{i} \neq \boldsymbol{h}^{\mathrm{anc}}, \end{aligned} \tag{8}$$

where $\mathrm{sim}(\cdot)$ calculates the similarity of representations using the cosine similarity. The final training objective is simply jointed as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{ce}} + \mathcal{L}_{\mathrm{ctr}}. \tag{9}$$

4 Experiments

4.1 Setups

Datasets We use English-centric datasets in our experiments, i.e., the training and validation data comprising translation pairs translation pairs translating from English and translating to English. We conduct our experiments with two datasets[6]. The first set is TED-19, which is a sub-collection of TED Talks (Ye et al., 2018) consisting of 6.5 million instances and 19 languages belonging to various language families, resulting in 32 supervised translation pairs and 306 zero-shot translation pairs. The second set is the revised version of OPUS-100

[6]Appendix E lists the information of datasets in detail.

(Zhang et al., 2020a; Yang et al., 2021), which consists of 95 languages and 92 million instances in total. However, the zero-shot translation of OPUS-100 only involves six languages, i.e., 30 pairs. Generally, each pair of validation and test sets in those two datasets contains 2,000 instances, but several pairs of OPUS-100 have fewer instances.

Evaluations We set beam size to 4 in inference, and evaluate the inference quality by four automatic evaluation metrics as follows: 1) BLEU (Papineni et al., 2002; Post, 2018) measures the overlap between inferences and references at the lexical level. 2) chrF++ (Popović, 2015, 2017) measures overlap at the character level and additionally considers a balance between precision and recall in its evaluation. 3) BERTScore (Zhang et al., 2020b) measures the similarity between inferences and references at the representation level.[7] 4) COMET (Rei et al., 2020) additionally considers the source text at the representation level for higher semantic relevance.[8] In addition, we employ *fasttext-langdetect*[9] to measure the target-off ratio on zero-shot pairs, i.e., the ratio that the source sentence does not translate to the correct target language, as a secondary metric.

Model settings of training from scratch Our model conforms to the manner of the Transformer (Vaswani et al., 2017). We have different settings[10] for two datasets. For TED-19, we set N to 6, d to 512, inner size of FFN to $4d$ for models trained on TED-19. Thus, the model with an encoder-decoder architecture has 70 million parameters, while the model with a decoder-only architecture has 63 million parameters. Moreover, the FFN in the adaptation module matches the dimensions of the FFN in the main part, so in this case, the model has 67 million parameters. For OPUS-100, we first increase N to 12, resulting in parameter counts of 121 million, 108 million, and 113 million, respectively. We also consider a wider model where N is 6 and d is 1024, resulting in parameter counts of 242 million, 217 million, and 234 million, respectively. Additionally, we consistently set $M = N$ and the layer index of InstruCL as $1.5N$ in the main experiments to ensure comparability across different architectures.

[7] In BERTScore, en is computed by *xlmr.large* (Conneau et al., 2019; Goyal et al., 2021) and other languages are computed by *bert-base-multilingual-cased* (Devlin et al., 2018).

[8] All COMET scores are computed by *Unbabel/wmt22-comet-da* (Rei et al., 2022).

[9] `https://pypi.org/project/fasttext-langdetect`

[10] Appendix F introduces the experimental settings in detail.

Model settings of fine-tuning We conduct fine-tuning experiments on TED-19 solely. Since pre-trained models in MNMT are mainly based on the encoder-decoder architecture, we train a model with parameters initialized from the decoder. We also froze the embedding layer in training. Our experiments include three pre-trained models: 1) M2M-418M (Fan et al., 2020), which has 12 decoder layers. so we set N to 6, resulting in parameter counts of 307 million, 282 million, and 299 million, respectively. 2) NLLB-600M (Team et al., 2022), which has the same configuration as M2M-418M but with a larger vocabulary size, leading to parameter counts of 439 million, 413 million, and 430 million, respectively. 3) M2M-1.2B (Fan et al., 2020), which has 24 decoder layers and a larger inner size of FFN compared to M2M-418M. We set N to 12, leading to parameter counts of 685 million, 635 million, and 668 million, respectively.

4.2 Results: Training from scratch

Table 1 shows the experimental results. The comparison between the basic architectures shows that, first, the prefix decoder-only consistently outperforms the causal decoder-only, which aligns with Raffel et al. (2023). Second, the decoder-only architecture consistently underperforms the encoder-decoder architecture in supervised pairs of all three settings, with maximum deficits of -4.17, -5.78, -1.14, and -5.16 on the BLEU, chrF++, BERTScore, and COMET respectively. On the other hand, while the decoder-only architecture shows weaker performance on TED-19 for zero-shot translation, it achieves higher scores in two settings on OPUS-100. This suggests that the zero-shot capability of the decoder-only architecture in MNMT relates to the amount of data and parameters.

In comparison with the encoder-decoder architecture, TDO, firstly, achieves competitively supervised capabilities using fewer parameters, and, specifically, TDO is slightly stronger when translating to en and slightly weaker when translating from en. Secondly, our method exhibits stronger zero-shot translation scores, achieving scores improvements of +2.49, +3.22, +1.57, and +4.81; +3.39, +6.99, +1.88, and +0.31; +2.41, +5.16, +0.76, +1.79 across three settings for the four main metrics respectively. We also find that the Adaptation module enhances both supervised and zero-shot translation performance.[11] On the other hand,

[11] Appendix G shows the improvement is not because of increased parameters.

					BLEU ↑			chrF++ ↑			BERTScore ↑			COMET ↑			off ↓
		Pref.	Adap.	CL	en→	→en	zero	en→	→en	zero	en→	→en	zero	en→	→en	zero	zero
TED N=6 d =512	Enc-Dec				25.46	28.31	12.32	45.96	50.86	32.13	84.10	93.37	78.03	80.49	78.15	67.26	3.82
	Dec-only				22.54	24.14	7.33	42.84	45.08	23.36	82.96	92.31	74.38	76.60	72.99	57.50	6.01
	Dec-only	✓			24.00	26.97	8.18	44.49	48.93	25.35	83.54	92.97	74.52	78.46	76.10	56.74	5.51
	TDO				25.47	28.88	13.56	45.98	51.33	34.04	84.11	93.45	78.90	80.41	78.42	69.74	3.54
	TDO		✓		25.55	**28.98**	13.61	46.03	**51.49**	34.11	84.15	93.50	78.94	80.56	**78.65**	70.09	3.49
	TDO			✓	25.37	28.46	13.95	45.99	51.13	34.41	84.09	93.40	79.15	80.35	78.26	70.43	3.45
	TDO		✓	✓	25.60	28.82	14.16	46.11	51.35	34.76	84.13	93.45	79.29	80.52	78.47	70.98	3.43
	TDO	✓			25.53	28.76	14.26	46.01	51.09	34.72	84.13	93.41	79.27	80.43	78.18	70.82	3.43
	TDO	✓	✓		25.61	28.52	14.51	46.04	50.89	35.01	**84.16**	93.40	79.41	80.60	78.16	71.48	3.49
	TDO	✓		✓	**25.62**	28.94	14.70	**46.15**	51.46	35.34	84.15	**93.47**	79.57	80.55	78.55	71.94	**3.39**
	TDO	✓	✓	✓	25.61	28.66	**14.81**	46.05	51.01	**35.35**	**84.16**	93.41	**79.60**	**80.61**	78.22	**72.07**	3.42
OPUS N=12 d =512	Enc-Dec				**25.18**	29.79	5.13	**44.75**	48.40	12.95	**82.98**	92.33	72.44	**76.59**	76.21	58.51	64.21
	Dec-only				23.09	26.80	5.42	42.18	45.05	13.55	82.19	91.72	72.48	74.66	73.65	58.17	60.22
	Dec-only	✓			23.96	28.41	6.62	42.98	47.22	15.36	82.47	92.06	73.57	75.48	75.34	**59.56**	58.91
	TDO	✓			24.88	**29.97**	5.32	44.72	**49.39**	13.29	82.91	**92.41**	72.50	76.26	**76.73**	58.30	51.56
	TDO	✓	✓		24.79	29.22	5.97	44.69	48.35	14.30	82.87	92.34	72.97	76.04	76.25	58.33	53.80
	TDO	✓		✓	24.35	29.52	7.93	44.44	48.74	18.65	82.84	92.37	73.97	75.93	76.23	58.71	48.37
	TDO	✓	✓	✓	24.73	29.70	**8.52**	44.60	48.72	**19.94**	82.90	92.38	**74.32**	76.16	76.59	58.82	**43.38**
OPUS N=6 d =1024	Enc-Dec				**27.71**	31.60	6.95	46.84	50.31	15.89	83.55	92.62	74.12	**78.10**	77.58	59.99	57.15
	Dec-only				26.09	29.09	7.55	44.51	47.44	16.98	82.93	92.12	73.94	76.77	75.80	61.21	63.80
	Dec-only	✓			26.79	30.42	8.15	45.48	48.92	17.65	83.21	92.37	74.17	77.53	76.69	**62.32**	55.67
	TDO	✓			27.22	31.58	7.06	46.54	**50.59**	15.96	83.44	92.64	73.78	77.68	**77.89**	60.60	52.43
	TDO	✓	✓		27.51	**31.64**	7.70	**46.87**	50.39	17.32	**83.58**	92.58	74.32	78.05	77.58	61.24	49.87
	TDO	✓		✓	27.12	31.49	9.28	46.55	50.23	21.33	83.50	**92.65**	**75.04**	77.63	77.64	60.84	**39.71**
	TDO	✓	✓	✓	27.45	31.36	**9.36**	46.79	50.06	**21.05**	83.52	92.64	74.88	77.97	77.75	61.78	43.36

Table 1: Averaged scores of results in the experiments of training from scratch. Enc-Dec and Dec-only are abbreviations of encoder-decoder and decoder-only, respectively. Pref., Adap., and Cl abbreviates Prefix, Adaption and InstruCL, respectively. ✓in the Prefix column means the masked self-attention mechanism follows Prefix manner, conversely, follows Causal manner. en→ and →en means the supervised pairs translating from English to non-central languages and translating from non-central languages to English, respectively. zero abbreviates zero-shot pairs, off abbreviates the target-off ratio. The best score in each column and block is in bold.

InstruCL significantly boosts zero-shot capability, though there is a degradation in supervised translation performance. Additionally, with the Adaptation module implemented, the degree of degradation in supervised performance is reduced.

Moreover, the prefix decoder-only architecture achieves the highest COMET score on OPUS-100, though, it remains weaker on BERTScore compared to TDO, where both two metrics are based on semantics. This phenomenon can be explained by the target-off ratio, in which models with decoder-only architecture still have a high target-off ratio with biasing towards English primarily (Chen et al., 2023) to hamper the evaluation of COMET by considering the source sentence at the same time.

4.3 Results: Fine-tuning

Table 2 shows the experimental results by fine-tuning the pre-trained models, which shows a similar tendency to Table 1 in general. First, since we initialize the model using parameters from the decoder, the training processes for the encoder-decoder, decoder-only, and TDO architectures are relatively fair. Thus, we can conclude that, when compared with the decoder-only architecture, the proposed TDO architecture supports an efficient transformation from pre-trained encoder-decoder models. Secondly, when compared with the encoder-decoder models, TDO models achieve the highest scores across four metrics, reaching up to +0.39, +0.48, +0.10, and +0.31 for pairs translating to en, up to +0.82, +1.00, +0.14, and +0.52 for pairs translating from en, and up to +0.47, +0.96, +0.29, and +0.88 for zero-shot pairs. Moreover, we observe that InstruCL does not show significant improvements in the case of NLLB-600M, whereas it remains effective in the two M2M cases. This may be attributed to that NLLB supports 205 languages, compared to 100 languages of M2M, implying a denser representational space that affects the effectiveness of InstruCL in aligning representations across languages. Additionally, the frozen embedding layer also potentially restricts the alignment.

5 Discussion

5.1 Representation Analysis

The limitation of the decoder-only architecture in MNMT is due to the lack of language transfer, which is shown in Figure 1b. To verify whether

		BLEU ↑			chrF++ ↑			BERTScore ↑			COMET ↑		
		en→	→en	zero	en→	→en	zero	en→	→en	zero	en→	→en	zero
M2M 418M	Enc-Dec	26.59	31.62	15.73	46.79	54.07	36.25	84.48	94.02	80.12	82.39	81.30	75.11
	Dec-only	25.72	30.06	14.67	45.88	52.52	34.51	84.12	93.70	79.45	81.61	79.89	73.33
	TDO	26.63	**32.44**	15.96	46.90	54.80	36.56	84.49	94.15	80.28	82.31	81.80	75.45
	+Adap.	**26.87**	31.93	16.12	**47.08**	54.21	36.73	**84.58**	94.08	80.35	**82.62**	81.54	75.80
	+CL	26.61	32.34	16.01	47.03	**55.07**	**36.87**	84.51	**94.16**	80.37	82.29	**81.82**	75.70
	+Adap.,+CL	26.75	31.83	**16.20**	46.98	54.09	36.82	84.56	94.07	**80.41**	82.56	81.52	**75.95**
NLLB 600M	Enc-Dec	26.39	32.04	15.44	46.90	54.51	36.09	84.46	94.07	79.96	81.98	81.16	74.05
	Dec-only	26.35	30.20	14.69	46.36	51.96	34.16	84.35	93.72	79.45	**82.20**	79.94	73.62
	TDO	25.82	32.15	15.48	46.42	54.76	36.35	84.30	94.10	80.09	81.34	81.28	74.17
	+Adap.	**26.60**	**32.47**	15.82	**47.04**	**54.83**	**36.62**	**84.54**	**94.15**	80.23	82.08	**81.48**	74.89
	+CL	25.87	32.29	15.48	46.44	54.71	36.21	84.31	94.11	80.09	81.43	81.27	74.18
	+Adap.,+CL	26.58	32.37	**15.85**	46.94	54.69	36.52	84.52	94.14	**80.24**	82.12	81.44	**74.93**
M2M 1.2B	Enc-Dec	27.02	31.75	16.21	47.05	53.82	36.51	84.60	94.03	80.29	82.93	81.38	76.13
	Dec-only	26.47	29.99	15.40	46.47	52.01	35.10	84.36	93.72	79.83	82.51	80.21	75.33
	TDO	27.17	**31.95**	16.45	47.37	**54.66**	37.24	84.64	**94.11**	80.48	82.96	81.71	76.47
	+Adap.	27.32	31.05	16.57	**47.53**	53.76	**37.47**	84.68	93.99	**80.56**	83.11	81.29	76.72
	+CL	27.27	31.83	16.57	47.32	54.42	37.08	84.67	**94.11**	80.54	83.04	**81.75**	76.72
	+Adap.,+CL	**27.41**	30.72	**16.60**	47.49	53.38	37.23	**84.70**	93.96	80.55	**83.24**	81.21	**76.88**

Table 2: Averaged scores of results in the experiments of fine-tuning. Abbreviations align with Table 2. Notably, the decoder-only and TDO architectures use Prefix masked self-attention only. The best score is in bold.

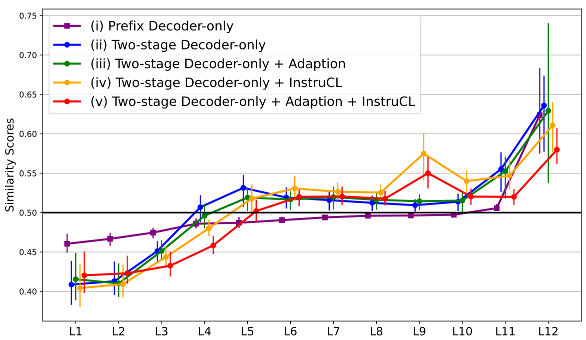
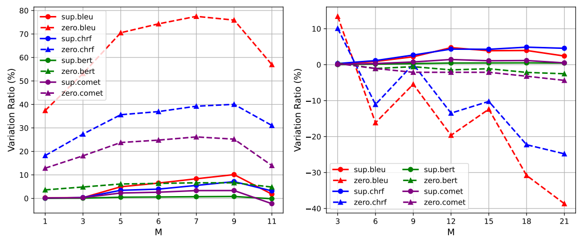


Figure 5: Illustration of linguistic preference, which follows Figure 1b. All cases in this figure use the Prefix manner for the masked self-attention mechanism. The marker of prefix decoder-only is square, and our proposed methods are round. The x-axis is the index of layers, and the vertical line indicates the value range.



(a) TED-19 (b) OPUS-100

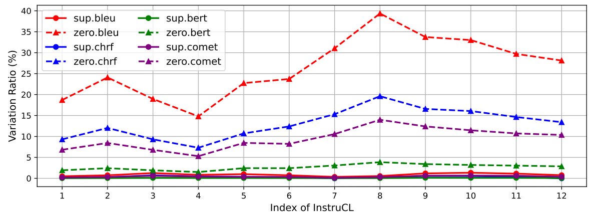
Figure 6: Variation in different values of M. The y-axis is the variation ratio compared to the performance of the model with prefix decoder-only architecture, and the x-axis is the value of M. The values of N are 6 and 12 in TED-19 and OPUS-100 respectively. Additionally, the line and the dotted line indicate supervised and zero-shot translations respectively.

our proposed methods can address this issue, we analyze the layer-wise representations of five models trained on TED-19: (i) a prefix decoder-only model with $N = 6$; (ii) a TDO model with $M = 6$; (iii) a TDO model with Adaption modules; (iv) a TDO model with InstrucCL; (v) a TDO model with Adaption modules and InstrucCL.

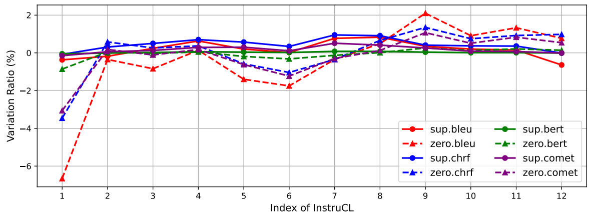
As illustrated in Figure 5, the representation of (i) only exhibits a preference for the target language in the last two layers. However, (ii) shows a preference for the target language from the fourth layer, and this trend continues into the second stage. Although (iii) exhibits a more stable layer-wise trend compared to (ii), it shows significant differences in the final output across languages. Meanwhile, (iv) exhibits smaller differences across languages. Finally, (v) incorporates all the advantages of (iii) and (iv). Therefore, we can conclude that the TDO enables better language transfer by aligning different languages in the representational subspace of the target language. Meanwhile, the Adaption module and InstrucCL improve the transferability of multilingual representations.

5.2 How to balance two stages?

In Section 4, we always set M equals N to ensure a fair comparison between the TDO and the encoder-decoder architectures. However, the balanced design is not optimal (Kasai et al., 2021; Pires et al., 2023). Thus, we test different M on TED-19 and OPUS-100 to investigate balancing two stages. As shown in Figure 6a, the performance is always improved with the increase of M on TED-19. On OPUS-100, as depicted in Figure 6b, the case with $M = 3$ achieves the best zero-shot translation scores, but there is a noticeable decline in zero-shot translation performance with



(a) Decoder-only



(b) Two-stage Decoder-only with $M = 6$

Figure 7: Variation in different layer index of InstruCL. The y-axis is the variation ratio compared to the performance of the model without InstruCL, and the x-axis is the index of the layer where employing InstruCL.

the increase of M, while supervised translation scores continue to rise.

Those results align with our expectations and experimental results that the first stage enhances language transfer. Moreover, as mentioned in Section 4.2, the decoder-only architecture scores better in zero-shot translation on OPUS-100 but exhibits a higher target-off ratio. Combining Figure 6b, we speculate that the second stage may focus more on the source language to align semantic information across languages, which is supported by Table 1 which shows a significant improvement in zero-shot translation scores of the TDO once InstruCL is applied to assist in aligning language features. Thus, we conclude that the first stage is crucial in small-scale datasets, whereas the second stage becomes more significant in large-scale datasets.

5.3 How to set layer index for InstruCL?

In Section 4, we set the layer index for InstruCL to $1.5N$ to prevent the degradation of language transfer in the second stage. Given that Section 5.2 shows the different roles of the first and second stages, we test the performance of models with different layer indexes of InstruCL for the decoder-only and the TDO models. Figure 7a demonstrates that InstruCL consistently yields positive gains for the decoder-only architecture. On the other hand, Figure 7b shows a decline in the first stage but benefits in the second stage. Moreover, in both cases, an excessively high index leads to reduced gains, which aligns with our expectations. These results indicate that InstruCL primarily affects layers that follow the decoder-only manner, namely, the second stage of TDO. This also indirectly shows that both InstruCL and the first stage enhance the alignment of multilingual representations.

6 Related Work

Research on applying the decoder-only architecture to MNMT is limited because the encoder-decoder architecture is more suitable for MNMT tasks in theory (Dabre et al., 2020; Raffel et al., 2023) and in practice (Fan et al., 2020; Team et al., 2022). Although Gao et al. (2022) exhibited that the decoder-only architecture does not have a distinct advantage in MNMT, the use of decoder-only architecture is highly appealing for MNMT, because the decoder-only architecture has been proven to have better capability in the zero-shot generalization (Radford et al., 2018; Brown et al., 2020; Wang et al., 2022), as zero-shot translation can significantly reduce the training costs of MNMT (Johnson et al., 2017; Aharoni et al., 2019; Arivazhagan et al., 2019; Gu et al., 2019; Qu and Watanabe, 2022; Chen et al., 2023). On the other hand, Kudugunta et al. (2019) pointed out that the MNMT model pairs different languages in the representational space, then Gu et al. (2019) stated that the pairing is weak. Recently, Qu et al. (2024) further demonstrated that the success of the MNMT model is because of aligning different source languages in the representational subspace of the target language, termed language transfer, by the encoder. This work proves that absenting this process limits the performance of decoder-only architecture in MNMT.

7 Conclusions

In this work, we first analyzed the reasons behind the poor performance of the decoder-only architecture in MNMT, identifying the lack of language transfer capability as the primary challenge. To address this, we introduced the Two-stage Decoder-only architecture. We also proposed Instruction-level Contrastive Learning to overcome the issue from the perspective of representation optimization. We conducted experiments on two settings, i.e., training from scratch and fine-tuning, using the TED-19 and OPUS-100 datasets, and the results validate the effectiveness of our approach. Through further representation analysis and further experiments, our study confirms that the advantages of our method are primarily derived from enhanced language transfer capabilities.

8 Limitations

This work preliminarily discussed achieving the multilingual neural machine translation (MNMT) task using a decoder-only architecture model. Although our model aligns with the standard implementation of the decoder-only architecture (Vaswani et al., 2017; Raffel et al., 2023), we train the model by the standard training objective of the MNMT task (Equation 1) rather than a language modeling task (Radford et al., 2018). Moreover, while relative position encoding (Su et al., 2023) has become de facto in the decoder-only architecture (Touvron et al., 2023), the MNMT task involves only sentence-level text. Thus, we exclusively use sinusoidal positional embeddings (Vaswani et al., 2017) to ensure a fair comparison with the encoder-decoder architecture.

9 Ethical Considerations

All datasets and toolkits used in this work are public, common, and general in the research on multilingual neural machine translation, meanwhile, the usage of those datasets and toolkits follows the license. Moreover, this work is foundational research and is not a report of specific applications. Therefore, this work is harmless and has no ethical risks.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. *Preprint*, arXiv:1903.00089.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. 2019. The missing ingredient in zero-shot neural machine translation. *Preprint*, arXiv:1903.07091.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Liang Chen, Shuming Ma, Dongdong Zhang, Furu Wei, and Baobao Chang. 2023. On the off-target problem of zero-shot multilingual neural machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9542–9558, Toronto, Canada. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Preprint*, arXiv:1905.03197.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *Preprint*, arXiv:2010.11125.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Yingbo Gao, Christian Herold, Zijian Yang, and Hermann Ney. 2022. Is encoder-decoder redundant for neural machine translation? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 562–574, Online only. 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.

Naman Goyal, Jingfei Du, Myle Ott, Giri Anantharaman, and Alexis Conneau. 2021. Larger-scale transformers for multilingual masked language modeling. *arXiv preprint arXiv:2105.00572*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1258–1268, Florence, Italy. Association for Computational Linguistics.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021. A survey on contrastive self-supervised learning. *Preprint*, arXiv:2011.00362.

Melvin Johnson, Mike Schuster, Quoc V. 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.

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

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

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.

Sneha Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual NMT representations at scale. 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)*, pages 1565–1575, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Preprint*, arXiv:2001.08210.

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.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 244–258, Online. Association for Computational Linguistics.

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

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

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

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

Zhi Qu, Chenchen Ding, and Taro Watanabe. 2024. Languages transferred within the encoder: On representation transfer in zero-shot multilingual translation. *Preprint*, arXiv:2406.08092.

Zhi Qu and Taro Watanabe. 2022. Adapting to non-centered languages for zero-shot multilingual translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5251–5265, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the*

31st International Conference on Neural Information Processing Systems, NIPS'17, page 6078–6087, Red Hook, NY, USA. Curran Associates Inc.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

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

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective work best for zero-shot generalization? *Preprint*, arXiv:2204.05832.

Liwei Wu, Shanbo Cheng, Mingxuan Wang, and Lei Li. 2021. Language tags matter for zero-shot neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3001–3007, Online. Association for Computational Linguistics.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2024. A paradigm shift in machine translation: Boosting translation performance of large language models. In *The Twelfth International Conference on Learning Representations*.

Yilin Yang, Akiko Eriguchi, Alexandre Muzio, Prasad Tadepalli, Stefan Lee, and Hany Hassan. 2021. Improving multilingual translation by representation and gradient regularization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7266–7279, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qi Ye, Sachan Devendra, Felix Matthieu, Padmanabhan Sarguna, and Neubig Graham. 2018. When and why are pre-trained word embeddings useful for neural machine translation. In *HLT-NAACL*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020a. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2023. Multilingual machine translation with large language models: Empirical results and analysis. *Preprint*, arXiv:2304.04675.

A Introduction of Illustrating Linguistic Preference

Overview In this work, we only quantify the linguistic preferences of the representations by the similarity scores, although the analysis of Qu et al. (2024) further quantified the semantic features of representations. Specifically, the score presents whether the representations at a certain state exhibit more features related to the target language or more features related to the source language.

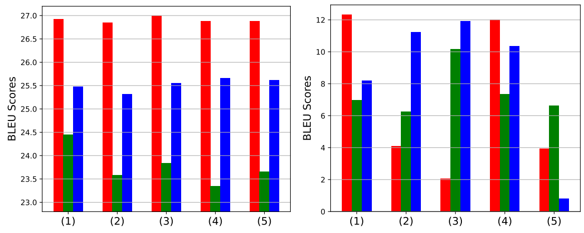
Setup First, computing the linguistic preferences of the representations requires a semantically parallel dataset. Therefore, we conduct analysis experiments on TED-19, which provides six fully parallel languages, including `ar`, `he`, `zh`, `hr`, `vi`, and `ja`. We connect these languages to generate 30 zero-shot translation pairs, each pair consisting of 967 sentences. The model setup is consistent with our main experiments (Section 4).

Computing the similarity score First, we follow the process of Qu et al. (2024) to measure representation similarity in MNMT, employing singular value canonical correlation analysis (Raghu et al., 2017). As the definition in Section 2, we obtain the token-wise hidden representations of the source sentence, i.e. $\mathbf{H}$, from a translation pair. Notably, for a decoder-only model, we cut out the source part, namely, $|\mathbf{H}|$ is always $I+1$. Then, we derive the sentence-level representation $\overline{\boldsymbol{h}}$ using average pooling $\overline{\boldsymbol{h}} = \frac{\sum_{i=1}^{q} \boldsymbol{h}_i}{q}$. Given $\mathbf{H}^a$ and $\mathbf{H}^b$ derived from two sentences, we first perform singular value decomposition on $\overline{\boldsymbol{h}}^a$ and $\overline{\boldsymbol{h}}^b$ to obtain subspace representations $\overline{\boldsymbol{h}}^{a} \in \mathbb{R}^{d^a}$ and $\overline{\boldsymbol{h}}^{b} \in \mathbb{R}^{d^b}$. Then we perform canonical correlation analysis to determine $\mathbf{W}^a \in \mathbb{R}^{d' \times d^a}$ and $\mathbf{W}^b \in \mathbb{R}^{d' \times d^b}$. Formally, we compute correlation ρ between $\overline{\boldsymbol{h}}^a$ and $\overline{\boldsymbol{h}}^b$ as

$$\rho = \frac{\langle \mathbf{W}^a \overline{\boldsymbol{h}}^a, \mathbf{W}^b \overline{\boldsymbol{h}}^b \rangle}{\|\mathbf{W}^a \overline{\boldsymbol{h}}^a\| \|\mathbf{W}^b \overline{\boldsymbol{h}}^b\|}, \tag{10}$$

where $\langle \cdot, \cdot \rangle$ indicates the inner product. We use ρ to represent the similarity of two sentences. Subsequently, we get the similarity $\rho_{\boldsymbol{x}}$ between $(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y})$ and $(l_{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{x})$ and the similarity $\rho_{\boldsymbol{y}}$ between $(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y})$ and $(l_{\boldsymbol{y}}, \boldsymbol{y}, \boldsymbol{y})$, respectively. Therefore, a similarity score of linguistic preference is computed as follows:

$$s_{(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y})} = \frac{\rho_{\boldsymbol{y}}}{\rho_{\boldsymbol{y}} + \rho_{\boldsymbol{x}}}, \tag{11}$$



(a) supervised (b) zero-shot

Figure 8: Averaged BLEU scores in different architectures. The palette follows Figure 1, i.e., red is encoder-decoder, green is causal decoder-only, and blue is prefix decoder-only.

where $s_{(l_{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y})}$ is the similarity score for the given translation pair. Finally, we compute the set-level score by taking the average scores of all sentences over the test set.

Meaning of the similarity score Equation 11 simply compares the importance of source information and target information in the representation. Therefore, a value higher than 0.5 means the representation prefers the target language, otherwise the representation prefers the source language. Moreover, the value reflects the degree of linguistic preference, for example, compared to 0.6, 0.7 means the representation presents much more features of the target language or fewer features of the source language. In addition, we also denote the highest and lowest values by the vertical lines on each point in Figures 1b and 5 to show the value range, which can present stability. Finally, we can find that models with decoder-only architecture cannot align the representation of the source tokens in the representational subspace of the target language, and they try to align source and target languages to be a language-agnostic state.

B Comparison between Different Instruction Strategies in MNMT

MNMT is sensitive to the strategy of translation instruction (Wu et al., 2021). We summarize the possible strategies as follows: (1) Adding a language tag specified to the target language at the beginning of source tokens; (2) Adding a language tag specified to the target language at the beginning of target tokens; (3) Based on the (2), using the language tag to replace the [eos] token, which is used to be the trigger of inference; (4) Adding two language tag specified to the target language at the beginning of source tokens and the beginning of target tokens, simultaneously; (5) Adding a language tag specified

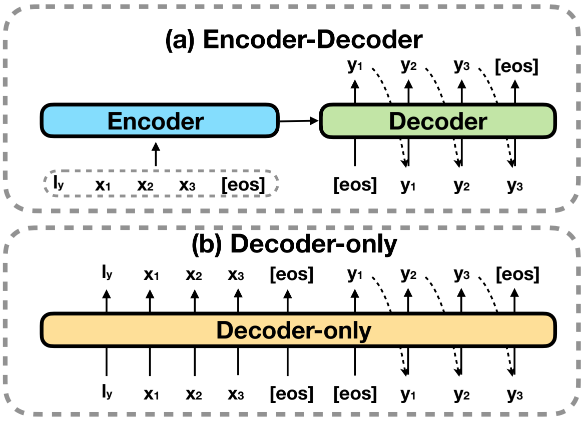


Figure 9: Illustration of different input forms. [eos] is a special token, which means the end of a sentence.

to the source language and a language tag specified to the target language at the beginning of source tokens and target tokens, respectively. Then, we conduct preliminary experiments on three architectures: encoder-decoder, causal decoder-only, and prefix decoder-only, to support the validity of using approach (1). As shown in Figure 8, the performance of encoder-decoder architecture meets the analysis of Wu et al. (2021). However, a language tag at the beginning of target tokens, i.e., (2), (3), and (4), is more beneficial for the zero-shot capability in Decoder-only architecture. Considering that (1) also benefits decoder-only architectures in the supervised translation, using (1) in this work is reasonable.

C Different Input Forms

Figure 9 illustrates different input forms for two architectures involved in this work. Initially, within the encoder-decoder architecture, the encoder receives parallel input from source tokens, including $l_{\boldsymbol{y}}$, $\boldsymbol{x}$, and [eos]. The decoder's input, however, is shifted. Specifically, in training, [eos] is placed at the beginning of the target tokens, and the output at each position always points to the token in the next position; in inference, [eos] serves as the trigger, and the model would generate the next token step by step until the predicted token is [eos]. On the other hand, the decoder-only architecture combines source tokens and target tokens. In this work, we only supervise the target tokens.

D Estimation of Parameters

We follow the notation in Section 4.1, that is, d is the dimension of the model and the inner size of FFN is $4d$. Therefore, each attention mechanism has $4d^2$ parameters because there are 4 matrices with dimensions of $d \times d$, and each FFN has $8d^2$ parameters (Vaswani et al., 2017). Then, all layers have the structure illustrated in Figure 2. Given N =1, the model with encoder-decoder architecture has $28d^2$ parameters and the model with Decoder-only architecture has $24d^2$ parameters. Thus, considering the fixed parameters of normalization modules and embedding layer, Decoder-only architecture is implemented with around 10% fewer parameters than encoder-decoder architecture.

E Detailed Information of Datasets

First of all, the language code in our descriptions follows ISO 639-1, referring to `https://www.loc.gov/standards/iso639-2/php/code_list.php`. We list the detailed information of TED-19 in Table 4, and of OPUS-100 in Table 5. Although Yang et al. (2021) has removed the repetition in the original version of OPUS-100 (Zhang et al., 2020a), we further remove noisy instances that only contain nonsense characters. Moreover, the zero-shot translation of OPUS-100 in this work only involves six languages, including `ar`, `nl`, `de`, `zh`, `ru`, and `fr`. Finally, we employ SentencePiece (Kudo and Richardson, 2018) to get the vocabulary for training, specifically, the vocabulary size is set to 50,000 of TED-19 and 64,000 of OPUS-100.

F Detailed Model Settings

We implement models by Fairseq (Ott et al., 2019), which is an open-source toolkit. First of all, in this work, we apply independent sinusoidal positional embeddings for source tokens and target tokens (Vaswani et al., 2017) for the input of the decoder-only architecture. In the case of training from scratch on TED-19, we set the learning rate to 0.0005 and the model is trained for 30 epochs on eight Nvidia V100 GPUs with a batch size of 4,000 per GPU to ensure full convergence. Moreover, we set the head number of the attention mechanism to 8, the dropout rate to 0.1, label smoothing to 0.1, and weight decay to 0.0001. We also employ Adam (Kingma and Ba, 2017) as our optimizer and set *share-all-embeddings* of Fairseq. We evaluate by averaging the top-5 best checkpoints selected based on validation loss. In the case of training from scratch on OPUS-100 with N =12, we set the number of gradient accumulation steps to 16 to in-

	d	d^1_{ffn}	d^2_{ffn}	en→	→en	zero
TDO+adapt.	512	2048	2048	**25.61**	28.52	**14.51**
TDO	544	2048	2048	25.55	28.28	14.22
	512	2432	2432	25.51	28.51	14.31
	512	2048	2816	25.32	27.98	13.89
	512	2816	2048	25.56	**28.95**	14.01

Table 3: Averaged BLEU scores of models with TDO architecture trained on TED-19. Abbreviations in this table follow Table 1. In addition, d^1_{ffn} is the inner size of FFN in the first stage, and d^2_{ffn} is in the second stage. The best score is in bold.

crease the batch size and train for 50,000 steps with a learning rate of 0.0007. For another setting of OPUS-100 with N =6, the d is increased to 1024, and the head number of the attention mechanism is 16. Therefore, we additionally set an attention dropout to 0.05. Moreover, we reduce the batch size per GPU to 2,000, set the number of gradient accumulation steps to 32, and train for 100,000 steps due to GPU memory constraints. For two cases of OPUS-100, we test the checkpoint with the best validation loss. Additionally, in training on OPUS-100, we set *encoder-normalize-before* and *decoder-normalize-before* in Fairseq and reduce the weight decay to 0, which lead to a quick convergence in a complex data condition (Liu et al., 2020; Fan et al., 2020; Team et al., 2022).

In the model settings of fine-tuning, M2M-418M has 12 layers for encoder and decoder, respectively. d of M2M-418M is 1024, the inner size of FFN is 4096, the label smoothing is 0.2, the dropout is 0.3, the attention dropout is 0.05, and the batch size and the learning rate keep the settings of training from scratch. However, we reduce the batch size to 2000 and set gradient accumulation to 2 for NLLB-600M because of the GPU memory constraints. In M2M-1.2B, our experiments are conducted on four NVIDIA A6000 GPUs, and we set gradient accumulation to 2. We also reduce the learning rate to 0.0002 and the number of training epochs to 10 because of more parameters.

G Experiments with Different Parameters

To verify the improvement brought by Adaption modules is not because of increased parameters, we run experiments with models that have different dimensions. We can find that models, which are shown in Table 3, have similar parameters. Therefore, the result of this table can prove our statement.

Code	Language	Family	Sub-Family	#Train	Code	Language	Family	Sub-Family	#Train
es	Spanish	Indo-European	Romance	196026	ar	Arabic	Afro-Asiatic	Semitic	214111
fr	French	Indo-European	Romance	192304	he	Hebrew	Afro-Asiatic	Semitic	211819
ro	Romanian	Indo-European	Romance	180484	ru	Russian	Indo-European	Slavic	208458
nl	Dutch	Indo-European	Germanic	183767	ko	Korean	Koreanic		205640
de	German	Indo-European	Germanic	167888	it	Italian	Indo-European	Romance	204503
pl	Polish	Indo-European	Slavic	176169	ja	Japanese	Japonic		204090
hr	Croatian	Indo-European	Slavic	122091	zh	Chinese	Sino-Tibetan	Sinitic	199855
cs	Czech	Indo-European	Slavic	103093	tr	Turkish	Turkic		182470
fa	Persian	Indo-European	Iranian	150965	vi	Vietnamese	Austroasiatic	Vietic	171995

Table 4: Detailed information of TED-19 datasets. #Train indicates the number of training instances.

Code	Language	Family	Sub-Family	#Train	Code	Language	Family	Sub-Family	#Train
fa	Persian	Indo-European	Iranian	934413	yi	Yiddish	Indo-European	Romance	1865
bn	Bengali	Indo-European	Iranian	724719	ga	Irish	Indo-European	Celtic	187967
ur	Urdu	Indo-European	Iranian	724226	br	Breton	Indo-European	Celtic	96951
si	Sinhala	Indo-European	Iranian	613702	cy	Welsh	Indo-European	Celtic	92615
hi	Hindi	Indo-European	Iranian	374472	gd	Scottish Gaelic	Indo-European	Celtic	11104
tg	Tajik	Indo-European	Iranian	183216	lt	Lithuanian	Indo-European	Baltic	797693
ne	Nepali	Indo-European	Iranian	144520	lv	Latvian	Indo-European	Baltic	779972
gu	Gujarati	Indo-European	Iranian	108564	tr	Turkish	Turkic		918838
ku	Kurdish	Indo-European	Iranian	107110	az	Azerbaijani	Turkic		237533
pa	Punjabi	Indo-European	Iranian	72160	uz	Uzbek	Turkic		148319
as	Assamese	Indo-European	Iranian	58009	tt	Tatar	Turkic		97746
mr	Marathi	Indo-European	Iranian	26117	ug	Uyghur	Turkic		71241
ps	Pashto	Indo-European	Iranian	14254	kk	Kazakh	Turkic		62227
or	Oriya	Indo-European	Iranian	13410	ky	Kyrgyz	Turkic		12724
de	German	Indo-European	Germanic	968252	tk	Turkmen	Turkic		98
nl	Dutch	Indo-European	Germanic	936611	ar	Arabic	Afro-Asiatic	Semitic	959868
sv	Swedish	Indo-European	Germanic	916259	he	Hebrew	Afro-Asiatic	Semitic	913493
no	Norwegian	Indo-European	Germanic	914187	mt	Maltese	Afro-Asiatic	Semitic	672134
da	Danish	Indo-European	Germanic	911156	ha	Hausa	Afro-Asiatic	Chadic	91869
is	Icelandic	Indo-European	Germanic	813820	am	Amharic	Afro-Asiatic	Semitic	64369
nn	Norwegian Nynorsk	Indo-European	Germanic	172187	el	Greek	Indo-European	Hellenic	932811
af	Afrikaans	Indo-European	Germanic	146600	sq	Albanian	Indo-European	Albanian	855095
nb	Norwegian Bokmål	Indo-European	Germanic	128374	ml	Malayalam	Dravidian		633920
fy	Frisian	Indo-European	Germanic	42372	ta	Tamil	Dravidian		184699
li	Limburgish	Indo-European	Germanic	3331	te	Telugu	Dravidian		37792
ru	Russian	Indo-European	Slavic	951611	kn	Kannada	Dravidian		13777
sr	Serbian	Indo-European	Slavic	935342	xh	Xhosa	Niger-Congo	Bantu	231708
hr	Croatian	Indo-European	Slavic	927541	rw	Kinyarwanda	Niger-Congo	Bantu	62159
pl	Polish	Indo-European	Slavic	926940	zu	Zulu	Niger-Congo	Bantu	6834
bg	Bulgarian	Indo-European	Slavic	925647	ig	Igbo	Niger-Congo	Volta-Niger	691
cs	Czech	Indo-European	Slavic	924282	fi	Finnish	Uralic	Finnic	938601
bs	Bosnian	Indo-European	Slavic	921232	et	Estonian	Uralic	Finnic	893074
sl	Slovenian	Indo-European	Slavic	912248	hu	Hungarian	Uralic	Finno-Ugric	920592
mk	Macedonian	Indo-European	Slavic	881176	se	Northern Sami	Uralic	Sami	32289
sk	Slovak	Indo-European	Slavic	878540	vi	Vietnamese	Austroasiatic	Vietic	883581
uk	Ukrainian	Indo-European	Slavic	759826	id	Indonesian	Austronesian	Malayo-Polynesian	881198
sh	Serbo-Croatian	Indo-European	Slavic	209379	ms	Malay	Austronesian	Malayo-Polynesian	819431
be	Belarusian	Indo-European	Slavic	61862	mg	Malagasy	Austronesian	Malayo-Polynesian	292520
fr	French	Indo-European	Romance	963140	km	Khmer	Austroasiatic	Khmeric	101294
es	Spanish	Indo-European	Romance	929677	zh	Chinese	Sino-Tibetan	Sinitic	954358
it	Italian	Indo-European	Romance	928427	my	Burmese	Sino-Tibetan	Lolo-Burmese	5326
pt	Portuguese	Indo-European	Romance	919755	th	Thai	Kra-Dai	Tai	892433
ro	Romanian	Indo-European	Romance	913451	ko	Korean	Koreanic		892064
ca	Catalan	Indo-European	Romance	633826	ja	Japanese	Japonic		886850
gl	Galician	Indo-European	Romance	353596	eu	Basque	Language isolate		786645
wa	Walloon	Indo-European	Romance	48894	eo	Esperanto	Constructed		257560
oc	Occitan	Indo-European	Romance	27773	ka	Georgian	Kartvelian		240335

Table 5: Detailed information of OPUS-100 datasets. #Train indicates the number of training instances.