Empirical Investigation of Optimization Algorithms in Neural Machine Translation

Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, Hermann Ney
bahar@i6.informatik.rwth-aachen.de

29th May, 2017
EAMT 2017, Prague, Czech Republic

Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University
Introduction

- Neural Machin Translation (NMT) trains a single, large neural network reading a sentence and generates a variable-length target sequence.
- Training an NMT system involves the estimation of a huge number of parameters in a non-convex scenario.
- Global optimality is given up and local minima in the parameter space are considered sufficient.
- Choosing an appropriate optimization strategy can not only obtain better performance, but also accelerate the training phase of neural networks and brings higher training stability.
Related work

- [Im & Tao\textsuperscript{+ 16}] try to show the performance of optimizers in the investigation of loss surface for image classification task
- [Zeyer & Doetsch\textsuperscript{+ 17}] investigate various optimization methods for acoustic modeling empirically
- [Dozat 15] compares different optimizers in language modeling
- [Britz & Goldie\textsuperscript{+ 17}] study a massive analysis of NMT hyperparameters aiming for better optimization being robust to the hyperparameter variations
- [Wu & Schuster\textsuperscript{+ 16}] utilize the combination of Adam and a simple Stochastic Gradient Descend (SGD) learning algorithm
This Work - Motivation

▶ A study of the most popular optimization techniques used in NMT
▶ Averaging the parameters of a few best snapshots from a single training run leads to improvement [Junczys-Dowmunt & Dwojak+ 16]
▶ An open question concerning training problem
▶ Either the model or the estimation of its parameters is weak
This work

- Empirically investigate the behavior of the most prominent optimization methods to train an NMT
- Investigate the combinations that seek to improve optimization
- Addressing three main concerns:
  - translation performance
  - convergence speed
  - training stability
- First, how well, fast and stable different optimization algorithms work
- Second, how a combination of them can improve these aspects of training
Given a source \( f = f_1^J \) and a target \( e = e_1^I \) sequence, NMT [Sutskever & Vinyals\textsuperscript{+} 14, Bahdanau & Cho\textsuperscript{+} 15] 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 \( \{ \langle f^{(s)}, e^{(s)} \rangle \}_{s=1}^S \)

\[
J(\theta) = \sum_{s=1}^{S} \sum_{i=1}^{I^{(s)}} \log p(e_i^{(s)} | e_{<i}^{(s)}, f^{(s)}; \theta)
\]
Stochastic Gradient Descent (SGD)  
[Robbins & Monro 51]

- SGD updates a set of parameters, $\theta$
- $g_t$ represents the gradient of the cost function $J$
- $\eta$ is called the learning rate, determining how large the update is
- Tunning of the learning carefully

Algorithm 1 : Stochastic Gradient Descent (SGD)

1: $g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
2: $\theta_{t+1} \leftarrow \theta_t - \eta g_t$
Adagrad
[Duchi & Hazan+ 11]

- The shared global learning rate $\eta$ is divided by the $l_2$-norm of all previous gradients, $n_t$
- Different learning rates for every parameter
- Larger updates for the dimensions with infrequent changes and smaller updates for those that have already large changes
- $n_t$ in the denominator is a positive growing value which might aggressively shrink the learning rate

Algorithm 2: Adagrad

1: $g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
2: $n_t \leftarrow n_{t-1} + g_t^2$
3: $\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{n_t} + \epsilon} g_t$
RmsProp
[Hinton & Srivastava+ 12]

▶ Instead of storing all the past squared gradients from the beginning of the training, a decaying weight of squared gradients is applied

Algorithm 3 : RmsProp

1: $g_t \leftarrow \nabla_{\theta_t} J(\theta_t)$
2: $n_t \leftarrow \nu n_{t-1} + (1 - \nu) g_t^2$
3: $\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{n_t + \epsilon}} g_t$
Adadelta
[Zeiler 12]

- Takes the decaying mean of the past squared gradients
- The squared parameter updates, $s_t$, is accumulated in a decaying manner to compute the final update
- 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

Algorithm 4: Adadelta

1: $g_t \leftarrow \nabla_{\theta_t} J (\theta_t)$
2: $n_t \leftarrow \nu n_{t-1} + (1 - \nu) g_t^2$
3: $r(n_t) \leftarrow \sqrt{n_t + \epsilon}$
4: $\Delta \theta_t \leftarrow \frac{-\eta}{r(n_t)} g_t$
5: $s_t \leftarrow \nu s_{t-1} + (1 - \nu) \Delta \theta_t^2$
6: $r(s_{t-1}) \leftarrow \sqrt{s_{t-1} + \epsilon}$
7: $\theta_{t+1} \leftarrow \theta_t - \frac{r(s_{t-1})}{r(n_t)} g_t$
Adam
[Kingma & Ba 15]

- The decaying average of the past squared gradients $n_t$
- Stores a decaying mean of past gradients $m_t$
- First and second moments

Algorithm 5 : Adam

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

- Two translation tasks, the WMT 2016 En→Ro and WMT 2015 De→En
- NMT model follows the architecture by [Bahdanau & Cho 2015]
- joint-BPE approach [Sennrich & Haddow 2016]
- Evaluate and save the models on validation sets every 5k iterations for En→Ro and every 10K iterations for De→En
- The models are trained with different optimization methods
  - the same architecture
  - the same number of parameters
  - identically initialized by the same random seed
Figure: log PPL and BLEU score of all optimizers on the val. sets.
Combination of Optimizers

- A fast convergence at the beginning, then reducing the learning rate.
- Take advantage of methods which accelerate the training and afterwards switch to the techniques with more control on the learning rate.
- Starting the training with any of the five considered optimizers, pick the best model, then continue training the network.

1. Fixed-SGD: simple SGD algorithm with a constant learning rate. Here, we use a learning rate of 0.01.
2. Annealing: annealing schedule in that the learning rate of optimizer is halved after every sub-epoch.

- Reaching 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 good local minima.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Optimizer</th>
<th>En→Ro</th>
<th>De→En</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>newsdev16 BLEU</td>
<td>newsdev11+12 BLEU</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SGD</td>
<td>23.3</td>
<td>22.8</td>
</tr>
<tr>
<td>2</td>
<td>+ Fixed-SGD</td>
<td>24.7 (+1.4)</td>
<td>23.8 (+1.0)</td>
</tr>
<tr>
<td>3</td>
<td>+ Annealing-SGD</td>
<td>24.8 (+1.5)</td>
<td>24.1 (+1.3)</td>
</tr>
<tr>
<td>4</td>
<td>Adagrad</td>
<td>23.9</td>
<td>22.6</td>
</tr>
<tr>
<td>5</td>
<td>+ Fixed-SGD</td>
<td>24.2 (+0.3)</td>
<td>22.4 (-0.2)</td>
</tr>
<tr>
<td>6</td>
<td>+ Annealing-SGD</td>
<td>24.3 (+0.4)</td>
<td>22.9 (+0.3)</td>
</tr>
<tr>
<td>7</td>
<td>+ Annealing-Adagrad</td>
<td>24.6 (+0.7)</td>
<td>22.6 (0.0)</td>
</tr>
<tr>
<td>8</td>
<td>Adadelta</td>
<td>23.2</td>
<td>22.9</td>
</tr>
<tr>
<td>9</td>
<td>+ Fixed-SGD</td>
<td>24.5 (+1.3)</td>
<td>23.8 (+0.9)</td>
</tr>
<tr>
<td>10</td>
<td>+ Annealing-SGD</td>
<td>24.6 (+1.4)</td>
<td>24.0 (+1.1)</td>
</tr>
<tr>
<td>11</td>
<td>+ Annealing-Adadelta</td>
<td>24.6 (+1.4)</td>
<td>24.0 (+1.1)</td>
</tr>
<tr>
<td>12</td>
<td>Adam</td>
<td>23.9</td>
<td>23.0</td>
</tr>
<tr>
<td>13</td>
<td>+ Fixed-SGD</td>
<td><strong>26.2 (+2.3)</strong></td>
<td>24.5 (+1.5)</td>
</tr>
<tr>
<td>14</td>
<td>+ Annealing-SGD</td>
<td><strong>26.3 (+2.4)</strong></td>
<td>24.9 (+1.9)</td>
</tr>
<tr>
<td>15</td>
<td>+ Annealing-Adam</td>
<td><strong>26.2 (+2.3)</strong></td>
<td><strong>25.4 (+2.4)</strong></td>
</tr>
</tbody>
</table>

**Table: Results in BLEU[%] on val. sets.**
# Results - Performance

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>En→Ro</th>
<th>De→En</th>
</tr>
</thead>
<tbody>
<tr>
<td>newstest16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td>20.3</td>
<td>26.1</td>
</tr>
<tr>
<td>+ Annealing-SGD</td>
<td>22.1</td>
<td>27.4</td>
</tr>
<tr>
<td>Adagrad</td>
<td>21.6</td>
<td>26.2</td>
</tr>
<tr>
<td>+ Annealing-Adagrad</td>
<td>21.9</td>
<td>25.5</td>
</tr>
<tr>
<td>Adadelta</td>
<td>20.5</td>
<td>25.6</td>
</tr>
<tr>
<td>+ Annealing-Adadelta</td>
<td>22.0</td>
<td>27.6</td>
</tr>
<tr>
<td>Adam</td>
<td>21.4</td>
<td>25.7</td>
</tr>
<tr>
<td>+ Annealing-Adam</td>
<td>23.0</td>
<td>29.0</td>
</tr>
</tbody>
</table>

Table: Results measured in BLEU[%] on the test sets.

- Shrinking the learning steps might lead to a finer search and prevent stumbling over a local minimum
- Adam followed by Annealing-Adam gains the best performance
Results - Convergence Speed

Figure: BLEU score of the best combination on the val. sets.

▶ Faster convergence in the training by Adam followed Annealing-Adam

(a) En→Ro  (b) De→En
## Results - Training Stability

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Best Model</th>
<th>Averaged-best</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>26.1</td>
<td>27.4</td>
</tr>
<tr>
<td>+ Annealing-SGD</td>
<td>27.4</td>
<td>27.2</td>
</tr>
<tr>
<td>Adagrad</td>
<td>26.2</td>
<td>26.0</td>
</tr>
<tr>
<td>+ Annealing-Adagrad</td>
<td>25.5</td>
<td>25.5</td>
</tr>
<tr>
<td>Adadelta</td>
<td>25.6</td>
<td>27.4</td>
</tr>
<tr>
<td>+ Annealing-Adadelta</td>
<td>27.6</td>
<td>27.4</td>
</tr>
<tr>
<td>Adam</td>
<td>25.7</td>
<td>28.9</td>
</tr>
<tr>
<td>+ Annealing-Adam</td>
<td>29.0</td>
<td>29.0</td>
</tr>
</tbody>
</table>

**Table**: Results measured in BLEU[%] for best and averaged-best models on the test sets.

- Pure Adam training is less regularized and stumbles on good cases
- Adam+Annealing-Adam is more regularized, leading to less varieties
Practically analyzed the performance of common gradient-based optimization methods in NMT
Ran alone or followed by the variations differing in the handling of the learning rate
The quality of the models in terms of BLEU scores as well as the convergence speed and robustness against stochasticity have been investigated on two WMT translation tasks
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 3.3% BLEU on newstest15 for De→En
It results to faster convergence as well as the training stability
Thank you for your attention

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

<surname>@i6.informatik.rwth-aachen.de
Figure: BLEU of optimizers followed by the combinations on the val. set for En→Ro.
Analysis - Combination

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

D. Bahdanau, K. Cho, Y. Bengio.
Neural machine translation by jointly learning to align and translate.

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow,
A. Bergeron, N. Bouchard, Y. Bengio.
Theano: new features and speed improvements.

D. Britz, A. Goldie, T. Luong, Q. Le.
Massive exploration of neural machine translation architectures.
K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio.  

Better hypothesis testing for statistical machine translation: Controlling for optimizer instability.  

T. Dozat.  
Incorporating nesterov momentum into adam.  
J. C. Duchi, E. Hazan, Y. Singer. 
Adaptive subgradient methods for online learning and stochastic optimization. 

Fbk’s neural machine translation systems for iwslt 2016. 
In *Proceedings of the ninth International Workshop on Spoken Language Translation, USA*, 2016.

I. Goodfellow, Y. Bengio, A. Courville. 
*Deep Learning*. 

G. Hinton, N. Srivastava, K. Swersky. 
Lecture 6a overview of mini–batch gradient descent. 
D. J. Im, M. Tao, K. Branson.  
An empirical analysis of deep network loss surfaces.  

Montreal neural machine translation systems for wmt’15.  

M. Junczys-Dowmunt, T. Dwojak, R. Sennrich.  
The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT.  

D. P. Kingma, J. Ba.  
Adam: A method for stochastic optimization.  


S. Ruder.
An overview of gradient descent optimization algorithms.

R. Sennrich, B. Haddow, A. Birch.
Neural machine translation of rare words with subword units.

A Study of Translation Edit Rate with Targeted Human Annotation.

I. Sutskever, O. Vinyals, Q. V. Le.
Sequence to sequence learning with neural networks.
Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, Klaus et al.

M. D. Zeiler.
ADADELTA: an adaptive learning rate method.

A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, H. Ney.
A comprehensive study of deep bidirectional LSTM rnns for acoustic modeling in speech recognition.