Can the Variation of Model Weights be used as a Criterion for Self-Paced Multilingual NMT?

Anonymous ACL submission

Abstract

Many-to-one neural machine translation systems improve over one-to-one systems when training data is scarce. In this paper, we design and test a novel algorithm for selecting the language of minibatches when training such systems. The algorithm changes the language of the minibatch when the weights of the model do not evolve significantly, as measured by the smoothed KL divergence between all layers of the Transformer network. This algorithm outperforms the use of alternating monolingual batches, but not the use of shuffled batches, in terms of translation quality (measured with BLEU and COMET) and convergence speed.

1 Introduction

Multilingual neural machine translation (MNMT) systems can be trained with several languages on the source side, or on the target side, or on both sides (Firat et al., 2016; Johnson et al., 2017). Many-to-one MNMT systems are particularly effective for low-resource languages (LRLs) on the source side, when they are accompanied by high-resource languages (HRLs) related to them (Gu et al., 2018). For instance, Neubig and Hu (2018) trained a many-to-one recurrent model on a multilingual dataset of almost 60 languages and showed that including HRLs in the training data reduces the chance of overfitting to the LRLs and improves translation quality. Aharoni et al. (2019) used Transformer models (Vaswani et al., 2017) to further improve over these results.

Many-to-one MNMT systems are usually trained with multilingual batches sampled from all source languages to avoid catastrophic forgetting (Jean et al., 2019), but the presence of several languages in a minibatch may ineffectively constrain the model and prevent it from training on the languages where training is most needed. An open question in many-to-one MNMT, therefore, is how the data from different source languages should be sampled during training, particularly when massive imbalances in sizes or difficulties occur across languages.

In this paper, we propose a dynamic scheduling approach which samples minibatches from the source languages based on the variation of weights in the layers of a Transformer. The main idea is the following one: when a model becomes competent for translating a certain source language, as indicated by a decreasing variation of a model's weights across training steps, then the language of the minibatches should be switched to a new one, in order to allocate more time to more challenging, hence useful tasks.

The main contributions of the paper are the precise formulation, implementation and testing of the idea. Specifically, we propose to: **(i)** measure variation of weights by comparing the weights of all layers of a Transformer across two consecutive steps with the same source language; **(ii)** compare weights by using symmetric KL divergence between softmaxes of layers, with exponential smoothing across time; **(iii)** trigger a change of task, i.e. source language, when weight variation decreases; **(iv)** compare translation quality and convergence speed for 2-to-1 and 8-to-1 MNMT on a dataset with four language families on the source side, and one HRL and one LRL for each of them (Neubig and Hu, 2018).

2 Related Work

Neubig and Hu (2018) study the upsampling of the HRL data when building minibatches, and observe that keeping the original proportions of HRL and LRL performs marginally better. Aharoni et al. (2019) also sample each batch uniformly from a concatenation of all language pairs. Arivazhagan et al. (2019) compare a simple concatenation with uniform balancing (Johnson et al., 2017), but observe better results for LRLs when translating into a HRL by using a temperature-based upsampling, which has been favored afterwards (Conneau et al.,

2020; Tang et al., 2021). As a method for dynamic scheduling of multitask training (Caruana, 1997), self-pacing consists in using the target model to quantify the difficulty of each sample or dataset – that is, measure the model's *competence* – and inform the scheduling module dynamically (Kumar et al., 2010). For MNMT, Jean et al. (2019) compare adaptively upsampling a language depending on various factors, observing best results on the LRLs when dynamically changing the gradient norm (Chen et al., 2018). Wang et al. (2020) adaptively balance the languages by learning language weights on the model's competence on a development set. Zhang et al. (2021) adaptively learn a sampling strategy by measuring per-language competence and LRL competence evaluated with a HRL's competence.

3 Method for Self-Paced MNMT

To train a many-to-one MNMT model, we consider M parallel datasets which correspond to as many tasks $\mathcal{T} = \{T_1, \ldots, T_M\}$ with different source languages and their respective English translations. Our algorithm chooses on which task T_c to train the MNMT model with parameters θ_t at each time step t, based on an estimation of the model's competence for each task (i.e., source language). The overall goal is to *increase time spent on tasks where the model is less competent, and to avoid overtraining on tasks where the model is already competent*.

We estimate the per-task competence of the model as the average variation of its weights in all layers (due to the back-propagation of gradients) at a given training step. We thus measure competence as the Kullback-Leibler divergence (D_{KL}) between the updated weights and the weights at the previous step at which the model was trained on the same task. Originally used to quantify the dissimilarity between two probability distributions P and Q, D_{KL} is defined as: $D_{\mathrm{KL}}(P||Q) = \sum_x P(x)\log(P(x)/Q(x))$ where x are the possible values of the P and Q random variables. To use D_{KL} as a distance measure between two sets of weights in a neural network, we apply softmax σ to convert the weights to probability distributions. Moreover, we take the logarithm of the first term in KL to handle the potential issue of capacity overflow and maintain the stability of divergence calculations (Liang et al., 2021). Finally, we symmetrize the distance by summing KL divergence in both directions. Therefore, we compute the average variation between two sets of values θ_{t-1} and θ_t of all the trainable weights of a Transformer network (layers 1 through L) as follows:

$$D(\theta_{t-1}, \theta_t) = \frac{1}{2L}\sum_{i=1}^{L} D_{\mathrm{KL}}(\log(\sigma(\theta^i_{t-1}))||\sigma(\theta^i_t)) + D_{\mathrm{KL}}(\log(\sigma(\theta^i_t))||\sigma(\theta^i_{t-1})). \quad (136)$$

Furthermore, we ensure that when the training switches to another task, the model trains on it for at least two updates, so that both θ_{t-1} and θ_t are the result of training on minibatches of the same source language: with this, we avoid measuring a large variation between weights simply as the result of switching between tasks. In order to obtain a task-switching schedule that is robust to local variations, we apply exponential smoothing and compute per-task competence, transforming D into D'_c as follows:

$$D'_c(\theta_{t-1}, \theta_t) = (1-w)D(\theta_{t-1}, \theta_t) + wD'_c(\theta_{t-k}, \theta_{t-1}) \quad (148)$$

where $k \geq 2$ is the smallest value such that $B_{t-k} \in T_c$ (in other words, $t-k$ is the latest step before $t-1$ for which $B_{t-k} \in T_c$). The smoothing weight was set at $w = 0.995$ after empirical analyses (see Appendix A.2).

The proposed algorithm for dynamic scheduling (Algorithm 1 below) has the following rationale. If the network is trained on a task T_c and the weight variation across consecutive steps increases, we consider that the network lacks competence on T_c and should keep training on it. Conversely, the less the weights change, the more competent the model is. So, if weight variation slows down, then training on the same task produces diminishing returns, and the network should switch to a task on which it is less competent. This condition appears in line 8 of Algorithm 1.

We define the model's per-task competences at step t as $\mathcal{C} = \{C_1, \ldots, C_M\}$, such that $C_c = D'_c(\theta_{j-1}, \theta_j)$, and $j \leq t$ is the last step such that minibatch $B_j \in T_c$. That is, for each $T_i \in \mathcal{T}$, C_i is the result of exponential smoothing over the weight variations of all the updates in which θ has trained on a minibatch from T_i. We define a sampling function – noted 'sample*' in line 10 of the algorithm – which, with the following role: in the initial phase, it randomly samples any of the $T_i \in \mathcal{T}$ on which the system has never been trained on; then, when all tasks have been seen at least once, it samples

a new $T_c \in \mathcal{T}$ based on the softmaxed per-task competence distribution $\sigma(\mathcal{C})$. Additionally, we introduce hyper-parameter α in order to compare the importance of previous weight variation versus the current one (line 8). However, after empirical analyses, we found that the best results were obtained with $\alpha = 1$ (see Appendix A.3).

Algorithm 1: Self-paced scheduling algorithm for MNMT using the variation of model weights.

Require: tasks $\mathcal{T} = \{T_1, .., T_M\}$, steps s
1 $T_c \leftarrow T_1$;
2 **for** $t \leftarrow 1, \ldots, s$ **do**
3 Sample minibatch B_t from T_c;
4 $\theta_{t+1} \leftarrow \theta_t - \eta \nabla_{\theta_t} L_{B_t}(\theta_t)$;
5 **if** changedTask **then**
6 changedTask $\leftarrow$ False;
7 **end**
8 **else if** $D'_c(\theta_{t-1}, \theta_t) < \alpha D'_c(\theta_{t-2}, \theta_{t-1})$ **then**
9 $C_c \leftarrow D'_c(\theta_{t-1}, \theta_t)$;
10 $T_c \leftarrow \text{sample}^*(\mathcal{T} - \{T_c\})$;
11 changedTask $\leftarrow$ True;
12 **end**
13 $t \leftarrow t + 1$;
14 **end**

In training, we use the 'noam' learning rate schedule (Vaswani et al., 2017, Eq. 3), which increases linearly from zero during the warmup steps, and afterwards decays proportionally to the inverse square root of the current step. Although the variation in weights throughout the entire training is strongly influenced by the learning rate schedule (Figure 2), we find that when comparing the smoothed weight variations between two near-consecutive steps, the influence of the learning rate variation is negligible. Finally, we note that our algorithm carries little computational overhead, since the self-assessed competence is obtained from the weight variation across standard updates.

4 Data and Systems

4.1 Corpora

We experiment on a subset of the multilingual TED corpus (Qi et al., 2018). As in previous multilingual studies (Neubig and Hu, 2018; Wang et al., 2019), we focus on four pairs of related LRL-HRL, as shown in Table 1, with the goal of translating them into English (EN).

4.2 Tokenization

As the data is already tokenized, we directly use Byte Pair Encoding (BPE) (Sennrich et al., 2016) for subword extraction and vocabulary construction. We learn a vocabulary by concatenating 10k random lines from each language in the training data, and upsample the LRL if it has fewer lines. For experiments involving only one LRL and one HRL in the source, we learn a vocabulary of 10k subwords. For experiments involving all four LRLs and all four HRL in the source, our vocabulary has 32k subwords. To facilitate language identification, we prefix the dataset of each language with a unique tag.

LRL	train	dev	test	**HRL**	train
BE	4.51k	248	664	RU	208k
AZ	5.94k	671	903	TR	182k
GL	10.0k	682	1.0k	PT	51.8k
SK	61.5k	2.2k	2.4k	CS	103k

Table 1: Data sizes for pairs of LRLs and HRLs.

4.3 System Architecture

We use Transformer models (Vaswani et al., 2017) from the OpenNMT-py library (Klein et al., 2017) version 3.1.1. In all our systems we use the following default values of hyper-parameters from Transformer-Base: 6 encoder/decoder layers, 8 attention heads, label smoothing of 0.1, hidden layer of 512 units, and FFN of 2,048 units. We use Adam optimizer (Kingma and Ba, 2014) and a batch size of 10k tokens. After empirical testing (Appendix A.6) we observe better quality with systems using more aggressive regularization. In particular, from the recommended values by OpenNMT-py, we increase the dropout rate to 0.3, the scaling factor to 10 and the number of warmup steps to 16k, and we re-normalize gradients if their norm is greater than 5. Our 2-to-1 and 8-to-1 models have 59M and 93M of parameters respectively. We train all our models on 2 GPUs (GTX 1080 or RTX 2080) for a maximum of 26 hours. In this study we trained approximately 40 models.

4.4 Evaluation

For each pair, we measure the BLEU score (Papineni et al., 2002) on the LRL test set using the SacreBLEU library[1] (Post, 2018) as well as the COMET score (Rei et al., 2020) using model `wmt22-comet-da`. We use bootstrap resampling from SacreBLEU to compute the 95% confi-

[1] github.com/mjpost/sacrebleu
signature: `nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.3.1.`

Task	System	2-to-1 BLEU	2-to-1 COMET	2-to-1 Updates	8-to-1 BLEU	8-to-1 COMET	8-to-1 Updates
BE→ EN	*shuffled*	**21.7** (±1.3)	**63.8**	48k	**20.0** (±1.4)	61.4	**64k**
	alternation	19.8 (±1.2)	61.5	44k	18.7 (±1.3)	61.3	120k
	self-paced	20.5 (±1.3)	62.8	**32k**	19.7 (±1.3)	**61.8**	128k
AZ→ EN	*shuffled*	**15.6** (±1.0)	**66.0**	**32k**	14.3 (±1.0)	**64.4**	144k
	alternation	14.4 (±1.0)	63.9	44k	**16.6** (±1.0)	62.9	**140k**
	self-paced	14.5 (±1.0)	64.9	48k	14.4 (±1.0)	64.0	150k
GL→ EN	*shuffled*	**30.2** (±1.2)	70.9	50k	**31.9** (±1.3)	**72.8**	**92k**
	alternation	30.0 (±1.2)	71.0	50k	30.4 (±1.2)	71.3	136k
	self-paced	**30.2** (±1.1)	**71.2**	**44k**	30.7 (±1.2)	72.0	150k
SK→ EN	*shuffled*	**33.6** (±0.8)	**76.2**	24k	**33.9** (±0.9)	**75.4**	**32k**
	alternation	33.4 (±0.8)	75.3	**20k**	31.8 (±0.9)	73.5	144k
	self-paced	33.3 (±0.9)	75.3	**20k**	31.9 (±0.9)	74.0	128k
Average of the four LRLs	*shuffled*	**25.3**	**69.2**	39k	**25.0**	**69.0**	**83k**
	alternation	24.4	68.0	40k	24.4	67.3	135k
	self-paced	24.6	69.0	**36k**	24.2	68	139k

Table 2: Results of the three methods compared on four 2-to-1 setups and an 8-to-1 setup, as well as the number of updates necessary to obtain the highest scores. We use our stronger regularization hyper-parameters (as in Table 6), and denote in bold the best score in each metric for each task.

dence interval around the mean of the BLEU score. We use a rolling ensemble of four checkpoints and select the best on the development set for the final translations.

5 Results

In Table 2 we present the scores of models trained on the four 2-to-1 setups (presented in Section 4.1) by order of increasing LRL size, as well as an 8-to-1 setup, which includes all the tasks. For each MNMT system, we compare three methods: first, we train a model on multilingual batches, by upsampling all the tasks until they are the same size and then *shuffling* them. Second, we apply a cyclical *alternation* of monolingual batches for each task, which results in the model being trained the same amount of time on all tasks. Third, we apply our *self-paced method* described in Section 3. Introducing more source languages does not improve scores in either of the three methods, although we observe a small negative effect on higher-resourced LRLs, which has been reported previously (Neubig and Hu, 2018; Aharoni et al., 2019). *Self-paced* tends to perform better than *alternation* on 2-to-1, but is more severely affected on an 8-to-1 setup. Both of these methods, which rely on monolingual updates, clearly underperform with respect to *shuffled*, which is trained with multilingual batches.

Additionally, in the 2-to-1 case, the difference between *shuffled* and *self-paced* decreases as the size of the dataset increases, but in the 8-to-1 case, the difference increases as the dataset size also increases. This indicates that with a small amount of training tasks, the more available data, the less important the sampling method is, but with many training tasks a more careful selection of the balancing of the data becomes more important for lower-resourced datasets. Regarding convergence speed, we measure the amount of updates that each model requires in order to reach its highest BLEU scores. We observe that all three methods train in nearly the same speed on the 2-to-1 case, but on the 8-to-1 case we observe that training with monolingual batches, regardless of the balancing of the tasks, results in a much slower training. This is likely due to either the model forgetting what it learned the last time it trained on a given task, or to the monolingual updates resulting in weights that are less useful to the other tasks.

6 Conclusion and Future Work

In this study we have presented a self-paced method to balance tasks in a many-to-one MNMT system by monitoring the average per-task weight variation across steps, with the objective of not over-training on tasks in which the model is competent, and better allocating resources to tasks in which the model is less competent. Our method carries no dedicated computational overhead. However, we have observed that a multilingual, uniform balancing of all tasks outperforms our method both on 2-to-1 and 8-to-1 setups.

7 Limitations

A limitation of our method may be that measuring the weight variation between two consecutive updates might result in too small a value, with too many oscillations even with the use of smoothing. Additionally, we have shown that as the amount of training tasks increases, performing single-task updates is counter-productive, both in quality and in speed. In the future, we hope to extend our method to assemble multilingual batches based on the per-task weight variation in order to solve this issue.

8 Ethics Statement

This study does not process personal or sensitive data. While MT in general may facilitate disclosure or cross-referencing of personal information, which may pose threats to minorities, the community appears to consider that the potential benefits far outweigh the risks, judging from the large number of studies for low-resource and unsupervised MT.

Acknowledgments

We are grateful for the support received from Armasuisse (UNISUB projet: Unsupervised NMT with Innovative Multilingual Subword Models) and from the Swiss National Science Foundation (DOMAT project: On-demand Knowledge for Document-level Machine Translation, n. 175693).

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.

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 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.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR.

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.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

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.

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

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.

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

M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 10890–10905. Curran Associates, Inc.

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.

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

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.

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

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.

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.

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

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. 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.

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

A Appendices: Design Choices and Hyper-parameter Setting

Firstly, we study the effect of several metrics to measure weight variation (Section A.1). Next, we perform various experiments in order to optimize our method, involving: the smoothing parameter w (Section A.2), the importance of the previous weight variation α (Section A.3), and training during the warmup steps only on the HRL, which simulates a pre-training regime (Section A.4). Experiments in Sections A.2 to A.4 are performed in a 2-to-1 setup (GL-PT-to-EN) using default hyper-parameters.

A.1 Weight Variation Metric

In order to measure the weight variation of a model between steps, firstly we train a model on a unidirectional low-resource NMT task (60k lines) and compare measuring the average of all weight matrices versus the last output layer, and using KL divergence as our metric, inverse cosine similarity, or L2 norm. We show in Figure 1 these six combinations, computing the variations every 10 steps and performing a rolling average with a window size of 100. We can observe in all of them the effect of the learning rate schedule (warmup steps and decay). Additionally, we also note a more regular pattern when measuring the change across all matrices versus the last layer. We decide on using KL divergence as our measure due to it striking a balance between the irregularity of the inverse cosine similarity and the L2 norm.

A.2 Setting of the Smoothing Weight

In Figure 2 we show the average weight variation between all weight matrices when experimenting

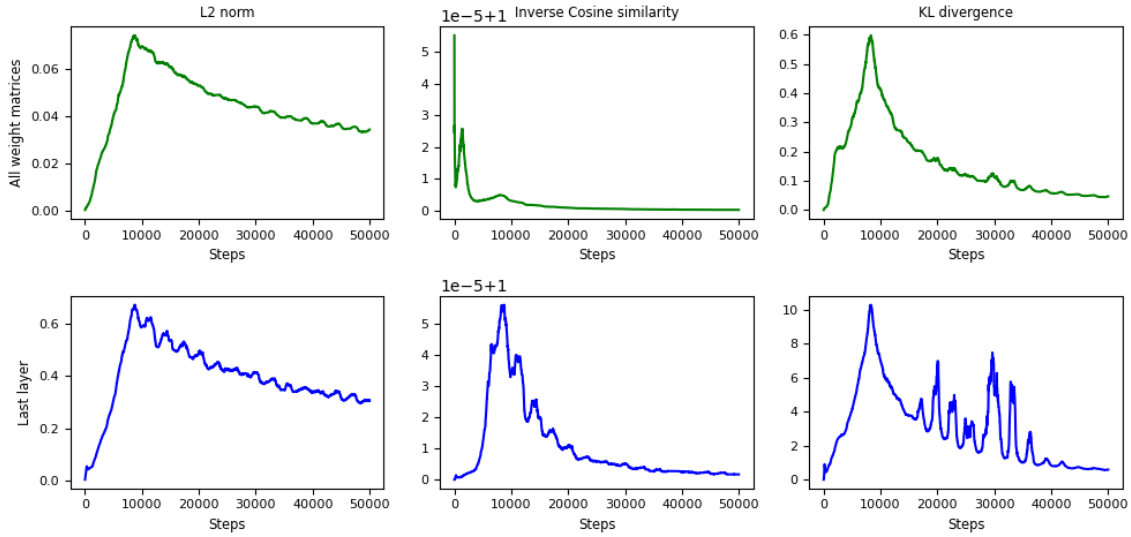

Figure 1: Comparison between three different metrics for model weight variation: L2 norm, inverse cosine similarity, and KL divergence. For each of them we compare monitoring the average over all weight matrices and only the final output layer.

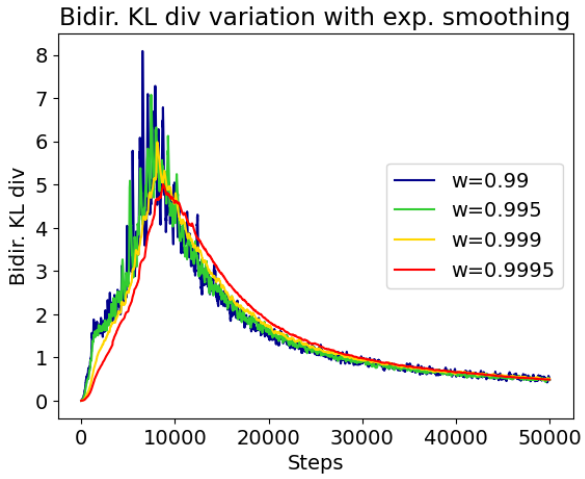

Figure 2: Evolution of the bidirectional Kullback-Leibler divergence for different values of the exponential smoothing coefficient w in an experiment on GL-PT→EN with dynamic sampling

#	System	w	Updates	BLEU
	shuffled	-	44k	**27.49**
	alternation	-	44k	25.64
	self-paced	0.99	48k	25.76
	self-paced	0.995	48k	25.92
	self-paced	0.999	40k	**26.28**
	self-paced	0.9995	28k	25.42

Table 3: BLEU scores on the LRL test set of our method with several values of the smoothing coefficient, w. We denote in bold the best result in the comparison methods, as well as in our method, and we underline our chosen value for w.

with various values for w, and in Table 3 the resulting scores, which we compare to *shuffled* and *alternation*. We can see that increasing w not only produces a more regular weight-variation curve, but also accelerates training without much loss in test score. Nonetheless, although some of the smoothing values produce better scores than a simple *alternation* of monolingual batches, none of them improve over the multilingual *shuffled* batches. We select a $w = 0.995$ for our main experiments as a balance between translation quality and regularity of the weight variation curve.

A.3 Importance of Previous Weight Variation

System	α	Updates	BLEU
shuffled	-	44k	**27.49**
alternation	-	44k	25.64
self-paced	0.9	48k	11.12
self-paced	0.95	48k	11.91
self-paced	1.0	48k	**25.92**
self-paced	1.1	40k	25.04
self-paced	1.2	44k	25.37

Table 4: BLEU scores on the LRL test set of our method with several values of the importance hyper-parameter α, with $w = 0.995$. We denote in bold the best result in the comparison methods, as well as in our method, and we underline our chosen value for α.

Similarly, we also experiment on the value of α, a hyper-parameter to weight the importance of the previous weight variation when comparing steps t and $t-1$. In Table 4 we show the results of training

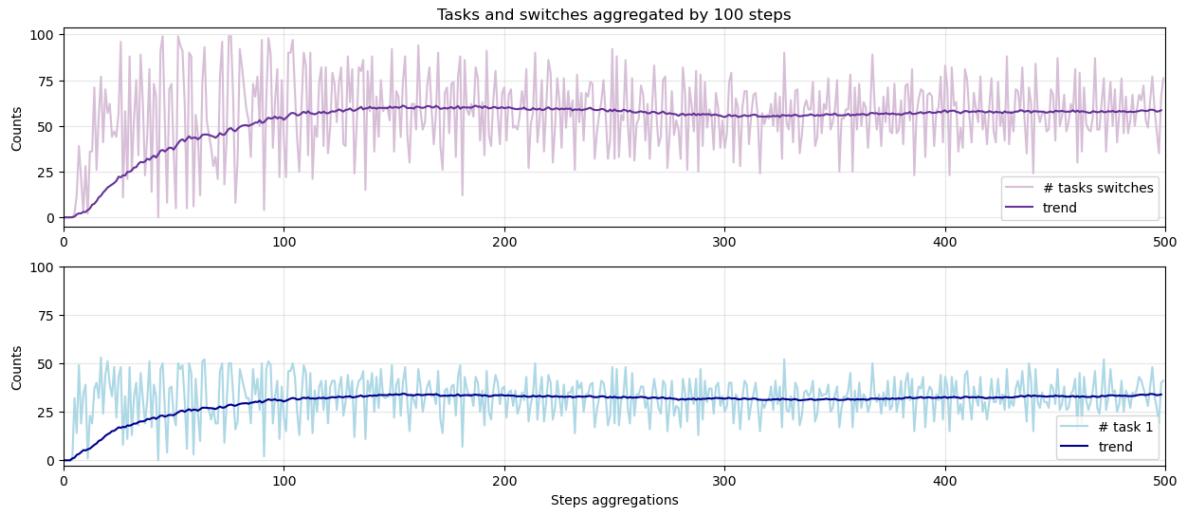

Figure 3: Amount of task switches and percentage of training on the LRL (*task 1*).

with our selected value of $w = 0.995$ and various values for α. We observe best results without any additional weighting of the previous variation, and so select for our main experiments $\alpha = 1$.

A.4 Training with HRL Warmup Steps

System	HRL warmup	Updates	BLEU
alternation	no	36k	24.92
alternation	yes	44k	**26.04**
self-paced	no	48k	25.92
self-paced	yes	48k	**26.16**

Table 5: BLEU scores on the LRL test set of our method when observing the role of HRL warmup, with $\alpha = 1$ and $w = 0.995$. We denote in bold the best result in the comparison methods, as well as in our method, and underline our chosen final technique.

Due to the effect that the learning rate warmup steps has on weight variation during the first 8k steps of training, we also consider starting training the two methods involving monolingual batches (*alternation* and *self-paced*) exclusively on the HRL, which simulates a pre-training regime. We show in Table 5 the effects of HRL warmup between each of these two methods. We observe *alternation* benefits significantly (+1 BLEU points) from HRL warmup, but our *self-paced* method much less noticeably. We do not consider HRL warmup to produce a positive balance between complexity and score improvement for our method, so we do not perform it in our main experiments.

A.5 Amount of Task Switches and Balancing

Finally, in Figure 3 we show the amount of task switches in training, aggregated by 100 steps, where *task 1* is the LRL. We can see that on this 2-to-1 case, after initial learning rate warmup steps, our method settles on a third of the training consisting of the LRL and two thirds on the HRL.

A.6 Hyper-Parameter Search

We search for the appropriate level of regularization to apply to our approach, by considering 2-to-1 MNMT systems, and compare the same methods as in Section 5. For each of the methods we compare a model trained with default hyper-parameters and more regularized ones (Atrio and Popescu-Belis, 2022).

System	BLEU	COMET	Updates
shuffled	22.1	64.7	36.0k
+ regularization	**25.3**	**69.2**	**38.5k**
alternation	20.9	63.2	35.0k
+ regularization	**24.4**	**68.0**	**39.5k**
self-paced	21.2	63.8	49.0k
+ regularization	**24.6**	**68.5**	**36.0k**

Table 6: Average BLEU scores on the test sets of the four LRLs on 2-to-1 setups for the three sampling strategies, with standard and increased regularization (best scores in bold).

The average BLEU scores over the four LRL tests sets of each model are shown in Table 6. Training with more regularization improves all three methods by 3 to 3.5 BLEU points. Additionally, the more regularized models improve over the scores

of previous studies on the same data (Neubig and Hu, 2018; Aharoni et al., 2019; Wang et al., 2020). On this 2-to-1 setup we obtain better results when training with multilingual *shuffled* batches, and a small improvement of *self-paced* versus an *alternation* of monolingual batches. All regularized models methods reach their highest BLEU score at a very similar number of updates, although when training with more aggressive regularization, we observe a noticeable improvement in speed in the *self-paced* method.