# Amortized Noisy Channel Neural Machine Translation

**Richard Yuanzhe Pang**
New York University
yzpang@nyu.edu

**He He**
New York University

**Kyunghyun Cho**
New York University
Genentech
CIFAR Fellow

## Abstract

Noisy channel models have been especially effective in neural machine translation (NMT). However, recent approaches like "beam search and rerank" (BSR) incur significant computation overhead during inference, making real-world application infeasible. We aim to study if it is possible to build an amortized noisy channel NMT model such that when we do greedy decoding during inference, the translation accuracy matches that of BSR in terms of reward (based on the source-to-target log probability and the target-to-source log probability) and quality (based on BLEU and BLEURT). We attempt three approaches to train the new model: knowledge distillation, 1-step-deviation imitation learning, and Q learning. The first approach obtains the noisy channel signal from a pseudo-corpus, and the latter two approaches aim to optimize toward a noisy-channel MT reward directly. For all three approaches, the generated translations fail to achieve rewards comparable to BSR, but the translation quality approximated by BLEU and BLEURT is similar to the quality of BSR-produced translations. Additionally, all three approaches speed up inference by 1–2 orders of magnitude.

## 1 Introduction

Noisy channel models have been traditionally used in many tasks, including speech recognition (Jelinek, 1997), spelling correction (Brill and Moore, 2000), question answering (Echihabi and Marcu, 2003), and statistical machine translation (Koehn et al., 2003). In machine translation (MT), the probability of the source sentence conditioned on the target-language generation is taken into account when generating a translation. In modern neural machine translation (NMT), the noisy channel approach is successful and often indispensable in many recent top-performing machine translation systems (Yee et al., 2019; Ng et al., 2019; Chen et al., 2020; Yu et al., 2020; Tran et al., 2021).

One widely used approach of noisy channel NMT is "beam search and rerank" (BSR). Assume a trained forward translator and a trained reverse translator,[1] BSR decoding consists of two steps: first, decode using beam search with a large beam size from the forward translation model and store the entire beam; second, rerank the beam using a reward which is the sum of the forward translation log probability and the reverse log probability. This approach incurs significant computational overhead, given the need to decode a large beam (usually with beam size 50–100) from the forward translator and the need to feed the large beam through the reverse translator. The computational cost is especially problematic if the practitioner has a large volume of translation requests, or if the system is mobile-based and requires offline translation.

We thus aim to learn a separate neural network with an identical architecture as the forward translator such that at inference time, when we do *greedy decoding* using this new network, we investigate how much translation accuracy would be sacrificed. Specifically, we investigate how forward/reverse rewards of the translations as well as the translation quality (approximated by BLEU and BLEURT) would compare to those of BSR-generated translations.[2]

The paper explores three approaches, with increasingly more exploration when optimizing the reward. (1) Knowledge distillation (KD) from a pseudo-training-corpus generated by BSR: we can treat the BSR-generated corpus as the oracle, and KD can be interpreted as behavioral cloning. (2) a one-step-deviation imitation learning strategy (IL) where given a fixed sequence of target-language

---

[1]Forward: from the source language to the target language; reverse: from the target language to the source language.

[2]Although we need time to train the separate network, at inference time and during the actual large-scale user-facing deployment, we would be able massively cut down computational cost in the long run. In this paper, we aim to investigate the accuracy of such decoded translations.

tokens, we adjust each time-step's probability distribution over the vocabulary such that the resulting distribution minimizes an energy function used in BSR reranking, and (3) Q learning which explicitly learns the scoring function used in BSR reranking.

We experiment on three datasets (IWSLT'14 De-En, WMT'16 Ro-En, and WMT'14 De-En). Experimental results show that all three approaches speed up inference by 50–100 times. The approaches fail to achieve comparable rewards to BSR, but compared to the non-BSR baselines, the approaches achieve much higher reverse rewards (i.e., $\log p_r(\boldsymbol{x} \mid \boldsymbol{y})$ where $p_r$ is the reverse translator) at the expense of forward rewards (i.e., $\log p_f(\boldsymbol{y} \mid \boldsymbol{x})$ where $p_f$ is the forward translator). Meanwhile, the approaches achieve a translation quality (approximated by BLEU and BLEURT) that is comparable to that of BSR. In particular, IL's BLEURT scores is significantly higher than those of beam search, across all three datasets; IL's BLEURT scores are not significantly different from BSR's scores, across three datasets.

## 2   Background

### 2.1   Neural Machine Translation

NMT systems usually model the distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ where $\boldsymbol{x} = (x_1, x_2, \ldots, x_{T_s})$ is a source-language sequence and $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$ is a target-language sequence. Most NMT systems use an autoregressive factorization:

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{t=1}^{T} \log p_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}),$$

where $\boldsymbol{y}_{<t} = (y_1, y_2, \ldots, y_{t-1})$, and $p_\theta$ is parameterized with a neural network. At test-time, to decode a translation given a source sentence, greedy decoding and beam search are most commonly used. Both are approximate search methods to find the highest-scoring translations.

### 2.2   Beam Search and Rerank (BSR)

BSR has appeared in a number of top-performing models, including many winning submissions of the WMT competitions (Ng et al., 2019; Chen et al., 2020; Yu et al., 2020; Tran et al., 2021). The intuition of BSR is to take advantage of the reverse translator during decoding. Specifically, we do beam search with a large beam size $b$ (usually 50–100) to obtain $b$ candidate translations. Then, we

rerank the candidates using the scoring function:

$$\log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y}) + \gamma' \log p_{lm}(\boldsymbol{y}),$$

where $\gamma$ and $\gamma'$ are tuned in $[0, 2]$. Without access to a language model trained on a huge target-language monolingual external corpus, if we use $\log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y})$ as the ranking criteria, BSR also provides a significant performance gain. With a large beam size, this approach performs better than the "two-step beam search" approach (Yu et al., 2017; Yee et al., 2019).

## 3   Amortized Noisy-Channel NMT

One common problem with the above approaches is the inference-time computation overhead. If a translation system needs to translate a high volume of texts, then the test-time computational efficiency is crucial. Thus, our goal is to use a network to approximate such a noisy channel NMT system, while having the same inference-time computational cost as *greedily decoding from $p_f$*. Specifically, we want our translations to maximize the following objective:

$$R(\boldsymbol{x}, \boldsymbol{y}) = \log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y}), \quad (1)$$

where $\gamma > 0$ is some fixed coefficient. Using the autoregressive factorization, the forward reward $\log p_f(\boldsymbol{y} \mid \boldsymbol{x})$ equals $\sum_{t=1}^{|\boldsymbol{y}|} \log p_f(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, and the reverse reward $\log p_r(\boldsymbol{x} \mid \boldsymbol{y})$ equals $\sum_{t=1}^{|\boldsymbol{x}|} \log p_r(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, \boldsymbol{y})$.

**Goal: Investigating if greedily decoding from our new models leads to successful amortization.** Three approaches are shown in this section. We do greedy decoding from the obtained models, and we investigate whether amortization is successful as follows.

- First, we examine if decoding is faster than BSR. This aspect is guaranteed given that we would do greedy decoding from our new network which has the same architecture as $p_f$.

- Next, we examine if both the forward and reverse rewards of the translations are close to the forward and reverse rewards of the translations generated by BSR, respectively.

- Finally, we examine the translation quality by checking if BLEU and BLEURT scores of our model's translations are close to those of BSR-produced translations.

## 3.1 Approach 1: Knowledge Distillation (KD)

KD has been used to amortize beam search (Chen et al., 2018). It is also effective in NMT in general (Kim and Rush, 2016; Freitag et al., 2017; Tan et al., 2019; Tu et al., 2020). Here we adapt a simple version of KD for amortized noisy-channel decoding.

First, train a forward translator $p_f$ and a reverse translator $p_r$ using maximum likelihood estimation. Then, do BSR on the entire training set to obtain the pseudo-corpus. In particular, we ignore the $p_{lm}$ term in this paper given that it usually requires a big language model, and the inclusion of the term is orthogonal to our goal of reducing inference time.[3]

Next, we train a separate "knowledge distilled" model $p_{KD}$ on this new pseudo-corpus (i.e., with the original source-language sentences and the BSR-generated target-language sentences). This objective is equivalent to minimizing the KL-divergence between the distribution induced by the pseudo-corpus obtained through BSR and our model distribution.

At inference time, we greedily decode from $p_{KD}$.

## 3.2 Approach 2: 1-Step-Deviation Imitation Learning (IL)

Define a network $A_\phi$ such that it takes in the source sentence and a target-language prefix, and $A_\phi(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ outputs a $|\mathcal{V}|$-dimensional probability distribution corresponding to the $t$-th time-step. Moreover, $A_\phi$ and $p_f$ have the same architecture. In autoregressive text generation, to learn $A_\phi$ such that it is close to an existing network $p_\theta$, imitation learning seeks to optimize $\phi$ as follows:

$$\arg\min_\phi \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}_{<t})} \mathcal{L}(A_\phi(\cdot|\boldsymbol{x}, \boldsymbol{y}_{<t}), p_\theta(\cdot|\boldsymbol{x}, \boldsymbol{y}_{<t})),$$

where one example of $\mathcal{L}$ is the cross entropy.

**Forward energy.** Inspired by ENGINE (Tu et al., 2020), in the context of noisy channel NMT, define the forward sub-energy $E_t^f$, which is a function of

$\phi$, as follows:[4]

$$E_t^f(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) = -A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})^\top$$
$$\log p_f(\cdot \mid A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<1}), \ldots, A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}), \boldsymbol{x}).$$

Suppose we have a source sentence $\boldsymbol{x}$ and a sequence of prefix distributions $\hat{\boldsymbol{y}}_{<1}, \ldots, \hat{\boldsymbol{y}}_{<T}$. We call $A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})$ the $t$-th step distribution according to $A_\phi$. Intuitively, given a source and a fixed sequence of prefixes, we learn $A_\phi$ such that the resulting $t$-th-step distribution matches the forward conditional probability (measured by $p_f$) − the latter depends on the source $\boldsymbol{x}$ and the prefix distributions.

**Reverse energy.** Next, we define the reverse sub-energy as follows:

$$E_t^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) = -\text{onehot}(\boldsymbol{x}_t)^\top \log p_r(\cdot \mid \boldsymbol{x}_{<t},$$
$$A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<1}), \ldots, A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<T})).$$

Intuitively, we also learn $A_\phi$ such that the one-hot distributions corresponding to the source words should match the reverse conditional probability (measured by $p_r$).

**Trajectories.** In the above equations, $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_T)$. $\hat{y}_t$ comes from two sources, with probability $p$ and $1 - p$ for each minibatch during training (Section 4.2): (i) $\hat{\boldsymbol{y}}_t = \arg\max_{v \in \mathcal{V}} A_\phi(v \mid \boldsymbol{x}, \hat{y}_{<t})$ and $\hat{y}_{<1} = \varnothing$; in other words, given that $A_\phi(\cdot \mid \boldsymbol{x}, \hat{y}_{<t})$ is a probability distribution, we use the most likely token as $\hat{y}_t$. (ii) For the second source, let $\hat{v}_t$ be the $t$-th token of the BSR-obtained sequence, so that we can expose our model to BSR-prefixes, which are the optimal prefixes.

**Final objective.** We train $A_\phi$ using the following objective:

$$\min_\phi \sum_{\boldsymbol{x}} \left[ \sum_{t=1}^T E_t^f(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) + \gamma \sum_{t'=1}^{|\boldsymbol{x}|} E_{t'}^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) \right].$$

During inference, we greedily decode from $A$.

---

[3]Generating the pseudo-corpora can be paralleled. If the system is deployed in the real world, we argue that the amount of computation used to generate the pseudo-corpus is negligible, compared to the aggregate amount of computation for inference.

[4]If we compute $p_f(\cdot \mid y_1, y_2, \ldots, y_{t-1}, \boldsymbol{x})$ where $y_i$'s correspond to token IDs of a partial translation in the target language, then we would first look up the $y_i$-th row of the embedding matrix $E_{emb}$ and use this row to embed $y_i$; equivalently, we can use the product $\text{onehot}(y_i)^\top E_{emb}$ to embed $y_i$. In the case of $E_t^f$, the prefixes used in $p_f$ are distributions instead of tokens. We can use $A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<i})^\top E_{emb}$ to represent the $i$-th token embedding. The embedding strategy is similar for $E_t^r$ below.

## 3.3 Approach 3: Q Learning

A well-motivated approach is to use Q learning (Watkins and Dayan, 1992; Sutton and Barto, 1998) to explicitly learn a reward function $Q$, with the goal that when we greedily decode from $Q$, the generations maximize the reward shown in Eq. (1).

Let us view machine translation as a sequential decision-making process. At time-step $t$, given a state $s_t = (\boldsymbol{y}_{<t}, \boldsymbol{x})$, a policy takes an action $a_t \in \mathcal{V}$, transits to the next state $s_{t+1} = (\boldsymbol{y}_{<(t+1)}, \boldsymbol{x})$,[5] and receives a reward $r_t$.

### 3.3.1 Background on Q Learning

In Q learning, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a function such that $Q^\pi(s_t, a_t)$ produces the expected return after seeing state $s_t$, taking action $a_t$, and following policy $\pi$; i.e., $Q^\pi(s_t, a_t) = \mathbb{E}[\sum_{t'=t}^{\infty} r_{t'} | s_t, a_t, \pi]$ assuming discount factor 1. We further define $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to be the optimal action-value function: $Q^*(s_t, a_t) = \max_\pi \mathbb{E}[\sum_{t'=t}^{\infty} r_{t'} | s_t, a_t, \pi]$, which is the maximum return achievable by following any strategy *after* seeing a state $s_t$ and taking an action $a_t$. In particular, $Q^*$ solves the Bellman Equation (Sutton and Barto, 1998):

$$Q^*(s_t, a_t) = r_t + \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}),$$

assuming discount factor 1 and given deterministic transition dynamics (in our machine translation scenario) after taking action $a_t$ given state $s_t$.

Traditionally, the $Q$ function is implemented as a matrix of size $|\mathcal{S}| \times |\mathcal{A}|$, which is intractable in the case of MT due to the combinatorial nature of the state space. We thus use function approximation to tackle this issue of intractability: we follow Mnih et al. (2015) and use a deep neural network trained with experience replay and target networks to approximate the Q learning.

Deep Q learning draws samples from a set of trajectories $\mathcal{B}$, and the neural network $Q$ aims to predict $Q^*$ by learning based on minimizing the following squared loss.

$$L(\phi) = \frac{1}{|\mathcal{B}|} \sum_{(s_t, a_t, s_{t+1}, r_t) \sim \text{Uniform}(\mathcal{B})}$$
$$\left[ (r_t + \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t))^2 \right],$$

where $\phi$ is the parameter to $Q$, and $Q'$ is a slightly old copy of $Q$.[6]

---

[5]$\boldsymbol{y}_{<(t+1)}$ equals $\boldsymbol{y}_{<t}$ concatenated with the action $a_t$.
[6]In other words, after a fixed number of optimization steps, we update $Q'$ by $Q$.

---

**Algorithm 1:** Q learning for amortized noisy channel NMT

Given $p_f$, $p_r$, and a parallel translation dataset $\mathcal{D}$.

**while** *not converged* **do**

Collect training trajectories (§3.3), and sample a mini-batch $\mathcal{B}$.

Compute target $R_t$:

- if $t < T$, then $R_t = r_t + \max_{a_{t+1}} Q'_\phi(s_{t+1}, a_{t+1})$;

- if $t = T$, then $R_t = r_T$.

Update $\phi$ (using gradient descent) by the objective

$$\arg\min_\phi \left[ Q_\phi(s_t, a_t) - R_t \right]^2.$$

Update $Q'_\phi$: $Q'_\phi \leftarrow Q_\phi$ every $K$ steps.

**end**

---

### 3.3.2 Q Learning for Amortized Noisy Channel MT

To model the noisy-channel NMT, given a target-language sequence $\boldsymbol{y}$ and its length $T$, we have reward $\boldsymbol{r} = (r_1, \dots, r_T)$, where

$$r_t = \begin{cases} \log p_f(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}), & \text{if } t < T, \\ \log p_f(y_T | \boldsymbol{y}_{<T}, \boldsymbol{x}) + \\ \qquad \gamma \cdot \log p_r(\boldsymbol{x} | \boldsymbol{y}), & \text{if } t = T. \end{cases} \quad (2)$$

We construct $Q$ to have the same architecture as $p_f$ without the final softmax layer. $Q$ is trained using Algorithm 1 which is adapted from deep Q learning originally applied to Atari games (Mnih et al., 2015), given that we aim to best leverage the existing off-policy trajectories from different sources. The full algorithm is shown in Algorithm 1.

In short, our algorithm says that given a trajectory $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{r})$, at time-step $t < T$, we want the scalar $Q(s_t, a_t)$ to be close to the sum of the $t$-th step reward and the *most optimistic* future return, had we taken action $a_t$ at time-step $t$. At time-step $T$, we want $Q(s_T, a_T) = Q((\boldsymbol{y}_{<T}, \boldsymbol{x}), \langle \text{eos} \rangle)$ to be close to $r_T$, as defined in Eq. (2).

To generate the $t$-th token at inference-time, we do greedy decoding using $Q$ as follows: $\hat{y}_t = \arg\max_{a_t \in \mathcal{V}} Q(s_t, a_t)$.

**Trajectories.** The off-policy algorithm shown in Algorithm 1 requires trajectories, i.e., $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{r})$ tuples. The trajectories come from two sources.

**(1) $Q$-based trajectories.** In this category, we have two ways of obtaining $\boldsymbol{y}$: (1a) Boltzmann exploration (Sutton, 1990)[7] and (1b) greedy decoding based on $Q$. At the start of the optimization, however, most of the $Q$-generated sequences are very far from target sequences. The lack of high-reward sequences prevents Q learning from efficient optimization. Therefore, we also inject reasonably good trajectories from the beginning of training by utilizing both ground-truth sequences as well as $p_f$-based sequences. We thus need the next category of sources.

**(2) $p_f$-based trajectories.** The target-language sequences are obtained by decoding using $p_f$; please find more details in Appendix A.1.[8]

# 4 Experimental Setup

## 4.1 Tasks and Models

We experiment on three translation tasks: IWSLT 2014 German to English (IWSLT'14 De-En; Cettolo et al., 2014) which has a small training set (train/dev/test size: 160,239/7,283/6,750), WMT 2016 Romanian to English (WMT'16 Ro-En; Bojar et al., 2016) which has a medium-sized training set (train/dev/test size: 608,319/1,999/1,999), and WMT 2014 German to English (WMT'14 De-En; Bojar et al., 2014) which has a moderately large training set (train/dev/test size: 4,500,966/3,000/3,003). Each of the transformer models (the $p_{\mathrm{KD}}$ in KD, the $A$ in IL, the $Q$ function in Q learning) has the same number of parameters as the original MLE-trained forward translator $p_f$. The model for IWSLT'14 De-En is the smallest, and the model for WMT'14 De-En is the largest. The detailed settings can be found in Appendix B. BLEU scores in this paper are computed with sacre-BLEU (Post, 2018). BLEURT scores are computed

---

[7]Recall that at time-step $t$, $Q(s_t, a_t) \in \mathbb{R}$ for each $a_t \in V$. Therefore, $Q(s_t, \cdot) \in \mathbb{R}^{|\mathcal{V}|}$. We turn the vector of real numbers to a categorical distribution by softmax with temperature $\gamma_b$. Then, the sequences in the trajectories are obtained by sampling from the aforementioned distribution. In practice, for each sequence, we use a temperature $\gamma_b$ sampled from $\mathrm{Uniform}(0, 1.5)$. One can think of this strategy as a variant of $\epsilon$-greedy which is typically used in Q learning.

[8]We have also experimented with gold-standard trajectories from the parallel translation dataset $\mathcal{D}$, but the inclusion of such trajectories do not lead to better rewards of $Q$-generated translations.

---

using BLEURT-20-D12 (Sellam et al., 2020), a recent RemBERT-based checkpoint that achieves high human agreement. The models we experiment on are shown in Table 1.

## 4.2 Hyperparameters

The architecture and optimization details of $p_f$ and $p_r$ are shown in Appendix B. When training $p_f$ and $p_r$, we validate the model performance after each epoch, and select the model that corresponds to the best dev set BLEU.

$\gamma$ is the coefficient multiplied to the reverse reward, when computing the total reward in Eq. (1); $\gamma$ and BSR beam size $b$ are tuned on dev set BLEU using BSR. We choose $\gamma = 0.9$ and $b = 100$ for IWSLT'14 De-En; $\gamma = 0.5$ and $b = 70$ for WMT'16 Ro-En; $\gamma = 0.5$ and $b = 50$ WMT'14 De-En. See Appendix B for details.

For training the IL-based network, the learning rate is selected from $\{10^{-6}, 5 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$. We use weight decay of $10^{-4}$. Dropout rate is selected from $\{0, 0.05, 0.1, 0.3\}$; we find that a dropout rate of 0 or 0.05 always works the best. We use a fixed max batch length (i.e., the max number of input tokens in a batch) of 4,096 tokens. The probability $p$, described in Section 3, is selected from $\{0, 0.1, 0.5, 0.9, 1\}$; we find that $p = 0.1$ or $p = 0.5$ usually works the best. We accumulate gradients and do gradient descent once every $k$ steps for computational reasons. $k$ is selected from $\{4, 8, 16\}$. We find that the IL approach relies on a good initialization, so we use $p_{\mathrm{KD/nc}}$ to initialize the new network.

For Q learning, the synchronization frequency $K$ in Algorithm 1 is selected from $\{10, 20, 30, 50, 150\}$. The learning rate is tuned in $\{10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$. We use weight decay of $10^{-4}$. Dropout rate is tuned in $\{0, 0.01, 0.05, 0.1\}$; we find that a dropout rate of 0 always works the best. We use a fixed max batch length 4096. We tune the number of steps per gradient update in $\{4, 8, 16\}$; a large number effectively increases the batch size. The ratio for different trajectories is described in Appendix A.1. Furthermore, we find that training $Q$ with a small $\gamma$ at the beginning stabilizes the training, so we first use $\gamma = 0.1$ and train till convergence, and then increase $\gamma$ by 0.2 increment, and we reiterate the process until reaching the desired $\gamma$.

We use the Adam optimizer (Kingma and Ba, 2014) for all experiments. We cap the maximum

| | IWSLT'14 De-En | | | WMT'16 Ro-En | | | WMT'14 De-En | | |
|---|---|---|---|---|---|---|---|---|---|
| | $b$ | fwd reward mean (std) | rvs reward mean (std) | $b$ | fwd reward mean (std) | rvs reward mean (std) | $b$ | fwd reward mean (std) | rvs reward mean (std) |
| $p_f$ | 1 | -9.1 (7.7) | -35.4 (39.9) | 1 | -9.5 (11.5) | -41.0 (50.1) | 1 | -11.0 (6.3) | -31.5 (24.6) |
| $p_f$ | 5 | -8.6 (7.0) | -34.2 (38.5) | 5 | -9.0 (8.5) | -40.2 (48.2) | 7 | -10.4 (5.5) | -29.9 (21.5) |
| BSR | 100 | -9.4 (6.8) | -25.7 (32.5) | 70 | -10.0 (6.0) | -29.7 (41.9) | 50 | -10.7 (5.3) | -23.6 (16.3) |
| KD | 1 | -13.8 (13.9) | -28.0 (32.7) | 1 | -17.2 (26.3) | -35.4 (44.6) | 1 | -14.8 (9.1) | -24.0 (16.7) |
| IL | 1 | -13.3 (13.2) | -27.9 (32.3) | 1 | -17.2 (30.9) | -34.3 (45.3) | 1 | -14.6 (8.9) | -23.6 (15.9) |
| Q learning | 1 | -13.7 (21.4) | -29.9 (35.1) | 1 | -11.6 (19.7) | -39.1 (52.9) | 1 | -14.4 (9.9) | -24.9 (17.5) |
| reference data | – | -38.8 (39.7) | -45.2 (46.6) | – | -55.3 (51.1) | -59.0 (54.2) | – | -36.8 (24.4) | -36.8 (23.0) |

Table 1: Mean and standard deviation (across sequences) of test set forward and reverse rewards for translations. $b$ refers to beam size during inference.

| | IWSLT'14 De-En | WMT'16 Ro-En | WMT'14 De-En |
|---|---|---|---|
| $p_f$ (greedy decoding) | 33.65 (0.06) | 33.23 (0.14) | 30.39 (0.13) |
| $p_f$ (beam search) | 34.54 (0.08) | 33.98 (0.15) | 31.78 (0.08) |
| BSR | 35.43 (0.06) | 34.81 (0.09) | 32.15 (0.14) |
| KD | 35.39 (0.04) | 33.95 (0.10) | 31.71 (0.05) |
| IL | 35.61 (0.09) | 34.65 (0.07) | 31.90 (0.07) |
| Q learning | 34.60 (0.08) | 34.31 (0.15) | 31.60 (0.19) |

Table 2: Test set sacreBLEU (mean & standard deviation of three runs using different random seeds). IL performs the best among the three proposed methods.

| | IWSLT'14 De-En | WMT'16 Ro-En | WMT'14 De-En |
|---|---|---|---|
| $p_f$ (greedy decoding) | 62.40 (0.04) | 61.14 (0.10) | 64.83 (0.10) |
| $p_f$ (beam search) | 63.21 (0.07) | 61.42 (0.15) | 65.79 (0.08) |
| BSR | 64.15 (0.05) | 62.67 (0.13) | 66.32 (0.12) |
| KD | 63.88 (0.04) | 61.78 (0.10) | 66.00 (0.07) |
| IL | 63.94 (0.13) | 62.35 (0.16) | 66.14 (0.08) |
| Q learning | 63.25 (0.07) | 61.70 (0.18) | 65.92 (0.14) |

Table 3: Test set BLEURT-20-D12 (mean & standard deviation of three runs). IL performs the best among the three proposed methods. Significance test is conducted in Table 8, which shows that IL's scores are significantly better than the scores by beam search; in addition, IL's scores are not significantly different from BSR's scores.

length of the translation at $1.2T_s + 20$ during decoding, where $T_s$ is the length of a source sentence. All implementation is based on fairseq (Ott et al., 2019). Each experiment is run on one NVIDIA RTX 8000 GPU.

## 5 Results

### 5.1 Preliminary Analysis

**Inference speed.** Using any of the three proposed approaches achieves a significant speedup, given that the three approaches all use greedy decoding. We quantify this speedup experimentally.

During inference, we maximize the memory usage of a single NVIDIA RTX 8000 GPU by finding the largest batch length in the form of $2^k$ where $k$ is a positive integer.[9] In the IWSLT'14 De-En task, the inference speed (sequences per second) for BSR is 11. The speed for "greedy by $p_f$" is around 1050, and the decoding speed for any of three proposed approaches is also similar.

**Rewards.** First, comparing the three approaches to greedy decoding or beam search from $p_f$, we see that the three approaches achieve smaller forward rewards, but much larger reverse rewards. This observation is expected given that the three approaches consider both the forward and reverse rewards, while greedy decoding or beam search from $p_f$ only consider forward rewards. Second, comparing the three approaches against BSR, the three approaches achieve both smaller forward rewards and smaller reverse rewards. However, we find this a reasonable trade-off to be made between decoding latency and rewards, as all these approaches are 1–2 orders of magnitude faster in decoding.

Among the three approaches, KD and IL achieve a better balance between forward and reverse rewards. This observation can be explained by the difference in how the reverse reward is presented among the three approaches. In KD and IL, the learning signal by reverse rewards is implicitly spread throughout all the steps in a sequence. In other words, changing the conditional distribution in each time-step would adjust the loss in KD and the reverse energies in IL. In Q learning, the reverse reward is sparse: it only appears at the end of the sequence, unlike the forward reward which is spread throughout all the steps. This makes it easier for Q learning to maximize the forward reward compared

---

[9]Batch length means the number of tokens in a batch (instead of the number of sequences).
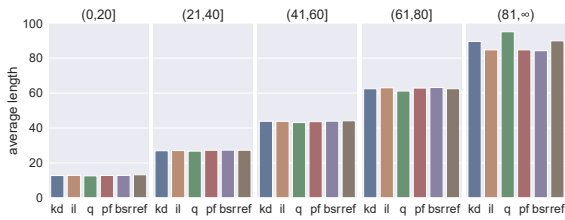
Figure 1: Average length bucketed by length of the source sentence. The five buckets contain 453, 877, 376, 92, 26 sentences, respectively. The six systems are KD, IL, Q learning, beam search by $p_f$, BSR, and reference translations, respectively. In the longest length bucket, Q learning produces translations that are longer than translations by other systems.



Figure 2: Repetition rate ("token rep") bucketed by length of the source sentence. The five buckets contain 453, 877, 376, 92, 26 sentences, respectively. N.B.: In the last bucket, "token rep" for $Q$-generated translations is around 0.31, and the bar is truncated.

to the reverse reward which requires many more updates to be propagated toward the earlier time steps.

**Translation quality.** The three approaches achieve BLEU and BLEURT scores that are comparable to those by BSR. Moreover, the three approaches achieve BLEU scores that are much better than "greedy decoding from $p_f$" which has the same computational budget; they are often better than "beam search from $p_f$" as well. In particular, Table 3 shows that IL's BLERUT scores are significantly higher than the scores of beam search across all three datasets. In addition, IL's BLEURT scores are not significantly different from BSR across all three datasets. Therefore, our approaches are able to generate translations with similar quality as those by BSR, while being 5–7 times as fast as beam search and 50–100 times as fast as BSR.

## 5.2 Analysis of Translations

In Q learning, the reverse reward is only presented as a learning signal at the end of each sequence. As observed earlier by Welleck et al. (2020), the length of the generations may inform us of the possible degeneracies, such as excessive repetitions.

Therefore, we analyze WMT'16 Ro-En translations generated by different systems, and we first examine the lengths of translations in different source length buckets. Figure 1 shows that the lengths by different systems are similar in the first four buckets, but in the longest source length bucket $(81, \infty)$, Q learning produces longer translations.

Closer examination of the translations reveal that Q learning produces degenerate translations with extensive repetitions when the source sentences are among the longest in the entire dev set; other
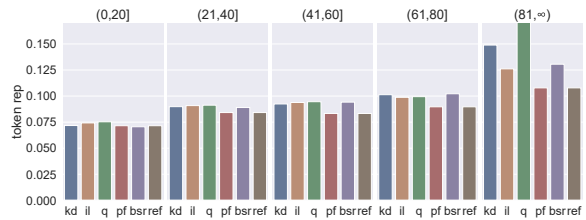
models do not have this issue. Some randomly selected examples are shown in Table 6.

To confirm this finding, we analyze repetitions by source-length buckets. We define "token rep" to be the percentage of tokens that have appeared in the immediately preceding 5-grams:

$$\frac{\sum_{i=1}^{N} \sum_{t=6}^{T^{(i)}} \mathbb{1}\left[ y_t^{(i)} \in \{y_{t-5}^{(i)}, \ldots, y_{t-1}^{(i)}\}\right]}{\sum_{i=1}^{N} \sum_{t=6}^{T^{(i)}} 1},$$

where the superscript indicates the $i$-th example, and $N$ indicates the number of translations.

We see from Figure 2 that for the longest source-sentence length bucket $(81, \infty)$, $Q$ produces translations with a significantly larger 5-gram repetition rate. Moreover, beam search from the forward only model $p_f$ exhibits a behavior most similar to reference translations. We leave it for the future to study the cause behind an elevated level of repetition in noisy-channel decoding.

Next, to compare translation similarity among different approaches, we examine the corpus-level BLEU score between each pair of approaches, averaged between two directions. By Table 5, translations by BSR is similar to those produced by $p_f$ and Q learning, compared to KD and IL. Now we compare the translations produced by the three approaches. Translations by KD are more similar to IL, compared to BSR and Q learning. This is in line with our intuition that KD and IL differ from Q learning, given that how the reverse reward is presented is different between KD/IL and Q learning.

## 5.3 Further Analysis

**KD.** One may wonder whether the improvements in KD arise from the KD procedure or because we use BSR when constructing the pseudo-corpus. We therefore experiment with another

| | IWSLT'14 De-En | | | | WMT'16 Ro-En | | | | WMT'14 De-En | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $b$ | fwd reward mean (std) | rvs reward mean (std) | BLEU | $b$ | fwd reward mean (std) | rvs reward mean (std) | BLEU | $b$ | fwd reward mean (std) | rvs reward mean (std) | BLEU | |
| $p_{\text{KD/beam}}$ trained by $(X, \widetilde{Y}_{\text{beam}})$ | 1 | -13.3 (13.4) | -31.6 (35.2) | 34.80 | 1 | -17.0 (17.4) | -38.9 (49.0) | 33.22 | 1 | -14.7 (9.2) | -28.0 (19.3) | 31.38 | |
| $p_{\text{KD/nc}}$ trained by $(X, \widetilde{Y}_{\text{NC}})$ | 1 | -13.8 (13.9) | -28.0 (32.7) | 35.39 | 1 | -17.2 (26.3) | -35.4 (44.6) | 33.95 | 1 | -14.8 (9.1) | -24.0 (16.7) | 31.71 | |

Table 4: The rewards and BLEU scores using two KD approaches: $p_{\text{KD/beam}}$ uses the pseudo-corpus generated by doing beam search from $p_f$. $p_{\text{KD/nc}}$ uses the pseudo-corpus generated by BSR.

| | system 2 | | | | |
|---|---|---|---|---|---|
| system 1 | $p_f$ (beam search) | BSR | KD | IL | Q learning |
| $p_f$ (beam search) | 100 | – | – | – | – |
| BSR | 81.2 | 100 | – | – | – |
| KD | 64.5 | 66.0 | 100 | – | – |
| IL | 64.4 | 66.2 | 70.8 | 100 | – |
| Q learning | 74.0 | 72.0 | 64.3 | 64.1 | 100 |

Table 5: Corpus-level BLEU between translations by pairs of systems. Each reported BLEU is averaged between two directions.

model $p_{\text{KD/beam}}$: we generate the pseudo-corpus $\widetilde{Y}_{\text{beam}}$ from the training set, by beam search from $p_f$, and then use MLE to train $p_{\text{KD/beam}}$ using the parallel corpora $(X, \widetilde{Y}_{\text{beam}})$. Table 4 suggests that the forward rewards of the two approaches are similar, but the reverse rewards for $p_{\text{KD/nc}}$ is much larger. Meanwhile, $p_{\text{KD/nc}}$ produces translations with higher BLEU. It is therefore necessary to use BSR to generate the pseudo-corpus, in order to amortize noisy-channel NMT using KD.

**Q learning.** Why does Q learning, the best understood approach among the three, fail to achieve rewards that are comparable to BSR? The two challenges of a general deep Q learning algorithm are exploration and optimization.

Exploration refers to whether we can find high-quality trajectories. We hypothesize that it is not an issue given the diversity of trajectories we use, as shown in Appendix A.1. We even attempt adding high-reward trajectories from BSR as well as trajectories from a deep ensemble of multiple $p_f$'s but neither BLEU nor reward improves.

We thus suspect optimization as a challenge. The reverse reward $\log p_r(\boldsymbol{x}|\boldsymbol{y})$ is *sparse* in that it is non-zero only at the terminal state $(\boldsymbol{y}_{1:T}, \boldsymbol{x})$ where $y_T = \langle \text{eos} \rangle$. The difficulty in maximizing the sparse reverse reward comes from using one-step bootstrapping in Q learning. Such bootstrapping

allows Q learning to cope with very long episodes or even an infinite horizon, but this slows down the propagation of future reward to the past. Because we always work with relatively short episodes only in machine translation, we should investigate other learning paradigms from reinforcement learning, such as R learning (Mahadevan, 1996). We leave this further investigation to the future.

## 6 Related Work

One of our approaches adapts knowledge distillation (KD) for the noisy channel NMT setting. KD (Hinton et al., 2015; Kim and Rush, 2016) has been shown to work well for sequence generation. Chen et al. (2018) propose trainable greedy decoding, in which they use knowledge distillation to train a greedy decoder so as to amortize the cost of beam search. More subsequent studies have demonstrated the effectiveness of KD in neural machine translation (Freitag et al., 2017; Tan et al., 2019); Gu et al. (2017) show that it is difficult for on-policy reinforcement learning (RL) to work better than KD. Recently, KD has greatly boosted performance of non-autoregressive MT models (Gu et al., 2018; Lee et al., 2018; Tu et al., 2020). KD is also used to speed up speech synthesis and the approach has been widely deployed in real products (van den Oord et al., 2018).

RL for sequence generation has been greatly inspired by Sutton and Barto (1998). Ranzato et al. (2016) and Bahdanau et al. (2016) apply on-policy RL (REINFORCE and actor-critic algorithms) to MT, but the major optimization challenge lingers given that the reward is usually sparse. Choshen et al. (2020) recently find that the improvements in MT performance may rely on a good initialization. To address the sparsity issue, Norouzi et al. (2016) attempt a hybrid maximum likelihood (ML) and RL approach. More recently, Pang and He (2021) attempt to use an offline RL setting with per-token reward based on the a translator trained

| |
|---|
| **source:** acum , insa , tsipras cere grecilor sa ii incredinteze din nou mandatul de premier , in cadrul unor alegeri despre care sustine ca ii vor intari pozitia politica .<br>**KD:** now , however , tsipras is urging greeks to entrust the prime minister &apos;s mandate again , in an election he claims will strengthen his political position .<br>**IL:** now , however , tsipras is asking greeks to reentrust them with the prime minister &apos;s term , in an election that they claim will strengthen his political position .<br>**Q learning:** now , however , tsipras is urging greeks to reentrust his term as prime minister in an election that he claims will strengthen his political position .<br>**beam search by** $p_f$**:** now , however , tsipras is urging greeks to re-entrust the prime minister &apos;s term in an election that he claims will strengthen his political position .<br>**BSR:** now , however , tsipras is urging greeks to reentrust the prime minister &apos;s term , in an election that he claims will strengthen his political stance .<br>**reference:** now , however , tsipras asks the greeks again to entrust him with the prime minister position , during an election which he says will strengthen his political position . |
| **source:** adomnitei a fost trimis in judecata de directia nationala antico \<unk\> ruptie ( dna ) , fiind acuzat de favorizarea faptuitorului si fals intelectual dupa ce , spun pro \<unk\> curorii , ar fi incercat sa mascheze un control de audit in urma caruia se descoperise o serie de nereguli cu privire la receptia dintr-un contranct public semnat intre cj si firma laser co .<br>**KD:** adomnitei was sued by the national anti-co nistelrooij ruptie ( dna ) as accused of favouring the perpetrator and false intellectual after , pro nistelrooij curorii says , he would have tried to disguise an audit control as a result of which a number of irregularities concerning reception in a public contranct signed between the cj and laser co were discovered .<br>**IL:** adomnitei was sued by the national directorate antico iel ruptie ( dna ) and accused of favouring the perpetrator and forgery an intellectual after , pro iel curorii says , he had tried to disguise an audit control that found a number of irregularities regarding the reception in a public conctrant signed between cj and laser .<br>**Q learning:** the runner the runner , the runner-the runner-in-ranging runner-up is given to the latter , as he is accused of promoting the perpetrator and faltering intellectual after , says pro or: curors , tried to disguise an audit control , as a result of which a number of irregularities concerning a reception signed between cj and laser had been discovered in a public cross-border convoy .<br>**beam search by** $p_f$**:** adomnitei was sued by the national anti-co nistelrooij rupture ( dna , accused of favouring the perpetrator and forgery intellectual after allegedly attempting to disguise an audit control line between cj and lasco .<br>**BSR:** adomnitei was sued by the national anti-co xiated department ( dna , accused of favouring the perpetrator and forgery intellectual after allegedly attempting to disguise an audit control line signed between cj and the lasco firm .<br>**reference:** adomni«unk» ei was indicted by the national anticorruption directorate ( dna ) , being accused of favouring the offender and forgery after , according to the prosecutors , he tried to mask an audit which discovered a number of irregularities regarding the acceptance of a public contract entered into by the county council and the company laser co . |

Table 6: WMT'16 Ro-En examples produced by different systems. The top example is randomly selected. The bottom example is an example with a long source, and Q learning produces repetitions.

using standard MLE.

In recent years, off-policy RL methods have been used to better leverage existing trajectories in text generation. For instance, in the chatbot setting (Serban et al., 2017; Zhou et al., 2017), the periodically-collected human feedback is treated as the trajectory. In our case, we leverage the expensive BSR-obtained trajectories as well as trajectories from many different models and sources, although the sparse reward issue still lingers.

Finally, we point out a recent endeavor to speed up noisy channel NMT inference (Bhosale et al., 2020). They reduce the size of the channel model, the size of the output vocabulary, and the number of candidates during beam search. Our solution is orthogonal: we aim to use a separate network to amortize decoding cost, while not changing the network's architecture.

# 7   Conclusion

We describe three approaches (KD, IL, Q learning) to train an amortized noisy-channel NMT model. We investigate whether greedily decoding from these models will lead to accurate translations in terms of reward and quality. Although all three approaches fail to achieve comparable rewards to BSR, the reverse rewards are much higher than those from non-BSR baselines, often at the ex-

pense of forward rewards. However, we found the translation quality (measured by BLEU and BLEURT) to be comparable to that of BSR, while inference is much faster. For future work, the research community could further investigate better ways to optimize toward a sparse reward in the language generation context. Another way to approach the Q learning optimization challenge is to find better reward functions including denser rewards.

# References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Shruti Bhosale, Kyra Yee, Sergey Edunov, and Michael Auli. 2020. Language models not just for pre-training: Fast online neural noisy channel modeling. In *Proceedings of the Fifth Conference on Machine Translation*, pages 584–593, Online. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, volume 57, Hanoi, Vietnam.

Peng-Jen Chen, Ann Lee, Changhan Wang, Naman Goyal, Angela Fan, Mary Williamson, and Jiatao Gu. 2020. Facebook AI's WMT20 news translation task submission. In *Proceedings of the Fifth Conference on Machine Translation*, pages 113–125, Online. Association for Computational Linguistics.

Yun Chen, Victor O.K. Li, Kyunghyun Cho, and Samuel Bowman. 2018. A stable and effective learning strategy for trainable greedy decoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 380–390, Brussels, Belgium. Association for Computational Linguistics.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2020. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Sapporo, Japan. Association for Computational Linguistics.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*.

Frederick Jelinek. 1997. *Statistical methods for speech recognition*. MIT press.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

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

Tom Kocmi, Tomasz Limisiewicz, and Gabriel Stanovsky. 2020. Gender coreference and bias evaluation at WMT 2020. In *Proceedings of the Fifth Conference on Machine Translation*, pages 357–364, Online. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. 2021. Faster depth-adaptive transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13424–13432.

Sridhar Mahadevan. 1996. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Richard Yuanzhe Pang and He He. 2021. Text generation by learning from demonstrations. In *International Conference on Learning Representations*.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Richard S Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021. Facebook AI's WMT21 news translation task submission. In *Proceedings of the Sixth Conference on Machine Translation*, pages 205–215, Online. Association for Computational Linguistics.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online. 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 of *Proceedings of Machine Learning Research*, pages 3918–3926. PMLR.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. 2020. Consistency of a recurrent language model with respect to incomplete decoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5553–5568, Online. Association for Computational Linguistics.

Kyra Yee, Yann Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. 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 5696–5701, Hong Kong, China. Association for Computational Linguistics.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefen-stette, and Tomas Kocisky. 2017. The neural noisy channel. In *International Conference on Learning Representations*.

Lei Yu, Laurent Sartran, Po-Sen Huang, Wojciech Stokowiec, Domenic Donato, Srivatsan Srinivasan, Alek Andreev, Wang Ling, Sona Mokra, Agustin Dal Lago, Yotam Doron, Susannah Young, Phil Blunsom, and Chris Dyer. 2020. The DeepMind Chinese–English document translation system at WMT2020. In *Proceedings of the Fifth Conference on Machine Translation*, pages 326–337, Online. Association for Computational Linguistics.

Li Zhou, Kevin Small, Oleg Rokhlenko, and Charles Elkan. 2017. End-to-end offline goal-oriented dialog policy learning via policy gradient. *arXiv preprint arXiv:1712.02838*.

## A More Information on Q learning for Amortized Noisy Channel NMT

### A.1 Details on trajectories

We have obtained trajectories from different sources in the off-policy algorithm (Algorithm 1). Each trajectory contains a source-language sequence $x$, a target-language sequence $y$, and the corresponding sequence of rewards $r = (r_1, \ldots, r_T)$.

One natural category of trajectories to consider is the ones obtained by $Q$ during training. Source (1a) and source (1b) correspond to $Q$-based trajectories.

Source (2) corresponds to $p_f$-obtained trajectories. Specifically, we split this category into a few sub-sources. (2a) The $y$ is obtained through sampling from $p_f$ with temperature sampled from $\text{Uniform}([0, 1])$. (2b) The $y$ is obtained through greedily decoding from $p_f$. (2c) The $y$ is obtained through beam search from $p_f$ with a beam size randomly chosen from 2 to 10. (2d) The $y$ is obtained through beam search from $p_f$: we first obtain 50 candidate sequences corresponding to largest $p_f$ probabilities using beam search with beam size 50; next, we pick a random sequence out of these 50 sentences.

We have also experimented with gold-standard trajectories from the parallel translation dataset $\mathcal{D}$, but the inclusion of such trajectories do not lead to better rewards (of translations generated from $Q$).

The probability for using (1a), (1b), (2a), (2b), (2c), (2d) sequences are 0.3, 0.2, 0.2, 0.1, 0.1, 0.1, respectively.

## B More Discussion on Experiments

**BSR hyperparameters.** $\gamma$ is tuned in $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$, and $b$ is tuned in $\{5, 10, 20, \ldots, 100\}$ for the first two datasets and $\{5, 10, 20, \ldots, 50\}$ for WMT'14 De-En due to memory constraints. The best $\gamma$ is 0.9, 0.5, 0.5, for IWSLT'14 De-En, WMT'16 Ro-En, WMT'14 De-En, respectively; the best $b$ is 100, 70, 50 for the three datasets, respectively.

**Details on $p_f$ and $p_r$.** Recall that $p_f$ is the forward translator (from the source language to the target language) and $p_r$ is the reverse translator (from the target language to the source language). We use transformer-based architectures for all experiments. Refer to Table 7 for the architecture.

**Number of parameters in the models.** The IWSLT'14 De-En transformer has 39,469,056 parameters, the WMT'16 Ro-En transformer has 62,046,208 parameters, and the WMT'14 De-En transformer has 209,911,808 parameters.

**Discussion on Q learning.** In Section 5.3, to investigate whether better trajectories can improve Q learning results, we attempt adding high-reward trajectories from BSR as well as trajectories from a deep ensemble of two $p_f$'s. Deep ensembling two models (using different seeds) can produce high-quality translations. In this case, we simply want to use deep ensembling to diversify the sources of high-reward and high-BLEU trajectories. However, the result is that neither BLEU nor reward improves.

## C Ethical Considerations

IWSLT and WMT datasets are standard machine translation benchmarks. The datasets come from a variety of sources: phone conversations, parliament proceedings, news, and so on. There may be naturally occurring social biases in the datasets which have not undergone thorough cleansing. Training on these potential biases may lead to biased generations. There has been recent work studying such biases (Kocmi et al., 2020).

This work deals with speeding up inference, but not pretraining or training (Liu et al., 2021; Hou et al., 2022). The standard practice of creating the pseudo-corpus requires a significant amount of computation. This step is optional, but it gives a boost in performance. We argue that if the MT system is put into production, then the benefit from the efficient inference will outweigh the cost of generating the pseudo-corpus.

|  | IWSLT'14 De-En | WMT'16 Ro-En | WMT'14 De-En |
|---|---|---|---|
| encoder embedding dimension | 512 | 512 | 1,024 |
| number of encoder attention heads | 4 | 8 | 16 |
| encoder ffn embedding dimension | 1,024 | 2,048 | 4,096 |
| encoder layers | 6 | 6 | 8 |
| decoder embedding dimension | 512 | 512 | 1,024 |
| number of decoder attention heads | 4 | 8 | 16 |
| decoder ffn embedding dimension | 1,024 | 2,048 | 4,096 |
| decoder layers | 6 | 6 | 8 |
| learning rate | 0.0005 | 0.0005 | 0.0005 |
| dropout rate | 0.3 | 0.1 | 0.1 |
| # tokens in a batch | 4,096 ($2^{12}$) | 65,536 ($2^{16}$) | 65,536 ($2^{16}$) |

Table 7: Settings for the forward model $p_f$ and the reverse (channel) model $p_r$.

| | IWSLT'14 De-En | WMT'16 Ro-En | WMT'14 De-En |
|---|---|---|---|
| $p_f$ (greedy) | 62.40 (0.04) | 61.14 (0.10) | 64.83 (0.10) |
| $p_f$ (beam) | 63.21 (0.07) | 61.42 (0.15) | 65.79 (0.08) |
| BSR | 64.15 (0.05) | 62.67 (0.13) | 66.32 (0.12) |
| KD | 63.88 (0.04) ∗ | 61.78 (0.10) ∗ | 66.00 (0.07) ∗ |
| IL | 63.94 (0.13) ∗† | 62.35 (0.16) ∗† | 66.14 (0.08) ∗† |
| Q learning | 63.25 (0.07) | 61.70 (0.18) | 65.92 (0.14) |

Table 8: Test set BLEURT-20-D12 (mean & standard deviation of three runs). IL performs the best among the three proposed methods. ∗: The score is significant (p-value smaller than 0.05) compared to the beam search results. †: The score is significantly higher (p-value smaller than 0.05) than BSR results, or the score is not significantly different (p-value larger than 0.05) from the BSR results.