# Optimizing the Training Schedule of Multilingual NMT using Reinforcement Learning

**Alexis Allemann[1], Àlex R. Atrio[2,\*], Andrei Popescu-Belis[1,3]**

[1]HEIG-VD / HES-SO, 1401 Yverdon-les-Bains, Switzerland
[2]Pi School, Rome, Italy
[3]EPFL, 1015 Lausanne, Switzerland
**Correspondence:** andrei.popescu-belis@heig-vd.ch

## Abstract

Multilingual NMT is a viable solution for translating low-resource languages (LRLs) when data from high-resource languages (HRLs) from the same language family is available. However, the training schedule, i.e. the order of presentation of languages, has an impact on the quality of such systems. Here, in a many-to-one translation setting, we propose to apply two algorithms that use reinforcement learning to optimize the training schedule of NMT: (1) Teacher-Student Curriculum Learning and (2) Deep Q Network. The former uses an exponentially smoothed estimate of the returns of each action based on the loss on monolingual or multilingual development subsets, while the latter estimates rewards using an additional neural network trained from the history of actions selected in different states of the system, together with the rewards received. On a 8-to-1 translation dataset with LRLs and HRLs, our second method improves BLEU and COMET scores with respect to both random selection of monolingual batches and shuffled multilingual batches, by adjusting the number of presentations of LRL vs. HRL batches.

## 1 Introduction

Multilingual neural machine translation (NMT) is particularly effective to enable the translation of low-resource languages (LRLs) when they are accompanied, in the training data, by related high-resource languages (HRLs) (Gu et al., 2018; Neubig and Hu, 2018). Including HRLs in the training data reduces the chance of overfitting to the LRLs and improves translation quality.

Many-to-one NMT systems can be trained either with monolingual or with multilingual batches. Monolingual batches include a single language on

the source side, while multilingual batches have their source side sampled from several source languages. Using multilingual batches helps avoiding catastrophic forgetting (Jean et al., 2019), but the mixture of languages in each batch may be ineffective at early stages of training. Here, we focus on monolingual batches, as they enable us to define the training schedule of a NMT system as the order of presentation of languages, but we also compare our results to those obtained with multilingual batches.

We propose to use reinforcement learning (RL) to optimize the training schedule of many-to-one NMT systems, i.e. to improve the training process and the resulting system compared to a fixed sampling strategy. We enable our systems to select the source language of the batch at each training step, based on a learned estimate of the model's competence on each language in terms of loss on a development set. Unlike fixed strategies, such as training on the hardest language, we leverage RL to let the model find better strategies.

We make the following contributions:[1]

- We apply the Teacher-Student Curriculum Learning algorithm (Matiisen et al., 2017) to NMT by modeling the expected return as the smoothed loss of the NMT system over a development set.

- Based on the Deep Q Network algorithm (DQN) (Mnih et al., 2013), we design a RL-based model in which the expected rewards are generated by an auxiliary neural network trained in parallel with the NMT system.

- We perform experiments on a dataset with four language families on the source side, with one HRL and one LRL for each family (Neubig and Hu, 2018); the target language is English.

- DQN outperforms in terms of BLEU and

---

[1]Source code is made available at https://github.com/alexis-allemann/OpenNMT-py/tree/curriculum_learning.

COMET scores previous training schedules used for multilingual NMT: monolingual mini-batches sampled equally, or in proportion of each language, or multilingual batches.

- Algorithms are robust to the setting of hyper-parameters, and increase the proportions of LRLs in the training schedule from less than 1% to at least 4% while decreasing those of HRLs.

## 2 Related Work

**Curriculum learning.** In many applications of machine learning, the order of presentation of items from the training set may influence the outcome of the training, i.e. the quality of the final model, or the training speed. For instance, presenting items by increasing levels of difficulty is often beneficial, an approach known as *curriculum learning* (Wang et al., 2021). The difficulty can be measured directly on the data, or it can be inferred from the observed competence of the model during training, an approach known as *self-paced learning* (Kumar et al., 2010; Jiang et al., 2015). The competence of a model can be estimated intrinsically, e.g. from its loss values on a subset of the data, or extrinsically, by using a teacher model that observes the behavior of the target model, called 'student' (Matiisen et al., 2017). Competence can be used by the teacher model to adjust the training schedule of the student model. In the case of systems that can perform several tasks, the training schedule consists of the selection of tasks and related data.

When the teacher model is in charge of the training schedule of the student, it may use reinforcement learning (RL), with the student model playing the role of the environment (Shen and Zhao, 2024). RL has proved particularly useful at training large language models to follow instructions (Ouyang et al., 2022), initially using PPO (Proximal Policy Optimization, Schulman et al., 2017) and then other algorithms (Rafailov et al., 2023; Ethayarajh et al., 2024), but these methods are not designed to optimize training schedules. While it is possible to use curriculum learning to train RL-based models (Narvekar et al., 2020), e.g. by presenting them with increasingly difficult problems, we focus here on the use of RL to train a teacher model, in the field of multilingual NMT.

**Training schedules for NMT.** Optimizing the training schedule of an NMT system depends in particular on its architecture. For a system with a single input language and domain, the training sentences can be presented by order of estimated difficulty, or by order of translation reliability or noisiness. When multiple domains must be considered, additional decisions must be made about which domain to use first, or how to mix them based on sizes of available data. Similar decisions must be made if there are multiple input languages, as in our case, or if one must train a multi-task system including NMT along with other tasks such as language modeling. We briefly review here previous work along these lines.

**Static scheduling in multilingual NMT.** Neubig and Hu (2018) study the upsampling of the LRL data when building minibatches, and observe that keeping the original proportions of HRL and LRL performs marginally better. However, Johnson et al. (2017) and Aharoni et al. (2019) sample each batch uniformly from a concatenation of all language pairs. Arivazhagan et al. (2019) compare simple concatenation with uniform balancing, and observe better results for LRLs when using temperature-based upsampling, which was favored afterwards (Conneau et al., 2020; Tang et al., 2021).

The translation capabilities of large language models (LLMs) have also been explored: Zhu et al. (2024) compares several recent LLMs and shows that they can achieve state-of-the-art results when translating HRLs, but highlights their limitations in translating LRLs compared to NMT models. One of the leading open-weights LLMs for MT, Tower Instruct (Alves et al., 2024), is fine-tuned on a large set of translation-related tasks in 10 HRLs, with no particular scheduling of the fine-tuning data, and no reinforcement learning.

**Curriculum learning in monolingual NMT.** Self-pacing has been used in NMT at the sample level, for instance by estimating learning confidence as the variance across dropout runs, with better performance and faster convergence compared to human-designed schedules (Wan et al., 2020). Similarly, Liu et al. (2020) design a self-paced curriculum based on the norm of a token's embedding. Zhang et al. (2018) adopt a probabilistic view of curriculum learning and improve the convergence time of a DE-EN NMT system at no loss in translation quality, but no gain either; moreover, they note a high sensitivity to hyperparameter settings. Platanios et al. (2019) propose a scheduling criterion combining the difficulty of samples and the competence of the NMT model,

the latter estimated as a linear or square root function of the number of steps. This reduces training time by up to 70% and improves BLEU scores by 1–2 points on three different language pairs. Wang et al. (2018) extend domain-specific data selection methods to denoise NMT training, which significantly improves NMT performance on noisy data. Wang et al. (2020a) introduce a method for multi-domain data selection in NMT, using instance-level domain-relevance features and an automated training curriculum to enhance performance across multiple domains.

**Curriculum learning in multilingual NMT.** Jean et al. (2019) compare adaptively upsampling a language depending on various criteria, observing best results on LRLs when dynamically changing the norm of the gradient. Wang et al. (2020b) adaptively balance the languages by learning their weights from the model's competence on a development set. Zhang et al. (2021) design a dynamic sampling strategy which measures per-language competence but also evaluates LRL competence through a related HRL's competence. Wu et al. (2021) also balance the data dynamically, but measure a model's uncertainty as the variance over several runs of Monte Carlo dropout. Estimates of competence using the evolution of the loss of the NMT system have been proposed by Zaremoodi and Haffari (2019), who use its absolute value, by Xu et al. (2020), who use its relative decrease, and by Atrio et al. (2024), who use Kullback-Leibler divergence between consecutive states of the weights of an entire Transformer network.

**RL-based curriculum learning in NMT.** In the field of machine translation, Kumar et al. (2019) propose a RL framework utilizing Q-Learning to automatically learn an optimal curriculum for heterogeneous data, matching state-of-the-art hand-designed curricula. Zhao et al. (2020) introduce a RL-based data selection framework using Deterministic Actor-Critic to improve pre-trained NMT models by re-selecting influential samples from the original training set. Kreutzer et al. (2021) use a multi-armed bandit to dynamically select training data, thus optimizing NMT model performance across different domains, data qualities, and language pairs without manual schedule design.

**Other applications of RL to NMT.** In machine translation, RL methods were employed by Edunov et al. (2018) to tackle the discrepancy between token-level likelihood optimization during training and corpus-level evaluations using metrics like BLEU, and to reduce exposure bias in autoregressive sequence generators (Ranzato et al., 2016; Wang and Sennrich, 2020; Wu et al., 2018b). Kiegeland and Kreutzer (2021) emphasize the importance of exploration strategies, reward scaling, and reward function design for improving translation quality, particularly with respect to domain adaptation. To enhance the effectiveness of RL in NMT, Yehudai et al. (2022) show the importance of reducing the size and dimensionality of the action space. Wang et al. (2024) introduce efficient sampling-based RL techniques for sequence generation models, with a strong focus, however, on instruction tuning of LLMs.

## 3 Two RL Algorithms for Optimizing Training Schedules

In the RL framework, an agent observes the state $S_t$ of the environment at each time step $t$, selects an action $A_t$ based on its policy $\pi$, executes the action, and receives a reward $R_t$ from the environment. Using the observed states, actions, and rewards, the goal is to learn an optimal policy, i.e. one that maximizes the cumulative reward over time. Bandit problems are those where the agent selects actions without considering state transitions.

In this study, as we use for training only monolingual batches, the set of possible actions $\mathcal{A}$ is simply the set of source languages. The states $\mathcal{S}$ of the system are the values of the parameters of the neural network and of the optimizer. However, these are too numerous to be sensibly observed at each step. Drawing inspiration from Wu et al. (2018a), we compute the current state of the model as the vector of cross-entropy loss values obtained from the NMT system over a development batch of sentences.[2] We use the score $X_t$ of the model at time step $t$ to compute the reward $R_t$ as the decrease of the loss of the NMT system between the last two time steps: $R_t = X_t - X_{t-1}$. The loss values are computed on a development minibatch of data selected from the current language, or, in some experiments, on a multilingual minibatch.

### 3.1 TSCL Algorithm for NMT

Our first proposal is an adaptation to NMT of the Teacher-Student Curriculum Learning (TSCL) al-

---

[2] Alternatively, MT-specific metrics such as BLEU or COMET could be used instead of the cross-entropy loss, but computing them is more costly, therefore we do not use them here.

gorithm, a bandit method introduced by Matiisen et al. (2017), who use it to add decimal numbers or to navigate Minecraft mazes. The gist of our adaptation of TSCL for multilingual NMT is represented in Figure 1, and the full algorithm is given in Appendix A.4.
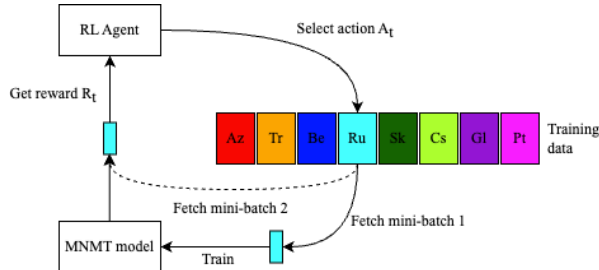


Figure 1: TSCL algorithm for NMT: relationship between the RL agent and the NMT system.

The action $A_t$ of the system at time step $t$ is the selection of a batch from a specific source language for training in the next step. The reward $R_t$ of the action is the decrease of the negative cross-entropy loss of the model, $X_t$, computed on a batch of 8k tokens from the current source language, with respect to the value $X_{t'}$ computed at the latest previous time step with the same language. Formally, $R_t = X_t - X_{t'}$ where $t' = \max\{s : s < t \text{ and } A_s = A_t\}$. The expected return $Q_t$ of the action is the exponentially weighted moving average of the rewards for the respective source language.

In some experiments, we start with a warm-up phase, a period during which all HRLs are randomly explored while the learning rate of the NMT model increases. Rewards of the RL agent only start to play a role after the warm-up phase, when the learning rate starts decreasing. In experiments without warm-up, each action is executed once at the beginning of the training, so that the model initiates training on the language that provides the highest reward from the start.

Additionally, to strike a balance between exploration and exploitation, we use an $\epsilon$-greedy policy with a fixed value of $\epsilon$. The action with the highest expected return is selected with probability $1 - \epsilon$, but with a small probability $\epsilon$ a random action is selected. In experiments with a warm-up period, this policy only starts after this period.

## 3.2 DQN Algorithm for NMT

The Deep Q Network (DQN) algorithm (Mnih et al., 2013) uses a neural network to approximate

the Q-function that represents the expected reward of an action in a given state. The algorithm iteratively updates the parameters of this network to minimize the difference between the predicted Q-values and the desired Q-values obtained from the target system. Moreover, DQN enables experience replay by storing past experiences in a replay buffer and sampling them randomly during the training of the Q network, a feature that was shown to improve training.

Our application of DQN to multilingual NMT is illustrated in Figure 2, and the full algorithm is given in Appendix A.5. The RL agent is the Q network, a feed-forward neural network with *tanh* activation functions. Its input is the state of the NMT model: specifically, each value in the input layer represents the cross-entropy loss of the NMT model over a batch of 10 sentences from a specific language. Thus, an input vector of size 200 corresponds to a prototype batch of 2,000 sentences, with 250 sentences from each of the 8 source languages. The input layer is followed by two hidden layers of size 512 and by an output layer with 8 units, corresponding to the possible actions (selection of a source language for the next training step). The Q network is trained with the RMSProp optimizer[3] and the Huber Loss (Huber, 1964), a loss function that reduces the influence of extreme values, to mitigate the issue of outliers during training.

At each timestep $t$, the RL agent retains a new transition in its experience replay buffer. A transition consists of the previous state of the system $S_{t-1}$, the selected action $A_{t-1}$, the obtained reward $R_t$, and the current state of the system $S_t$. These transitions are used to train the Q network so that it predicts the action with the best estimated reward given the state of the NMT model.

We use an $\epsilon$-greedy policy to balance between exploring actions and exploiting the Q network, like for TSCL. During the warm-up period, which is always applied to DQN, actions are randomly selected, but after it, actions are selected by the Q network with a probability of $1 - \epsilon$ or thy are randomly selected with a probability of $\epsilon$. However, unlike TSCL, we follow Kumar et al. (2019) and start with $\epsilon = 1$ during warm-up, then gradually decrease this value at the end of warm-up to a minimum of 0.01 after 50k steps. This allows the network to randomly explore actions during

---

[3] A variant of stochastic gradient descent, proposed by G. Hinton, which adapts the learning rate for each parameter based on recent gradient averages (Ruder, 2016).
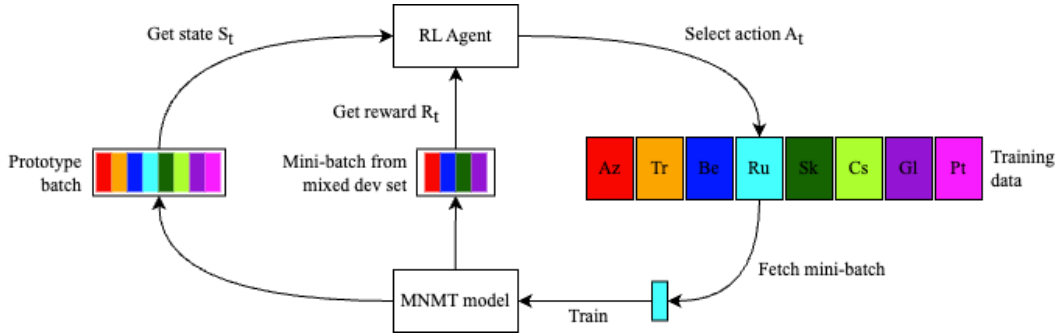
Figure 2: DQN algorithm for NMT: relationship between the RL agent (Q network) and the NMT model.

the warm-up period before exploiting the learned knowledge more and more. The schedule of $\epsilon$ is represented in Figure 5 of Appendix A.1.

## 4 Experimental Settings

**Data.** Experiments were conducted using a subset of the multilingual TED corpus collected by Qi et al. (2018), with four HRLs and four LRLs.[4] For comparability with prior research on multilingual NMT (Neubig and Hu, 2018; Wang et al., 2019; Zhang et al., 2021), we consider a 8-to-1 translation task with English as the target language. We are especially interested in the translation quality of the four LRLs of the dataset: Belarusian (BE), Azerbaijani (AZ), Galician (GL) and Slovak (SK), which are respectively paired with a HRL from the same family: Russian (RU), Turkish (TR), Portuguese (PT) and Czech (CS). Three language families are thus represented (Romance, Slavic and Turkic) but all scripts are Latin-based.

The numbers of sentences of the training and testing sets for each of the 8 languages are shown in Table 1. These numbers show that the distinction of LRLs vs. HRLs made in previous studies is to some extent arbitrary. Indeed, there are fewer PT sentences (considered nevertheless as a HRL with respect to GL) than SK sentences (considered as a LRL with respect to CS).

**Preprocessing.** The original data is already tokenized into words. We use Byte Pair Encoding (BPE) for subword extraction and vocabulary construction (Sennrich et al., 2016).[5] A vocabulary of 32k subwords is generated over a multilingual corpus obtained by combining 10k random lines from the training data of each language, with upsampling for AZ and BE which have fewer than

| Language | train | dev | test |
|---|---|---|---|
| AZ | 5.9k (0.95%) | 671 | 903 |
| BE | 4.5k (0.72%) | 248 | 664 |
| GL | 10.0k (1.60%) | 682 | 1.0k |
| SK | 61.5k (9.79%) | 2.2k | 2.4k |
| TR | 182k (29.07%) | 4.0k | 5.0k |
| RU | 208k (33.21%) | 4.8k | 5.5k |
| PT | 51.8k (8.25%) | 1.2k | 1.8k |
| CS | 103k (16.42%) | 3.5k | 3.8k |

Table 1: Numbers of sentences for LRLs and HRLs.

10k lines. For source language identification by the NMT model, each sentence is prefixed with a language tag.

**NMT Models.** We experiment with Transformer models from the OpenNMT-py library version 3.4.3 (Klein et al., 2017).[6] All models are trained for 150k steps. The hyperparameter values are the default ones from the Transformer-Base model (Vaswani et al., 2017): 6 layers for the encoder and 6 for the decoder, 8 attention heads, label smoothing of 0.1, hidden layers with 512 units, and feedforward networks with 2,048 units. The Adam optimizer (Kingma and Ba, 2014) is used. Following Atrio and Popescu-Belis (2022), we use a batch size of 8k tokens and the regularization parameters are: dropout rate of 0.3, scaling factor of 10, and gradients are re-normalized if their norm exceeds 5. In experiments with warm-up, there are 16k steps during which the learning rate increases from 0 to its maximum.

**RL Agents.** Several hyperparameters must be set for RL Agents. Their default values are given here, while the behavior of the systems when these values are modified are studied in Section 5.4.

---

[4]github.com/neulab/word-embeddings-for-nmt
[5]github.com/rsennrich/subword-nmt

[6]github.com/OpenNMT/OpenNMT-py

The TSCL algorithm is run with a smoothing co-efficient $\alpha = 0.1$. The warm-up period is 16k steps, during which batches from HRLs are presented in a random order. For the $\epsilon$-greedy policy, $\epsilon = 0.1$. These values correspond to those used by Matiisen et al. (2017).

The DQN algorithm is also run with a warm-up period of 16k steps on HRLs only. Unlike TSCL, a new action is selected every 10 steps, and not at every step, to reduce computing time, with no significant differences in observed results. The Q network underlying the RL agent has an input layer with 200 units, two fully connected subsequent layers with 512 units each, and an output layer with 8 units. As explained in Section 3.2, each value in the input layer corresponds to the cross-entropy loss of the NMT model over a batch of 10 sentences from a specific language.

The training of the Q network has a learning rate $lr = 2.5\mathrm{e}-4$ and a soft update smoothing coefficient $\tau = 0.005$. The discount factor, which influences the importance of future rewards in the agent's decision-making process, is $\gamma = 0.99$.[7] The experience replay buffer has minimal/maximal sizes of 1k/10k. These values are those used by Kumar et al. (2019).

**Evaluation Metrics.** Translation quality is measured using the BLEU and COMET metrics. BLEU scores are computed with the SacreBLEU library (Post, 2018).[8] COMET scores are computed using the `wmt22-comet-da` model (Rei et al., 2022).[9] Scores are computed using a rolling ensemble of four checkpoints. The best ensemble in terms of average BLEU score on the LRLs development sets is used to translate the test set.

## 5 Results and Analysis

### 5.1 Baselines

We compare TSCL and DQN to baseline training schedules in which source languages are selected randomly at each step, either with a uniform distribution ($P = 1/8$ for each language) or with a distribution that is proportional to the number of sentences of the respective language in the training data – hence between 0.95% for Az and 33.21% for Ru, as shown in Table 1. Moreover, a warm-up

period of 16k on HRLs can be used or not. This results in four baseline schedules, shown in the first four lines of Tables 2, 3 and 4. While these baselines and the TSCL and DQN algorithms use monolingual batches, a fifth baseline uses multilingual shuffled ones, with sentences drawn randomly from the source languages in proportion to their frequency, and a warm-up period of 16k on HRLs. Shuffled batches were found to perform particularly well on this dataset (Neubig and Hu, 2018; Atrio et al., 2024).

### 5.2 Translation Performance

The BLEU and COMET scores of the TSCL and DQN algorithms, in comparison to the baselines, are presented in Table 2 for the LRLs and in Table 3 for the HRLs. The averages of BLEU and of COMET scores over the 8 languages are presented in Table 4, giving the same importance to each language, regardless of its frequency in the training data (macro-average).

The DQN algorithm outperforms on average all baselines, as well as the simpler TSCL algorithm, both in terms of BLEU and of COMET (Table 4). The TSCL algorithm is second for BLEU, but third for COMET, slightly behind the uniform training schedule with warm-up. Considering Table 2 with LRLs, we see that DQN often outperforms the other methods: it ranks first on COMET for Az and Be and second for GL (but first on BLEU). Moreover, DQN ranks first on COMET for PT, as seen in Table 3. Therefore, DQN ranks first on three of the four least represented languages in the dataset.[10] This shows that DQN improves learning of the LRLs at the price of a small degradation in HRLs, though still improving their macro-average. As for TSCL, although it is competitive on average with the baselines, it lags behind the best ones when it comes to individual languages.

The baseline that is most often ranked first is the one that selects batches in proportion to the frequency of the language in the training data, with no warm-up. This has best BLEU and COMET scores on three HRLs (Tr, Ru, Cs) and one LRL[10] (Sk), likely because each of these languages constitutes more than 10% of the training data. However, in this case, the NMT model struggles to learn LRLs because it does not see enough data from them. As a result, when considering the macro-average, this

| Training schedule | Warm up | AZ → EN | | BE → EN | | SK → EN | | GL → EN | |
|---|---|---|---|---|---|---|---|---|---|
| | | BLEU | COMET | BLEU | COMET | BLEU | COMET | BLEU | COMET |
| Uniform | no | 13.86 | 62.99 | 19.75 | 60.80 | 32.85 | 75.09 | 31.14 | 72.08 |
| Proportional | no | **15.82** | 65.66 | 19.81 | 61.85 | **35.11** | **76.51** | 31.07 | 72.65 |
| Uniform | 16k | 15.42 | 65.19 | 20.29 | 62.13 | 34.11 | 76.45 | 32.74 | **73.84** |
| Proportional | 16k | 15.14 | 65.70 | 19.22 | 61.48 | 34.97 | 76.26 | 31.79 | 72.70 |
| Shuffled batch | 16k | 14.37 | 64.37 | 20.08 | 62.15 | 33.92 | 76.28 | 32.15 | 72.99 |
| TSCL | 16k | 14.89 | 65.04 | 20.10 | 61.96 | 34.35 | 76.23 | 32.64 | 73.59 |
| DQN | 16k | 15.62 | **65.86** | **21.11** | **62.82** | 34.54 | 76.15 | **33.02** | 73.73 |

Table 2: Results of TSCL and DQN compared to baselines on LRLs.

| Training schedule | Warm up | TR → EN | | RU → EN | | CS → EN | | PT → EN | |
|---|---|---|---|---|---|---|---|---|---|
| | | BLEU | COMET | BLEU | COMET | BLEU | COMET | BLEU | COMET |
| Uniform | no | 27.26 | 75.41 | 26.95 | 72.18 | 30.87 | 74.30 | 39.76 | 78.41 |
| Proportional | no | **29.40** | **77.48** | **28.14** | **74.04** | **32.47** | **75.79** | 37.98 | 76.56 |
| Uniform | 16k | 28.29 | 76.72 | 27.53 | 73.32 | 31.76 | 75.60 | 41.32 | 79.57 |
| Proportional | 16k | 28.97 | 77.25 | 27.96 | 73.53 | 32.16 | 75.38 | 38.64 | 76.47 |
| Shuffled batch | 16k | 28.25 | 76.76 | 27.31 | 73.20 | 31.30 | 75.37 | 40.58 | 79.25 |
| TSCL | 16k | 28.50 | 76.64 | 27.56 | 72.99 | 31.76 | 75.15 | **42.38** | 79.52 |
| DQN | 16k | 28.11 | 76.45 | 27.66 | 73.28 | 31.89 | 75.31 | 42.09 | **79.73** |

Table 3: Results of TSCL and DQN compared to baselines on HRLs.

| Training schedule | Warm up | Average | |
|---|---|---|---|
| | | BLEU | COMET |
| Uniform | no | 27.81 | 71.40 |
| Proportional | no | 28.73 | 72.57 |
| Uniform | 16k | 28.93 | 72.85 |
| Proportional | 16k | 28.61 | 72.35 |
| Shuffled batch | 16k | 28.49 | 72.55 |
| TSCL | 16k | 29.02 | 72.64 |
| DQN | 16k | **29.30** | **72.92** |

Table 4: Macro-averages over all languages of the scores of TSCL and DQN compared to baselines.

the baseline with proportional batches (both at 60k). The baseline with uniformly-drawn batches needs twice more steps to converge.
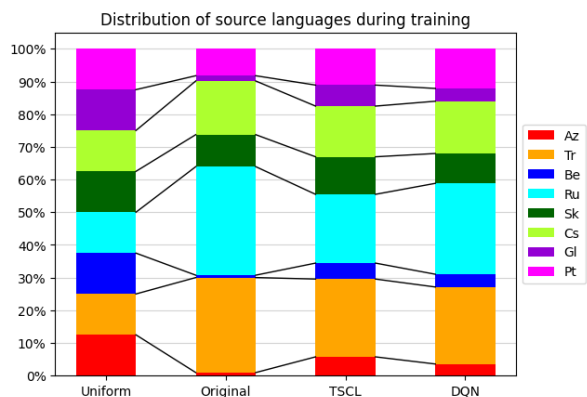


Figure 3: Proportions of data seen during training, as optimized by the TSCL and DQN algorithms, in comparison to uniform (1/8) or original proportions.

baseline is slightly behind the one using warm-up on HRLs followed by selection of actions with uniform probability, which also has better COMET scores for BE, GL and PT.

Moreover, the DQN and TSCL algorithms are efficient in terms of convergence speed, defined as the number of steps needed to reach their best scores (the macro-average of BLEU on the LRLs of the development set). As shown more fully in Table 7 of Appendix A.2, DQN reaches best performance after 52k steps, followed closely by TSCL and by

### 5.3 Optimized Training Schedule

We claim that the improved average scores with respect to the baselines are due to an optimized training schedule, which can be observed by considering the total amount of data from each lan-

guage seen during training, shown in Figure 3. The 'uniform' and 'proportional' baselines are shown in the first two columns. In the first case, the NMT model likely overfits to the LRL data, which is seen too often (12.5% of the times per language) with respect to its diversity (ca. 1% for three LRLs). In the second case, the number of times each LRL batch is seen during training is insufficient.

Our two algorithms strike a balance between these two extremes, as they are able to automatically determine more suitable proportions of batches of LRLs vs. HRLs for training. We see in Figure 3, third column, how the proportions of three LRLs are increased by TSCL (Az in dark orange, Be in dark blue, and Gl in dark purple). Two other languages with similar original proportions (Pt in light purple and Sk in dark green) see their proportions increased too, though less than the previous ones. Conversely, the proportions of HRLs decrease, especially for Ru and Tr.

In comparison to TSCL, the DQN algorithm appears to reach a slightly smaller proportion of LRLs, as seen in the fourth column of Figure 3, where proportions of the darker colors are shrunk with respect to the third column. These proportions are found quite quickly during training, as can be seen from Figure 6 in Appendix A.3, where we aggregate the proportions of actions every 1000 steps.

### 5.4 Role of Hyperparameters

In this section, we study the influence of hyperparameters on the scores of NMT systems trained with the TSCL and DQN algorithms. We present the scores obtained with significant variations of one parameter at a time in Table 5 for TSCL and in Table 6 for DQN. Globally, the scores of the algorithms do not vary much, which shows that they are robust with respect to the variations of the hyperparameters, but also confirms that the algorithms behave consistently from run to run. For both algorithms, BLEU and COMET scores lead to similar rankings.

For TSCL, we observe first that adding LRLs during warm-up (with uniform frequencies), or skipping warm-up entirely (thus starting with the highest learning rate), are not good options (second and third lines of Table 5). Instead, cross-lingual transfer from HRLs to LRLs becomes fully beneficial only with a 16k step warm-up on HRLs. Moreover, convergence is twice slower without warm-up. The smoothing coefficient $\alpha$ can vary around the

default value of 0.3 with a small decrease in performance ($\alpha = 0.1$ is shown in the 4th line) and so can $\epsilon$ for the $\epsilon$-greedy policy ($\epsilon = 0.3$ instead of 0.1 is shown in the 5th line). Finally, whether an action is selected every 10 steps or at every step results in comparable scores.

For DQN, we examine first if the Q network is over-parameterized, by reducing the size of the two hidden layers from 512 to 128 (2nd line of Table 6). This brings only a moderate decrease in average scores, but slightly better COMET scores for Az, Sk and Gl.

If we vary $\tau$, the smoothing rate of the updates of the Q network (see line 20 of Algorithm 2) within a large range between 0 and 1, the scores remain stable or even increase for some LRLs (3rd and 4th lines of Table 6, values of 0.5 and 0.995 with respect to default of 0.005).

Similarly, if we vary $\gamma$, the discount factor for the importance of future rewards, within a large interval between 0 and 1, the scores also remain stable (5th and 6th lines of Table 6, values of 0.5 and 0.01 with respect to default of 0.99). In fact, the value with the lowest scores is $\gamma = 0.01$, i.e. a system that gives only a marginal importance to long-term rewards. Conversely, this is also the system with the fastest convergence, although no particular variant seems to be particularly slow to converge (see Table 8 in Appendix A.2), and none achieves highest scores on all LRLs.

We can thus conclude that the default values of hyperparameters of TSCL and DQN used in Section 5.2 above, inspired respectively by Matiisen et al. (2017) and by Kumar et al. (2019), perform well and that both algorithms are stable when these hyperparameters vary.

### 5.5 Analysis of the Q Network

In this section, we propose a method to analyze the Q network of the DQN algorithm, which predicts on what language to train next, given a vector of 200 scores of an NMT model. Specifically, these scores are the cross-entropy loss values on 200 monolingual batches of 10 sentences each from the prototype set. At a given moment during training, the Q network can be probed with a specific vector as input, for instance a vector that represents a specific state of the NMT system. We propose to probe the Q network with a state in which one language is poorly learned. This is mimicked by assigning high loss values to the coefficients of the vector that represent scores on batches of this

| Hyperparameter values | AZ → EN | | BE → EN | | SK → EN | | GL → EN | |
|---|---|---|---|---|---|---|---|---|
| | BLEU | COMET | BLEU | COMET | BLEU | COMET | BLEU | COMET |
| Default | **14.89** | **65.04** | **20.10** | **61.96** | **34.35** | **76.23** | **32.64** | **73.59** |
| Warm-up LRL+HRL | 14.33 | 64.01 | 19.93 | 61.86 | 33.21 | 75.80 | 31.83 | 72.59 |
| No warm-up | 14.50 | 64.36 | 19.41 | 61.02 | 33.34 | 75.46 | 31.75 | 72.17 |
| $\alpha = 0.3$ | 14.28 | 64.72 | 19.71 | 61.74 | 33.51 | 75.68 | 31.79 | 72.56 |
| $\epsilon = 0.3$ | 14.10 | 64.37 | 19.95 | 61.80 | 33.36 | 75.75 | 31.16 | 72.38 |
| $n = 10$ | 14.74 | 64.92 | 19.83 | 61.30 | 33.66 | 75.93 | 31.71 | 72.82 |

Table 5: MT performance using the TSCL algorithm when hyperparameters vary.

| Hyperparameter values | AZ → EN | | BE → EN | | SK → EN | | GL → EN | |
|---|---|---|---|---|---|---|---|---|
| | BLEU | COMET | BLEU | COMET | BLEU | COMET | BLEU | COMET |
| Default | 15.62 | 65.86 | **21.11** | **62.82** | 34.54 | 76.15 | 33.02 | 73.73 |
| Hidden size $= 128$ | 15.86 | **66.15** | 20.38 | 62.64 | 34.40 | 76.21 | 32.59 | 73.79 |
| $\tau = 0.5$ | 15.55 | 66.13 | 20.17 | 62.47 | 34.52 | 76.25 | **33.13** | **73.81** |
| $\tau = 0.995$ | **16.06** | 66.06 | 20.19 | 62.18 | **34.55** | **76.31** | 32.45 | 73.46 |
| $\gamma = 0.5$ | 15.78 | 65.85 | 20.51 | 62.64 | 34.49 | 76.16 | 32.94 | 73.63 |
| $\gamma = 0.01$ | 15.34 | 65.38 | 20.59 | 62.36 | 33.87 | 75.70 | 32.43 | 73.39 |

Table 6: MT performance using the DQN algorithm when hyperparameters vary.

language. To avoid an entirely synthetic vector, we pick an actual vector occurring during training and multiply by 5 the loss values of all 25 batches from the targeted language.

We probe the Q network with each of the 8 source languages in turn, pretending that this language is not well learned and observing the action selected by the network, i.e. the language that it requires the NMT model to see next. Rather than observing the single selected language, we considered the softmaxed output activations for all 8 languages. The result is thus an 8-by-8 matrix, represented in Figure 4 at 28k and respectively 56k training steps. The X-axis represents the softmaxed output activations (predicted Q-values for each language), while the lines of the Y-axis correspond to each probed language (the one for which loss values were amplified).
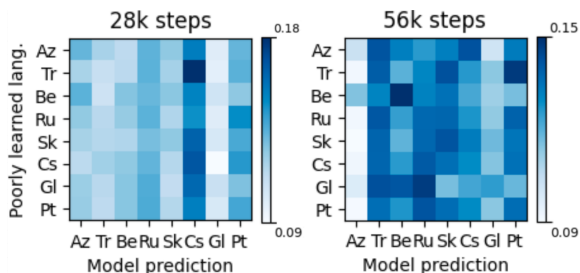


Figure 4: Behavior of the Q network at two stages during training.

The state at 28k steps is typical of incomplete training. The Q network appears to favor one HRL language (CZ) regardless of the language that is the least well learned according to its synthetic vector. The Q network selects one language for a large number of steps and gradually switches to another. The state at 56k steps (when the NMT trained with DQN reaches its best score) demonstrates a more balanced behavior: if one language is insufficiently learned, especially a LRL, then the network predicts that more training should be done on that language. Indeed, for several lines (though not all), the cell on the diagonal is one of the darkest of the line (e.g. for TR, BE, SK or PT). These observations suggest that the DQN model's decisions are complex and evolve over time, rather than always favoring the language that is currently the least well learned.

## 6 Conclusion and Perspectives

In this paper, we presented two algorithms for optimizing the training schedule of multilingual NMT models when a mixture of HRLs and LRLs must be learned on the source side. The TSCL algorithm models the expected return of each action by smoothing past observations, while DQN trains a neural network to perform this estimation and to select the optimal action.

Both algorithms strike a balance between a uniform distribution of training batches across lan-

guages and a distribution purely based on the respective frequencies of these languages in the actual data. The algorithms increase the proportions of LRLs and reduce those of HRLs, while still enabling cross-lingual transfer from HRLs to related LRLs. The better balance of HRLs and LRLs avoids too great a focus on the more abundant HRL data (which would sacrifice LRLs) or too great a focus on LRLs (which would lead to overfitting). Without such algorithms, it would be difficult to find extrinsic criteria to optimize the presentation frequencies of batches. Moreover, the optimized training schedules lead to improved macro-average BLEU and COMET scores.

We leave for future work the study of other ways to construct batches. One option is to use multilingual batches – though, as shown above, shuffled batches underperform with respect to an optimized balance of LRLs and HRLs. Another option is to define actions as specific batches or groups of batches, which would enable the model to prioritize certain batches over others, but would also increase the number of possible actions and hence the learning complexity of the RL agent.

The relevance of our proposal should be tested with additional datasets combining HRLs and related LRLs, and with other neural architectures for which cross-lingual transfer may be important to ensure acceptable performance on LRLs, particularly LLMs fine-tuned on translation tasks (Alves et al., 2024). In such cases, an optimized training schedule across available resources may also be beneficial.

## Acknowledgments

## References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Duarte M. Alves, José Pombal, Nuno M. Guerreiro, Pedro H. Martins, João Alves, Amin Farajian, Ben Peters, Ricardo Rei, Patrick Fernandes, Sweta Agrawal, Pierre Colombo, José G. C. de Souza, and André F. T. Martins. 2024. Tower: An open multilingual large language model for translation-related tasks. *arXiv preprint arXiv:2402.17733*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Àlex R. Atrio, Alexis Allemann, Ljiljana Dolamic, and Andrei Popescu-Belis. 2024. Can the variation of model weights be used as a criterion for self-paced multilingual NMT? *arXiv 2410.04147*.

Àlex R. Atrio and Andrei Popescu-Belis. 2022. On the interaction of regularization factors in low-resource neural machine translation. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 111–120, Ghent, Belgium. European Association for Machine Translation.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana. Association for Computational Linguistics.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. *arXiv 2402.01306*.

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

Peter J. Huber. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101.

Sébastien Jean, Orhan Firat, and Melvin Johnson. 2019. Adaptive scheduling for multi-task learning. *arXiv 1909.06434*.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. 2015. Self-paced curriculum learning. *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 1:2694–2700.

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

Samuel Kiegeland and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Julia Kreutzer, David Vilar, and Artem Sokolov. 2021. Bandits don't follow rules: Balancing multi-facet machine translation with multi-armed bandits. *arXiv 2110.06997*.

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. 2019. Reinforcement learning based curriculum optimization for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2054–2061, Minneapolis, Minnesota. Association for Computational Linguistics.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in Neural Information Processing Systems (NeurIPS)*, 23.

Xuebo Liu, Houtim Lai, Derek F. Wong, and Lidia S. Chao. 2020. Norm-based curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 427–436, Online. Association for Computational Linguistics.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2017. Teacher–student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, arXiv:1707.00183.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with deep reinforcement learning. *arXiv 1312.5602*.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: a framework and survey. *Journal of Machine Learning Research*, 21(1).

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Long Ouyang et al. 2022. Training language models to follow instructions with human feedback. *arXiv 2203.02155*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv 2305.18290*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *arXiv 1511.06732*.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova,

Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv 1609.04747*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv 1707.06347*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yingli Shen and Xiaobing Zhao. 2024. Reinforcement learning in natural language processing: A survey. In *Proceedings of the 2023 6th International Conference on Machine Learning and Natural Language Processing*, MLNLP '23, page 84–90, New York, NY, USA. Association for Computing Machinery.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2021. Multilingual translation from denoising pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3450–3466, Online. Association for Computational Linguistics.

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*, volume 30. Curran Associates, Inc.

Yu Wan, Baosong Yang, Derek F. Wong, Yikai Zhou, Lidia S. Chao, Haibo Zhang, and Boxing Chen. 2020. Self-paced learning for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1074–1080, Online. Association for Computational Linguistics.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Chenglong Wang, Hang Zhou, Yimin Hu, Yifu Huo, Bei Li, Tongran Liu, Tong Xiao, and Jingbo Zhu. 2024. ESRL: Efficient sampling-based reinforcement learning for sequence generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19107–19115.

Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. 2020a. Learning a Multi-Domain Curriculum for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7711–7723, Online. Association for Computational Linguistics.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising Neural Machine Translation Training with Trusted Data and Online Data Selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143, Brussels, Belgium. Association for Computational Linguistics.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. *arXiv 1902.03499*.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020b. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online. Association for Computational Linguistics.

Jiawei Wu, Lei Li, and William Yang Wang. 2018a. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1252–1262, New Orleans, Louisiana. Association for Computational Linguistics.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018b. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

Minghao Wu, Yitong Li, Meng Zhang, Liangyou Li, Gholamreza Haffari, and Qun Liu. 2021. Uncertainty-aware balancing for multilingual and multi-domain neural machine translation training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7291–7305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chen Xu, Bojie Hu, Yufan Jiang, Kai Feng, Zeyang Wang, Shen Huang, Qi Ju, Tong Xiao, and Jingbo Zhu. 2020. Dynamic curriculum learning for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3977–3989, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Asaf Yehudai, Leshem Choshen, Lior Fox, and Omri
Abend. 2022. Reinforcement Learning with Large
Action Spaces for Neural Machine Translation. In
*Proceedings of the 29th International Conference
on Computational Linguistics*, pages 4544–4556,
Gyeongju, Republic of Korea. International Com-
mittee on Computational Linguistics.

Poorya Zaremoodi and Gholamreza Haffari. 2019.
Adaptively scheduled multitask learning: The case of
low-resource neural machine translation. In *Proceed-
ings of the 3rd Workshop on Neural Generation and
Translation*, pages 177–186, Hong Kong. Association
for Computational Linguistics.

Mingliang Zhang, Fandong Meng, Yunhai Tong, and
Jie Zhou. 2021. Competence-based curriculum learn-
ing for multilingual machine translation. In *Find-
ings of the Association for Computational Linguis-
tics: EMNLP 2021*, pages 2481–2493, Punta Cana,
Dominican Republic. Association for Computational
Linguistics.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Ken-
ton Murray, Jeremy Gwinnup, Marianna J Mar-
tindale, Paul McNamee, Kevin Duh, and Marine
Carpuat. 2018. An empirical exploration of curricu-
lum learning for neural machine translation. *arXiv
1811.00739*.

Mingjun Zhao, Haijiang Wu, Di Niu, and Xiaoli Wang.
2020. Reinforced curriculum learning on pre-trained
neural machine translation models. In *Proceedings
of the AAAI Conference on Artificial Intelligence*,
volume 34(05), pages 9652–9659.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu,
Shujian Huang, Lingpeng Kong, Jiajun Chen, and
Lei Li. 2024. Multilingual machine translation with
large language models: Empirical results and anal-
ysis. In *Findings of the Association for Computa-
tional Linguistics: NAACL 2024*, pages 2765–2781,
Mexico City, Mexico. Association for Computational
Linguistics.

# A Appendix

## A.1 Epsilon Scheduling for DQN

The RL agent in the DQN algorithm follows, as explained in Section 3.2, an $\epsilon$-greedy policy: with a probability of $1-\epsilon$, actions (i.e. the source language of a batch) are selected using the Q network, but with a probability of $\epsilon$, a random action is selected. The exact schedule of $\epsilon$ is shown in Figure 5.
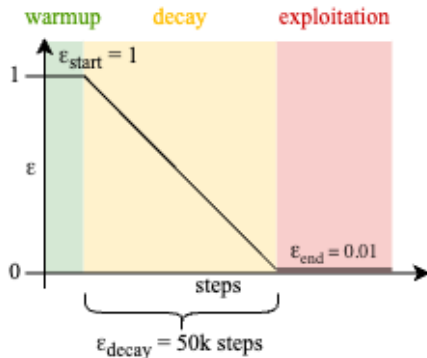


Figure 5: Evolution of $\epsilon$ during training with DQN.

During the warm-up period of 16k a value of 1 means that the Q network is not used and source HRLs (in this case) are drawn randomly. Then, during a decay of 50k steps, the importance of the Q network in deciding the actions grows progressively, while random choices decrease to a minimal probability of 0.01 after 66k steps. This approach, inspired by Kumar et al. (2019), achieves a balance between exploiting the Q network and exploring new actions.

## A.2 Convergence Speed

In the experiments presented above, the scores were computed using a rolling ensemble of 4 checkpoints, and the best score was selected as the highest macro-average of BLEU achieved on the development data of the LRLs. We mentioned at the end of Section 5.2 that the DQN was the method that reached optimal scores after the smallest number of steps, followed by TSCL and then by the 'proportional' scheduling. The exact numbers of steps are given in Table 7, showing that DQN accelerates convergence with respect to the other schedules. Moreover, this behavior is stable when varying some of the hyperparameters of the algorithm, as shown in Table 8.

## A.3 Learned Policies

We presented in Section 5.3 the total numbers of actions of each type (i.e. source language of the

| Training | Warm up | Best checkpoint |
|---|---|---|
| Uniform | no | 136k |
| Proportional | no | 60k |
| Uniform | 16k | 124k |
| Proportional | 16k | 60k |
| Shuffle batch | 16k | 128k |
| TSCL | 16k | 56k |
| DQN | 16k | 52k |

Table 7: Comparison of the number of steps required by the NMT model to achieve the best scores on the LRLs.

| Parameter values | Best checkpoint |
|---|---|
| Default | 52k |
| $\tau = 0.5$ | 60k |
| $\tau = 0.995$ | 76k |
| $\gamma = 0.5$ | 48k |
| $\gamma = 0.01$ | 36k |
| Hidden layer: 128 | 76k |

Table 8: Comparison of the number of steps required for the NMT model using the DQN algorithm to achieve the best scores on the LRLs. The parameter values are the default ones, except the changes shown for each line.

batch) selected during training for the TSCL and DQN algorithms, in comparison to the 'uniform' and 'proportional' training schedules. Here, we show in Figure 6 the evolution of the proportion of actions during training with the DQN algorithm, aggregated every 1000 steps.

In this representation, we first observe that the initial 16k steps are performed only on the HRLs, as configured. When the DQN algorithm starts playing a role, a random selection of languages is observed. As the algorithm learns, the proportions of LRLs decrease, while the proportions of HRLs increase, and tend to stabilize towards steady-state values. The proportions averaged over the entire training period are provided in the legend of the chart. These are the proportions compared between the systems in Figure 3.

## A.4 The TSCL Algorithm

The full specification of the TSCL algorithm in pseudo-code is provided hereafter as Algorithm 1.

## A.5 The DQN Algorithm

The full specification of the DQN algorithm in pseudo-code is provided hereafter as Algorithm 2.
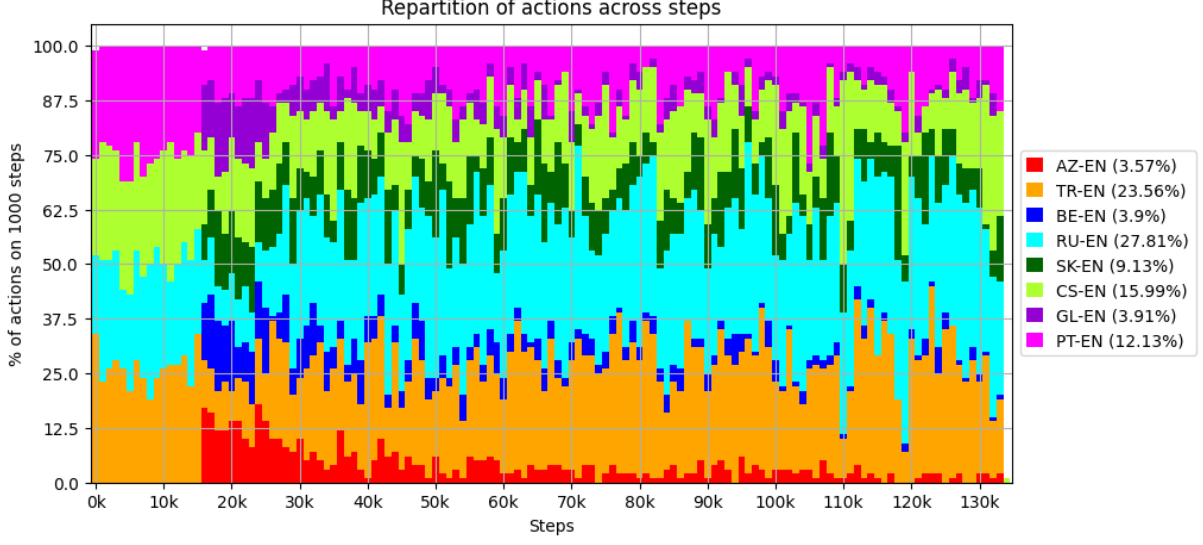
Figure 6: Evolution of the proportions of actions (i.e. language of batches) chosen during the training of the DQN system. Aggregation windows of 1000 steps are computed. The first 16k steps are the warmup on the HRLs only.

---

**Algorithm 1:** TSCL algorithm for NMT.

**Require:** actions $\mathcal{A} \leftarrow \{A_1, .., A_k\}$, number of training steps $ts$, number of consecutive actions $n$, number of warm-up steps $w$, $\epsilon$-greedy policy exploration parameter $\epsilon$, smoothing coefficient $\alpha$

1  Initialize NMT model
2  Initialize action index $i \leftarrow 1$
3  Initialize unvisited actions indexes $U \leftarrow \{2, .., k\}$
4  Initialize estimated return $Q(A_k) \leftarrow 0$ for all $k$ actions
5  Initialize rewards history $H(A_k) \leftarrow 0$ for all $k$ actions
6  **for** $t \leftarrow 1, \ldots, ts$ **do**
7      Sample mini-batch $B_t$ from action $A_i$
8      Train NMT model using mini-batch $B_t$
9      **if** $t \bmod n = 0$ **then**
10         Observe reward $R_t \leftarrow X_t - H(A_i)$
11         Update reward history $H(A_i) \leftarrow X_t$
12         Exponentially smooth estimated return $Q(A_i) \leftarrow \alpha R_t + (1 - \alpha)Q(A_i)$
13         **if** $|U| \neq 0$ **then**
14             Choose action index $i \leftarrow U[0]$
15             Update $U \leftarrow U - \{i\}$
16         **end**
17         **else**
18             Choose random number $r$ between 0 and 1
19             **if** $t < w$ **or** $r < \epsilon$ **then**
20                 Choose action index $i$ randomly between 1 and $k$
21             **end**
22             **else**
23                 Set $i$ as the index of the max arg. of absolute values in $Q$
24             **end**
25         **end**
26     **end**
27 **end**

**Algorithm 2:** DQN Algorithm for NMT

---

**Require:** actions $\mathcal{A} \leftarrow \{A_1, .., A_k\}$, number of training steps $ts$, number of consecutive actions $n$, number of warm-up steps $w$, $\epsilon$-greedy policy exploration parameter $\epsilon$, soft update coefficient $\tau$, replay memory capacity $c$, minimum replay memory capacity $c_{min}$

1   Initialize NMT model learning algorithm
2   Initialize RL agent's online model
3   Initialize replay memory deque $\mathcal{D}$ with capacity $c$
4   Initialize RL agent's target network with same weights as RL agent's online model
5   Initialize action index $i \leftarrow 1$
6   **for** $t \leftarrow 1, \ldots, T$ **do**
7     Sample mini-batch $B_t$ from action $A_i$
8     Train NMT model using mini-batch $B_t$
9     **if** $t \bmod n = 0$ **then**
10       **if** $t < w$ **then**
11         Choose action index $i$ randomly between 1 and $k$
12       **end**
13       **else**
14         Observe current state $S_t$
15         Observe reward $R_t \leftarrow X_t - X_{t-1}$
16         Store transition $(S_{t-1}, i, R_t, S_t)$ in replay memory $\mathcal{D}$
17         **if** $|D| >= c_{min}$ **then**
18           Sample mini-batch of transitions $T$ from replay memory $\mathcal{D}$
19           Train RL agent's online model using mini-batch $T$
20           Soft update RL agent's target model weights with RL agent's online model weights
            $\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$
21         **end**
22         Choose random number $r$ between 0 and 1
23         **if** $r < \epsilon$ **then**
24           Choose action index $i$ randomly between 1 and $k$
25         **end**
26         **else**
27           Predict Q values at state $S_t$ with RL agent target network
28           Set $i$ as the index of the arg. max in $Q$
29         **end**
30         Decrease $\epsilon$ according to decay schedule
31       **end**
32     **end**
33   **end**