PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 108 JUNE 2017 13–25

Empirical Investigation of Optimization Algorithms in Neural Machine Translation

Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter,
Christopher Jan-Steffen Brix, Hermann Ney

Human Language Technology and Pattern Recognition Group,
RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany

Abstract

Training neural networks is a non-convex and a high-dimensional optimization problem. In this paper, we provide a comparative study of the most popular stochastic optimization techniques used to train neural networks. We evaluate the methods in terms of convergence speed, translation quality, and training stability. In addition, we investigate combinations that seek to improve optimization in terms of these aspects. We train state-of-the-art attention-based models and apply them to perform neural machine translation. We demonstrate our results on two tasks: WMT 2016 En→Ro and WMT 2015 De→En.

1. Introduction

Training a neural network involves the estimation of a huge number of parameters. Ideally, optimization seeks to find the global optima, but in such a non-convex problem, global optimality is given up and local minima in the parameter space are considered sufficient to obtain the models that generalize beyond the training data (Goodfellow et al., 2016, Chapter 8). Besides obtaining better performance, choosing an appropriate optimization strategy could accelerate the training phase of neural networks and brings higher training stability.

Modeling and training problems are two major issues involved in Neural Machine Translation (NMT) systems. (Junczys-Dowmunt et al., 2016) state that averaging the parameters of a few best models from a single training run, considered as a single model, leads to improvement in terms of both translation metrics and perplexity. This

 Corresponding author: bahar@i6.informatik.rwth-aachen.de
Cite as: Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, Hermann Ney. Empirical Investigation of Optimization Algorithms in Neural Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 108, 2017, pp. 13–25. doi: 10.1515/pralin-2017-0005.

indicates that we might have a training problem since the model and the number of parameters are exactly the same in this scenario. We call this averaging averaged-best. On the other hand, building ensembles requires training several models which is time consuming, however, it is common to do that in NMT (Jean et al., 2015). Thus, an investigation is needed to discover whether either the model or the estimation of its parameters is weak.

In this work, we empirically investigate the most prominent first-order stochastic optimization methods to train an NMT system and exclusively investigate their behavior in NMT. We address three main concerns. a) translation performance, b) training stability and c) convergence speed. On one hand, how well, fast and stable different optimization algorithms are able to find appropriate local minima and on the other hand, how a combination of them can solve these aspects of training problems. The results show that applying these combinations leads to faster convergence, translation performance boost and more regularized behavior compared to running an optimizer alone. In this work, we follow the same standalone attention-based NMT proposed by (Bahdanau et al., 2015) but with different optimization schemes.

1.1. Related Work

There are many works in which researchers interpret the characteristics of different optimization techniques theoretically (Goodfellow et al., 2016; Ruder, 2016). Moreover, some other works try to show the performance of optimizers in the investigation of loss surface for image classification task such as (Im et al., 2016). (Zeyer et al., 2017) investigate various optimization methods for acoustic modeling empirically. (Dozat, 2015) compares different optimizers in language modeling. Furthermore, (Britz et al., 2017) study a massive analysis of NMT hyperparameters aiming for better optimization being robust to the hyperparameter variations.

To the best of our knowledge, there is no work comparing different optimization algorithms for NMT. Most of the works in this area focus on the modeling problem and rely on Adadelta used in vanilla NMT (Cho et al., 2014; Bahdanau et al., 2015).

Recently, (Wu et al., 2016) utilized the combination of Adam and a simple Stochastic Gradient Descend (SGD) learning algorithm. They run Adam for a fixed number of iterations after which they switch to SGD to slow down the training phase. Furthermore, (Farajian et al., 2016) optimize the networks with both Adagrad and Adadelta and show that using Adagrad leads to faster convergence and better performance.

2. Neural Machine Translation

Given a source $\mathbf{f} = f_1^I$ and a target $\boldsymbol{e} = e_1^I$ sequence, NMT (Sutskever et al., 2014; Bahdanau et al., 2015) models the conditional probability of target words given the source sequence. The NMT training objective function is to minimize the cross-entropy over the S training samples $\left\{\left\langle \mathbf{f}^{(s)}, \boldsymbol{e}^{(s)} \right\rangle\right\}_{s=1}^{S}$ which is defined as below:

$$J(\theta) = \sum_{s=1}^{S} \sum_{i=1}^{I^{(s)}} \log p(e_i{}^{(s)} | e_{<i}{}^{(s)}, f^{(s)}; \theta) \quad (1)$$

Since computing the objective function for the whole training data is expensive, we randomly select a small number of samples and take the average over them. This is so-called mini-batch training, resulting in mini-batch gradient calculations. We leave out the corresponding notations for mini-batches for simplicity.

3. Optimization

Fast convergence and robustness against stochasticity are important aspects desired in an optimizer so that it finds the global optimum. In practice, local optima can be sufficient and gradient-based techniques are able to find it (Goodfellow et al., 2016).

3.1. Stochastic Gradient Descent

The commonly used gradient-based algorithm in neural network is Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951) which updates a set of parameters, θ, as shown in Algorithm 1. η is called the learning rate, determining how large the update is and g_t represents the gradient of cost function J. Then, the parameters in the direction of the gradients are updated. For simplicity, we refer to g_{θ_t} as g_t. Through this paper, we use the term SGD to state the simple SGD defined here.

Algorithm 1 : Stochastic Gradient Descent (SGD)

1: $g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
2: $\theta_{t+1} \leftarrow \theta_t - \eta g_t$

SGD usually uses scheduling-based step size selection and the learning rate is one of the important hyperparameters of training that should be carefully tuned. Unlike simple SGD, a number of methods have been introduced to adapt the separate learning rate for each parameter, called adaptive optimizer. It is still necessary to choose proper hyperparameters for these methods, but less sensitive. The most prominent first-order gradient-based optimizers are Adagrad, RmsProp, Adadelta and Adam that are briefly discussed in the following.

3.2. Adagrad

Adagrad (Duchi et al., 2011) is a gradient-based method in which the shared global learning rate η is divided by the l_2-norm of all previous gradients, n_t, as seen in Algorithm 2, line 3. Hence, it introduces different learning rates for every parameter

at each time step, so that larger gradients have smaller learning rates and vice versa. This property helps to perform larger updates for the dimensions with infrequent changes and smaller updates for those that have already large changes. On the other hand, n_t in the denominator is a positive growing value which might aggressively shrink the learning rate. ϵ is the stabilizing numerical constant.

Algorithm 2 : Adagrad

$g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
$n_t \leftarrow n_{t-1} + g_t^2$
$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{n_t}+\epsilon} g_t$

Algorithm 3 : RmsProp

$g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2$
$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{n_t}+\epsilon} g_t$

3.3. RmsProp

Instead of storing all the past squared gradients from the beginning of the training, one can restrict a window over the recent gradients to acquire local information. An efficient way to do it is RmsProp in which instead of using the sum of squared gradients, a decaying weight of squared gradients is applied (Algorithm 3) (Hinton et al., 2012).

Algorithm 4 : Adadelta

$g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2$
$r(n_t) \leftarrow \sqrt{n_t + \epsilon}$
$\Delta\theta_t \leftarrow \frac{-\eta}{r(n_t)} g_t$
$s_t \leftarrow \nu s_{t-1} + (1-\nu)\Delta\theta_t^2$
$r(s_{t-1}) \leftarrow \sqrt{s_{t-1} + \epsilon}$
$\theta_{t+1} \leftarrow \theta_t - \frac{r(s_{t-1})}{r(n_t)} g_t$

Algorithm 5 : Adam

$g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2$
$\hat{n}_t \leftarrow \frac{n_t}{1-\nu^t}$
$m_t \leftarrow \mu m_{t-1} + (1-\mu) g_t$
$\hat{m}_t \leftarrow \frac{m_t}{1-\mu^t}$
$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{\hat{n}_t}+\epsilon} \hat{m}_t$

3.4. Adadelta

Similar to RmsProp, Adadelta takes the decaying mean of the past squared gradients. As shown in Algorithm 4, n_t accumulates this quantity, and its square root becomes the r_t of past squared gradients up to the time t. The obtained parameter update is stored in $\Delta\theta_t$. Then the squared parameter updates, s_t, is accumulated in a decaying manner to compute the final update (Zeiler, 2012). Since $\Delta\theta_t$ is unknown for the current time step, its value is estimated by the r_t of parameter updates up to the last time step. Eventually, the update rule requires no default learning rate to set.

3.5. Adam

Adaptive Moment Estimation (Adam) is another gradient-based approach that has been proposed recently (Kingma and Ba, 2014). It not only accumulates the decaying average of the past squared gradients $\mathbf{n}_t$, like RmsProp and Adadelta, but also stores a decaying mean of past gradients $\mathbf{m}_t$. There are two terms which can be considered as the first and second moments. In Algorithm 5, $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{n}}_t$ are the bias corrected terms for instability against zero initialization.

3.6. Combination of Optimizers

Because the learning trajectory significantly affects training process, it is required to control the learning rate. Many research attempts show that simple SGD is able to find a minimum, but it might take long and it relies on the initial learning rate (Goodfellow et al., 2016, Chapter 8). Therefore, at the beginning, a fast convergence to the zone in which local minima located is desired. Then, by reducing the decay rate, the model has better opportunity to find the best critical point within that area. To do so, one can combine different optimization algorithms to take advantage of methods which accelerate the training and afterwards switch to the techniques with more control on the learning rate (Wu et al., 2016). Here, we combine adaptive optimizers with the simple SGD not only to regulate the learning phase, but also to accelerate the whole process. We start the training with any of the five considered optimizers, then run the variants of reducing the learning rate. These variations are:

1. Fixed-SGD: means the training is carried on by the simple SGD algorithm with a constant learning rate. Thus, it is easy to apply and there is no need to have any schedules or thresholds in advance. Here, we use a learning rate of 0.01.
2. Annealing: refers to the scheduling scheme in which the learning rate of the associated optimizer is decreased based on a pre-defined schedule between epochs. We use a schedule in that the learning rate is halved after every sub-epoch.

4. Experiments

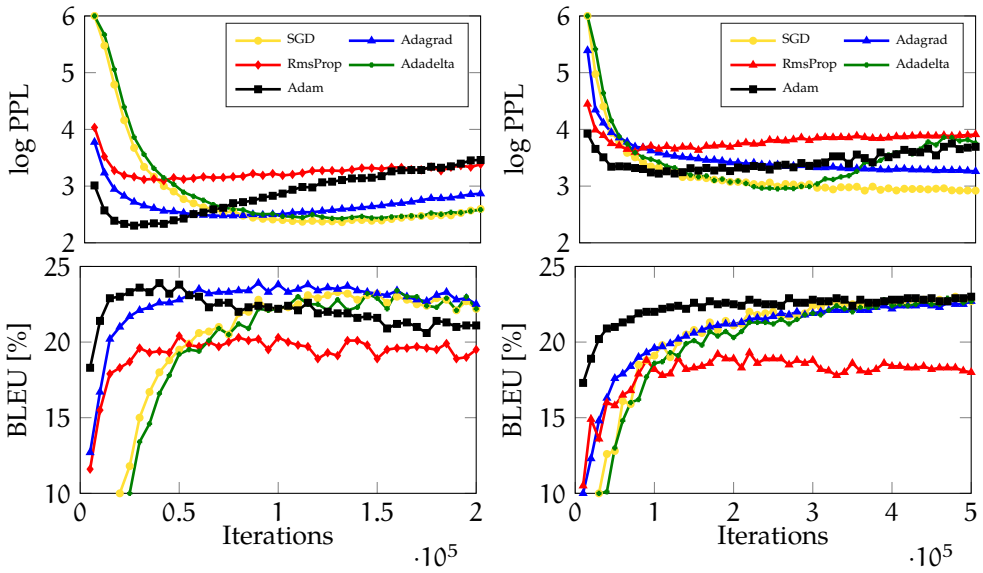
We have carried out the experiments on two translation tasks. The WMT 2016 En→Ro and WMT 2015 De→En. All experiments use the bilingual data, without any monolingual data. All systems follow the architecture by (Bahdanau et al., 2015). We use an implementation based on Blocks (Merriënboer et al., 2015)[1] which is a framework on top of Theano (Bastien et al., 2012). To deal with OOVs, we use the joint-BPE approach (Sennrich et al., 2016) to have a sequence of subwords in both the source and the target sides. In both tasks, the number of joint-BPE operations is 20K. All words are projected into a 620-dimensional embedding space. Both encoder and decoder are equipped with LSTMs with peephole connections with 1000 cells. We shuffle the training samples once before training and use mini-batches of 50 sentence pairs

[1] https://github.com/mila-udem/blocks-examples

and remove sentences longer than 75 subwords. Decoding is performed using beam search with a beam size of 12. The models are trained with different optimization schemes (see § 3) using the same architecture, the same number of parameters and all are identically initialized by the same random seed. The total number of parameters are 73M and 76M for En→Ro and De→En respectively. The systems are evaluated using case-sensitive BLEU and case-sensitive TER (Papineni et al., 2002; Snover et al., 2006) computed by MultEval (Clark et al., 2011).

For WMT 2016 En→Ro, the training data consists of 604K pairs of bilingual sentences with 16.8M English and 17.7M Romanian subwords. Validation is performed on 1000 sentences of the `newsdev2016` corpus. We stop training after 200K iterations and evaluate them every 5K. One iteration is one mini-batch. The `newstest16` corpus consisting of 1999 sentences is used as our test set.

For WMT 2015 De→En translation task, the bilingual training data includes 4.2M sentence pairs. The data set is composed of 133M German and 125M English words. The concatenation of `newstest2011` and `newstest2012` is used as our validation set named (`newsdev11+12`) resulting to 5984 sentences. We evaluate and save the models every 10K and stop training after 500K iterations. `newstest2014` and `newstest2015` are used as the test set including 3003 and 2169 samples respectively. For adaptive-based algorithms, which adapt the learning rate during training, we use the default hyperparameters proposed by the original publications (see Algorithms 2-5).



(a) En→Ro *(b) De→En*

Figure 1: log PPL and BLEU score of all optimizers on validation sets.

5. Analysis

Figure 1 shows the behavior of five optimizers for both En→Ro and De→En tasks in terms of BLEU and perplexity (PPL). As it is shown, Adam and Adagrad are faster at the beginning and converge to a relatively good area in the parameter space in terms of the cost function. At the beginning of the training, Adam outperforms the other methods in terms of both BLEU and perplexity. As it is seen in the Figure 1a, Adam reaches 23.9% in BLEU after only 40K iterations for En→Ro and 22.8% BLEU after 220K iterations on De→En compared to the others (see Fig. 1b). Its aggressive movement diverts from the local minima afterward.

In Adagrad, the denominator accumulates the sum of square of past gradients over training iterations leading to a significantly small learning step which slows down the training phase. Although RmsProp moves fast initially, it converges to a worse point compared to the other optimizers. We leave out RmsProp in the rest of our experiments since it has not shown promising results. Adadelta and simple SGD (with a constant learning rate) have a similar smooth pattern. Both have a moderate behavior for the first iterations and move slowly towards saturation. The same patterns for all of the optimizers in terms of log PPL can be seen in Figure 1. Again Adam converges to a proper point faster than the others.

In our experiments, we continue the training of the best model using different combinations described in Section 3.6. We monitor the perplexity and BLEU score on the validation set during training. We pick the best model among all based on BLEU to continue training the network by one of the explained combinations. In this case, our intuition is that we have already reached an appropriate region in the parameter space and it is a good time to slow down the training. By means of finer search, the optimizer has better chance not to skip a good local minima.

Figures 2 and 3 show the BLEU on the validation sets for En→Ro and De→En translation tasks respectively using these combinations. For example, the network is firstly trained by Adam and followed by Fixed-SGD, Annealing-SGD and Annealing-Adam (Fig. 2d and 3d). As the name suggests in Fixed-SGD, we continue training with the simple SGD and a fixed learning rate is used. While in the annealing-based strategies, the network continues having an annealing schedule. The difference between the two last variants is that in the former (Annealing-SGD), the learning rate of the simple SGD is reduced, whereas in the latter (Annealing-Adam) the learning rate of the adaptive-based optimizer, Adam, is decreased.

One can find the detailed results of the individual configuration on the validation set for each task in Table 1. In this Table, on one hand, the performance of each strategy has been compared with its base optimizer (e.g. line 12 compared to line 15) and on the other hand, the overall analogy among different groups has been shown. For each group, the improvement over the base optimizer has been written in the parenthesis and the best performance is marked by (†). As depicted, for all of the optimizers, applying these combinations improves both BLEU and TER. We also observed the same performance boost in terms of PPL. The smallest boost is associated with the

Adagrad optimizer on De→En. We speculate that the learning rate of Adagrad is already too small and annealing it makes the entire term much smaller leading to the slow training. Therefore, it is not possible to find better optima in a proper time.

Overall analogy states that Adam followed by Annealing-Adam gives the best results on `newsdev11+12` for De→En up to 25.4% in BLEU and 56.7% in TER. Moreover, the boost of this approach is the same as the Adam plus the other configurations on `newsdev16` for En→Ro and obtains 26.2% in BLEU and 55.9% in TER. We believe that the small fluctuations in BLEU and TER scores might be the noise.

Since the performance of the third strategy (results listed in the line 3, 7, 11 and 15) is as good as or better than the rest, we choose it to narrow down the results and verify the results on the test sets. In this case, for each parameter, we have an individual learning rate.

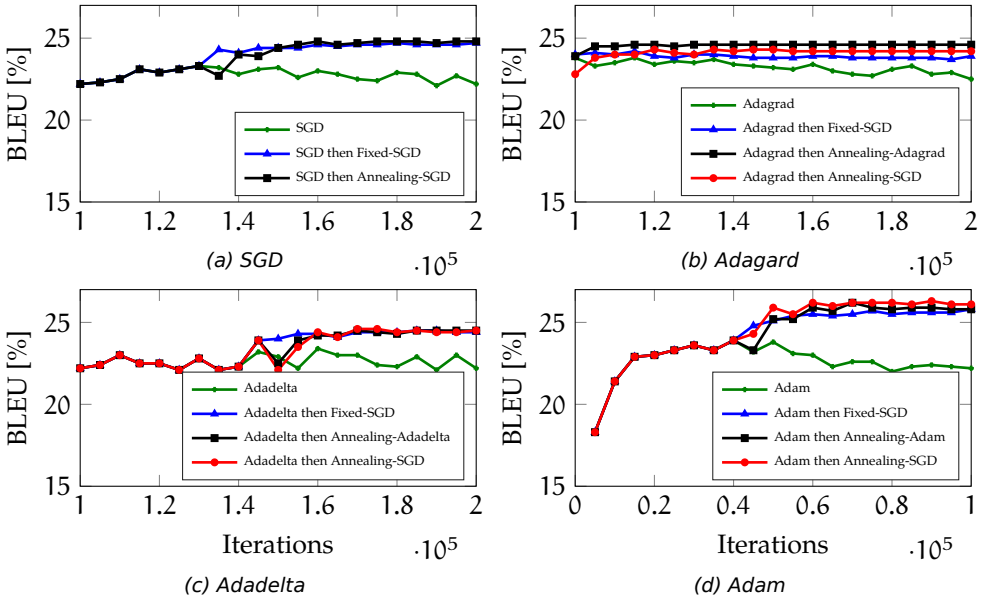
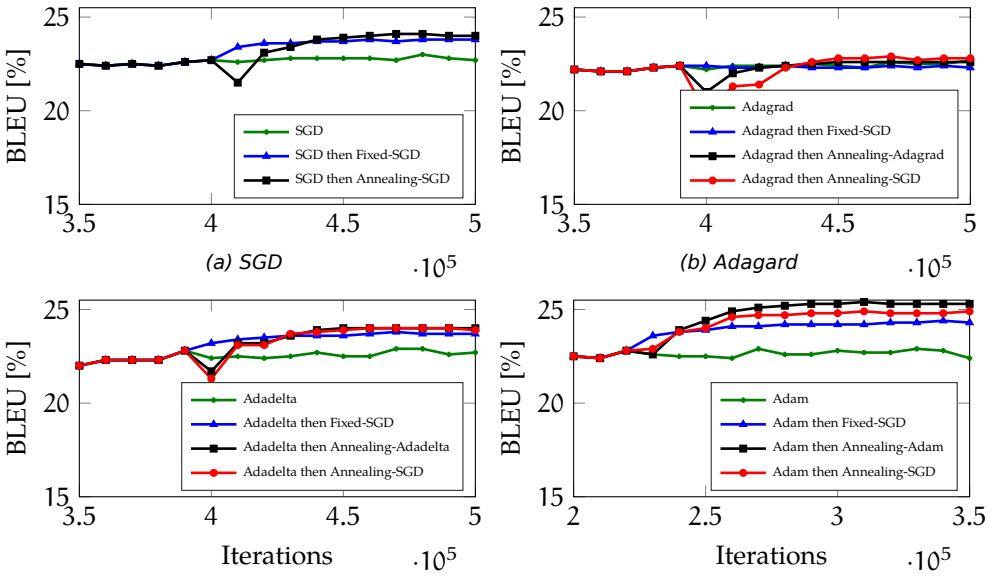


Figure 2: BLEU of optimizers followed by the combinations on the val. set for En→Ro.



(a) SGD *(b) Adagard* *(c) Adadelta* *(d) Adam*

Figure 3: BLEU of optimizers followed by the combinations on the val. set for De→En. The representation of x-axis of Adam is different as it is faster.

		En→Ro		De→En	
	Optimizer	newsdev16		newsdev11+12	
		BLEU	TER	BLEU	TER
1	SGD	23.3	59.5	22.8	59.5
2	+ Fixed-SGD	24.7 (+1.4)	57.0 (-2.5)	23.8 (+1.0)	58.4 (-1.1)
3	+ Annealing-SGD	24.8 (+1.5)	57.0 (-2.5)	24.1 (+1.3)	58.1 (-1.4)
4	Adagrad	23.9	58.1	22.6	60.0
5	+ Fixed-SGD	24.2 (+0.3)	57.7 (-0.4)	22.4 (-0.2)	60.3 (+0.3)
6	+ Annealing-SGD	24.3 (+0.4)	57.4 (-0.7)	22.9 (+0.3)	59.7 (-0.3)
7	+ Annealing-Adagrad	24.6 (+0.7)	57.0 (-1.1)	22.6 (0.0)	59.9 (-0.1)
8	Adadelta	23.2	59.1	22.9	59.8
9	+ Fixed-SGD	24.5 (+1.3)	57.3 (-1.8)	23.8 (+0.9)	58.5 (-1.3)
10	+ Annealing-SGD	24.6 (+1.4)	57.6 (-1.5)	24.0 (+1.1)	58.2 (-1.6)
11	+ Annealing-Adadelta	24.6 (+1.4)	57.5 (-1.6)	24.0 (+1.1)	58.4 (-1.4)
12	Adam	23.9	58.2	23.0	59.4
13	+ Fixed-SGD	26.2 (+2.3)	55.6 (-2.6)	24.5 (+1.5)	57.7 (-1.7)
14	+ Annealing-SGD	26.3 (+2.4)†	55.8 (-2.4)†	24.9 (+1.9)	57.4 (-2.0)
15	+ Annealing-Adam	26.2 (+2.3)	55.9 (-2.3)	25.4 (+2.4)†	56.7 (-2.7)†

Table 1: Results in BLEU[%] and TER[%] on val. sets. † shows the best results.

5.1. Translation Quality

Table 2 lists the results of the test sets for En→Ro and De→En tasks. We also report the results of the Annealing method for each optimizers. In addition to the performance of the best model, the results of the averaged-best model which is the average of the four best training points which is also a single model is shown. Clearly, the Adam followed by Annealing-Adam outperforms the rest of combinations. An interesting point to be highlighted is that the averaged-best leads to improvements as good as the annealing strategy except for one case in the table. On `newstest16` En→Ro[2], the averaged-best of Adam outperforms the Annealing-Adam. If one does not follow any schedule scheme, he can easily run the pure Adam, save the intermediate points and average the weights. We observed the same pattern of improvements for TER on both tasks. As a summary, we reach the conclusion that the model has the opportunity to find the better critical point within that located area, if the learning step is reduced. Applying these variations differing in the handling of the learning rate can be helpful to focus more on an area containing the local minima. We conclude that shrinking the learning steps might lead to a finer search and prevent stumbling over a local minimum. By comparing the performance of different optimization approaches, we showed that Adam followed by Annealing-Adam gains the best performance.

		En→Ro		De→En			
		Best	Averaged-best	Best		Averaged-best	
	Optimizer	`newstest16`		`newstest14`	`newstest15`	`newstest14`	`newstest15`
1	SGD	20.3	22.5	25.2	26.1	26.7	27.4
2	+ Annealing-SGD	22.1	21.9	26.9	27.4	26.9	27.2
3	Adagrad	21.6	21.7	24.8	26.2	24.9	26.0
4	+ Annealing-Adagrad	21.9	21.9	24.6	25.5	24.6	25.5
5	Adadelta	20.5	22.2	25.2	25.6	26.6	27.4
6	+ Annealing-Adadelta	22.0	22.0	26.4	27.6	26.5	27.4
7	Adam	21.4	24.6	25.0	25.7	28.0	28.9
8	+ Annealing-Adam	23.0†	23.1	28.1†	29.0†	28.2	29.0

Table 2: Results measured in BLEU[%] for best and averaged-best models on test sets. † shows the best performance for the Best models.

5.2. Training Stability

As shown in Table 2, using one of these configurations to slow down the learning phase narrows the gap between averaged-best and the best model. For example for En→Ro, averaged-best model gains 1.7% in BLEU using Adadelta (line 5) while applying Annealing-Adadelta has already covered this offset and it does not get more boost from averaging (line 6). Moreover, this gap has been compensated by decreasing the learning steps. This property holds for all of the cases in Table 2. We conclude that pure Adam training is less regularized, therefore the model is allowed to navigate

[2]Note that, the best performing En→Ro NMT system in the WMT 2016 shared task has been used the synthetic data which is not the case in our work.

varying areas in the parameter space. It stumbles on good cases. Thus it is beneficial to average the best cases. Whereas in the Adam+Annealing-Adam the parameter space navigated is more regularized, leading to less varieties.

5.3. Convergence Speed

The momentum in Adam optimizer regulating the directions causes a fast descent towards the minimum. By adding the previous updates into the current update, momentum enforces the updates in a particular direction. The convergence of Adam followed by Annealing-Adam obtained after 70K iterations for En→Ro shown in figure 2d as well as 310K iterations for De→En pictured in figure 3d. This results to 50% faster convergence in the training on average on both tasks. As the number of training samples for En→Ro is seven times smaller than those for De→En, the convergence of this task is faster.

6. Conclusion

We practically analyzed the performance of five common first-order gradient-based optimization methods in NMT which are either run alone or followed by the variations differing in the handling of the learning rate. We benefited from the methods accelerating the training at the beginning and then switched to the techniques with more control on the learning rate to find the better local minimum in parameter space. The quality of the models in terms of BLEU and TER scores as well as the convergence speed and robustness against stochasticity have been investigated on two WMT translation tasks. We concluded that in order to speed up the training and enhance the performance in terms of both BLEU and TER, one could apply Adam followed by Annealing-Adam. Experiments done on WMT 2016 En→Ro and WMT 2015 De→En show that the mentioned technique leads to 1.6% BLEU improvements on `newstest16` for En→Ro, and 3.1% BLEU on `newstest15` for De→En. Moreover, it results to faster convergence of 50% as well as the training stability. We showed that using Annealing-Adam compensates the offset between the best model and the averaged-best. We recommend that, if someone does not utilize the annealing scheme to reduce the learning rate, he should average the best training points to increase the translation performance. Similar to NMT, we hope that the proposed techniques would help other neural network training including non-sequential models.

Acknowledgements

The work reported in this paper results from two projects, SEQCLAS and QT21. SEQCLAS has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement n° 694537. QT21 has received funding from the European Union's

Horizon 2020 research and innovation program under grant agreement nº 645452. The work reflects only the authors' views and neither the European Commission nor the European Research Council Executive Agency is responsible for any use that may be made of the information it contains. Tamer Alkhouli was partly funded by the 2016 Google PhD Fellowship for North America, Europe and the Middle East.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2015.

Bastien, Frédéric, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

Britz, Denny, Anna Goldie, Thang Luong, and Quoc Le. Massive Exploration of Neural Machine Translation Architectures. *arXiv preprint arXiv:1703.03906*, 2017.

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Qatar*, pages 103–111, 2014.

Clark, Jonathan H., Chris Dyer, Alon Lavie, and Noah A. Smith. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *49th Annual Meeting of the Association for Computational Linguistics*, pages 176—181, USA, 2011.

Dozat, Timothy. Incorporating Nesterov momentum into Adam. Technical report, 2015.

Duchi, John C., Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Farajian, M Amin, Rajen Chatterjee, Costanza Conforti, Shahab Jalalvand, Vevake Balaraman, Mattia A Di Gangi, Duygu Ataman, Marco Turchi, Matteo Negri, and Marcello Federico. FBK's Neural Machine Translation Systems for IWSLT 2016. In *Proceedings of the ninth International Workshop on Spoken Language Translation, USA*, 2016.

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

Hinton, Geoffrey, N Srivastava, and Kevin Swersky. Lecture 6a overview of mini–batch gradient descent. *Coursera Lecture slides https://class.coursera.org/neuralnets-2012-001/*, 2012.

Im, Daniel Jiwoong, Michael Tao, and Kristin Branson. An Empirical Analysis of Deep Network Loss Surfaces. *CoRR*, abs/1612.04010, 2016.

Jean, Sébastien, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. Montreal Neural Machine Translation Systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT 2015, Portugal*, pages 134–140, 2015.

Junczys-Dowmunt, Marcin, Tomasz Dwojak, and Rico Sennrich. The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT. In *Proceedings of the First Conference on Machine Translation, WMT 2016, Germany*, pages 319–325, 2016.

Kingma, Diederik P. and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

Merriënboer, Bart, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and Fuel: Frameworks for deep learning. 2015.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, USA, 2002.

Robbins, Herbert and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Ruder, Sebastian. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Germany*, 2016.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, USA, 2006.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Canada*, pages 3104–3112, 2014.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016.

Zeiler, Matthew D. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

Zeyer, Albert, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition. *CoRR*, abs/1606.06871, 2017.

Address for correspondence:
Parnia Bahar
bahar@i6.informatik.rwth-aachen.de
Human Language Technology and Pattern Recognition Group,
RWTH Aachen University,
Ahornstraße 55, 52074 Aachen, Germany