On the Interaction of Regularization Factors in Low-resource Neural Machine Translation

Àlex R. Atrio[1,2] and **Andrei Popescu-Belis**[1,2]

[1]HEIG-VD / HES-SO
Yverdon-les-Bains
Switzerland

[2]EPFL
Lausanne
Switzerland

{alejandro.ramirezatrio, andrei.popescu-belis}@heig-vd.ch

Abstract

We explore the roles and interactions of the hyper-parameters governing regularization, and propose a range of values applicable to low-resource neural machine translation. We demonstrate that default or recommended values for high-resource settings are not optimal for low-resource ones, and that more aggressive regularization is needed when resources are scarce, in proportion to their scarcity. We explain our observations by the generalization abilities of sharp vs. flat basins in the loss landscape of a neural network. Results for four regularization factors corroborate our claim: batch size, learning rate, dropout rate, and gradient clipping. Moreover, we show that optimal results are obtained when using several of these factors, and that our findings generalize across datasets of different sizes and languages.

1 Introduction

The training of neural machine translation (NMT) models is governed by many hyper-parameters, which play a central role in the performances of the trained models, especially their generalization abilities. While most of the NMT frameworks recommend default values for the hyper-parameters, when it comes to low-resource settings, fewer guidelines are available.

This study systematically explores the roles and interactions of a subset of hyper-parameters in low-resource NMT settings, namely those acting as *regularization factors*. Regularizers do not fall under a single theoretical definition: Goodfellow et al. (2016, page 224) view them as a collection of methods “intended to reduce generalization error but not training error.” We present here a unified perspective on several regularizers which act upon the estimation of the gradients during back-propagation. Using the distinction made by Keskar et al. (2016) between flat and sharp basins in the loss landscape, we argue that noisier estimates of the gradients can increase the likelihood of finding flatter minima, which have better generalization abilities. Specifically, we defend three claims:

1. NMT models benefit from more aggressive regularization when the amount of training data is small. We demonstrate this for four different regularizers: batch size, learning rate, dropout, and gradient clipping. We compare the default regularization hyper-parameters of the OpenNMT-py framework for mid-to-high resources – comparable to those of the original Transformer (Vaswani et al., 2017) – with the ones we optimized for a low-resource setting (Sections 4-7).

2. The combination of different regularization sources is preferable over their individual use. When used together, an amount of regularization from each of the four factors under study outperforms the use of any single one alone, and the best scores are robust with respect to the variation of each factor (Section 8).

3. Regularization factors optimized on one low-resource dataset are beneficial for low-resource datasets in other languages, and benefit from more aggressive regularization as the amount of training data decreases. We demonstrate this by comparing our default and optimized settings on data samples of varying sizes from our main corpus and four additional low-size datasets (Section 9).

2 Background and Related Work

2.1 Regularizers and the Loss Landscape

In the absence of a general treatment of regularization factors, most studies combine them empirically and search only a very small part of the hyper-parameter space. Kukačka et al. (2017) provide a taxonomy of regularization factors, but continue to define them simply as techniques to improve generalization. Similarly, in their survey, Moradi et al. (2020) consider as regularization any "component of the learning process or prediction procedure that is added to alleviate data shortage," but do not provide a common measure of regularization or consider the combination of factors.

Peng et al. (2015) study regularization techniques independently as well as in combination, still without a common theoretical underpinning. On two NLP tasks, they observe that using two factors – namely, L2 norm of weights and embeddings, and dropout – is better than using either by itself. Moreover, when using both factors, if one is set to its optimal value obtained when used alone, the other one must be lowered.

We adopt here the perspective put forward by Keskar et al. (2016), among others, who explain the *generalization gap* between values of regularization factors in terms of the topography of the loss landscape. Given a minimum of the loss function, the slower this function varies around its neighborhood (hence creating flat basins in the topography), the *flatter* (or less sharp) is the region. Models that are optimized in flatter regions tend to generalize better, and moderately less accurate gradients give models a higher probability of finding these flatter regions.

Here, we narrow down our perspective to a set of regularization factors that concern the estimation of the gradients of the loss function, as they are used during training with back-propagation. According to the above perspective, models trained with noisier gradient estimates are more likely than models trained with precise ones to find *flat* minima of the loss function, as their identification requires less precision. Additionally, a moderate amount of noise confers "exploratory abilities" that allow the search to exit sharper basins. Therefore, there is an optimal amount of noise in the gradient estimation: with too much noise, training is hampered or becomes impossible, but with too little noise, the model is likely to get trapped into sharp minimizers with low generalization abilities.

For instance, in the case of batch size (a frequently studied regularization factor), Goodfellow et al. (2016, Chapter 8.1.3) explain that models trained with smaller batch sizes tend to optimize into low-precision regions because they use noisier gradient estimates than when training with larger batch sizes.

Hypothesizing that noisier gradients improve the chance of a model to optimize into flatter regions, Smith and Le (2017) and Smith et al. (2017) propose a gradient noise scale to measure how learning rate (another regularization factor) should be adjusted to the batch size, on image data. They estimate the average gradient noise g for each batch as $g = \epsilon(N/B - 1) \approx \epsilon N/B$ where ϵ is the learning rate, N the size of the training set, and B the batch size, assuming that $N \gg B$. This shows that "increasing the batch size and decaying the learning rate are quantitatively equivalent" (Smith et al., 2017, Sec. 1).

Jastrzębski et al. (2018) also note that the proportionality of batch size and learning rate is crucial for gradient descent convergence, and the ability of the resulting model to generalize well. In particular, higher ratios seem to lead to flatter minima, which lead to better generalization, similar to what Keskar et al. (2016) observed. Specifically whether the relation between batch size and learning rate is linear, squared, or otherwise, has not been conclusively determined (Krizhevsky, 2014; Hoffer et al., 2017; Popel and Bojar, 2018). The roles of the batch size and learning rate have often been discussed from the perspective of computer vision, but different studies have made different observations, and the debate has not been settled yet (Dinh et al., 2017; Hoffer et al., 2017; Goyal et al., 2017; Li et al., 2017; Kawaguchi et al., 2017). As for dropout and gradient clipping, which are additional regularization factors, they have not been considered yet in relation to flat and sharp minimizers. We will consider here that the claim that less accurate gradients lead to flatter minima applies to them too: for dropout, due to removing some components of the sums; and for clipping, by affecting the norm of the gradient.

2.2 Regularization Factors for NMT

Recent NMT models are based on the Transformer (Vaswani et al., 2017), a deep encoder-decoder neural network which is quite sensitive to the hyper-parameters governing regularization factors during training. We discuss here the four parameters that we study in this paper.

Batch size. As we saw, models trained with smaller batch sizes have better generalization capabilities. However, batch size is not only a regularization factor, but has an influence on training speed: larger batches accelerate training by making a better use of the GPU memory.

Learning rate is a positive scalar that controls how much the weights are updated. We use the dynamic learning schedule known as 'noam' (Vaswani et al., 2017, Eq. 3). During its initial steps, known as *warmup*, the learning rate increases linearly from zero, reaching its highest value at the last warmup step w. Afterwards, it decays proportionally to the inverse square root of the step number s. At each step, this is multiplied by a factor based on the output size of the embedding layer d_{model} (512 in Transformer-Base). Moreover, following OpenNMT-py's recommendation, we include a *scaling factor* (sf), which we set by default to 2. The learning rate lr at each step s:

$$lr(s) = sf \cdot d_{model}^{-0.5} \cdot min\left(s^{-0.5},\ s \cdot w^{-1.5}\right) \quad (1)$$

Dropout (Srivastava et al., 2014) consists of a masking noise: a probability that a unit is randomly turned off during training. It is applied on the output of each hidden layer, including the output of the attention layers, but not the embedding layer, so no loss of input or output data occurs. This encourages each hidden unit to perform well regardless of other units (Goodfellow et al., 2016, Chapter 7.12).

Gradient clipping consists of renormalizing the gradient g to a threshold v if it exceeds it, i.e. if $||g|| > v$, then $g \leftarrow gv/||g||$ (same direction but bounded norm). Therefore, the smaller the value of v, the more aggressively we clip the gradients, and the more regularization is applied (Goodfellow et al., 2016, Chapter 10.11.1).

2.3 Role of Regularization for NMT

Popel and Bojar (2018) report that BLEU scores increase with batch size in a Transformer-based NMT system, although with diminishing returns, and recommend setting a large batch size. They observed moderate changes across a large range of learning rates, and found thresholds beyond which training was much slower or diverged. They made similar observations for warmup steps, concluding that the search space for learning rate and warmup steps was wide. Their experiments were performed on large datasets, leaving their questions open for low-resource settings.

Ott et al. (2018) observe that training time with very large datasets can be shortened when using larger batch sizes: they accumulate batches from 25k tokens per batch to 400k. When paired with an increased learning rate schedule (noam's times two) they do not report performance loss.

Sennrich and Zhang (2019) found that smaller batch sizes (1k-4k) were beneficial for low-resource NMT, and studied a variety of regularization factors for recurrent neural networks. However, the regularization factors were not disentangled, and their effects on Transformer-based NMT are difficult to extrapolate.

Araabi and Monz (2020) studied the Transformer's hyper-parameters in several low-resource settings. They observed improvements for larger batch sizes on the larger datasets, but did not observe improvements with smaller batch sizes on smaller datasets, or changes to optimal number of warmup steps or learning rate. They concluded to the need for larger batches from the Transformer. However, due to the late position of the batch size and learning rate in their order of optimization of the hyper-parameters, their regularizing effects cannot be precisely determined.

Xu et al. (2020) computed gradients while accumulating minibatches, and observed that increasing batch size stabilizes gradient direction up to a certain point, which allowed them to dynamically adjust batch sizes while training. Miceli Barone et al. (2017) observed improvements when combining dropout with L2-norm during fine-tuning, and concluded that "multiple regularizers outperform a single one."

In previous work, we observed improvements of scores and training time when using smaller batch sizes, with a Transformer on a low-resource dataset (Atrio and Popescu-Belis, 2021). We found a minimum value of the batch size below which training diverged, but did not study other regularization factors and interactions between them.

Studies on the optimization and effects of regularization factors thus remain scarce. Many previous studies optimize parameters in sequence. While this strategy is certainly a faster approach to optimization, it does not shed full light on each factor in isolation, as we do below in Sections 4 to 7, or in combination, as we study in Sections 8

Dataset	Src-tgt	Lines	Words (tgt)
WMT20 Low-res	HSB-DE	60k	823k
=	=	40k	550k
=	=	20k	273k
NewsComm. v13	DE-EN	120k	3M
TED Talks	SK-EN	61k	1.3M
=	SL-EN	19k	443k
=	GL-EN	10k	214k

Table 1: Numbers of lines of the original corpora used in our experiments. Sections 4-8 use only the first dataset. We do not use monolingual or back-translated data, and train our tokenizers using only each parallel corpus.

and 9.

3 Data and Systems

We train our NMT systems with the Upper Sorbian (HSB) to German (DE) training data of the WMT 2020 Low-Resource Translation Task (Fraser, 2020). We also use the HSB-DE development and test sets provided by the WMT 2020 and 2021 Low-Resource Translation Tasks (Fraser, 2020; Libovický and Fraser, 2021), each consisting of 2k sentences. As length-based filtering does not show significant differences, we do not filter our data. Additionally, in Section 9, we train systems for translation from Galician (GL), Slovenian (SL), and Slovak (SK) into English (EN), using tokenized and cleaned transcriptions of TED Talks (Qi et al., 2018).[1] Finally, we train a larger German to English system using 120k lines from News Commentary v13 (Bojar et al., 2018), and sample 1,500 lines each for development and testing. Table 1 presents these resources.

Tokenization into subwords is done with a Unigram LM model (Kudo, 2018) from SentencePiece.[2] For each language pair we build a shared vocabulary of 10k subwords using only the parallel corpus, with character coverage of 0.98, *nbest* of 1 and *alpha* of 0.

We use the Transformer-Base architecture (Vaswani et al., 2017) implemented in OpenNMT-py (Klein et al., 2017; Klein et al., 2020).[3] Our default setting of hyper-parameters is the one recommended by OpenNMT-py[4] which is close to the original Transformer (Vaswani et al., 2017). The regularization factors appear with relatively low strengths in this setting, as is usual when large datasets are available. The setting includes the 'noam' learning rate schedule with a scaling factor of 2 and a dropout rate of 0.1. For Adam, $\beta_1 = 0.9$, $\beta_2 = 0.998$ and $\epsilon = 10^{-8}$.

We train our models for a maximum of 100 hours, although they generally converge earlier. When comparing batch sizes in Section 4, it could be argued that epochs might provide a fairer comparison, but we measure real clock time as the most relevant measure for practitioners.

A batch consists of lines (tokenized sentences) that are translated one by one, with a fixed maximum length of 512 tokens for Transformer-Base. Lines are padded if shorter, and filtered out if longer. We train all models on two GPUs with 11 GB of memory each (GeForce RTX 1080Ti). Each device processes several batches, depending on the batch size, which are afterwards accumulated and used to update the model. The *effective batch size* and the `batch_size` parameter of OpenNMT-py are two different values: the former is $G \times A \times \texttt{batch\_size}$, where G is the number of GPUs and A the number of accumulated batches, here equal to two.[5] Throughout the paper, we report the `batch_size` parameter, but the effective batch size is in fact four times larger.

We generate translations with a beam size of seven, with consecutive ensembles of four checkpoints. For each model we report the highest BLEU score (Papineni et al., 2002) calculated with SacreBLEU (Post, 2018) on detokenized text[6] as well as the chrF score (Popović, 2015). We test the statistical significance of differences in scores at the 95% confidence level using paired bootstrap resampling from SacreBLEU.

4 Batch Size

In this section we train models with batch sizes ranging from 500 to 10,000, with all other hyper-parameters set to default. Models with batch sizes of 100 and 250 were also trained, but did not converge. The largest tested batch size is the largest value supported by our GPUs.

The BLEU and chrF scores in Table 2 show that lowering the batch size improves quality of NMT,

[1] `https://github.com/neulab/word-embeddings-for-nmt`

[2] `https://github.com/google/sentencepiece`

[3] We make public our configuration files and package requirements at `https://github.com/AlexRAtrio/reg-factors`.

[4] `https://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model`

[5] `https://forum.opennmt.net/t/epochs-determination/3119`

[6] `https://github.com/mjpost/sacrebleu` with the signature nrefs:1|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0.

Batch Size	train		dev		test	
	Xent	Acc.	BLEU	chrF	BLEU	chrF
0.5k	0.02	99.93	50.54*	73.35	43.95^	69.25
1k	0.01	99.94	**52.02**	**74.63**	**44.40**^	**70.02**
3k	0.01	99.96	50.16*	73.38	43.91^	69.16
6k	0.01	99.97	49.66+	73.09	42.55–	68.85
9k	0.01	99.96	49.42+	73.10	42.22–	68.40
10k	0.01	99.97	48.46	72.49	42.19–	68.38

Table 2: HSB-DE scores with various batch sizes, all other settings being default ones. Values with the same color or symbol are *not* significantly different. The highest scores are in bold.

likely due to the regularizing effect of a less accurate gradient, according to our theoretical perspective. In particular, we observe improved results with a batch size smaller than 3,000 (+1.71 BLEU) and an optimal size around 1,000 (+2.21), with scores gradually decreasing as batch size increases. These results are in line with previous observations (Sennrich and Zhang, 2019; Atrio and Popescu-Belis, 2021).

There is no clear correlation between the training accuracy or cross-entropy loss and the generalization capacity, i.e. the scores on the development and test sets. The lower scores of models trained with larger batch sizes are likely not due to overfitting, because the *testing* curves of these models do not show any decrease late in the training. This further supports the claim that better generalization abilities are due to flat minima (Keskar et al., 2016, Section 2.1).

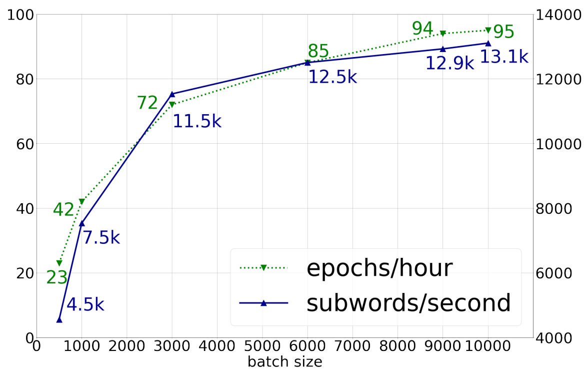

Figure 1: Throughput (subwords/second, in blue) and speed (epochs/hour, in green) for the tested batch sizes.

Our results are competitive with the comparable baselines from the WMT20 shared task on low-resource NMT for HSB-DE (Fraser, 2020), which used the same parallel data.[7] The baseline BLEU scores of Knowles et al. (2020), Libovický et al. (2020) and Kvapilíková et al. (2020) were respectively 44.1, 43.4, and 38.7 on the test set.

Regularization through smaller batch sizes thus provides visible improvements with respect to the default setting. Larger batch sizes, however, exploit more fully the memory of the GPUs, which enables higher throughput in terms of subwords processed per second, as illustrated in Figure 1, although this does not increase linearly: instead, we observe diminishing returns as batch size increases. Still, while a batch size of 10k has the lowest BLEU scores, it nearly doubles the throughput with respect to the highest-scoring batch size (1k). Due to differences in hardware and software, these values are difficult to compare to other studies, but the trends are similar to those observed by Popel and Bojar (2018, Section 4.1).

If the regularization attained with lower batch sizes can also be obtained by using other regularization factors, this would allow the use of larger batch sizes for a more efficient training. Therefore, in the next sections, we will compare a large batch size (10k) and the optimal, regularized one (1k), and verify that none of the other regularization factors that will be optimized have an impact on speed.

5 Learning Rate

Previous studies by Smith et al. (2017) and Smith and Le (2017) have shown that the regularization effects of the batch size and of the learning rate may be comparable. In this section, we study the role of varying schedules of the learning rate (5.1) and the effect of resetting the schedule in mid-training, i.e. suddenly increasing the learning rate before another decrease (5.2).

5.1 Regularization through Learning Rate

Since all our models have the same dimension of embeddings (d_{model} in Eq. 1 above), the only variables influencing the learning rate in the 'noam' schedule are the number of warmup steps and the scaling factor (Vaswani et al., 2017, Eq. 3). We test two different values for the former: 8k (default) and 16k. For the latter, we test even values from 2 (default) to 14. Figure 2 displays some tested schedules, including our default one (8k, 2) and the 'noam' original one (4k, 1).

The results in Table 3 show that both batch sizes reach similar maximal scores (46.20 and 46.29),

[7]Some of these systems used in fact larger monolingual HSB, DE and/or CS datasets for training their tokenizers, while we only used 60k lines of parallel HSB-DE text.

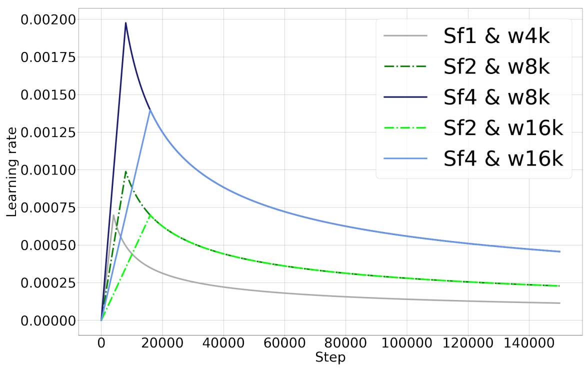

Figure 2: 'Noam' learning rate schedules with different scaling factors (*sf*) and numbers of warmup steps (*w*).

although with different scaling factors: 6 for a batch size of 1k vs. 10 for a batch size of 10k. The improvement is 1.8 BLEU points for a batch size of 1k, and 4.1 for 10k. As a batch size of 1k is already a strong regularization factor, a smaller value of the learning rate (hence less regularization through this factor) is sufficient, compared to the case of a larger batch size.

War mup	**Scaling factor**						
	2	4	6	8	10	12	14
8k	44.40	**45.42**	38.90	0.65	0.18	0.05	0.60
16k	43.96	45.74	**46.20***	46.07*	45.79*	45.24*	42.24
8k	42.19	44.59	45.27⋆	**45.93**–	45.87–	45.34⋆	45.31⋆
16k	41.70	44.36	45.32+	45.89^	**46.29**^	45.69+	45.69+

Table 3: BLEU scores on the HSB-DE test set for batch sizes of 1k (top) and 10k (bottom) and various learning schedules. We denote scores that are *not* significantly different row-wise with the same color or symbol.

The models trained with the larger batch size (10k) are more stable when learning rates increase (larger scaling factors) likely due to more accurate estimates of the gradients (compare lines 1 vs. 3, and 2 vs. 4). Similarly, these models have a higher maximal learning rate beyond which they diverge (compare in Table 3 the large difference between lines 1 and 2 with the small difference between lines 3 and 4). This shows the importance of increasing the number of warmup steps as the scaling factor increases, to avoid reaching high maxima of the learning rate (the peaks visible on the schedules in Figure 2). Moreover, the regularization provided by other factors (in this case, batch size) needs to be taken into account when increasing the amount of regularization from the learning rate. Finally, as long as the maximal learning rate remains below the values that make a model diverge, the BLEU scores do not change significantly when the scaling factor increases above a certain value, as also observed by Popel and Bojar (2018, 4.6, Fig. 7).

5.2 Resetting the LR during Training

From the perspective of the loss landscape, we hypothesize that introducing more noise into the gradient when the scores have already leveled-off, namely by resetting the learning rate schedule, should increase the probability for the weights to escape the sharp minima basins and avoid falling back into them, which should improve the generalization abilities of the trained model. Since a model trained with a smaller batch size has a higher chance, during the first part of training, to fall into flat minima due to an increased gradient noise (Smith et al., 2017), we expect the larger batch sizes to benefit more from this strategy than the smaller ones.

Batch size		**Hours** 50	100 no lr reset	100 reset lr
1k	BLEU	44.25	44.40	**45.85**
	chrF	69.78	70.02	70.84
	Train. Acc.	99.93	99.94	99.84
	Xent	0.02	0.01	0.02
	Δ		*+0.15*	***+1.60***
10k	BLEU	41.60	42.19	**45.25**
	chrF	68.03	68.38	70.57
	Train. Acc.	99.94	99.97	99.92
	Xent	0.01	0.01	0.01
	Δ		*+0.59*	***+3.65***

Table 4: BLEU and chrF scores on the HSB-DE test set, training accuracy and cross-entropy on the training set, and change of BLEU scores when continuing training until 100 hours vs. resetting the learning rate at 50h.

In Table 4 we provide the scores after training for 50 hours (half of their training time); the scores after 100 hours when continuing to train from the 50-hour checkpoint; and the final score after training for 50 hours with a schedule reset at the 50-hour checkpoint. The results corroborate our hypothesis: both batch sizes benefit significantly from the strategy of resetting the learning rate, and the large batch size more than the smaller one ((+3.65 vs. +1.6 BLEU points). As both models reached their highest BLEU scores before 25 hours, the difference is likely not due to that fact that the first model saw more times the training data thanks to its higher throughput. Furthermore, after increasing the learning rate mid-training, both the loss and training accuracy worsen or remain stable, while BLEU scores improve, likely due to reaching flatter basins, not lower minima.

6 Dropout Rate

The dropout of a certain proportion of neurons during training is another frequent source of regularization. As this amounts to removing certain terms from the summation of gradients, its role can also be considered from the perspective of flat vs. sharp minimizers.

Dropout							
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
44.40*	45.35+	**45.39**+	44.87*	44.54*	42.58	37.69	19.83
42.19	43.76	44.74	**45.40**^	45.39^	45.26^	42.91	35.52

Table 5: Dropout scores on the HSB-DE test set for 1k (top) and 10k (bottom) batch sizes. We denote row-wise lack of significant differences with the same color or symbol. Dropout rates of 0.9 have considerably lower scores.

BLEU scores in Table 5 show that the model trained with a larger batch size – hence subject to less regularization – requires a more aggressive dropout of around 0.4–0.6 in order to reach its highest scores, with respect to a model trained with a smaller batch size, which reaches its highest score for 0.2–0.3. This is consistent with our previous findings from Section 5.1 and Table 3, which also showed that the model subject to less regularization from a factor (larger batch size) required more regularization from another factor in order to reach its highest scores.

7 Gradient Clipping

Finally, we experiment with our fourth regularization factor: gradient clipping. Since it directly involves constraining the norm of the gradient, the perspective based on flat vs. sharp basins in the loss landscape also holds for it.

Batch size	Drop out	Gradient Clipping None	20	10	5	2.5
1k	0.1	44.40	44.75	**44.92**	44.74	44.54
10k	0.1	42.19	**42.41**	42.01	42.30	42.20
	0.2	43.76	44.15	**44.34**	43.98	43.85
	0.3	44.74	**45.36**	44.72	44.75	44.99
	0.4	45.40	**45.56**	45.30	45.45	45.48

Table 6: BLEU scores on the HSB-DE test set for batch sizes of 10k and 1k on the test set, with a dropout rate of 0.1 (default), for several upper limits of the gradients.

As in the previous sections, we compare models trained with batch sizes of 1k and 10k, but observe no statistically significant differences between them when using default values for other hyper-parameters, with BLEU scores shown in Table 6 – although values of 10 or 20 are always among the best. This is likely because default settings do not feature enough regularization (i.e., they do not increase enough the gradient's norm) for the gradients to be affected by clipping. For this reason, we perform additional experiments with a batch size of 10k (due to its advantage for speed) with more regularizing dropout values of 0.2, 0.3, and 0.4, and scaling factor of 6 and 10. Regarding the models with increasing dropout rate, we only observe a statistically significant difference between the best and worst results (for dropout of 0.2), the best and two worst results (for 0.3), and no differences at all (for 0.4). We conclude that gradient clipping only marginally affects training in these settings.

8 Combining Regularization Factors

We will now show that a combination of regularization factors can produce higher scores than individual factors used separately, and that the maximal scores are stable when varying the strengths of regularizers around their optimal values. The batch size is fixed at 10k, since this enables a higher training speed than 1k with similar best scores, provided that other regularization factors are used, as shown in Tables 3, 4 and 5. The number of warmup steps is fixed at 16k since we showed in Section 5.1 that this parameter mainly limits the peaks of the learning rate and thus prevents models from diverging early in the training. Our search space for the other regularization factors is shown in Table 8.

Factor	Value	Xent	Tr. acc.	BLEU	chrF	Δ
Defaults	-	0.01	99.97	42.19	68.38	-
Batch size	1k	0.02	99.94	44.40	70.02	+2.21
S.f.	10	0.01	99.94	45.93	70.74	+3.74
S.f. + w.s.	10+16k	0.01	99.94	46.29	71.22	+4.10
L.r. reset	50%	0.01	99.92	45.25	70.57	+3.06
Dropout	0.4	0.07	99.46	45.40	71.00	+3.21
Clipping	10	0.01	99.96	42.41	68.43	+0.22
Combination	Table 8	0.03	99.78	**47.11**	**71.88**	**+4.92**
+ l.r. reset	-	0.06	99.30	**47.20**	**71.80**	**+5.01**

Table 7: HSB-DE scores on the test set when the regularization factors are used either independently (lines 2–6) or in combination (line 7), in the latter case with the optimal values from Table 8. The last column shows increases in BLEU scores over the default settings.

We present in Table 7 the highest scores achieved using individual regularization factors, along with those from the default setup (first line) and from the combination of factors (last two lines). Regularization factors are already present in the default setup, but at low strengths.

The comparison of scores in Table 7 shows that each factor used independently allows the model to outperform the default setting by 2–4 BLEU points. However, the use of a combination of factors achieves the highest score of 47.20 BLEU points (+ 5.01), which is significantly above all others. In the case of resetting the learning rate, although this has a visible effect when used with default parameters, its effect is much smaller when used jointly with other regularization factors, likely because a flat basin is found before the reset. Moreover, the combination of factors results in a higher loss and a lower accuracy on the train set than the default setup or factors used individually, which supports our interpretation of the improvement based on flatter minima.

Table 8 shows that the best scores reached with increased regularization are quite stable when varying the intensity of the factors. The optimal region of the scaling factor is around 10, with a relatively flat neighborhood, similar to the case when it was optimized individually (Section 5). Optimal dropout rates are now around 0.3–0.5, compared to 0.4–0.6 when used individually (Section 6). Finally, gradient clipping has only a marginal effect in combination with other factors, presumably because it cannot help to increase the gradients.

9 Testing on Additional Corpora

In this section, we confirm our claims using additional low-resource datasets. We consider two smaller samples with 40k and 20k lines from the HSB-DE corpus, as well as parallel datasets for Galician, German, Slovak and Slovenian (see Section 3). We do not optimize regularization factors on each dataset, but only use the optimal hyper-parameters found above on HSB-DE with 60k lines.

Table 9 demonstrates that these hyper-parameter values bring significant improvements of BLEU and chrF scores over the baseline for all datasets (four different source languages). When comparing HSB-DE datasets of different sizes, we find that as the amount of data decreases, the positive effects of our regularization parameters increase, with up to 21% improvement in BLEU scores for the smallest subset. Furthermore, we also observe an increase in the loss over all datasets with the optimized setup, which shows that the reason why their less accurate gradients generalize better is not due to finding lower but rather flatter minima of loss.

Grad clipping	**Scaling factor**	**Dropout** 0.1	0.3	0.5	0.7
None	2	42.19	44.74	45.39	42.91
	6	45.32	46.70	46.22	43.66
	10	46.29	47.06	46.93	43.18
	14	45.69	46.84	47.07	43.61
	18	45.26	46.89	46.67	43.19
5	2	41.39	44.47	45.05	43.48
	6	45.20	46.62	46.70	43.88
	10	45.65	**47.11**	46.76	44.04
	14	45.57	**47.11**	47.06	43.63
	18	44.72	46.59	47.02	42.72

Table 8: HSB-DE BLEU scores for a combination of the scaling factor, gradient clipping, and dropout rate, for a batch size of 10k and 16k warmup steps. The highest scores are in bold.

10 Conclusion

We presented a unified perspective on the role of four regularization factors in low-resource settings: batch size, learning schedule, gradient clipping and dropout rate. The results support our claim that more regularization is beneficial in such settings, with respect to the default values that are recommended for high-resource settings. We first substantiated the claim for each factor taken individually, and then showed that a combination of factors leads to improved scores and is robust when factors vary. Finally, we showed that our findings generalize across different low-resource sizes and

Corpus	**Lines**	**Default** Xent	Tr. Acc.	BLEU	chrF	**Optimized** Xent	Tr. Acc.	BLEU	chrF	**%Δ** BLEU
HSB-DE	*60k*	*0.01*	*99.97*	*42.19*	*68.38*	*0.06*	*99.30*	***47.20***	*71.80*	***+11.87***
HSB-DE	40k	0.01	99.98	32.38	60.68	0.03	99.80	**37.63**	65.12	**+16.21**
HSB-DE	20k	0.01	99.98	22.93	51.42	0.02	99.93	**27.84**	56.27	**+21.41**
DE-EN	120k	0.10	98.20	29.94	56.81	0.60	84.71	**35.77**	61.44	**+19.47**
SK-EN	61k	0.02	99.89	25.61	46.42	0.40	89.29	**29.71**	49.67	**+16.01**
SL-EN	19k	0.01	99.93	15.53	34.99	0.09	98.89	**18.43**	37.75	**+18.67**
GL-EN	10k	0.01	99.98	16.00	34.52	0.04	99.69	**19.04**	37.84	**+19.00**

Table 9: BLEU scores on test sets of different corpora and subsets of our main HSB-DE corpus (first line), comparing our default setup and our optimized setup as presented in Section 8.

languages. Overall, we interpreted the results from the perspective of the loss landscape, and argued that more regularization is beneficial because the noise it introduces in the estimation of gradients leads to finding flatter minima of the loss, which have better generalization abilities. We hope that better insights on the loss landscape of the Transformer will confirm our theoretical interpretation, and that the observations put forward in this paper will also help practitioners with setting hyperparameters for low-resource NMT systems.

11 Acknowledgments

We thank the Swiss National Science Foundation (DOMAT grant n. 175693, On-demand Knowledge for Document-level Machine Translation) and Armasuisse (FamilyMT project). We especially thank Dr. Ljiljana Dolamic (Armasuisse) for her support in the FamilyMT project. We are also grateful to the anonymous reviewers and to Giorgos Vernikos for their helpful suggestions.

References

Araabi, Ali and Christof Monz. 2020. Optimizing Transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain.

Atrio, Àlex R. and Andrei Popescu-Belis. 2021. Small batch sizes improve training of low-resource neural MT. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, Patna, India.

Bojar, Ondřej, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels.

Dinh, Laurent, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. 2017. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028.

Fraser, Alexander. 2020. Findings of the WMT 2020 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 765–771.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: Training Imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Hoffer, Elad, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1729–1739.

Jastrzębski, Stanislaw, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. 2018. Width of minima reached by stochastic gradient descent is influenced by learning rate to batch size ratio. In *Proceedings of 27th International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science, pages 392–402. Springer, Cham.

Kawaguchi, Kenji, Leslie Pack Kaelbling, and Yoshua Bengio. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Keskar, Nitish Shirish, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for NMT. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 67–72.

Klein, Guillaume, François Hernandez, Vincent Nguyen, and Jean Senellart. 2020. The OpenNMT neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas*, pages 102–109.

Knowles, Rebecca, Samuel Larkin, Darlene Stewart, and Patrick Littell. 2020. NRC systems for low resource German-Upper Sorbian machine translation 2020: Transfer learning with lexical modifications. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1112–1122.

Krizhevsky, Alex. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*.

Kudo, Taku. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

Kukačka, Jan, Vladimir Golkov, and Daniel Cremers. 2017. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*.

Kvapilíková, Ivana, Tom Kocmi, and Ondřej Bojar. 2020. CUNI systems for the unsupervised and very low resource translation task in WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1123–1128.

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2017. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

Libovický, Jindřich, Viktor Hangya, Helmut Schmid, and Alexander Fraser. 2020. The LMU Munich system for the WMT20 very low resource supervised MT task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1104–1111.

Libovický, Jindřich and Alexander Fraser. 2021. Findings of the WMT 2021 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Sixth Conference on Machine Translation*.

Miceli Barone, Antonio Valerio, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, Copenhagen, Denmark.

Moradi, Reza, Reza Berangi, and Behrouz Minaei. 2020. A survey of regularization strategies for deep models. *Artificial Intelligence Review*, 53(6):3947–3986.

Ott, Myle, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Belgium, Brussels.

Papineni, Kishore, 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, PA, USA.

Peng, Hao, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2106–2111, Lisbon, Portugal.

Popel, Martin and Ondřej Bojar. 2018. Training tips for the Transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70, 4.

Popović, Maja. 2015. chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal.

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

Qi, Ye, 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*, pages 529–535, New Orleans, LA, USA.

Sennrich, Rico and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy.

Smith, Samuel L and Quoc V Le. 2017. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*.

Smith, Samuel L, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Xu, Hongfei, Josef van Genabith, Deyi Xiong, and Qiuhui Liu. 2020. Dynamically adjusting Transformer batch size by monitoring gradient direction change. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3519–3524.