AdvAug: Robust Adversarial Augmentation for Neural Machine Translation

Yong Cheng, Lu Jiang, Wolfgang Macherey and Jacob Eisenstein
Google Research
{chengyong, lujiang, wmach, jeisenstein}@google.com

Abstract

In this paper, we propose a new adversarial augmentation method for Neural Machine Translation (NMT). The main idea is to minimize the vicinal risk over virtual sentences sampled from two vicinity distributions, of which the crucial one is a novel vicinity distribution for adversarial sentences that describes a smooth interpolated embedding space centered around observed training sentence pairs. We then discuss our approach, *AdvAug*, to train NMT models using the embeddings of virtual sentences in sequence-to-sequence learning. Experiments on Chinese-English, English-French, and English-German translation benchmarks show that *AdvAug* achieves significant improvements over the Transformer (up to 4.9 BLEU points), and substantially outperforms other data augmentation techniques (*e.g.* back-translation) without using extra corpora.

1 Introduction

Recent work in neural machine translation (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) has led to dramatic improvements in both research and commercial systems (Wu et al., 2016). However, a key weakness of contemporary systems is that performance can drop dramatically when they are exposed to input perturbations (Belinkov and Bisk, 2018; Cheng et al., 2019), even when these perturbations are not strong enough to alter the meaning of the input sentence. Consider a Chinese sentence, "zhejia feiji meiyou zhuangshang zhujia **huo** yiyuan, shizai shi qiji". If we change the word "**huo** (或)" to its synonym"**ji** (及)", the Transformer model will generate contradictory results of "It was indeed a miracle that the plane **did not touch down** at home or hospital." versus "It was a miracle that the plane **landed** at home and hospital." Such perturbations can readily be found in many public benchmarks and real-world applications. This lack of stability not only lowers translation quality but also inhibits applications in more sensitive scenarios.

At the root of this problem are two interrelated issues: first, machine translation training sets are insufficiently diverse, and second, NMT architectures are powerful enough to overfit — and, in extreme cases, memorize — the observed training examples, without learning to generalize to unseen perturbed examples. One potential solution is data augmentation which introduces noise to make the NMT model training more robust. In general, two types of noise can be distinguished: (1) continuous noise which is modeled as a real-valued vector applied to word embeddings (Miyato et al., 2016, 2017; Cheng et al., 2018; Sato et al., 2019), and (2) discrete noise which adds, deletes, and/or replaces characters or words in the observed sentences (Belinkov and Bisk, 2018; Sperber et al., 2017; Ebrahimi et al., 2018; Michel et al., 2019; Cheng et al., 2019; Karpukhin et al., 2019). In both cases, the challenge is to ensure that the noisy examples are still semantically valid translation pairs. In the case of continuous noise, it only ensures that the noise vector lies within an L_2-norm ball but does not guarantee to maintain semantics. While constructing semantics-preserving continuous noise in a high-dimensional space proves to be non-trivial, state-of-the-art NMT models are currently based on adversarial examples of discrete noise. For instance, Cheng et al. (2019) generate adversarial sentences using discrete word replacements in both the source and target, guided by the NMT loss. This approach achieves significant improvements over the Transformer on several standard NMT benchmarks. Despite this promising result, we find that the generated adversarial sentences are unnatural, and, as we will show, suboptimal for learning robust NMT models.

In this paper, we propose *AdvAug*, a new adversarial augmentation technique for sequence-to-sequence learning. We introduce a novel vicinity distribution to describe the space of adversarial examples centered around each training example. Unlike prior work (Cheng et al., 2019), we first generate adversarial sentences in the discrete data space and then sample *virtual* adversarial sentences from the vicinity distribution according to their interpolated embeddings. Our intuition is that the introduced vicinity distribution may increase the sample diversity for adversarial sentences. Our idea is partially inspired by *mixup* (Zhang et al., 2018), a technique for data augmentation in computer vision, and we also use a similar vicinity distribution as in *mixup* to augment the authentic training data. Our *AdvAug* approach finally trains on the embeddings sampled from the above two vicinity distributions. As a result, we augment the training using virtual sentences in the feature space as opposed to in the data space. The novelty of our paper is the new vicinity distribution for adversarial examples and the augmentation algorithm for sequence-to-sequence learning.

Extensive experimental results on three translation benchmarks (NIST Chinese-English, IWSLT English-French, and WMT English-German) show that our approach achieves significant improvements of up to 4.9 BLEU points over the Transformer (Vaswani et al., 2017), outperforming the former state-of-the-art in adversarial learning (Cheng et al., 2019) by up to 3.3 BLEU points. When compared with widely-used data augmentation methods (Sennrich et al., 2016a; Edunov et al., 2018), we find that our approach yields better performance even without using extra corpora. We conduct ablation studies to gain further insights into which parts of our approach matter most. In summary, our contributions are as follows:

1. We propose to sample adversarial examples from a new vicinity distribution and utilize their embeddings, instead of their data points, to augment the model training.

2. We design an effective augmentation algorithm for learning sequence-to-sequence NMT models via mini-batches.

3. Our approach achieves significant improvements over the Transformer and prior state-of-the-art models on three translation benchmarks.

2 Background

Neural Machine Translation. Generally, NMT (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) models the translation probability $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ based on the encoder-decoder paradigm where $\mathbf{x}$ is a source-language sentence, $\mathbf{y}$ is a target-language sentence, and $\boldsymbol{\theta}$ is a set of model parameters. The decoder in the NMT model acts as a conditional language model that operates on a shifted copy of $\mathbf{y}$, i.e., $\langle sos \rangle, y_0, ..., y_{|\mathbf{y}|-1}$ where $\langle sos \rangle$ is a start symbol of a sentence and representations of $\mathbf{x}$ learned by the encoder. For clarity, we use $e(\mathbf{x}) \in \mathbb{R}^{d\times|\mathbf{x}|}$ to denote the feature vectors (or word embeddings) of the sentence $\mathbf{x}$ where d is dimension size.

Given a parallel training corpus $\mathcal{S}$, the standard training objective for NMT is to minimize the empirical risk:

$$\mathcal{L}_{clean}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{P_\delta(\mathbf{x},\mathbf{y})}[\ell(f(e(\mathbf{x}), e(\mathbf{y});\boldsymbol{\theta}), \ddot{\mathbf{y}})], \quad (1)$$

where $f(e(\mathbf{x}), e(\mathbf{y});\boldsymbol{\theta})$ is a sequence of model predictions $f_j(e(\mathbf{x}), e(\mathbf{y});\boldsymbol{\theta}) = P(y|\mathbf{y}_{<j}, \mathbf{x};\boldsymbol{\theta})$ at position j, and $\ddot{\mathbf{y}}$ is a sequence of one-hot label vectors for $\mathbf{y}$ (with label smoothing in the Transformer). ℓ is the cross entropy loss. The expectation of the loss function is summed over the empirical distribution $P_\delta(\mathbf{x},\mathbf{y})$ of the training corpus:

$$P_\delta(\mathbf{x},\mathbf{y}) = \frac{1}{|S|}\sum_{(\mathbf{x}',\mathbf{y}')\in\mathcal{S}} \delta(\mathbf{x}=\mathbf{x}', \mathbf{y}=\mathbf{y}'), \quad (2)$$

where δ denotes the Dirac delta function.

Generating Adversarial Examples for NMT. To improve NMT's robustness to small perturbations in the input sentences, Cheng et al. (2019) incorporate adversarial examples into the NMT model training. These adversarial sentences $\mathbf{x}'$ are generated by applying small perturbations that are jointly learned together with the NMT model:

$$\hat{\mathbf{x}} = \mathop{\mathrm{argmax}}_{\hat{\mathbf{x}}:\mathcal{R}(\hat{\mathbf{x}},\mathbf{x})\le\epsilon} \ell(f(e(\hat{\mathbf{x}}), e(\mathbf{y});\boldsymbol{\theta}), \ddot{\mathbf{y}}), \quad (3)$$

where $\mathcal{R}(\hat{\mathbf{x}},\mathbf{x})$ captures the degree of semantic similarity and ϵ is an upper bound on the semantic distance between the adversarial example and the original example. Ideally, the adversarial sentences convey only barely perceptible differences to the original input sentence yet result in dramatic distortions of the model output.

Cheng et al. (2019) propose the *AdvGen* algorithm, which greedily replaces words with their top k most probable alternatives, using the gradients of their word embeddings. Adversarial examples are designed to both attack and defend the NMT model. On the encoder side, an adversarial sentence $\hat{\mathbf{x}}$ is constructed from the original input $\mathbf{x}$ to attack the NMT model. To defend against adversarial perturbations in the source input $\hat{\mathbf{x}}$, they use the *AdvGen* algorithm to find an adversarial target input $\hat{\mathbf{y}}$ from the decoder input $\mathbf{y}$. For notational convenience, let π denote this algorithm, the adversarial example $\hat{\mathbf{s}}$ is stochastically induced by π as $\hat{\mathbf{s}} \leftarrow \pi(\mathbf{s}; \mathbf{x}, \mathbf{y}, \xi)$ where ξ is the set of parameters used in π including the NMT model parameters $\boldsymbol{\theta}$. For a detailed definition of ξ, we refer to (Cheng et al., 2019). Hence, the set of adversarial examples originating from $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}$, namely $A_{(\mathbf{x},\mathbf{y})}$, can be written as:

$$\begin{aligned} A_{(\mathbf{x},\mathbf{y})} = \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}) | \hat{\mathbf{x}} \leftarrow \pi(\mathbf{x}; \mathbf{x}, \mathbf{y}, \xi_{src}), \\ \hat{\mathbf{y}} \leftarrow \pi(\mathbf{y}; \hat{\mathbf{x}}, \mathbf{y}, \xi_{tgt})\}, \end{aligned} \tag{4}$$

where ξ_{src} and ξ_{tgt} are separate parameters for generating $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, respectively. Finally, the robustness loss $\mathcal{L}_{robust}$ is computed on $A_{(\mathbf{x},\mathbf{y})}$ with the loss $\ell(f(e(\hat{\mathbf{x}}), e(\hat{\mathbf{y}}); \boldsymbol{\theta}), \ddot{\mathbf{y}})$, and is used together with $\mathcal{L}_{clean}$ to train the NMT model.

Data Mixup. In image classification, the *mixup* data augmentation technique involves training on linear interpolations of randomly sampled pairs of examples (Zhang et al., 2018). Given a pair of images $(\mathbf{x}', \mathbf{y}')$ and $(\mathbf{x}'', \mathbf{y}'')$, where $\mathbf{x}'$ denotes the RGB pixels of the input image and $\mathbf{y}'$ is its one-hot label, *mixup* minimizes the sample loss from a vicinity distribution (Chapelle et al., 2001) $P_v(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ defined in the RGB-pixel (label) space:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}' + (1 - \lambda)\mathbf{x}'', \tag{5}$$

$$\tilde{\mathbf{y}} = \lambda \mathbf{y}' + (1 - \lambda)\mathbf{y}''. \tag{6}$$

λ is drawn from a Beta distribution Beta(α, α) controlled by the hyperparameter α. When $\alpha \rightarrow 0$, $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is close to any one of the images $(\mathbf{x}', \mathbf{y}')$ and $(\mathbf{x}'', \mathbf{y}'')$. Conversely, $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ approaches the middle interpolation point between them when $\alpha \rightarrow +\infty$. The neural networks g parameterized by ψ can be trained over the mixed images $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ with the loss function $\mathcal{L}_{mixup}(\boldsymbol{\theta}) = \ell(g(\tilde{\mathbf{x}}; \boldsymbol{\psi}), \tilde{\mathbf{y}})$. In practice, the image pair is randomly sampled from the same mini-batch.

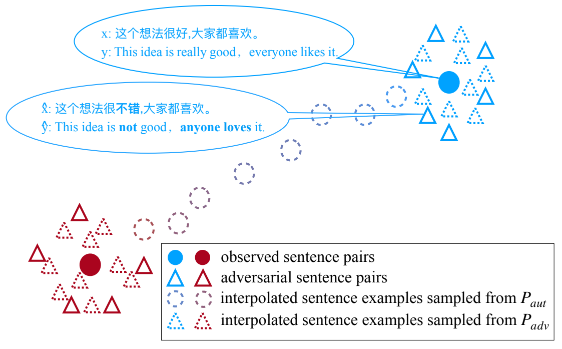
x: 这个想法很好,大家都喜欢。
y: This idea is really good，everyone likes it.
$\hat{x}$: 这个想法很**不错**,大家都喜欢。
$\hat{y}$: This idea is **not** good，**anyone loves** it.
observed sentence pairs
adversarial sentence pairs
interpolated sentence examples sampled from P_{aut}
interpolated sentence examples sampled from P_{adv}

Figure 1: Illustration of training examples sampled from vicinity distributions of P_{adv} and P_{aut}. Solid circles are observed sentences in the training corpus $\mathcal{S}$. Solid triangles are adversarial sentences generated by replacing words in their corresponding observed sentences. Dashed points are virtual sentences obtained by interpolating the embeddings of the solid points. The dashed triangles define the data space of adversarial examples from P_{adv}. The circles (solid and dashed) constitute P_{aut}.

3 Approach: AdvAug

In our approach *AdvAug*, the goal is to reinforce the model over virtual data points surrounding the observed examples in the training set.

We approximate the density of $P(\mathbf{x}, \mathbf{y})$ in the vicinities of the generated adversarial examples and observed training examples. To be specific, we design two vicinity distributions (Chapelle et al., 2001) to estimate the joint distribution of $P(\mathbf{x}, \mathbf{y})$: P_{adv} for the (dynamically generated) *adversarial* examples and P_{aut} for the (observed) *authentic* examples in $\mathcal{S}$. Given the training set $\mathcal{S}$, we have:

$$P_{adv}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{S}} \mu_{adv}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} | A_{(\mathbf{x},\mathbf{y})}), \tag{7}$$

$$P_{aut}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{S}} \mu_{aut}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} | \mathbf{x}, \mathbf{y}), \tag{8}$$

where $A_{(\mathbf{x},\mathbf{y})}$ is the set of adversarial examples originated from $(\mathbf{x}, \mathbf{y})$ defined in Eq. (4). We will discuss μ_{adv} and μ_{aut} in detail which define the probability functions, but first we give some high-level descriptions:

- P_{adv} is a new vicinity distribution for virtual adversarial sentences of the same origin. It captures the intuition that the convex combination of adversarial sentences should have the same translation. It is the most important factor for improving the translation quality in our experiments.

- P_{aut} is a distribution to improve the NMT's robustness by "mixing up" observed sentences of different origins. This distribution is similar to *mixup*, but it is defined over linear interpolations of the sequence of word embeddings of the source and target sentences. Although P_{aut} by itself yields marginal improvements, we find it is complementary to P_{adv}.

We train the NMT model on two vicinity distributions P_{adv} and P_{aut}. Figure 1 illustrates examples sampled from them. As shown, a solid circle stands for an observed training example (*i.e.* a sentence-pair) in $\mathcal{S}$ and a solid triangle denotes an adversarial example in $A_{(\mathbf{x},\mathbf{y})}$. For P_{adv}, we construct virtual *adversarial* examples (dashed triangles) to amend the sample diversity by interpolating the word embeddings of solid triangles. Likewise, we interpolate the word embeddings of solid circles to model P_{aut} for the (observed) *authentic* examples. This results in the dashed circles in Figure 1.

Unlike prior works on vicinal risk minimization (Chapelle et al., 2001; Zhang et al., 2018), we do not directly observe the virtual sentences in P_{adv} or P_{aut}. This also distinguishes us from Cheng et al. (2019), who generate actual adversarial sentences in the discrete word space. In the remainder of this section, we will discuss the definition of P_{adv} and P_{aut} and how to optimize the translation loss over virtual sentences via mini-batch training.

3.1 P_{adv} for Adversarial Data

To compute μ_{adv}, we employ π similar as in (Cheng et al., 2019) to generate an adversarial example set $A_{(\mathbf{x},\mathbf{y})}$ from each instance $(\mathbf{x},\mathbf{y}) \in \mathcal{S}$ (see Eq. (4)). Let $(\mathbf{x}',\mathbf{y}')$ and $(\mathbf{x}'',\mathbf{y}'')$ be two examples randomly sampled from $A_{(\mathbf{x},\mathbf{y})}$. We align the two sequences by padding tokens to the end of the shorter sentence. Note that this operation aims for a general case (particularly for P_{aut}) although the lengths of $\mathbf{y}'$ and $\mathbf{y}''$ in $A_{(\mathbf{x},\mathbf{y})}$ are same. To obtain $e(\tilde{\mathbf{x}}) = [e(\tilde{x}_1), \ldots, e(\tilde{x}_{|\tilde{\mathbf{x}}|})]$, we apply the convex combination $m_\lambda(\mathbf{x}',\mathbf{x}'')$ over the aligned word embeddings, which is:

$$e(\tilde{x}_i) = \lambda e(x'_i) + (1-\lambda)e(x''_i), \forall i \in [1, |\tilde{\mathbf{x}}|], \quad (9)$$

where $\lambda \sim \text{Beta}(\alpha,\alpha)$. We use $m_\lambda(\cdot,\cdot)$ for the interpolation. Similarly, $e(\tilde{\mathbf{y}})$ can also be obtained with $m_\lambda(\mathbf{y}',\mathbf{y}'')$.

All adversarial examples in $A_{(\mathbf{x},\mathbf{y})}$ are supposed to be translated into the same target sentence, and the convex combination still lies in space of the adversarial search ball defined in Eq. (3). As a result, all virtual sentence pairs $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \in A_{(\mathbf{x},\mathbf{y})}$ of the same origin can be fed into NMT models as source and target inputs which share the same soft target label for $(\mathbf{x},\mathbf{y})$.

μ_{adv} in P_{adv} can be calculated from:

$$\mu_{adv}(\tilde{\mathbf{x}},\tilde{\mathbf{y}}|A_{(\mathbf{x},\mathbf{y})}) = \frac{1}{|A_{(\mathbf{x},\mathbf{y})}|^2} \sum_{(\mathbf{x}',\mathbf{y}')\in A_{(\mathbf{x},\mathbf{y})}} \sum_{(\mathbf{x}'',\mathbf{y}'')\in A_{(\mathbf{x},\mathbf{y})}} \mathop{\mathbb{E}}_{\lambda}[\delta(e(\tilde{\mathbf{x}}) = m_\lambda(\mathbf{x}',\mathbf{x}''), e(\tilde{\mathbf{y}}) = m_\lambda(\mathbf{y}',\mathbf{y}''))]. \quad (10)$$

Hence, the translation loss on vicinal adversarial examples $\mathcal{L}_{adv}(\boldsymbol{\theta})$ can be integrated over P_{adv} as:

$$\mathcal{L}_{adv}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{P_{adv}(\tilde{\mathbf{x}},\tilde{\mathbf{y}})}[\ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), \boldsymbol{\omega})], \quad (11)$$

where $\boldsymbol{\omega}$ is a sequence of output distributions (denoted as a sequence of label vectors, *e.g.* $\ddot{\mathbf{y}}$) as the soft target for the sentence $\mathbf{y}$.

We employ two useful techniques in computing the loss $\mathcal{L}_{adv}$ in Eq. (11). First, we minimize the KL-divergence between the model predictions at the word level:

$$\sum_{j=1}^{|\mathbf{y}|} D_{KL}(f_j(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}}) || f_j(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta})), \quad (12)$$

where $\hat{\boldsymbol{\theta}}$ means a fixed copy of the current parameter set and no gradients are back-propagated through it. Removing constant values from Eq. (12) yields an equivalent solution of:

$$\begin{aligned} &\ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), \boldsymbol{\omega}) \\ =&\ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), f(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}})). \end{aligned} \quad (13)$$

Eq. (13) indicates that $f(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}})$ can be used as the soft target $\boldsymbol{\omega}$ in Eq. (11) for virtual adversarial example $(\tilde{\mathbf{x}},\tilde{\mathbf{y}})$. KL-divergence enforces the model on virtual adversarial examples to indirectly learn from the soft target of the observed examples over large vocabularies. This justifies the use of $\boldsymbol{\omega}$ in Eq. (11) and turns out to be more effective than directly learning from the ground-truth label.

Besides, Eq. (11) needs to enumerate numerous pairs of adversarial examples in $A_{(\mathbf{x},\mathbf{y})}$ while in practice we only sample a pair at a time inside each mini-batch for training efficiency. We hence employ curriculum learning to do the importance sampling. To do so, we re-normalize the translation loss and employ a curriculum from (Jiang et al., 2018) to encourage the model to focus on the difficult training examples.

Formally, for a mini-batch of the training losses $\mathbf{L} = \{\ell_i\}_{i=1}^m$, we re-weigh the batch loss using:

$$\mathbf{L} = \frac{1}{\sum_{i=1}^m I(\ell_i > \eta)} \sum_{i=1}^m I(\ell_i > \eta)\ell_i, \qquad (14)$$

where $I(\cdot)$ is an indicator function and η is set by a moving average tracking the p-th percentile of the example losses of every mini-batch. In our experiments, we set the p-th percentile to be $100 \times (1 - r_t)$ for the training iteration t, and gradually anneal r_t using $r_t = 0.5^{t/\beta}$, where β is the hyperparameter.

3.2 P_{aut} for Authentic Data

We define the μ_{aut} in the vicinity distribution P_{aut} for authentic examples as follows:

$$\mu_{aut}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}|\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}',\mathbf{y}') \in \mathcal{S}} \mathop{\mathbb{E}}_{\lambda}[$$
$$\delta(e(\tilde{\mathbf{x}}) = m_\lambda(\mathbf{x}, \mathbf{x}'), e(\tilde{\mathbf{y}}) = m_\lambda(\mathbf{y}, \mathbf{y}'),$$
$$\tilde{\boldsymbol{\omega}} = m_\lambda(\boldsymbol{\omega}, \boldsymbol{\omega}'))]. \qquad (15)$$

The translation loss on authentic data is integrated over all examples of the vicinity distribution P_{aut}:

$$\mathcal{L}_{aut}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{P_{aut}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})} [\ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), \tilde{\boldsymbol{\omega}})]. \qquad (16)$$

In our experiments, we select the value of λ in Eq. (15) twice for every $(\mathbf{x}, \mathbf{y})$: (1) a constant 1.0 and (2) a sample from the Beta distribution. The former is equivalent to sampling from the empirical distribution P_δ whereas the latter is similar to applying *mixup* in the embedding space of the sequence model. In other words, $\mathcal{L}_{aut}(\boldsymbol{\theta})$ equals the sum of two translation losses, $\mathcal{L}_{clean}(\boldsymbol{\theta})$ computed on the original training examples when λ is 1.0 and $\mathcal{L}_{mixup}(\boldsymbol{\theta})$ computed on virtual examples when λ is sampled from a Beta distribution. Accordingly, when λ is 1.0 we set $\tilde{\boldsymbol{\omega}}$ to be the interpolation of the sequences of one-hot label vectors for $\mathbf{y}$ and $\mathbf{y}'$, *i.e.* $\boldsymbol{\omega} = \ddot{\mathbf{y}}$ and $\boldsymbol{\omega}' = \ddot{\mathbf{y}}'$. Otherwise $\tilde{\boldsymbol{\omega}}$ is the interpolation of model output vectors of $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$, that is, $\boldsymbol{\omega} = f(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}})$ and $\boldsymbol{\omega}' = f(e(\mathbf{x}'), e(\mathbf{y}'); \hat{\boldsymbol{\theta}})$.

Algorithm 1: Proposed *AdvAug* function.

```
Input: A batch of source and target sentences
       $(\mathbf{X}, \mathbf{Y})$; the selection ratio $r_t$; the
       hyperparameter $\alpha$.
Output: Mini-batch losses $\mathcal{L}_{adv}$ and $\mathcal{L}_{aut}$
1  Function AdvAug($\mathbf{X}, \mathbf{Y}$):
2      foreach $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}, \mathbf{Y})$ do
3          $\boldsymbol{\omega} \leftarrow f(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}})$;
4          Sample two adversarial examples $(\mathbf{x}', \mathbf{y}')$
             and $(\mathbf{x}'', \mathbf{y}'')$ from $A_{(\mathbf{x},\mathbf{y})}$ by Eq. (4) ;
5          $\lambda \leftarrow \mathrm{Beta}(\alpha, \alpha)$ ;
6          $e(\tilde{\mathbf{x}}) \leftarrow m_\lambda(\mathbf{x}', \mathbf{x}'')$, $e(\tilde{\mathbf{y}}) \leftarrow m_\lambda(\mathbf{y}', \mathbf{y}'')$;
7          $\ell_i \leftarrow \ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), \boldsymbol{\omega})$ ;
8      end
9      Compute $\mathcal{L}_{adv}$ using $r_t$ and $\{\ell_i\}$ by Eq. (14) ;
10     $(\mathbf{X}', \mathbf{Y}') \leftarrow$ Shuffle $(\mathbf{X}, \mathbf{Y})$ ;
11     foreach $(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') \in (\mathbf{X}, \mathbf{Y}, \mathbf{X}', \mathbf{Y}')$ do
12         $\boldsymbol{\omega} \leftarrow f(e(\mathbf{x}), e(\mathbf{y}); \hat{\boldsymbol{\theta}})$;
13         $\boldsymbol{\omega}' \leftarrow f(e(\mathbf{x}'), e(\mathbf{y}'); \hat{\boldsymbol{\theta}})$;
14         $\lambda \leftarrow \mathrm{Beta}(\alpha, \alpha)$ ;
15         $e(\tilde{\mathbf{x}}) \leftarrow m_\lambda(\mathbf{x}, \mathbf{x}')$, $e(\tilde{\mathbf{y}}) \leftarrow m_\lambda(\mathbf{y}, \mathbf{y}')$ ;
16         $\tilde{\boldsymbol{\omega}} \leftarrow m_\lambda(\boldsymbol{\omega}, \boldsymbol{\omega}')$ ;
17         $\ell_i \leftarrow \ell(f(e(\tilde{\mathbf{x}}), e(\tilde{\mathbf{y}}); \boldsymbol{\theta}), \tilde{\boldsymbol{\omega}})$ +
             $\ell(f(e(\mathbf{x}), e(\mathbf{y}); \boldsymbol{\theta}), \ddot{\mathbf{y}})$ ;
18     end
19     Compute $\mathcal{L}_{aut}$ by averaging $\{\ell_i\}$ ;
20     return $\mathcal{L}_{adv}$, $\mathcal{L}_{aut}$
```

3.3 Training

Finally, the training objective in our *AdvAug* is a combination of the two losses:

$$\boldsymbol{\theta}^* = \mathop{\mathrm{argmin}}_{\boldsymbol{\theta}} \{\mathcal{L}_{aut}(\boldsymbol{\theta}) + \mathcal{L}_{adv}(\boldsymbol{\theta})\}. \qquad (17)$$

Here, we omit two bidirectional language model losses for simplicity, which are used to recommend word candidates to maintain semantic similarities (Cheng et al., 2019).

In practice, we need to compute the loss via mini-batch training. For the P_{aut}, we follow the pair sampling inside each mini-batch in *mixup*. It can avoid padding too much tokens because sentences of similar lengths are grouped within a mini-batch (Vaswani et al., 2017). For the P_{adv}, we sample a pair of examples from $A_{(\mathbf{x},\mathbf{y})}$ for each $(\mathbf{x}, \mathbf{y})$ and cover the distribution over multiple training epochs. The entire procedure to calculate the translation losses, $\mathcal{L}_{adv}(\boldsymbol{\theta})$ and $\mathcal{L}_{aut}(\boldsymbol{\theta})$, is presented in Algorithm 1. In a nutshell, for each batch of training examples, we firstly sample virtual examples from P_{adv} and P_{aut} by interpolating the embeddings of the adversarial or authentic training examples. Then we calculate the translation loss using their interpolated embeddings.

Method	Loss Config.	MT06	MT02	MT03	MT04	MT05	MT08
Vaswani et al. (2017)	$\mathcal{L}_{clean}$	44.57	45.49	44.55	46.20	44.96	35.11
Miyato et al. (2017)	-	45.28	45.95	44.68	45.99	45.32	35.84
Sato et al. (2019)	-	45.75	46.37	45.02	46.49	45.88	35.90
Cheng et al. (2019)	-	46.95	47.06	46.48	47.39	46.58	37.38
Sennrich et al. (2016a)*	-	46.39	47.31	47.10	47.81	45.69	36.43
Edunov et al. (2018)*	-	46.20	47.78	46.93	47.80	46.81	36.79
Ours	$\mathcal{L}_{mixup}$	45.12	46.32	44.81	46.61	46.08	36.00
	$\mathcal{L}_{aut}$	46.73	46.79	46.13	47.54	46.88	37.21
	$\mathcal{L}_{clean} + \mathcal{L}_{adv}$	47.89	48.53	**48.73**	48.60	48.76	39.03
	$\mathcal{L}_{aut} + \mathcal{L}_{adv}$	**49.26**	**49.03**	47.96	**48.86**	**49.88**	**39.63**
Ours*	$\mathcal{L}_{aut} + \mathcal{L}_{adv}$	**49.98**	**50.34**	**49.81**	**50.61**	**50.72**	**40.45**

Table 1: Baseline comparison on NIST Chinese-English translation. * indicates the model uses extra corpora and - means not elaborating on its training loss.

4 Experiments

4.1 Setup

We verify our approach on translation tasks for three language pairs: Chinese-English, English-French, and English-German. The performance is evaluated with the 4-gram BLEU score (Papineni et al., 2002) calculated by the *multi-bleu.perl* script. We report case-sensitive tokenized BLEU scores for English-French and English-German, and case-insensitive tokenized BLEU scores for Chinese-English. Note that all reported BLEU scores in our approach are from a single model rather than averaging multiple models (Vaswani et al., 2017).

For the Chinese-English translation task, the training set is the LDC corpus consisting of 1.2M sentence pairs. The NIST 2006 dataset is used as the validation set, and NIST 02, 03, 04, 05, 08 are used as the test sets. We apply byte-pair encoding (BPE) (Sennrich et al., 2016b) with 60K merge operations to build two vocabularies comprising 46K Chinese sub-words and 30K English sub-words. We use the IWSLT 2016 corpus for English-French translation. The training corpus with 0.23M sentence pairs is preprocessed with the BPE script with 20K joint operations. The validation set is test2012 and the test sets are test2013 and test2014. For English-German translation, we use the WMT14 corpus consisting of 4.5M sentence pairs. The validation set is newstest2013 whereas the test set is newstest2014. We build a shared vocabulary of 32K sub-words using the BPE script.

We implement our approach on top of the Transformer (Vaswani et al., 2017). The size of the hidden unit is 512 and the other hyperparameters are set following their default settings. There are three important hyperparameters in our approach, α in the Beta distribution and the word replacement ratio of $\gamma_{src} \in \xi_{src}$, and $\gamma_{tgt} \in \xi_{tgt}$ detailed in Eq. (4). Note that γ_{src} and γ_{tgt} are not new hyperparameters but inherited from (Cheng et al., 2019). We tune these hyperameters on the validation set via a grid search, *i.e.* $\alpha \in \{0.2, 0.4, 4, 8, 32\}$, $\gamma_{src} \in \{0.10, 0.15, 0.25\}$ and $\gamma_{tgt} \in \{0.10, 0.15, 0.30, 0.5\}$. For the *mixup* loss $\mathcal{L}_{mixup}$, α is fixed to 0.2. For the loss $\mathcal{L}_{aut}$ and L_{adv}, the optimal value of α is 8.0. The optimal values of $(\gamma_{src}, \gamma_{tgt})$ are found to be $(0.25, 0.50)$, $(0.15, 0.30)$ and $(0.15, 0.15)$ for Chinese-English, English-French and English-German, respectively, while it is set to $(0.10, 0.10)$ only for back-translated sentence pairs. β in Eq. (14) is set to 250K, 100K, 1M for Chinese-English, English-French and English-German. Unlike Cheng et al. (2019), we remove the learning of target language models to speed up the training. For each training batch, we introduce a batch of augmented adversarial examples and a batch of augmented authentic examples, which costs twice the vanilla training. For constructing adversarial examples, we solely compute the gradients for word embeddings which takes little time. After summing up the time of all steps, our total training time is about 3.3 times the vanilla training.

4.2 Main Results

Chinese-English Translation. Table 1 shows results on the Chinese-English translation task, in comparison with the following six baseline methods. For a fair comparison, we implement all these

Method	Loss Config.	English-French		English-German	
		test2013	test2014	newstest13	newstest14
Vaswani et al. (2017)	$\mathcal{L}_{clean}$	40.78	37.57	25.80	27.30
Sato et al. (2019)	–	41.67	38.72	25.97	27.46
Cheng et al. (2019)	–	41.76	39.46	26.34	28.34
Ours	$\mathcal{L}_{mixup}$	40.78	38.11	26.28	28.08
	$\mathcal{L}_{aut}$	41.49	38.74	26.33	28.58
	$\mathcal{L}_{aut} + \mathcal{L}_{adv}$	**43.03**	**40.91**	**27.20**	**29.57**

Table 2: Results on IWSLT16 English-French and WMT14 English-German translation.

methods using the Transformer backbone or report results from those papers on the same corpora.

1. The seminal Transformer model for NMT (Vaswani et al., 2017).
2. Following Miyato et al. (2017), we use adversarial learning to add continuous gradient-based perturbations to source word embeddings and extend it to the Transformer model.
3. Sato et al. (2019) leverage Miyato et al. (2017)'s idea into NMT by incorporating gradient-based perturbations to both source and target word embeddings and optimize the model with adversarial training.
4. Cheng et al. (2019) generate discrete adversarial examples guided by the gradients of word embeddings. Adversarial examples are used to both attack and defend the NMT model.
5. Sennrich et al. (2016a) translate monolingual corpora using an inverse NMT model and then augment the training data with them.
6. Based on Sennrich et al. (2016a), Edunov et al. (2018) propose three improved methods to generate back-translated data, which are *sampling*, *top10* and *beam+noise*. Among those, we choose *beam+noise* as our baseline method, which can be regarded as an approach to incorporating noise into data.

We first verify the importance of different translation losses in our approach. We find that both $\mathcal{L}_{aut}$ and $\mathcal{L}_{adv}$ are useful in improving the Transformer model. $\mathcal{L}_{adv}$ is more important and yields a significant improvement when combined with the standard empirical loss $\mathcal{L}_{clean}$ (cf. Eq. (1)). These results validate the effectiveness of augmenting with virtual adversarial examples. When we use both $\mathcal{L}_{aut}$ and $\mathcal{L}_{adv}$ to train the model, we obtain the best performance (up to 4.92 BLEU points on MT05). We also compare with the *mixup* loss. However, $\mathcal{L}_{mixup}$ is only slightly better than the standard empirical loss $\mathcal{L}_{clean}$.

Compared with the baseline methods without using extra corpora, our approach shows significant improvements over the state-of-the-art models. In particular, the superiority of $\mathcal{L}_{clean} + \mathcal{L}_{adv}$ over both Cheng et al. (2019) and Sato et al. (2019) verifies that we propose a more effective method to address adversarial examples in NMT. We also directly incorporate two adversarial examples to NMT models without interpolating their embeddings, but we do not observe any further gain over Cheng et al. (2019). This substantiates the superior performance of our approach on the standard data sets.

To compare with the approaches using extra monolingual corpora, we sample 1.25M English sentences from the Xinhua portion of the GIGAWORD corpus and list our performance in the last row of Table 1. When the back-translated corpus is incorporated, our approach yields further improvements, suggesting our approach complements the back-translation approaches.

English-French and English-German Translation. Table 2 shows the comparison with the Transformer model (Vaswani et al., 2017), Sato et al. (2019) and Cheng et al. (2019) on English-French and English-German translation tasks. Our approach consistently outperforms all three baseline methods, yielding significant 3.34 and 2.27 BLEU point gains over the Transformer on the English-French and English-German translation tasks, respectively. We also conduct similar ablation studies on the translation loss. We still find that the combination of $\mathcal{L}_{adv}$ abd $\mathcal{L}_{aut}$ performs the best, which is consistent with the findings in the Chinese-English translation task. The substantial gains on these two translation tasks suggest the potential applicability of our approach to more language pairs.

Input	但（但是）协议执行过程一波三折，致使和平进程一再受挫
Reference	however, implementation of the deals has witnessed ups and downs, resulting in continuous setbacks in the peace process
Vaswani et al. on Input	however, the process of implementing the agreement was full of twists and turns, with the result that the peace process suffered setbacks again and again.
on Noisy Input	the process of the agreement has caused repeated setbacks to the peace process.
Ours on Input	however, the process of implementing the agreement experienced twists and turns, resulting in repeated setbacks in the peace process.
on Noisy Input	however, the process of implementing the agreement experienced twists and turns, resulting in repeated setbacks in the peace process.

Table 3: Translation Examples of Transformer and our model for an input and its adversarial input.

Loss	$\alpha =$				
	0.2	0.4	4	8	32
$\mathcal{L}_{mixup}$	45.28	45.38	45.64	45.09	-
$\mathcal{L}_{aut}$	45.95	45.92	46.70	46.73	46.54
$\mathcal{L}_{clean}+\mathcal{L}_{adv}$	47.06	46.88	47.60	47.89	47.81

Table 4: Effect of α on the Chinese-English validation set. "-" indicates that the model fails to converge.

Method	0.00	0.05	0.10	0.15
Vaswani et al.	44.59	41.54	38.84	35.71
Miyato et al.	45.11	42.11	39.39	36.44
Sato et al.	45.75	44.04	41.25	38.78
Cheng et al.	46.95	44.20	41.71	39.89
Ours	**49.26**	**47.53**	**44.71**	**41.76**

Table 5: Results on artificial noisy inputs. The column lists results for different noise fractions.

4.3 Effect of α

The hyperparameter α controls the shape of the Beta distribution over interpolation weights. We study its effect on the validation set in Table 4. Notable differences occur when $\alpha < 1$ and $\alpha > 1$, this is because the Beta distribution show two different shapes with $\alpha = 1$ as a critical point. As we see, both $\mathcal{L}_{aut}$ and $\mathcal{L}_{adv}$ prefer a large α and perform better when $\alpha = 8$. Recall that when α is large, m_{λ} behaves similarly to a simple average function. In $\mathcal{L}_{mixup}$, $\alpha = 4$ performs slightly better, and a large $\alpha = 32$ will fail the model training. Although the result with $\alpha = 4$ appears to be slightly better, it consumes more iterations to train the model to reach the convergence, *i.e.* , 90K for $\alpha = 4$ vs. 20K for $\alpha = 0.2$. These indicate the differences between the proposed vicinity distributions and the one used in *mixup*.

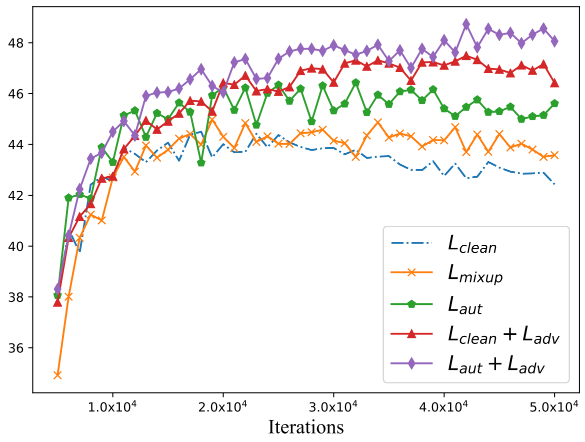

Figure 2: BLEU scores over iterations on the Chinese-English validation set.

4.4 Robustness to Noisy Inputs and Overfitting

To test robustness on noisy inputs, we follow Cheng et al. (2019) to construct a noisy data set by randomly replacing a word in each sentence of the standard validation set with a relevant alternative. The relevance between words is measured by the similarity of word embeddings. 100 noisy sentences are generated for each of them and then re-scored to pick the best one with a bidirectional language model. Table 5 shows the results on artificial noisy inputs with different noise levels. Our approach shows higher robustness over all baseline methods across all noise levels.

Figure 2 shows the evolution of BLEU scores during training. For $\mathcal{L}_{clean}$, the BLEU score reaches its peak at about 20K iterations, and then the model starts overfitting. In comparison, all of the training losses proposed in this paper are capable of resisting overfitting: in fact, even after 100K iterations, no significant regression is observed (not shown in this figure). At the same iteration, our results are consistently higher than both the empirical risk ($\mathcal{L}_{clean}$) and *mixup* ($\mathcal{L}_{mixup}$).

As shown in Table 3, the baseline yields an incorrect translation possibly because the word "danshi(但是)" seldom occurs in this context in our training data. In contrast, our model incorporates embeddings of virtual sentences that contain "danshi(但是)" or its synonym "dan(但)". This encourages our model to learn to push their embeddings closer during training, and make our model more robust to small perturbations in real sentences.

5 Related Work

Data Augmentation. Data augmentation is an effective method to improve machine translation performance. Existing methods in NMT may be divided into two categories, based upon extra corpora (Sennrich et al., 2016a; Cheng et al., 2016; Zhang and Zong, 2016; Edunov et al., 2018) or original parallel corpora (Fadaee et al., 2017; Wang et al., 2018; Cheng et al., 2019). Recently, *mixup* (Zhang et al., 2018) has become a popular data augmentation technique for semi-supervised learning (Berthelot et al., 2019) and overcoming real-world noisy data (Jiang et al., 2019). Unlike prior works, we introduce a new method to augment the representations of the adversarial examples in sequence-to-sequence training of the NMT model. Even without extra monolingual corpora, our approach substantially outperforms the widely-used back-translation methods (Sennrich et al., 2016a; Edunov et al., 2018). Furthermore, we can obtain even better performance by including additional monolingual corpora.

Robust Neural Machine Translation. It is well known that neural networks are sensitive to noisy inputs (Szegedy et al., 2014; Goodfellow et al., 2014), and neural machine translation is no exception. Thus improving the robustness of NMT models has become a popular research topic (e.g., Belinkov and Bisk, 2018; Sperber et al., 2017; Ebrahimi et al., 2018; Cheng et al., 2018, 2019; Karpukhin et al., 2019; Li et al., 2019). Many of these studies focus on augmenting the training data to improve robustness, especially with adversarial examples (Ebrahimi et al., 2018; Cheng et al., 2019; Karpukhin et al., 2019; Michel et al., 2019). Others also tried to deal with this issue by finding better input representations (Durrani et al., 2019), adding adversarial regularization (Sato et al., 2019) and so on. In contrast to those studies, we propose the vicinity distribution defined in a smooth space by interpolating discrete adversarial examples. Experimental results show substantial improvements on both clean and noisy inputs.

6 Conclusion

We have presented an approach to augment the training data of NMT models by introducing a new vicinity distribution defined over the interpolated embeddings of adversarial examples. To further improve the translation quality, we also incorporate an existing vicinity distribution, similar to *mixup* for observed examples in the training set. We design an augmentation algorithm over the virtual sentences sampled from both of the vicinity distributions in sequence-to-sequence NMT model training. Experimental results on Chinese-English, English-French and English-German translation tasks demonstrate the capability of our approach to improving both translation performance and robustness.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. 2001. Vicinal risk minimization. In *Advances in neural information processing systems*, pages 416–422.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Association for Computational Linguistics*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *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 *Association for Computational Linguistics*.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov, and Preslav Nakov. 2019. One size does not fit all: Comparing nmt representations of different granularities. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *Proceedings of COLING*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Empirical Methods in Natural Language Processing*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Association for Computational Linguistics*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*.

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*.

Lu Jiang, Di Huang, and Weilong Yang. 2019. Synthetic vs real: Deep learning on controlled noise. *arXiv preprint arXiv:1911.09781*.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*.

Xian Li, Paul Michel, Antonios Anastasopoulos, Yonatan Belinkov, Nadir Durrani, Orhan Firat, Philipp Koehn, Graham Neubig, Juan Pino, and Hassan Sajjad. 2019. Findings of the first shared task on machine translation robustness. *arXiv preprint arXiv:1906.11943*.

Paul Michel, Xian Li, Graham Neubig, and Juan Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a methof for automatic evaluation of machine translation. In *Association for Computational Linguistics*.

Motoki Sato, Jun Suzuki, and Shun Kiyono. 2019. Effective adversarial regularization for neural machine translation. In *Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Association for Computational Linguistics*.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation*.

Christian Szegedy, Wojciech Zaremba, Sutskever Ilya, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Empirical Methods in Natural Language Processing*.

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

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Empirical Methods in Natural Language Processing*.