Flow-Adapter Architecture for Unsupervised Machine Translation

Yihong Liu[1], **Haris Jabbar**[2] and **Hinrich Schütze**[2]
[1]Department of Informatics, Technical University of Munich
[2]Center for Information and Language Processing, LMU Munich
yihong.liu@tum.de
jabbar@cis.lmu.de

Abstract

In this work, we propose a *flow-adapter architecture* for unsupervised NMT. It leverages normalizing flows to explicitly model the distributions of sentence-level latent representations, which are subsequently used in conjunction with the attention mechanism for the translation task. The primary novelties of our model are: (a) capturing language-specific sentence representations separately for each language using normalizing flows and (b) using a simple transformation of these latent representations for translating from one language to another. This architecture allows for unsupervised training of each language independently. While there is prior work on latent variables for supervised MT, to the best of our knowledge, this is the first work that uses latent variables and normalizing flows for unsupervised MT. We obtain competitive results on several unsupervised MT benchmarks.

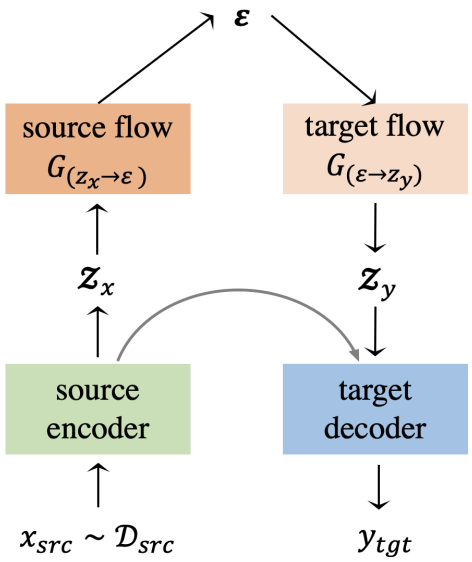

Figure 1: Inference pipeline of proposed flow-adapter based model for source-to-target translation. The decoder also uses the attentional input (shown as the gray arrow between the encoder and the decoder).

1 Introduction

Recent advances in deep learning have boosted the development of neural machine translation (NMT). Typical NMT models leverage an encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014). However, NMT models have been shown to be data-hungry, as the number of parallel sentences significantly influences the performance (Zoph et al., 2016). Unfortunately, large-scale bilingual corpora are limited to a relatively small subset of languages (Al-Onaizan et al., 2002). In contrast to bilingual corpora, monolingual corpora are much easier to obtain.

Unsupervised NMT, compared with supervised NMT, aims to train a model without parallel data. Some early works (Irvine and Callison-Burch, 2016; Sennrich et al., 2016b; Cheng et al., 2016) used monolingual corpora to boost performance when parallel data is not abundant. Lample et al. (2018a) and Artetxe et al. (2018) explored the possibility of training a model relying only on monolingual corpora. They both leveraged a shared-encoder architecture in order to generate universal representations, trained with techniques such as initial word-by-word translation through bilingual dictionaries (Lample et al., 2018b; Artetxe et al., 2017), denoising auto-encoding (DAE) (Vincent et al., 2008) and iterative back-translation (BT) (Hoang et al., 2018). However, Yang et al. (2018) argued that it is a bottleneck in such shared-encoder models to use a shared encoder that maps pairs of sentences of different languages to the same shared latent space. They proposed to use two independent encoders sharing part of their weights and achieved better results. But all of those aforementioned approaches trained the translation models almost from scratch (with only some prior knowledge in the pre-trained embeddings) and therefore it is hard to further advance their performance.

More recently, with the advance in pre-trained

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1253 - 1266

models (Peters et al., 2018; Devlin et al., 2019), researchers have begun to explore the possibility of using pre-trained models for unsupervised NMT. Conneau and Lample (2019) extended the pre-training from a single language to multiple languages, referred to as cross-lingual pre-training. By using pre-trained cross-language models (XLMs) to initialize encoder and decoder, they achieved good unsupervised MT performance on multiple language pairs. In related work, Song et al. (2019) proposed masked sequence to sequence pre-training (MASS), which directly pre-trains a whole encoder-decoder model. Üstün et al. (2021) proposed a language-specific denoising-adapter architecture to increase the multilingual modeling capacity of the pre-trained model mBART (Liu et al., 2020) and used these adapters for multilingual unsupervised NMT. Although these adapters are trained with monolingual data only, the fine-tuning step relies on parallel data.

Current NMT frameworks rely heavily on the attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017) to capture alignments. However, attention-based context vectors can fail to extract sufficiently accurate sentence-level semantics and thus result in incorrect translations or translation ambiguity (Tu et al., 2016; Zhang et al., 2016). To tackle this issue, several variational frameworks for modeling the translation process have been proposed (Zhang et al., 2016; Eikema and Aziz, 2019; Setiawan et al., 2020). These approaches incorporate sentence-level latent representations into NMT. A latent representation, in the context of this paper, is a fixed-size continuous vector from an unknown distribution that captures the semantics of a source sentence. The target sentence is then generated from this latent representation using a simple transformation along with the attention mechanism commonly found in transformer architectures. In this way, when the attention mechanism learns incorrect alignments, the latent representation plays a complementary role in guiding the translation.

Prior work in this vein has only been conducted in supervised NMT. In this paper, we propose a flow-adapter architecture for unsupervised NMT. Similar to variational methods, we model the distribution of sentence-level representations. However, unlike variational methods, which model the distribution in an implicit way, we use a pair of normalizing flows to explicitly model the distributions of source and target languages. Secondly, different from some previous unsupervised NMT models that assume that the representations of source and target sentences share a common semantic space, we assume the representations are different because of language-specific characteristics. Hence they are modeled separately for each language. Subsequently a simple transformation converts source representations into target representations. This makes it possible to better capture sentence semantics in a language-specific manner. Lastly, instead of minimizing KL loss, the flows are directly trained by maximum likelihood estimation (MLE) of sentence-level latent representations. This gives the latent representations more flexibility.

Our main contributions:
(1) We propose a novel flow-adapter architecture. It uses normalizing flows to explicitly model the distributions of sentence-level representations and performs a latent representation transformation from source to target. To the best of our knowledge, this is the first work that uses latent variables and normalizing flows for unsupervised NMT.
(2) Experiments show the validity and effectiveness of our flow-adapter architecture. It performs very well in unsupervised NMT on several language pairs on the Multi30K dataset. When additionally using pre-trained models, we achieve results competitive with the state of the art on WMT datasets, especially for *en-fr* (WMT'14) and *en-ro* (WMT'16).

2 Background

2.1 Normalizing Flows

Normalizing flows (NFs) are a special type of deep generative model. Different from generative adversarial networks (GAN) (Goodfellow et al., 2014) and variational auto-encoding (VAE) (Kingma and Welling, 2014), NFs allow for not only sampling but also exact density estimation. Due to such desirable properties, in recent years, they have been successfully applied to fields such as image (Ho et al., 2019; Kingma and Dhariwal, 2018), audio (Esling et al., 2019; van den Oord et al., 2018) and video generation (Kumar et al., 2019). In addition to significant achievements in modeling continuous data, NFs have also been used for modeling discrete data, either by directly modeling the data in discrete space (Tran et al., 2019; Hesselink and Aziz, 2020) or by transforming the discrete data into continuous space (Ziegler and Rush, 2019; Tang et al., 2021).

NFs transform between two distributions based on the following change-of-variables formula (we follow the introduction of (Dinh et al., 2015, 2017)):

$$\log p_x(\boldsymbol{x}) = \log p_z(\boldsymbol{z}) + \log \left| \det \frac{\partial f_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}} \right|^{-1} \quad (1)$$

where $\boldsymbol{z} \sim p_z(\boldsymbol{z})$ and $\boldsymbol{x} \sim p_x(\boldsymbol{x})$ denote two vectors from a simple latent distribution $p_z(\boldsymbol{z})$ and a complex distribution of the observed data $p_x(\boldsymbol{x})$, f_θ is an invertible and differentiable function (neural network with parameters θ), $f_\theta(\boldsymbol{z}) = \boldsymbol{x}$ and $\det \frac{\partial f_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}}$ denotes the determinant of the Jacobian matrix of f_θ. The idea of NFs is to learn an f_θ such that f_θ and f_θ^{-1} transform between the latent space $p_z(\boldsymbol{z})$ and the observed space $p_x(\boldsymbol{x})$.

Constructing a single arbitrarily complex invertible and differentiable function is usually cumbersome. Therefore, a generally adopted approach is to stack multiple transformations f_i together, i.e., $\boldsymbol{x} = f_\theta(\boldsymbol{z}) = f_K \circ \cdots \circ f_1(\boldsymbol{z})$. Similarly, for the reverse direction we have $\boldsymbol{z} = f_\theta^{-1}(\boldsymbol{x}) = f_1^{-1} \circ \cdots \circ f_K^{-1}(\boldsymbol{x})$, whose Jacobian matrix is efficient to compute. Here K denotes the number of sequential flows (e.g., $K = 3$ in Table 1).

Normalizing flows are usually optimized by MLE of the parameters θ, i.e., $\log p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \log p_x(\boldsymbol{x}^{(n)}|\theta)$, where N is the data size. By applying a variant of the change-of-variable formula in Equation (1), i.e., $\log p_x(\boldsymbol{x}) = \log p_z(f_\theta^{-1}(\boldsymbol{x})) + \log \left| \det \frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|$, the MLE objective can be reformulated as follows:

$$\begin{aligned} \log p(\mathcal{D}|\theta) = & \sum_{n=1}^{N} \log p_z(f_\theta^{-1}(\boldsymbol{x}^{(n)})|\theta) \\ & + \log \left| \det \frac{\partial f_\theta^{-1}(\boldsymbol{x}^{(n)})}{\partial \boldsymbol{x}^{(n)}} |\theta \right| \end{aligned} \quad (2)$$

2.2 Latent-variable (variational) NMT

Compared with standard encoder-decoder based NMT models, latent-variable (variational) approaches (Zhang et al., 2016; Eikema and Aziz, 2019; Ma et al., 2019; Calixto et al., 2019; Setiawan et al., 2020; Shu et al., 2020) additionally leverage latent random variables.

Let $\boldsymbol{x}$ be a sentence from the source language and $\boldsymbol{y}$ be its translation in the target language. Then, the variational NMT framework introduces a continuous random latent variable $\boldsymbol{z}$ for the translation modeling, i.e., $p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})$. With the introduction of $\boldsymbol{z}$, the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ can then be reformulated as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x}) p(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} \quad (3)$$

In this way, $\boldsymbol{z}$ serves as a global semantic signal that is helpful to counteract incorrect alignments the model has learned and uses through attention. However, the integration of $\boldsymbol{z}$ poses challenges for inference. To address this problem, variational NMT adopts techniques from VAE (Kingma and Welling, 2014; Rezende et al., 2014), namely, neural approximation and the reparameterization trick.

Neural approximation leverages a neural network to approximate the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ with $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$, where ϕ denotes the parameters of the neural network. In most works, $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ is designed as a diagonal Gaussian $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, where the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$ are parameterized with neural networks.

Reparameterization means that the latent random variable $\boldsymbol{z}$ is parameterized as a function of the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$. In this way, the gradient with respect to the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ can be computed. The reparameterization of $\boldsymbol{z}$ is often carried out in a location-scale manner: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$

With these two techniques, the learning objective of variational NMT is the evidence lower-bound or ELBO of the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$:

$$\begin{aligned} \mathcal{L}(\theta, \phi; \boldsymbol{x}, \boldsymbol{y}) = & -\text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) || p_\theta(\boldsymbol{z}|\boldsymbol{x})) \\ & + E_{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})] \end{aligned} \quad (4)$$

where $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is the prior distribution modeled by a neural network and $p_\theta(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})$ is modeled by the decoder given the input source sentence $\boldsymbol{x}$ and the latent variable $\boldsymbol{z}$. The KL term minimizes the discrepancy between the prior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ and the posterior $q_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$. In the inference step, $\boldsymbol{z}$ can therefore be sampled from the prior, which only requires $\boldsymbol{x}$ instead of the posterior that requires both $\boldsymbol{x}$ and $\boldsymbol{y}$. Although this variational framework leverages latent variables, which are helpful for translation, it still has some flaws: **1)** training a variational NMT framework requires parallel corpora to construct the posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ and such parallel corpora are not available for unsupervised MT; **2)** the distribution family of the latent variables, e.g., $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, is pre-defined, e.g., a Gaussian, which might restricts the advantage of using a complex posterior; **3)** as variational NMT leverages $\boldsymbol{z}$

sampled from $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ for inference, an underlying assumption is that $\boldsymbol{z}$ should be the same whether only $\boldsymbol{x}$ is considered or both $\boldsymbol{x}$ and $\boldsymbol{y}$ are considered. In other words, this framework assumes $\boldsymbol{z}$ is language-agnostic, which might not be true since language-specific characteristics can influence the generation of $\boldsymbol{z}$.

3 Flow-Adapter Based Framework

In this work, we want to reap the benefits of introducing latent variables into unsupervised MT while at the same time avoiding the flaws of variational NMT we just discussed. Therefore, we propose a flow-adapter based framework that uses two NFs to explicitly model the distribution of the sentence-level latent representations of the source and target sentences. In this way, we can take account of multilinguality in unsupervised MT and make use of language-specific sentence-level representations. During the translation process, a latent code transformation is performed to transform the source-language representation into the target-language representation so that the decoder can leverage them to generate a better target-language sentence. We will first introduce the sentence-level representation as well as the latent code transformation in Section 3.1, followed by the description of the flow-adapter based framework for unsupervised MT in Section 3.2.

3.1 Modeling Representation by NFs & Latent Code Transformation

As previously mentioned, variational methods such as (Zhang et al., 2016; Setiawan et al., 2020) assume that the semantics of the source sentence $\boldsymbol{x}$ and target sentence $\boldsymbol{y}$ are the same and thus the generated latent variable $\boldsymbol{z}$ is the same regardless of whether we only consider $\boldsymbol{x}$ or consider both $\boldsymbol{x}$ and $\boldsymbol{y}$. Unsupervised NMT methods such as (Lample et al., 2018a; Conneau and Lample, 2019) similarly assume that a shared encoder maps source and target sentences into a shared latent space.

In this work, however, we diverge from this assumption and follow Yang et al. (2018) in adopting the desideratum that the unique and internal characteristics of each language be respected. One could think that the semantics of a pair of sentences should theoretically be the same; but in reality, because of language-specific characteristics, the latent representations $\boldsymbol{z}$ obtained by an encoder can be different for source and target sentences. Differences in vocabulary, pragmatics and other linguistic properties all influence the generation of the latent representations. Therefore, we consider the latent representations from a different perspective as follows. We can view $\boldsymbol{z}_x$ and $\boldsymbol{z}_y$ as expressions of the sentence-level representations in two distinct languages based on the same semantics $\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is truly language-agnostic. $\boldsymbol{z}_x$ and $\boldsymbol{z}_y$ are obtained by applying parameter-free techniques such as pooling to the output of the encoder fed with source and target languages (see Section 3.2 for details).

Modeling by NFs. For our unsupervised scenario, we propose to explicitly model the distributions of the sentence-level representations of both source and target sentences – i.e., $p_{\boldsymbol{z}_x}(\boldsymbol{z}_x)$ and $p_{\boldsymbol{z}_y}(\boldsymbol{z}_y)$ – using NFs with K sequential flows:

$$p_{\boldsymbol{z}_x}(\boldsymbol{z}_x) = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}) \prod_{i=1}^{K} \left| \det \frac{\partial f_x^{(i)}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{z}^{(i)}} \right|^{-1} \quad (5)$$

$$p_{\boldsymbol{z}_y}(\boldsymbol{z}_y) = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}) \prod_{i=1}^{K} \left| \det \frac{\partial f_y^{(i)}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{z}^{(i)}} \right|^{-1} \quad (6)$$

where $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ is a base distribution, e.g., the standard normal distribution; $f_x^{(i)}$ and $f_y^{(i)}$ are the i^{th} transformations for the source and target languages, respectively; and $\boldsymbol{z}^{(i)}$ is the intermediate variable where we define $\boldsymbol{z}^{(1)} = \boldsymbol{\epsilon}$ and $\boldsymbol{z}^{(K)} = \boldsymbol{z}_x$ or $\boldsymbol{z}_y$ for notational convenience. The base distribution can be viewed as the "true" underlying semantic space, abstracting away from language specifics.

Our transformation to the sentence-level representations is similar to (Li et al., 2020). They argued that BERT induces a non-smooth anisotropic semantic space of sentences, which can harm its accurate representation of semantic similarity. Therefore, they also used NFs to transform the anisotropic BERT sentence-level distribution to a standard Gaussian distribution that is smooth and isotropic and reported better performance on some sentence-level similarity tasks. By using this type of sentence-level representation, the semantics of sentences from different languages can therefore be aligned in a simple common space in an unsupervised way, which we show is effective for unsupervised MT.

For simplicity, we denote the NFs for transforming the distributions of source and target sentence-level representations to the base distribution as mappings $G_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})}$ and $G_{(\boldsymbol{z}_y \to \boldsymbol{\epsilon})}$. Because of the

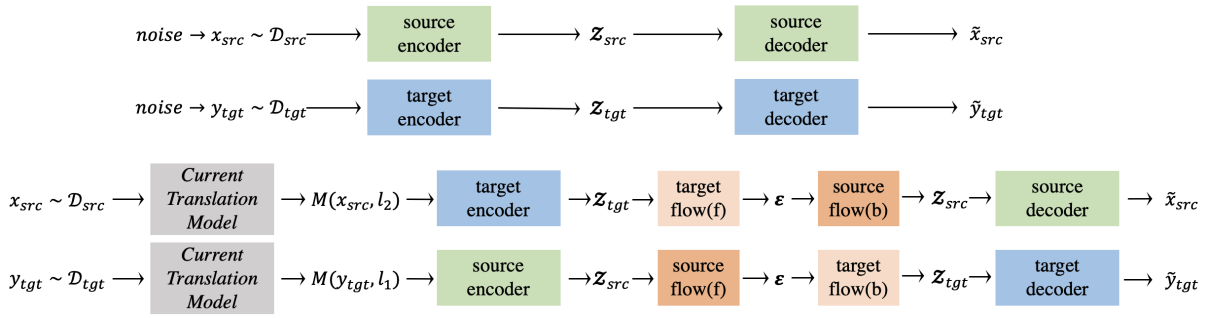

Figure 2: Top two diagrams: denoising auto-encoding for source and target sentences. Bottom two diagrams: iterative back-translation for source and target sentences. $M(\boldsymbol{x}, l_2)$ (resp. $M(\boldsymbol{y}, l_1)$) denotes the target-language (resp. source-language) sentence generated by applying the current translation model M to the source-language sentence $\boldsymbol{x}$ (resp. the target-language sentence $\boldsymbol{y}$). l_1 (the source language) and l_2 (the target language) are the parameters specifying the aim of the translation direction of M. $\tilde{\boldsymbol{x}}_{src}$ (resp. $\tilde{\boldsymbol{y}}_{tgt}$) is the reconstruction of $\boldsymbol{x}_{src}$ (resp. $\boldsymbol{y}_{tgt}$). (f) indicates the flow transforms forward from $\boldsymbol{z}$ to $\boldsymbol{\epsilon}$ while (b) for backward transformation, i.e., from $\boldsymbol{\epsilon}$ to $\boldsymbol{z}$.

invertibility property of NFs, these mappings are also invertible, and we have $G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_x)} = G^{-1}_{(\boldsymbol{z}_x\to\boldsymbol{\epsilon})}$ and $G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_y)} = G^{-1}_{(\boldsymbol{z}_y\to\boldsymbol{\epsilon})}$.

Latent Code Transformation. Inspired by AlignFlow (Grover et al., 2020), we consider the cross-domain transformation between $\boldsymbol{z}_x$ and $\boldsymbol{z}_y$. In this way, we can formulate a language-specific latent code for the decoder. We formalize the cross-language latent code transformation from the source to the target language as follows:

$$G_{(\boldsymbol{z}_x\to\boldsymbol{z}_y)} = G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_y)} \circ G_{(\boldsymbol{z}_x\to\boldsymbol{\epsilon})} \quad (7)$$

The target-to-source latent code transformation is then the composition of $G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_x)}$ and $G_{(\boldsymbol{z}_y\to\boldsymbol{\epsilon})}$. As $G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_y)}$ and $G_{(\boldsymbol{\epsilon}\to\boldsymbol{z}_x)}$ are the inverse mappings of $G_{(\boldsymbol{z}_y\to\boldsymbol{\epsilon})}$ and $G_{(\boldsymbol{z}_y\to\boldsymbol{\epsilon})}$, we can easily obtain them with normalizing flows, such as realNVP (Dinh et al., 2017) and Glow (Kingma and Dhariwal, 2018). We also note that $G_{(\boldsymbol{z}_x\to\boldsymbol{z}_y)}$ and $G_{(\boldsymbol{z}_y\to\boldsymbol{z}_x)}$ are both invertible since they are compositions of two invertible mappings. Moreover, $G_{(\boldsymbol{z}_x\to\boldsymbol{z}_y)}$ is the inverse of $G_{(\boldsymbol{z}_y\to\boldsymbol{z}_x)}$ and vice versa (see Appendix A.1 for details).

3.2 Flow-Adapter Based Unsupervised Machine Translation

The general architecture is shown in Figure 1. The transformer architecture (Vaswani et al., 2017) is used for both encoder and decoder. We use source encoder/decoder to denote the encoder/decoder for encoding/generating the source-language sentence. Similarly, target encoder/decoder refer to the encoder/decoder encoding/generating the target-language sentence. The decoders work in an autoregressive way. Source flow and target flow are NFs for modeling the sentence-level latent representations of the source and target language, respectively, as introduced in Section 3.1.

Encoding. The source encoder and the target encoder work in the same way; for brevity, we only describe the procedure of encoding the source sentence and how $\boldsymbol{z}_x$ is generated. The source encoder takes the source sentence $\boldsymbol{x} = \{x_0, \cdots, x_S\}$ as input and generates the hidden representations $\{\boldsymbol{h}_0, \cdots, \boldsymbol{h}_S\}$. These hidden representations will be used as encoder-decoder attentional inputs. In addition, we use the hidden representations to generate a sentence-level representation for the source sentence by applying max-pooling and mean-pooling to the token-level representations. After that, we sum up the results with the CLS representation $\boldsymbol{h}_0$, which usually encodes some global information. Finally, we use a projection matrix $\boldsymbol{W}$ to project the resulting vector to a latent space. The output is referred to as $\boldsymbol{z}_x$, i.e., the sentence-level representation of the source sentence (see Appendix A.2 for equation and illustration).

Cross-lingual Translation. We hypothesize that the decoder can better leverage language-specific latent representations (i.e., $\boldsymbol{z}_x$ for the source decoder and $\boldsymbol{z}_y$ for the target decoder) than indiscriminately using the same representational space for source and target, e.g., $\boldsymbol{z}_x$ for the target decoder. Therefore, we propose to perform a latent code transformation for cross-language translation as shown in Figure 1. If the model is performing the translation in the source-to-target direction, the source flow first transforms the source latent representation $\boldsymbol{z}_x$ into $\boldsymbol{\epsilon}$, which is a vector in the

semantic base space. Then the target flow transforms ϵ back into $\boldsymbol{z}_y$, which is in the target latent representation space. Then $\boldsymbol{z}_y$ is used in the target decoder for generating the target sentence.

Denoising Auto-Encoding (DAE) and Back Translation (BT) Processes. The DAE reconstructs a sentence from its noised version. For inducing noise, we use the same strategy which is used by (Lample et al., 2018a) (For more details, please refer to Appendix A.3). Since we train the DAEs separately for source and target languages, hence we don't need a latent code transformation there. For BT, however, such a latent code transformation is performed twice; taking BT for the source language as an example: first in the source-to-target direction, then in the target-to-source direction as shown in Figure 2.

Decoding. To enable the decoder to capture the global semantics and mitigate improper alignments, we use the procedure outlined in (Setiawan et al., 2020), and incorporate the latent representation $\boldsymbol{z}$ into the output of the last layer of the decoder $\{\boldsymbol{s}_0, \cdots, \boldsymbol{s}_T\}$:

$$\boldsymbol{o}_i = (1 - \boldsymbol{g}_i) \odot \boldsymbol{s}_i + \boldsymbol{g}_i \odot \boldsymbol{z} \tag{8}$$

where $\boldsymbol{g}_i = \sigma([\boldsymbol{s}_i; \boldsymbol{z}])$, $\sigma(\cdot)$ is the sigmoid function, $\odot$ denotes Hadamard product between two vectors, and $\boldsymbol{o}_i$ is the logit vector used to generate a prediction at the i^{th} position. The values in $\boldsymbol{g}_i$ control the contribution of $\boldsymbol{z}$ to $\boldsymbol{o}_i$. In case the dimension of the latent representation does not match the dimension of the decoder output, a linear projection maps $\boldsymbol{z}$ to the desired dimension.

Training. Our flow-adapter framework has three learning objectives: DAE, BT and MLE of the sentence-level representations. The description of DAE and BT is omitted here as they are well known from related work (Lample et al., 2018a; Artetxe et al., 2018). A single training iteration consists of a DAE step followed by a BT step as shown in Figure 2. MLE computation is integrated into the DAE step to calculate the likelihood of the sentence-level representations. Our MLE learning objective for the source monolingual dataset can be formulated as follows (similar for the target dataset, omitted):

$$\mathcal{L}_{MLE}(G_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})}) = E_{z \sim p_{\boldsymbol{z}_x}}[\log p_{\boldsymbol{z}_x}(z)] \tag{9}$$

where

$$p_{\boldsymbol{z}_x}(z) = p_{\boldsymbol{\epsilon}}(G_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})}(z)) \left| \det \frac{\partial G_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})}}{\partial \boldsymbol{z}_x} \right|_{\boldsymbol{z}_x = z} \tag{10}$$

by definition of the source NFs in Equation 5. $E_{z \sim p_{\boldsymbol{z}_x}}$ is approximated via mini-batches of sentence-level latent representations generated by the encoder in the training process. By training the source flow and the target flow with this MLE loss, the flows can therefore transform between the language-specific latent space of the representations and the base semantic space. In this way, the latent code transformations, i.e., $G_{(\boldsymbol{z}_x \to \boldsymbol{z}_y)}$ and $G_{(\boldsymbol{z}_y \to \boldsymbol{z}_x)}$ can be constructed.

4 Experiments

4.1 Datasets

Multi30K task1 dataset (Elliott et al., 2016, 2017).[1] This is a multi-modal dataset that has 30,000 images annotated with captions in English, German and French. Similar to (Lample et al., 2018a), we only use the caption of each image. The officially provided train, validation and test sets are used. We use this dataset as a small-scale test for validating the effectiveness of our methods.

WMT datasets.[2] Our experiments are run with the settings that were used for XLM (Conneau and Lample, 2019). XLM uses the monolingual data from the WMT News Crawl datasets[3]. We report results on *newstest2014 en-fr*, *newstest2016 en-de* and *newstest2016 en-ro*.

4.2 Setups

Preprocessing. We tokenize the sentences with the Moses script (Koehn et al., 2007). For the Multi30K dataset, we process it similar to Lample et al. (2018a). Specifically, the sentences are randomly divided into two parts. The source-language monolingual dataset is built from the source-language sentences in the first part and the target-language dataset from the second part. In this way, there will be no exact translations of any sentences in the datasets. For the WMT datasets, we use the preprocessing methods from (Conneau and Lample, 2019). For the English-Romanian dataset, we remove the diacritics as done by Sennrich et al. (2016a) to avoid their inconsistent usage in the Romanian part of the dataset.

Metric & Performance. We use BLEU as metric (Papineni et al., 2002) for all our experiments. Although Artetxe et al. (2020) recommended to use

[1] https://github.com/multi30k/dataset

[2] http://www.statmt.org/

[3] https://github.com/facebookresearch/XLM/blob/main/get-data-nmt.sh

Models	en-de	de-en	en-fr	fr-en	de-fr	fr-de
baseline	11.87	19.31	16.52	19.24	11.03	8.36
3-scf	**12.25**	19.83	**16.98**	**20.12**	**11.67**	**8.98**
3-glow	11.91	**20.14**	16.86	19.55	11.49	8.61

Table 1: BLEU of our flow-adapter model for multilingual translation on Multi30K. Baseline: model without our proposed flow-adapter architecture. 3-scf or 3-glow models: baseline models with the flow-adapter architecture constructed by two realNVP NF models or Glow NF models, each of which consists of 3 sequential flows, for performing the latent code transformation in that translation direction.

unsupervised validation criteria for systematic tuning, we follow the setting of (Conneau and Lample, 2019; Song et al., 2019) and use the provided parallel validation sets for tuning hyperparameters. We report the results on the test sets of the models that achieve best performance on the validation sets.

Pre-trained Embeddings & Models. We use the pre-trained MUSE[4] (Lample et al., 2018b) embeddings for the multilingual unsupervised MT experiment (Table 1). We also leverage pre-trained cross-lingual models in the experiment of shared & separate decoder(s) (Table 2). Specifically, XLM models from HuggingFace[5] (Wolf et al., 2020) are used to initialize the encoder. Moreover, we also incorporate our flow-adapter architecture directly into the codebase of the original implementation of XLM[6] for the WMT dataset experiment (Table 3). In this case, the encoder and decoder are both initialized with pre-trained models. Details of these models can be found in Appendix A.3.

4.3 Results of Multilingual Unsupervised Machine Translation on Multi30K

As Multi30K provides parallel test data for English, French and German, we first conduct experiments to show the multilingual translation ability of our flow-adapter models. The results are shown in Table 1. The baseline model (without flow-adapter architecture) is trained with only DAE loss, while the flow-adapter based models (3-scf and 3-glow) are additionally trained with MLE loss for the NFs. 3-scf (resp. 3-glow) is the baseline model with two realNVP NF models (Dinh et al., 2017) (resp. Glow NF models (Kingma and Dhariwal, 2018)) , each of which consists of 3 sequential flows. Each NF model is used to model the sentence-level representations of one specific language, and two NF models then construct a flow-adapter for that translation direction (as shown in Figure 1). The flow-adapter based models additionally perform the latent code transformation to generate a language-specific representation while the baseline model does not perform such a transformation.

For this experiment, we use the pre-trained cross-lingual word embeddings (MUSE embeddings) and randomly initialize a shared encoder and a shared decoder for all three languages. It is worth noting that the training objective does not contain the iterative back-translation. For further research where there are far more languages accommodated, random online back-translation (ROBT) proposed by Zhang et al. (2020) could be considered.

Table 1 shows improvements over all six translation directions by using the flow-adapter architecture. Notably, our 3-scf and 3-glow models achieve 19.83 and 20.14 BLEU scores, respectively, on *de-en*, which is 0.52 and 0.83 higher than the baseline model. Similar improvements can also be seen on other translation directions. Our 3-scf model has BLEU scores that are 0.46 to 0.88 higher than the baselines while our 3-glow model has BLEU scores that are 0.04 to 0.83 higher than the baselines. The overall improvements show that the flow-adapter can generate more suitable sentence-level representations by performing the latent code transformation, which is helpful for the decoder to capture the semantics and generate more suitable translations.

We also find that the translation performance is closely related to the language pair and the translation direction for both the baseline models and flow-adapter models. Our models obtain very good performance on *en-fr*, with performances in both the *en-fr* or *fr-en* directions better by 16 BLEU points. For other language pairs (including *en-fr*), there is always one direction showing better performance than the other. Specifically, *de-en* achieves more than 19 BLEU points compared with 12 points for *en-de*, and *de-fr* achieves more than 11 BLEU points compared with 8.5 for *fr-de*.

4.4 Results of Shared-Decoder & Separate-Decoder Models on Multi30K

We present the performance of our flow-adapter models under the shared-decoder and separate-decoder settings on Multi30K. For this experiment, the encoder is initialized with the pre-trained XLM model and fixed; the decoder parameters are ran-

[4]https://github.com/facebookresearch/MUSE
[5]https://github.com/huggingface
[6]https://github.com/facebookresearch/XLM

Models	en-de	de-en	en-fr	fr-en
baseline (shared decoder)	0.25	0.17	0.13	0.11
3-scf (shared decoder)	25.80	28.92	39.26	**36.84**
3-glow (shared decoder)	26.09	29.48	39.21	36.66
baseline (separate decoders)	27.54	28.97	39.17	36.27
3-scf (separate decoders)	28.24	**30.63**	**39.64**	36.45
3-glow (separate decoders)	**28.79**	30.45	39.31	36.29
UNMT (Lample et al., 2018a)	22.74	26.26	32.76	32.07

Table 2: BLEU of the flow-adapter models and unsupervised SOTA model, i.e., UNMT (Lample et al., 2018a), on Multi30K. Baseline models use pre-trained XLMs from HuggingFace as the encoder and randomly initialized decoder(s) without the flow-adapter. (separate decoders): two independent and randomly initialized decoders are used, each for decoding a specific language. (shared decoder): a single shared decoder for decoding both languages is used. 3-scf and 3-glow (as defined in Table 1 and Section 4.3) denote the baseline models with the flow-adapter architecture. We report the results of UNMT from their original paper.

domly initialized and then trained. We also report the performance of a previous SOTA model, i.e., UNMT (Lample et al., 2018a).[7] The results are shown in Table 2. First, we notice that the shared-decoder baseline model obtains very low BLEU scores. By checking the translation generated, we find the model only copies the input as translation. This phenomenon shows that this baseline, which does not perform the latent code transformation, cannot model two languages simultaneously well, and thus cannot generate translations in the desired language domains. However, by incorporating the flow-adapter, the models will no longer have this limitation. Both shared-decoder models, i.e., 3-scf and 3-glow, achieve very good performance on all translation pairs. For example, the 3-scf model obtains BLEU scores of 25.80, 28.92, 39.26 and 36.84 on *en-de*, *de-en*, *en-fr* and *fr-en*, which are much higher than the baseline.

Compared with the shared-decoder scenario, the models under the separate-decoder setting do not suffer from the copying problem, because different decoders are used to specifically model and generate sentences in distinct language domains. The downside, however, is that using multiple decoders at the same time can substantially increase the number of trainable parameters. Within the separate-decoder models, the flow-adapter models generally perform better than the baseline model, with about 1 BLEU increase on *en-de* and *de-en* directions and relatively smaller improvements on *en-fr* and *fr-en*. Those improvements demonstrate that the model can benefit from the flow-adapter architectures as language-specific latent representations are used, thus advancing the translation quality.

We also observe that the separate-decoder models generally perform better than the shared-decoder models. The separate-decoder baseline is much better than its counterpart as it avoids the copying problem. For the 3-scf flow-adapter models, we find that the separate-decoder model outperforms the shared-decoder model by 2.44, 1.71, 0.38 on *en-de*, *de-en* and *en-fr*. However, on *fr-en*, the shared-decoder model achieves a BLEU socre that is by 0.39 BLEU points better. A similar phenomenon can also be seen for the 3-glow model. We conjecture this is due to the similarity between languages. As English and French share common vocabulary, some common features can therefore be captured by a shared decoder, thus improving its generalization.

Lastly, when compared with UNMT, our models show superiority, improving performance by more than 4 BLEU points in each direction. We attribute the improvements to the usage of the pre-trained model and incorporation of language-specific sentence-level representations obtained by our latent code transformation.

4.5 Results on WMT datasets

We further integrate our flow-adapter architecture into the original implementation of XLM (Conneau and Lample, 2019) and conduct experiments on the WMT datasets. To fully leverage the pre-trained models, we initialize both the encoder and decoder with XLM models and set them trainable. In contrast to the experiment in Section 4.4, a single shared decoder is used for this experiment, since the decoder is also initialized with the pre-trained model and has far more parameters compared with the randomly initialized transformer decoder we use in Section 4.4. We report the performance of the flow-adapter based models (5-scf and 5-

[7] UNMT did not use pre-trained models. The results are therefore not strictly comparable to ours.

Models	en-de	de-en	en-ro	ro-en	en-fr	fr-en
XLM (EMD + EMD) (Conneau and Lample, 2019)	21.30	27.30	27.50	26.60	29.40	29.40
XLM (MLM + MLM) (Conneau and Lample, 2019)	26.40	**34.30**	33.30	**31.80**	33.40	33.30
5-scf	**26.50**	32.63	**34.11**	31.69	**35.77**	**33.72**
5-glow	26.43	32.04	33.87	31.32	35.25	33.12
MASS (Song et al., 2019)	28.30	35.20	35.20	33.10	37.50	34.90
CSP and fine-tuning (Yang et al., 2020)	28.70	35.70	-	-	37.90	34.50

Table 3: BLEU of the flow-adapter models (5-scf and 5-glow) and SOTA models on WMT datasets. XLM (MLM + MLM) is the baseline in this table, as 5-scf and 5-glow use it as the base model for the flow-adapter architecture. We report the results of XLM, MASS and CSP from the original paper.

glow[8]) as well as the performance of the SOTA models, namely XLM, MASS and CSP.[9] The results are shown in Table 3. Noticeably, both of our flow-adapter models achieve remarkable performance on all language pairs. Compared with the results of XLM (EMD + EMD), which uses the pre-trained cross-lingual embeddings instead of pre-trained models, both 5-scf and 5-glow achieve overall better performance. For example, 3-scf achieves BLEU scores higher by 5.20, 5.33, 6.61, 5.09, 6.37 and 4.32 on *en-de*, *de-en*, *en-ro*, *ro-en*, *en-fr* and *fr-en*, respectively. While not being as good as 5-scf, 5-glow still shows superiority over XLM (EMD + EMD). These improvements can be contributed to (1) the usage of pre-trained models and (2) the introduction of the flow-adapter.

We further compare our flow-adapter based models with XLM (MLM + MLM), which is also initialized with pre-trained models. We find the performance of *x-en* directions is consistently lower than *en-x* directions for our models except for *en-de*. This pattern is not limited to our architecture but is consistently present in prior work. We, again, speculate this is relating to the complexity of languages as well as similarity between languages. We leave this finding for future investigation. Our flow-adapter based models, though achieving similar or relatively worse BLEU scores on *de-en* and *ro-en* compared with XLM (MLM + MLM), obtain higher scores on other directions, i.e., *en-de* and *en-ro*, suggesting that our models might be more helpful on specific translation directions, as the flow-adapter generates language-specific representations. Lastly, 5-scf achieves scores by 2.37 and 0.42 better than XLM (MLM + MLM) on *en-fr* and *fr-en*. As in the other experiments, the improvement due to flow adapters seems to be related to the languages involved in that language pair and the translation directions. We would like to investigate this phenomenon in future research.

Finally, out models are competitve with MASS and CSP, with only small differences in BLEU. In general, the experiments shows the validity and effectiveness of our flow-adapter architecture integrated into pre-trained models.

5 Conclusion

In this work, we propose a novel flow-adapter architecture for unsupervised NMT. The flow-adapter employs a pair of NFs to explicitly model the distributions of the sentence-level representations. A latent code transformation is performed in translation, which enables the decoder to better capture the semantics of sentences in certain language domains. Through extensive experiments, we show the flow-adapter can improve multilingual translation ability. Moreover, it can alleviate the copying problem. By integrating the flow-adapter into pre-trained XLM models, we achieve results competitive to state-of-the-art models on WMT datasets.

In the future, we would like to explore the possibility of pre-training the flow-adapter simultaneously when pre-training the language models so that the flows can learn more information. Moreover, we would like to extend normalizing flows to language generation. By using different flows for different languages, multilingual language generation of the same semantics can be performed.

Acknowledgements

We are grateful to Alex Fraser and Alexandra Chronopoulou for their insightful input. This work was funded by the European Research Council (ERC #740516).

[8]Preliminary experiments showed that using 5 flows provides slightly better results than 3 flows for WMT as WMT has many more sentences than Multi30K and therefore more powerful generative models (by adding more intermediate flows) are needed to model the sentence-level representations.

[9]We follow prior convention and compare directly with MASS and CSP even though dataset processing for MASS and CSP (e.g., filtering, sampling) are not strictly the same as for XLM. But the difference is small and results would not be much different as Yang et al. (2020) mentions.

References

Yaser Al-Onaizan, Ulrich Germann, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Daniel Marcu, and Kenji Yamada. 2002. Translation with scarce bilingual resources. *Machine translation*, 17(1):1–17.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Mikel Artetxe, Sebastian Ruder, Dani Yogatama, Gorka Labaka, and Eneko Agirre. 2020. A call for more rigor in unsupervised cross-lingual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7375–7388, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Iacer Calixto, Miguel Rios, and Wilker Aziz. 2019. Latent variable model for multi-modal translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6392–6405, Florence, Italy. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, 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. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Bryan Eikema and Wilker Aziz. 2019. Auto-encoding variational neural machine translation. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 124–141, Florence, Italy. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation*, pages 215–233, Copenhagen, Denmark. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany. Association for Computational Linguistics.

Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, et al. 2019. Universal audio synthesizer control with normalizing flows. *arXiv preprint arXiv:1907.00971*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Aditya Grover, Christopher Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. 2020. Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The*

Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 4028–4035. AAAI Press.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging non-linearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Rob Hesselink and Wilker Aziz. 2020. Latent transformations for discrete-data normalising flows. *arXiv preprint arXiv:2006.06346*.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2722–2730.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2016. End-to-end statistical machine translation with zero or small parallel texts. *Natural Language Engineering*, 22(4):517–548.

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

Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10236–10245.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. 2019. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2(5).

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018b. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*.

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. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. 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 4282–4292, Hong Kong, China. 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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In

Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, page II–1278–II–1286. JMLR.org.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Hendra Setiawan, Matthias Sperber, Udhyakumar Nallasamy, and Matthias Paulik. 2020. Variational neural machine translation with normalizing flows. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7771–7777, Online. Association for Computational Linguistics.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8846–8853. AAAI Press.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 5926–5936. PMLR.

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(56):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*, page 3104–3112. MIT Press.

Zineng Tang, Shiyue Zhang, Hyounghun Kim, and Mohit Bansal. 2021. Continuous language generative flow. 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 4609–4622, Online. Association for Computational Linguistics.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. 2019. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Ahmet Üstün, Alexandre Berard, Laurent Besacier, and Matthias Gallé. 2021. Multilingual unsupervised neural machine translation with denoising adapters. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6650–6662, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. 2018. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3918–3926.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010. Curran Associates Inc.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55, Melbourne, Australia. Association for Computational Linguistics.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. CSP:code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636, Online. Association for Computational Linguistics.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. 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.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

Zachary M. Ziegler and Alexander M. Rush. 2019. Latent normalizing flows for discrete sequences. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 7673–7682. PMLR.

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, Austin, Texas. Association for Computational Linguistics.

A Appendix

A.1 Proof of the Invertibility

The following proof is based on the proof by Grover et al. (2020) and shows the source-to-target latent code transformation is the inverse of the target-to-source latent code transformation, and vice versa:

$$\begin{aligned} G^{-1}_{(\boldsymbol{z}_x \to \boldsymbol{z}_y)} &= (G_{(\boldsymbol{\epsilon} \to \boldsymbol{z}_y)} \circ G_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})})^{-1} \\ &= G^{-1}_{(\boldsymbol{z}_x \to \boldsymbol{\epsilon})} \circ G^{-1}_{(\boldsymbol{\epsilon} \to \boldsymbol{z}_y)} \\ &= G_{(\boldsymbol{\epsilon} \to \boldsymbol{z}_x)} \circ G_{(\boldsymbol{z}_y \to \boldsymbol{\epsilon})} \\ &= G_{(\boldsymbol{z}_y \to \boldsymbol{z}_x)} \end{aligned} \tag{11}$$

A.2 Generation of Sentence-level Representation

The following formula shows the process of how the sentence-level representation is generated:

$$\begin{aligned} \boldsymbol{z} = \text{Linear}(&\text{max-pool}([\boldsymbol{h}_0, \cdots, \boldsymbol{h}_S]) \\ &+ \text{mean-pool}([\boldsymbol{h}_0, \cdots, \boldsymbol{h}_S]) \\ &+ \boldsymbol{h}_0) \end{aligned} \tag{12}$$

where the pooling operation generates a vector that has the same dimension as $\boldsymbol{h}_0$, so the three vectors have the same shape and therefore are additive. An illustration can be seen in Figure 3.

A.3 Details of the Experiments: Models & Hyperparameters

A.3.1 Multi30K Experiment

For the multilingual machine translation tasks, we use the cross-lingual MUSE (Lample et al., 2018b). The embeddings were learned using fastText[10] (Bojanowski et al., 2017) on Wikipedia data and then aligned in a common space by the method proposed by Lample et al. (2018b). The results shown in Table 1 is the average over 10 runs on the test sets. Denosing auto-encoding is used to train the baseline model. The flow-adapter based (3-scf and 3-glow) models are additionally trained with MLE loss. We follow the denoising auto-encoding hyperparameter settings used by Lample et al. (2018a). Specifically, word drop and word shuffling are used. For word drop, every word in a sentence (except <bos> and <eos>) can be dropped with a probability p_{wd}, which we set 0.1 in our experiments. For word shuffling, a random permutation σ is applied to the input sentence, which satisfy the condition: $\forall i \in \{1, n\}, |\sigma(i) - i| \leq k$, where i is the index of a word in the sequence, n is the length of the sequence and k is a hyperparameter that controls the degree of the permutation which we set 3 in our experiments. The dimension of the pre-trained embedding is 300. The randomly initialized shared encoder and decoder use transformer architecture with 512 hidden units, 4 heads and 3 layers by default. We use separate embedding layers for each language and tie their weights with the output layers for each language. The size of the sentence-level latent representation is set to 100. And the weight of the MLE loss for the flows is set to 0.01. We use dropout (Srivastava et al., 2014) probability of 0.2 for the transformers and 0 for the flows. The batch size is set to 32. The whole model is trained in an end-to-end manner with Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.0001.

For the shared-decoder & separate-decoder experiments, we use the pre-trained language models *xlm-mlm-enfr-1024*, *xlm-mlm-ende-1024*, *xlm-mlm-enro-1024* from HuggingFace[11] (Wolf et al., 2020)

[10] https://github.com/facebookresearch/fastText

[11] https://github.com/huggingface

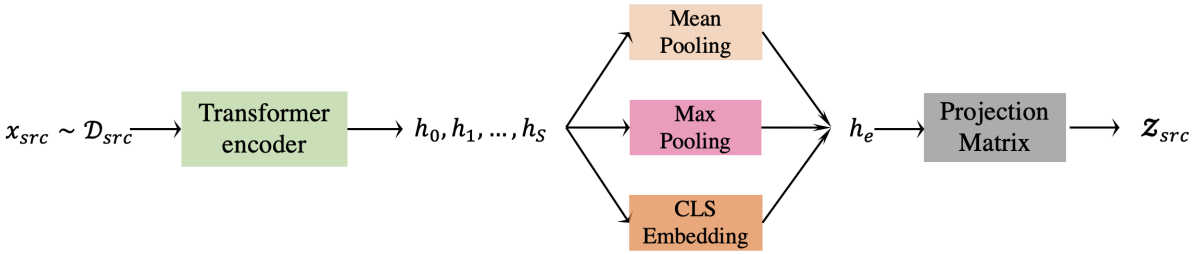

Figure 3: The illustration of generation of the sentence-level representations. CLS embedding refers to the first vector output by the transformer encoder, i.e., $\boldsymbol{h}_0$.

to initialize a shared encoder and randomly initialize the decoder(s).using pre-trained models. Denosing auto-encoding and iterative back-translation are used to train the baseline model. The flow-adapter based (3-scf and 3-glow) models are additionally trained with MLE loss. The same denoising auto-encoding hyperparameters as above are used. For iterative back-translation, greedy decoding is used to generate synthetic parallel sentences as well as the reconstructions. A single embedding layer (from the pre-trained model) is used for both the source and target languages and its weight is tied with the output layer. The parameters of the encoder are fixed except for its embedding layer which is also used by the decoder(s). The size of the sentence-level latent representation is set to 256. The pre-trained encoder uses 1024 as the embedding size and GELU activations (Hendrycks and Gimpel, 2016), and has 4096 hidden units, 8 heads and 6 layers. The randomly initialized decoder has 512 hidden units, 8 heads and 3 layers. The models are firstly trained with DAE loss (and MLE loss for flow-adapter models) for the first 3 epochs, then trained with all losses (including the iterative back-translation) for the rest epochs. The rest hyperparameters are the same as above.

A.3.2 WMT Experiment

We insert our implementation of flow-adapter architecture into the codebase of XLM[12] and use the pre-trained model of *en-fr*, *en-de* and *en-ro* from them. We also follow their recommended unsupervised training settings. For the flow-related hyperparameters, we use 256 as the size of the sentence-level latent representation. The weight of the MLE loss is set to 0.01.

[12]https://github.com/facebookresearch/XLM#iii-applications-supervised–unsupervised-mt