Natural Language Engineering (2023), **29**, pp. 316–336
doi:10.1017/S135132492200002X

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

An empirical study of cyclical learning rate on neural machine translation

Weixuan Wang, Choon Meng Lee, Jianfeng Liu, Talha Colakoglu and Wei Peng*

Artificial Intelligence Application Research Center, Huawei Technologies, Co., Ltd., Shenzhen, People's Republic of China
*Corresponding author. E-mail: peng.wei1@huawei.com

(Received 17 September 2020; revised 8 November 2021; accepted 8 November 2021; first published online 9 February 2022)

Abstract

In training deep learning networks, the optimizer and related learning rate are often used without much thought or with minimal tuning, even though it is crucial in ensuring a fast convergence to a good quality minimum of the loss function that can also generalize well on the test dataset. Drawing inspiration from the successful application of cyclical learning rate policy to computer vision tasks, we explore how cyclical learning rate can be applied to train transformer-based neural networks for neural machine translation. From our carefully designed experiments, we show that the choice of optimizers and the associated cyclical learning rate policy can have a significant impact on the performance. In addition, we establish guidelines when applying cyclical learning rates to neural machine translation tasks.

Keywords: Neural machine translation; Cyclical learning rate; Optimizer; Adam; Batch size

1. Introduction

There have been many interests in deep learning optimizer research recently (Kingma and Ba 2014; Reddi, Kale, and Kumar 2018; Luo *et al*. 2019; Zhang *et al*. 2019; Liu *et al*. 2019). These works attempt to answer the question: what is the best step size to use in each step of the gradient descent? With the first-order gradient descent being the *de facto* standard in deep learning optimization, the question of the optimal step size or learning rate in each step of the gradient descent arises naturally. The difficulty in choosing a good learning rate can be better understood by considering the two extremes: (1) when the learning rate is too small, training takes a long time; (2) while overly large learning rate may cause training to diverge instead of converging to a satisfactory solution.

The two main classes of optimizers commonly used in deep learning are the momentum based Stochastic Gradient Descent (SGD) (Bottou 2010) and adaptive momentum-based methods (Duchi, Hazan, and Singer 2010; Kingma and Ba 2014; Reddi *et al*. 2018; Luo *et al*. 2019; Liu *et al*. 2019). The difference between the two lies in how the newly computed gradient is updated. In SGD with momentum, the new gradient is updated as a convex combination of the current gradient and the exponentially averaged previous gradients. For the adaptive case, the current gradient is further weighted by a term involving the sum of squares of the previous gradients. For a more detailed description and convergence analysis, please refer to Reddi *et al*. (2018).

SGD maintains a fixed learning rate for each step gradient. The frequent updates in the training process produce high variance parameters, causing the objective function to fluctuate with different intensity. This fluctuation may enable SGD to find a new and potentially better local minimum in non-convex optimization problems. However, SGD suffers in loss surfaces with sharp minima

due to frequent overshooting. The SGD method can diverge or converge incredibly slowly if its learning rate is set inappropriately (Ruder 2016).

With adaptive learning rate, Adam is well-suited for optimization problems with noisy gradients. Adam, with a decaying schedule, tends to avoid overshooting problems due to loss function fluctuation. The experiments conducted on the MNIST and CIFAR10 dataset showed that Adam has the fastest convergence property, compared to other optimizers, in particular SGD with Nesterov momentum (Kingma and Ba 2014). Adam has been popular with the deep learning community due to the speed of convergence. However, Adabound (Luo *et al*. 2019), a proposed improvement to Adam by clipping the gradient range, showed in the experiments that given enough training epochs, SGD can converge to a better quality solution than Adam. To quote from the future work of Adabound, "why SGD usually performs well across diverse applications of machine learning remains uncertain." The choice of optimizers is by no means straight forward or cut and dry.

Another critical aspect of training a deep learning model is the batch size. Once again, while the batch size was previously regarded as a hyperparameter, recent studies such as Keskar *et al*. (2016) have shed light on the role of batch size when it comes to generalization, that is, how the trained model performs on the test dataset. Research works (Keskar *et al*. 2016; Hochreiter and Schmidhuber 1997a) explored the idea of sharp versus flat minima when it comes to generalization. From experimental results on convolutional networks, for example, AlexNet (Krizhevsky, Sutskever, and Hinton 2017), VggNet (Simonyan and Zisserman 2014), Keskar *et al*. (2016) demonstrated that overly large batch size tends to lead to sharp minima while sufficiently small batch size brings about flat minima. Dinh *et al*. (2017), however, argues that sharp minima can also generalize well in deep networks, provided that the notion of sharpness is taken in context. The computer vision (CV) community tends to advocate training schedules with large batch sizes (Smith *et al*. 2018). There is a lack of empirical study investigating interaction effects between learning rate scheduling and batch size. Increasing the learning rate and scaling the batch size Smith *et al*. (2018) achieves good performance with fewer parameter updates, but it has not been proven effective when applied to neural machine translation (NMT).

While the aforementioned works have helped to contribute to our understanding of the nature of the various optimizers, they are mainly focused on computer vision (CV)-related deep learning networks and datasets. In contrast, the rich body of works in NMT and other natural language processing (NLP)-related tasks have been largely left untouched. Recall that CV deep learning networks and NMT deep learning networks are very different. For instance, the convolutional network that forms the basis of many successful CV deep learning networks is translation invariant, for example, in a face recognition network, the convolutional filters produce the same response even when the same face is shifted or translated. On the other hand, recurrent neural networks (RNN) (Hochreiter and Schmidhuber 1997b; Chung *et al*. 2014) and transformer-based deep learning networks (Vaswani *et al*. 2017; Devlin *et al*. 2019) for NMT are specifically looking patterns in sequences. It is often assumed that using the mainstream optimizer (Adam) with the default settings is good enough. However, our empirical study demonstrated that there is significant room for improvement. While there is some awareness in the NMT and NLP community of the cyclical learning rate (CLR), there is still a lack of understanding of the effect of applying CLR to the NMT task. In addition, how the combined effect of learning rate policies and batch size affect the NMT loss is also largely unexplored.

1.1 The contributions

The contributions of this study are to:

- Explore the use of cyclical learning rates for NMT in our empirical study and provide detailed practical insights into the application of CLR.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

- Disclose the interaction of learning rate and batch size on the loss in NMT. From our experiments, we show how the use of CLR enables NMT to work with smaller batch sizes while achieving comparable validation loss compared to large batch sizes. This empirical finding will have implications for NMT training in resource-constrained scenarios.
- Establish an explanation via visualizing the loss error surface, optimizer trajectory for the NMT training process. At the same time, the rationale behind the performance leap when applying CLR to NMT is unveiled.[a]

2. Related works

Our work is mainly related to the CLR study (Smith 2017), which addresses the learning rate issue by having repeated cycles of linearly increasing and decreasing learning rates, constituting the triangle policy for each cycle. CLR draws its inspiration from curriculum learning (Bengio *et al*. 2009) and simulated annealing (Aarts and Korst 2003). Bengio *et al*. (2009) clarify when and why a curriculum or "starting small" strategy can benefit machine learning algorithms. They contribute to this question by showing several cases involving vision and language tasks in which very simple multi-stage curriculum strategies give rise to improved generalization and faster convergence. Smith (2017) demonstrated the effectiveness of CLR on standard computer vision (CV) datasets CIFAR-10 and CIFAR-100, using well-established CV architecture such as ResNet (He *et al*. 2015) and DenseNet (Huang, Liu, and Weinberger 2016).

One interesting aspect of CLR is the need to balance regularizations such as weight decay, dropout, and batch size, as pointed out in Smith and Topin (2017). The experiments verified that various regularizations need to be toned down when using CLR to achieve good results. In particular, the generalization results using the small batch size from the above-mentioned studies no longer hold for CLR. This is interesting because the use of CLR allows training to be accelerated by using a larger batch size without the sharp minima generalization concern. A related work (McCandlish *et al*. 2018) sets a theoretical upper limit on the speed up in training time with increasing batch size. There is no uplift in training time beyond this theoretical upper limit, even with increased batch size.

Li *et al*. (2018) proposes various visualization methods for understanding the loss landscape defined by the loss functions and how the various deep learning architectures affect the landscape. The proposed visualization techniques allow a depiction of the optimization trajectory, which is particularly helpful in understanding the behavior of the various optimizers and how they eventually reach their local minima.

NMT is based on an end-to-end framework without the need to handle SMT (statistical machine translation)-specific problems like word alignments, translation rules, and complicated decoding methods. Bahdanau, Cho, and Bengio (2015) proposed a sequence-to-sequence NMT model based on RNN, leading to significant improvements in translation quality. The words of the input sentences from a source language are first encoded into vectors of intermediate representation and passed to the decoder. The context vector is derived by applying an attention mechanism that measures the degree of alignment between the source sentences and target sentences. And the decoder converts the context vector to the target translated word based on the previously translated words. However, RNN suffers from seriality resulting in long training time for the NMT model. Vaswani *et al*. (2017) proposed a more efficient transformer architecture. Both encoders and decoders of a transformer contain self-attention and pointwise fully connected layers without relying on an RNN architecture. Decoder layers have another sublayer that computes attention values over the encoder's output, enabling learning alignment patterns between the source and the target sentences.

[a]The codes are available at https://github.com/Vicky-Wil/NLE_CLR_NMT.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

In order to get better performance for NMT, most of the existing works mainly focus on modifying the framework of the NMT model and designing better attention mechanisms. There are few research works on the optimization methods of model parameters. Wiseman and Rush (2016) propose a model using a beam search training scheme to get sequence-level scores. This structured approach avoids classical biases associated with local training and unifies the training loss with the test-time usage while preserving the proven model architecture of seq2seq (sequence-to-sequence) and its efficient training approach. Hoang, Haffari, and Cohn (2017) convert the decoding from a discrete optimization problem to a continuous optimization problem. The significant difference between our work and these studies lies in that we focus on improving the NMT training process from another perspective. As far as we know, CLR has not been applied to NMT. The methodology, best practices, and experiments are mainly based on results from CV architecture and datasets. It is by no means apparent or straightforward that the same approach can be directly carried over to NMT. Therefore, we explore the use of cyclical learning rates policy for NMT.

3. Neural machine translation

The NMT models are implemented using an encoder-decoder framework with attention mechanisms to learn a mapping between two sequences of symbolic representations. Given a sentence pair (X, Y), X is the source sentence and Y is the target sentence, where $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$ are the words in the sentence pair. An NMT model contains two parts, an encoder and d decoder, to decode a target sequence given the encoded source sequence probabilistically. An encoder transforms the source sentence X into the context vectors. And a decoder generates target translation Y from the context vectors by maximizing the probability of $p(y_j|y_{<j})$. During the training process, an NMT models the translation probability as:

$$p(y|x) = \prod_{j=1}^{n} p\left(y_j|y_{<j}, x, \theta\right) \tag{1}$$

where θ is the model parameters and $y_{<j} = (y_1, y_2, ..., y_{j-1})$ is the partial translation. The generation conditional probability of y_j is produced by the decoder, representing the final output of the decoder.

Given the sentence-aligned bilingual training data $D = \{< X^{(k)}, Y^{(k)} >\}_{k=1}^{K}$, the cost function can be defined as the following conditional log-likelihood:

$$L(\theta, D) = \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{n} log\left(p\left(y_j^{(k)}|y_{<j}^{(k)}, X^{(k)}, \theta\right)\right) \tag{2}$$

in which the corresponding gradient $\nabla L(\theta, D)$ can be calculated as the sum of the gradients of the sentences in minibatch B:

$$\nabla L(\theta, D) = \sum_{k=1}^{B} \nabla L\left(\theta, \left(X^{(k)}, Y^{(k)}\right)\right) \tag{3}$$

The gradients of each sentence in a parallel corpus can be calculated as a sum of gradients per step:

$$\nabla L\left(\theta, \left(X^{(k)}, Y^{(k)}\right)\right) = \sum_{j=1}^{n} \nabla log\left(p\left(y_j^{(k)}|y_{<j}^{(k)}, X^{(k)}, \theta\right)\right) \tag{4}$$

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

We can then use maximum likelihood estimation with SGD or Adam optimization strategy and backpropagation to get the optimal parameters. For SGD, the calculation with learning rate η is:

$$\theta \leftarrow \theta - \eta \times \nabla L(\theta, D) \tag{5}$$

and the final parameter optimization method can be defined as follows:

$$\theta \leftarrow \theta - \eta \times \sum_{k=1}^{B} \sum_{j=1}^{n} \nabla log \left(p \left(y_j^{(k)} | y_{<j}^{(k)}, X^{(k)}, \theta \right) \right. \tag{6}$$

For Adam, the calculation with learning rate η is as follows, β_1 is set to 0.9 representing the weighted average, β_2 is set to 0.99 that is the RMSprop term:

$$g = \frac{1}{|B|} \sum_{k \in B} \nabla L \left(\theta, \left(X^{(k)}, Y^{(k)} \right) \right) \tag{7}$$

$$m \leftarrow \beta_1 m + (1 - \beta_1) g \tag{8}$$

$$v \leftarrow \beta_2 V + (1 - \beta_2) g^2 \tag{9}$$

$$\widehat{m} \leftarrow \frac{m}{1 - \beta_1} \tag{10}$$

$$\widehat{v} \leftarrow \frac{v}{1 - \beta_2} \tag{11}$$

$$\theta \leftarrow \theta - \eta \times \frac{\widehat{m}}{\sqrt{\widehat{v}} + \epsilon} \tag{12}$$

4. The proposed approach

Our main approach in the NMT-based learning rate policy is based on CLR's triangular learning rate policy. Figure 1 depicts the learning rate decay policy, which is the way the learning rate changes over training epochs. For various optimizers, the learning rate is usually decayed to a small value to ensure convergence. The commonly used decay schemes include piece-wise constant step function, inverse (reciprocal) square root. This study adopts two learning rate decay policies:

- Fixed decay (shrinking) policy where the max learning rate is halved after each learning rate cycle, and
- No decay. This is unusual because for both SGD and adaptive momentum optimizers, a decay policy is required to ensure convergence.

Our adopted learning rate decay policy is interesting because experiments in Smith (2017) showed that using a decay rate is detrimental to the resultant accuracy.

The CLR decay policy should be contrasted with the standard inverse square root policy (INV) that is commonly used in deep learning platforms, for example, in fairseq (Ott *et al.* 2019). The INV typically starts with a warm-up phase where the learning rate is linearly increased to a maximum value. The learning rate is decayed as the reciprocal of the square root of the number of epochs from the above-mentioned maximum value.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

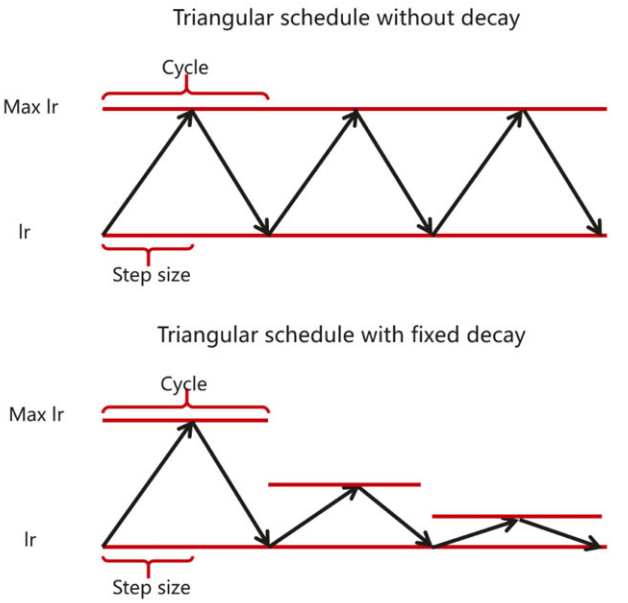


Figure 1. The learning rate decay used in our experiments.

Given the base learning rate and the max learning rate, the learning rate update is as follows:

$$c = 1 + \frac{i}{2 \times s} \tag{13}$$

$$x = \left| \frac{i}{s} - 2 \times c + 1 \right| \tag{14}$$

$$\eta = \eta_{base} + (\eta_{\max} - \eta_{base}) \times \max(0, 1 - x) \tag{15}$$

where i is the current number of iterations of training, s is the stepsize that denotes half the period or cycle length, c denotes the current cycle, $\eta_{\max}$ and η_{base} represent the max learning rate η and the base learning rate, respectively. This policy varies the learning rate linearly between the minimum η_{base} and the maximum $\eta_{\max}$. The no decay policy calculation is presented above. For the fixed decay policy, the max learning rate is no longer a constant. $\eta_{\max}$ is halved with the decay rate d each learning rate cycle:

$$\eta_{\max} = \eta_{\max} \times d \tag{16}$$

The learning rate is subsequently applied in the gradient calculation to update the parameters of NMT. Compared with the traditional NMT backpropagation process, CLR varies the learning rate within a range of values rather than adopting a stepwise fixed or exponentially decreasing value. A short run of only a few epochs is sufficient to estimate the CLR policies' boundary learning rates in a range test (Section 4.2). Then, a policy where the learning rate cyclically varies between these bounds is adequate to obtain near-optimal results, often with fewer iterations. Unlike adaptive learning rate methods, this policy is easy to implement and incurs essentially no additional computational expense.

4.1 CLR-based NMT

Algorithm 1 summarizes the training of the NMT model using CLR. First, we train the NMT model with the base learning rate η_{base} on the range test set for some epoch n_1, deriving the

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Algorithm 1 CLR-based NMT

Input:
a NMT model $p_{\theta_0}(y|x)$, a range test dataset T, training dataset D, the base learning rate η_{base}, decay rate d, cycle length C, range test epoch number n_1
Output: a better performing NMT model $p_{\theta_k}(y|x)$
for $t = 0, 1, ..., n_1$ **do**
  Sample examples from T
  Train NMT model with the base learning rate η_{base}
end for
Get the maximum learning rate η_{max}
for $i = 0, 1, ...,$ **do**
  Sample examples from D
  Train NMT model with SGD or Adam, using the η_{base}, η_{max} based CLR policy
  if $t = m \times C, m = 1, 2, ...$ **then**
    $\eta_{max} = \eta_{max} \times d$
  end if
  Update $p_{\theta_t}(y|x)$ to produce $p_{\theta_{t+1}}(y|x)$
end for

maximum learning rate $\eta_{\max}$. The parameters of NMT model are subsequently updated using the $\eta_{\max}$ and η_{base} based CLR. If the current iteration i is an integer m that is a multiple of the cycle length C which is the twice of step size, $\eta_{\max}$ will be by multiplying with the decay rate d.

4.2 Learning rate range test

For CLR, some pertinent parameters need to be determined: the base learning rate (η_{base}), the maximum learning rate ($\eta_{\max}$), and cycle length C. As suggested in CLR, we perform the range test to set the base/maximum learning rate while the cycle number is some multiples of the number of epochs. The range test is designed to select the base/maximum learning rate in CLR. Without the range test, the base/maximum learning rate in CLR will need to be tuned as hyperparameters which is difficult and time consuming. The purpose of the learning rate range test is to search for the largest learning rate without divergence as a large learning rate leads to faster model convergence than a small learning rate.

In a range test, the network is trained for several epochs with the learning rate linearly increased from an initial rate. For instance, the range test for the IWSLT2014 (DE2EN) dataset was run for 35 epochs, with the initial learning rate set to some small values, for example, $1e^{-5}$ for Adam and increased linearly over the 35 epochs. Given the range test curve, for example, Figure 2, the base learning rate (η_{base}) is set to the point where the loss starts to decrease while the maximum learning rate ($\eta_{\max}$) is selected as the point where the loss starts to plateau or to increase. We observe the qualitative different range test curves for CV and NMT datasets, as we can see from Figures 2 and 3. The CV range test curve looks more well-defined in terms of choosing the maximum learning rate from the point where the curve starts to be ragged. For NMT, the range curve exhibits a smoother, more plateau characteristic. As shown in Figure 2, the base learning rate is selected as the initial learning rate for the range test, and the initial value should be a small value to ensure model convergence otherwise searching for a maximum learning rate to speed up training convergence is meaningless. The maximum learning rate is the point where the loss stagnates. For the step size, following the guideline given in Smith (2017) to select the step size between 2 and

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

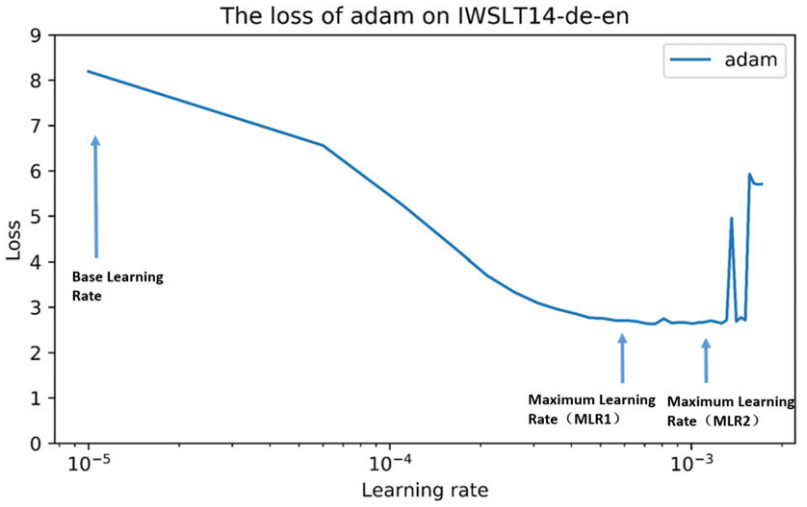


Figure 2. Range test curve for the IWSLT2014-de-en dataset, showing the chosen base and maximum learning rate for the triangular policy.

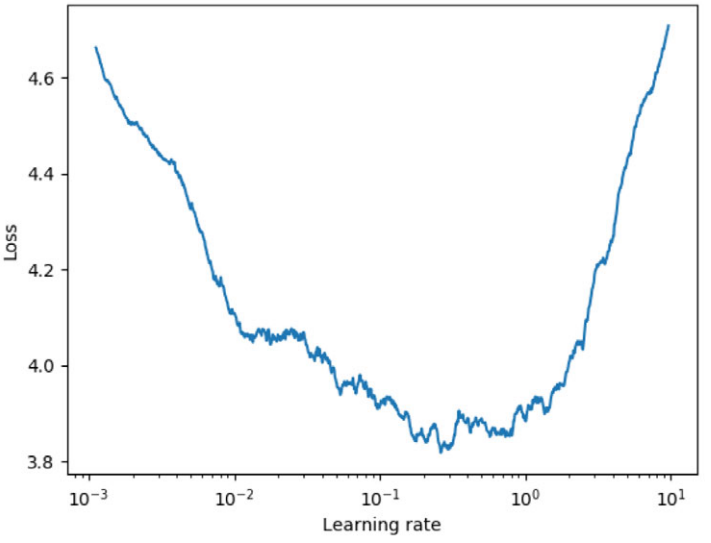


Figure 3. Range test curve for the CV CIFAR-100 dataset.

10 times the number of iterations in an epoch and set the step size to 4.5 epochs. And we also perform the range test of SGD, as shown in Figure A1 in Appendix A.

An intuitive understanding of why CLR methods work comes from considering the loss function topology in NMT model. The difficulty in minimizing the loss arises from saddle points rather than poor local minima. Saddle points have small gradients that slow the learning process. However, increasing the learning rate allows more rapid traversal of saddle point plateaus. A more practical reason as to why CLR works is that, by following the methods in Section 5, it is likely the optimal learning rate will be between the bounds and near-optimal learning rates will be used throughout training.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Table 1. Datasets used for the experiment

Corpus	Training	Valid.	Test	Source Vocab.	Target Vocab.
IWSLT2014-de-en (DE2EN)	160,239	7283	6750	8844	6628
IWSLT2014-fr-en (FR2EN)	166,045	4818	4800	8508	7308
IWSLT2017-de-en (DE2EN)	192,347	4829	4822	13,156	10,108

4.3 Batch size

The other point of interest is how to deal with batch size when using CLR in NMT. While it is well known in the NMT community that larger batch size improves performance, how CLR impacts the batch size performance relationship is one of the central themes in our study. Following the lead in Smith and Topin (2017), we look at how the NMT tasks perform when varying the batch size on top of the CLR policy. Compared to Smith and Topin (2017), we stretch the batch size range, going from batch size as small as 256 to as high as 8192. Only through examining the extreme behaviors can we better understand the effect of batch size superimposed on CLR.

5. Experiments

The purpose of this section is to explore the effects of applying CLR and various batch sizes to train NMT models in a series of experiments.

5.1 Experiment settings

The experiments are performed on two translation directions (DE → EN and FR → EN) for IWSLT2014 and IWSLT2017 (Cettolo, Girardi, and Federico 2012).

The data are pre-processed using functions from Moses (Koehn *et al.* 2007). The punctuation is normalized into a standard format. After tokenization, byte pair encoding (BPE) (Sennrich, Haddow, and Birch 2016) is applied to the data to mitigate the adverse effects of out-of-vocabulary (OOV) rare words. The sentences with a source-target sentence length ratio greater than 1.5 are removed to reduce potential errors from sentence misalignment. Long sentences with a length greater than 250 are also removed as a common practice. The split of the datasets produces the training, validation (valid.), and test sets with the count of sentence pairs and the size of vocabulary presented in Table 1.

The transformer architecture (Vaswani *et al.* 2017) from fairseq (Ott *et al.* 2019)[b] is used for all the experiments. The hyperparameters are presented in Table 2. We compared training under CLR with an inverse square for two popular optimizers used in machine translation tasks, Adam and SGD. All models are trained using one NVIDIA V100 GPU.

The learning rate boundary of the CLR is selected by the range test (shown in Figure 2). The base and maximum (max) learning rates adopted in this study are presented in Table 3. Shrink strategy is applied when examining the effects of CLR in training NMT. The optimizers (Adam and SGD) are assigned with two options: (1) without shrink (as "nshrink"); (2) with shrink at a rate of 0.5 ("yshrink"), which means the maximum learning rate for each cycle is reduced at a decay rate of 0.5.

[b]https://github.com/pytorch/fairseq.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Table 2. Hyperparameters for the experiments

Hyperparameters	Values
Encoder/Decoder layers	6
Embedding units	512
Attention heads	4
Feed-forward hidden units	1024
Batch size (default)	4096
Training epoch (default)	50

Table 3. Learning rate boundary for CLR

	Adam		SGD	
Corpus	Max	Base	Max	Base
IWSLT2014-de-en	$5e^{-4}$	$1e^{-5}$	6.9	$1e^{-3}$
IWSLT2014-fr-en	$8e^{-4}$	$1e^{-5}$	–	–
IWSLT2017-de-en	$7.6e^{-4}$	$1e^{-5}$	8	$1e^{-3}$

5.2 Effects of applying CLR to NMT training

A hypothesis we hold is that NMT training under CLR may result in a better local minimum than that achieved by training with the default learning rate schedule. A comparison experiment is performed for training NMT models for "IWSLT2014-de-en" corpus using CLR and INV with a range of initial learning rates on two optimizers (Adam and SGD), respectively. It can be observed that both Adam and SGD are very sensitive to the initial learning rate under the default INV schedule before CLR is applied (as shown in Figures 4 and 5). In general, SGD prefers a bigger initial learning rate when CLR is not applied. The initial learning rate of Adam is more concentrated towards the central range.

Applying CLR has positive impacts on NMT training for both Adam and SGD. When applied to SGD, CLR exempts the needs for a big initial learning rate as it enables the optimizer to explore the local minima better. Shrinking on CLR for SGD is not desirable as a higher learning rate is required (Figure 5). Furthermore, we observe that applying CLR on SGD with a shrink option leads to a high convergence error (Figure 6). In this sense, the "nshrink" option is mandatory for SGD. It is noted that applying CLR to Adam produces consistent improvements regardless of shrink options (Figure 4). Furthermore, it can be observed that the effects of applying CLR to Adam are more pronounced than those of SGD, as shown in Figure 6. Similar results are obtained from our experiments on "IWSLT2017-de-en" and "IWSLT2014-fr-en" corpora (Figures A2 and A3 in Appendix A). The corresponding BLEU scores are presented in Table 4, in which the above-mentioned effects of CLR on Adam can also be established. The training takes fewer epochs to converge to reach a local minimum with better BLEU scores (i.e., bold fonts in Table 4).

5.3 Effects of batch size on CLR

Batch size is regarded as a significant factor influencing deep learning models from the various CV studies detailed in Section 1. It is well known to CV researchers that a large batch size is often associated with a poor test accuracy.

The situation is reversed for the case of NMT, as demonstrated in Popel and Bojar (2018), where the use of a larger batch size generally helps improve training BLEU. Figure 5 in Popel and Bojar

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

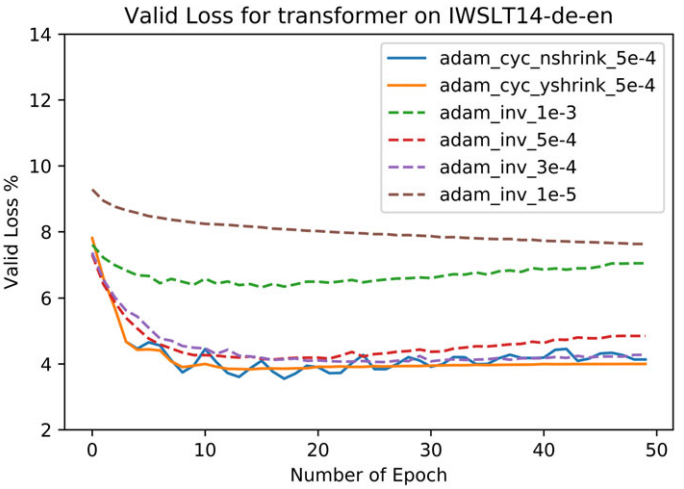


Figure 4. A comparison study of training NMT models on IWSLT2014-de-en using CLR and INV with a range of initial learning rate on Adam. The learning rate policy "adam_cyc_nshink_5e-4" denotes the optimizer Adam is trained under CLR with the no shrink option and a maximum learning rate of $5e^{-4}$.

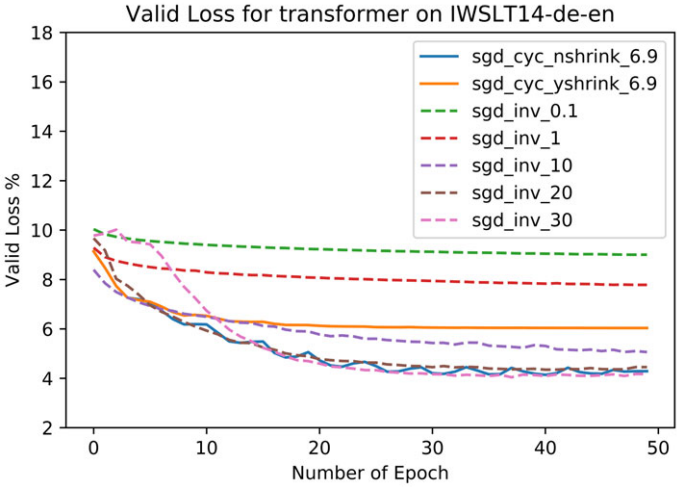


Figure 5. A comparison study of training NMT models on IWSLT2014-de-en using CLR and INV with a range of initial learning rates on SGD. The learning rate policy "sgd_cyc_yshink_5e-4" denotes the optimizer SGD that is trained under CLR with the shrink option and a maximum learning rate of $5e^{-4}$.

(2018) illustrates the benefits of large batch sizes, showing steady training BLEU improvement going from a batch size of 1000–4500, eventually plateauing at a batch size of 6000. In this paper, Figure 7 shows that the trend of CLR with a larger batch size for NMT training does indeed lead to better performance. Furthermore, when the batch size reaches a certain value, it is no longer a critical factor; for example, we observe similar validation losses for the training with batch sizes of 4096 and 8192. The effect of the various learning rate strategies is shown in Figure 8. The various strategies share similar good performance under large batch sizes, in line with Figure 5 in Popel and Bojar (2018). Interestingly, as the batch size decreases to batch size as small as 256, the difference in the various strategies starts to show. Both the triangle and sine with no shrinking strategies

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Table 4. The best BLEU for various learning rate policies when training NMT models on IWSLT2014-de-en, IWSLT2017-de-en and IWSLT2014-fr-en. The total number of training epochs for all the experiments is 50. The table is sorted by the best BLEU in descending order

Corpus	Learning rate policy	Best BLEU	Epoch
IWSLT2014-de-en	adam_cyc_nshrink_5e-4	**32.65**	**18**
	adam_cyc_yshrink_5e-4	**31.29**	**18**
	adam_inv_5e-4	30.88	16
	sgd_inv_30	30.78	42
	adam_inv_3e-4	30.46	34
	sgd_cyc_nshrink_6.9	30.16	45
IWSLT2017-de-en	adam_cyc_nshrink_7.6e-4	**33.00**	**18**
	adam_cyc_yshrink_7.6e-4	**31.56**	**19**
	sgd_inv_30	30.82	49
	adam_inv_3e-4	30.78	35
	adam_inv_5e-4	30.70	19
	sgd_cyc_nshrink_8	30.40	49
	adam_inv_7.6e-4	28.94	40
IWSLT2014-fr-en	adam_cyc_nshrink_8e-4	**37.82**	**17**
	adam_cyc_yshrink_8e-4	**36.91**	**17**
	adam_inv_5e-4	36.43	17
	adam_inv_3e-4	36.25	35
	sgd_inv_30	35.51	45
	adam_inv_8e-4	6.20	43

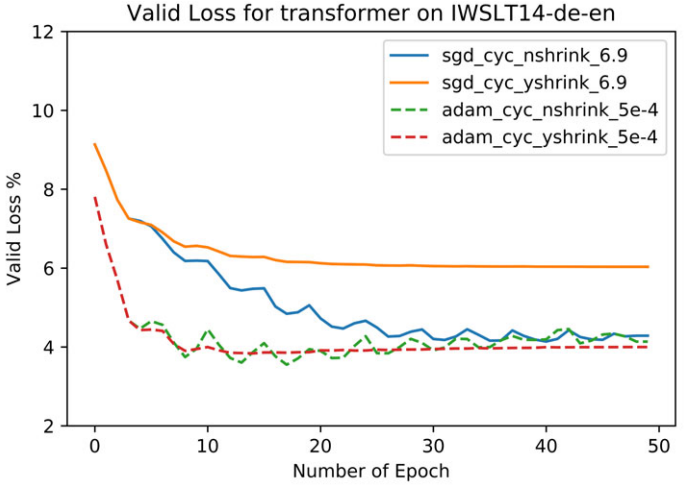


Figure 6. A view of effects of applying CLR to Adam and SGD when training the NMT on IWSLT2014-de-en.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

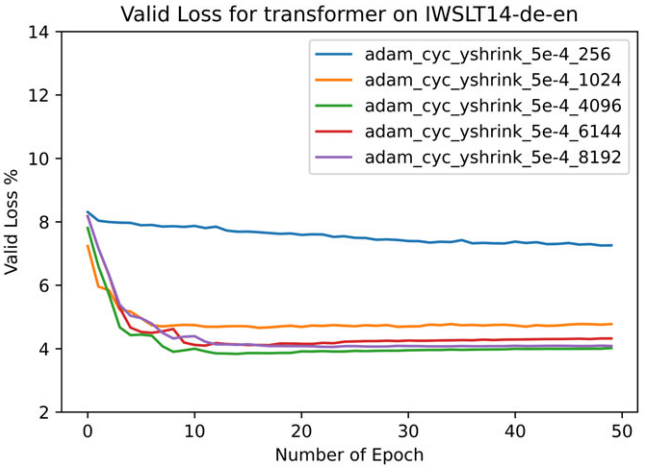


Figure 7. Effects of various batch sizes when training the NMT on IWSLT2014-de-en corpus with CLR.

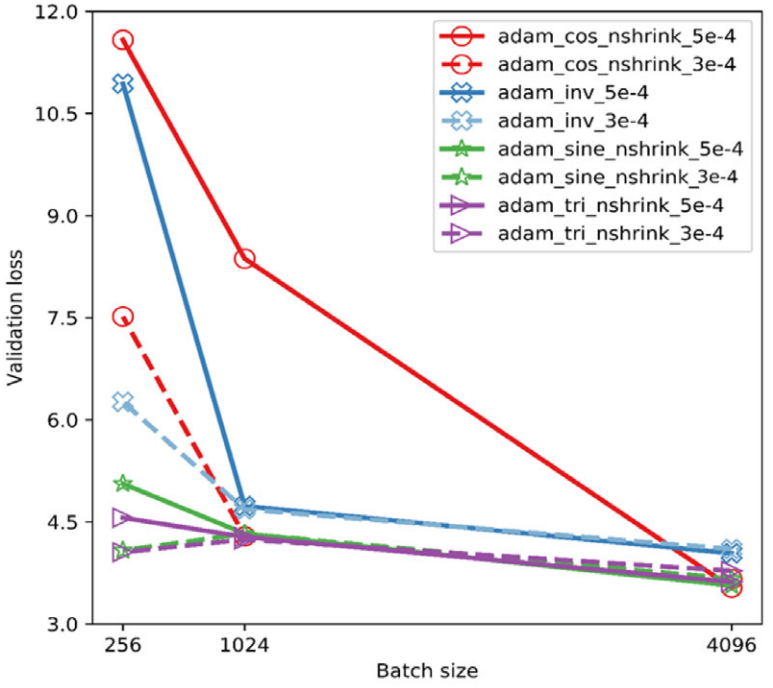


Figure 8. Effects of various batch sizes when training the NMT on IWSLT2014-de-en corpus with TRI, COS, SINE, and INV under different learning rates. The validation loss shown in this figure is the minimal validation loss when training each model with a different batch size and learning rate configuration. The training was run for a fixed number of epochs (50).

show exceptional performance even at a batch size of 256, comparable to using a large batch size of 4096. Our findings in Figure 8 have a direct bearing on NMT training with constrained GPU resources, particularly GPU memory. We run the same experiment on one GPU and four GPUs and show the results in Table A1 (Appendix A). We find out the effect of the large batch size has a roof effect after extending the experiment to 4096 × 4 GPUs. Based on the experimental results, a batch size of 4096 in a single GPU produces slightly better results than those obtained from the

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

batch size of 4096 × 4 GPUs. For memory-constrained GPU training, both the Adam triangle and sine with "nshrink" strategies present themselves as promising learning rate scheduler candidates.

A further experiment is performed to unveil whether the effect of batch size on CLR is sensitive to the learning rate scheduler. We test four learning rate schedulers, namely the triangular (TRI), a full-phase cosine (COS),[c] a sine (SINE), and a standard INV, for batch sizes of 256, 1024, and 4096, respectively, on learning rates of $3e^{-4}$ and $5e^{-4}$. A clear trend emerges from the results of this experiment (Figure 8). All four learning rate schedulers can achieve reasonable low validation loss using a large batch size (4096) given enough training epochs. All three learning rate schedulers from the CLR family outperform INV in this batch size. A TRI learning rate scheduler is a consistent winner performing well across batch bands. The larger batch size effect afforded by CLR appears to be immune to learning rates. The larger learning rate ($5e^{-4}$) is associated with a higher validation loss at a smaller batch size (256) though. An immediate question that may arise from Figure 8 is the significant performance difference between the sine and cosine forms. One possible explanation is that the sine form allows warm-up, whereas the cosine form starts with a large learning rate before decay. As many previous studies have shown, having a warm-up stage helps achieve better training results.

6. Further analysis and discussions

An unsuitable maximum learning rate for the triangular policy of CLR-based NMT often leads to non-convergence. The learning rate range test illustrated in Section 4.2 is further discussed in this section to unveil the practical details. It is suggested to validate the effect of CLR on a larger scale dataset. We perform a comparison test on training NMT models using Adam with a TRI and an INV scheduler on WMT2014-de-en to this end. We also discuss the rationale why CLR works in this section.

6.1 How to apply CLR to NMT training matters

As mentioned in Section 4.2, the value of the base learning rate is required to be a small value to guarantee convergence during the training. We believe that setting the minimum learning rate to be $1e^{-5}$ is reasonable. It is a value empirically ensuring convergence when training transformer-based NMT (note that the observations and analysis are based on IWLST dataset). Training with a base learning rate of a value of $1e^{-8}$ generates an almost identical result to that from $1e^{-5}$ (Figure 9). The experiment shows that the training may not be sensitive to selecting the base learning rate as long as it is a value ensuring convergence.

A range test is performed to identify the maximum learning rates for the triangular policy of CLR (Figure 2). The experiments show the training is sensitive to the selection of $\eta_{\max}$. As the range curve for training NMT models is distinctive to that obtained from a typical case of computer vision, it is not clear how to choose the $\eta_{\max}$ when applying CLR. A comparison experiment is performed to try $\eta_{\max}$ with different values. It can be observed that the first $\eta_{\max}$ (MLR1) is a preferable option for both SGD and Adam (Figures 10 and 11). Choosing a larger learning rate on the extreme right end as the triangular policy's maximum learning rate often leads to the loss not converging due to gradient divergence. It is better to be more conservative and choose the point where the loss stagnates as the maximum learning rate for the triangular policy.

The "nshrink" option is mandatory for SGD, but this constraint can be relaxed for Adam. Adam is sensitive to excessive learning rate, that is, the second $\eta_{\max}$ (MLR2). The pairing of CLR with Adam produces consistent improvements regardless of shrink rate, but such enhancement is not

[c]This scheduler uses a full-phase cosine wave. This is different than the one described in Loshchilov and Hutter (2016) which uses the half cosine wave and warm restarts.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

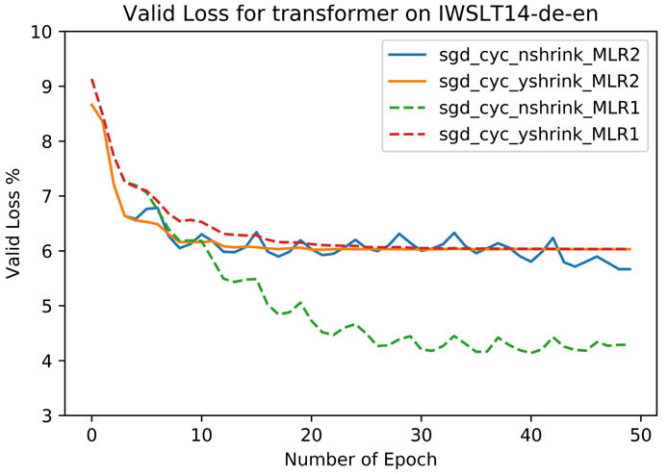


Figure 9. The first $\eta_{\max}$ (MLR1) with "nshrink" is a preferable option for SGD when applying CLR to train NMT models on IWSLT2014-de-en.

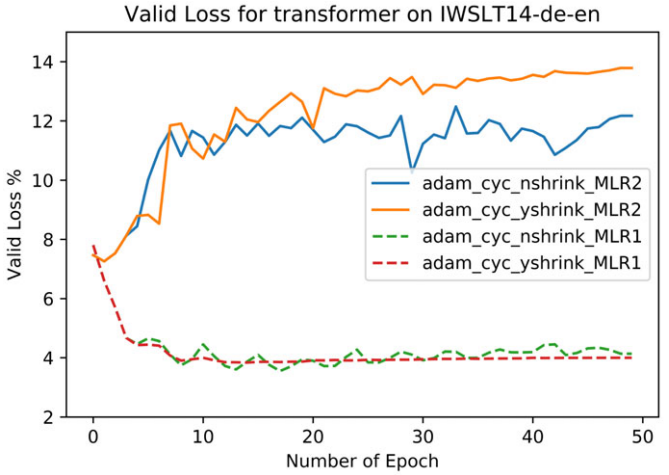


Figure 10. The first $\eta_{\max}$ (MLR1) is a preferable option for Adam when applying CLR to train NMT models on IWSLT2014-de-en.

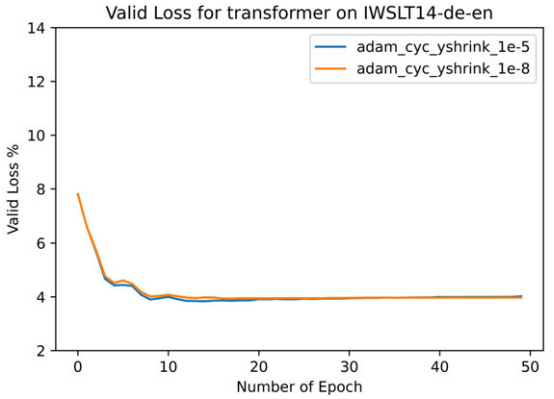


Figure 11. Comparison of the base learning rate with values of $1e^{-5}$ and $1e^{-8}$ for the IWSLT2014-de-en dataset.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Table 5. WMT 2014 DE2EN dataset used for the experiment

Corpus	Training	Valid.	Test	Source Vocab.	Target Vocab.
WMT2014-de-en (DE2EN)	3,961,179	40,058	3003	43,460	43,460

Table 6. The best BLEU scores for training NMT models on WMT14-de-en dataset using TRI and INV learning rate schedulers

Learning rate policy	Best BLEU	Epoch
adam_cyc_yshrink_1e-3	31.43	17
adam_inv_1e-3	30.95	45

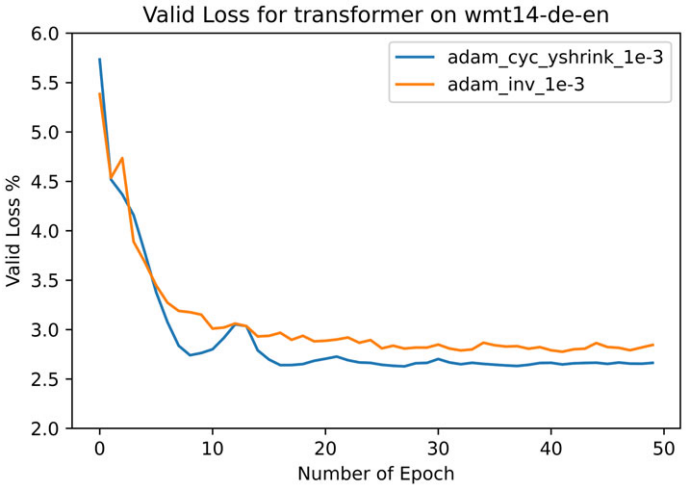


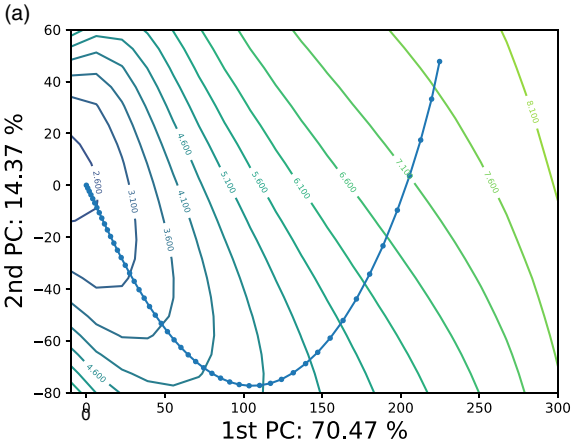
Figure 12. Comparison of the validation loss with a maximum learning rate of $1e^{-3}$ for the WMT2014-de-en dataset.

the same for different datasets. This indicates the pre-defined shrink rate (i.e., 0.5) can further be optimized.

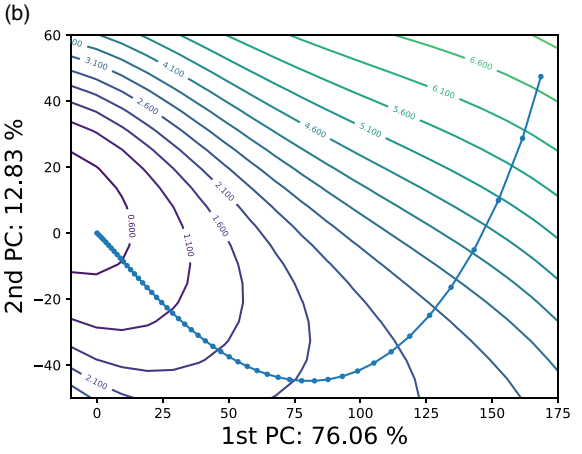
6.2 Validating the effect of CLR on NMT training in a large scale dataset

As shown in Table 5, the WMT2014-de-en data is involved to train NMT models using Adam with a TRI and an INV learning rate scheduler. We perform a range test and produce a maximum learning rate ($\eta_{\max}$) of $1e^{-3}$ from the initial rate $1e^{-5}$. A step size of 4.5 epochs is chosen to set the TRI learning rate scheduler with a shrink option. The experimental results demonstrate that the NMT model trained with a TRI learning rate scheduler can reach a lower validation loss compared to the model with an INV scheduler (Figure 12). The corresponding best BLEU scores for this comparison study are presented in the Table 6. It can be observed that the NMT model trained with a TRI learning rate scheduler can produce a higher BLEU score than that using INV scheduler in a much fewer number of epochs.

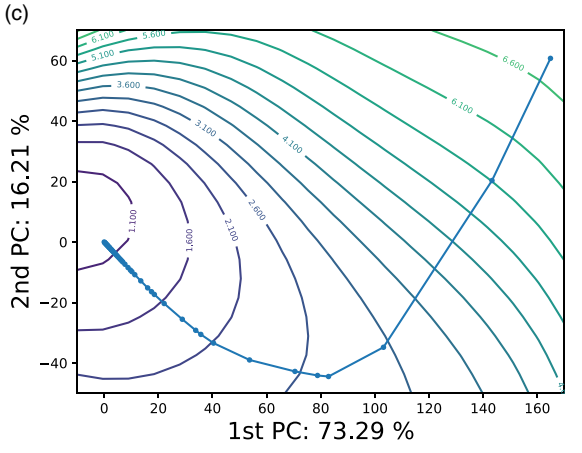
https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press



adam-inv-1e-3



adam-inv-5e-4



adam-cyc-yshrink

Figure 13. Loss surface, optimizer trajectory, and learning rates visualization for training NMT models on IWSLT2014-de-en.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

6.3 Rationale behind applying CLR to NMT training

As pointed out in Dauphin, de Vries, and Bengio (2015), the difficulty in optimizing deep learning networks is due to saddle points, not local minima. One theoretical perspective of why CLR works is that the increasing learning rate helps the optimizer to escape from saddle point plateaus (Smith 2017). The other more intuitive reason is that the learning rates covered in CLR are likely to include the near-optimal learning rate which will be would ideally be used throughout the training (Smith 2017). This near-optimal learning rate during the cycle may allow training to escape from a saddle the training is stuck in.

The other benefit that CLR brings is fast convergence. The mental model we have is that of the training being in the mouth of a near global minimum valley and the near-optimal learning rate encountered during the cycle allows training to make rapid progress towards the valley bottom. Figures 4 and 5 lend credence to this hypothesis, where we see CLR converging much faster compared to INV. Leveraging the visualization techniques proposed by Li *et al.* (2018), we take a peek at the error surface, optimizer trajectory, and learning rate. The first thing to note is the smoothness of the error surface. This is perhaps not so surprising given the abundance of skip connections in transformer-based networks. Referring to Figure 13(c), we see the cyclical learning rate greatly amplifying Adam's learning rate in flatter region while nearer the local minimum, the cyclical learning rate policy does not harm convergence to the local minimum. This is in contrast to Figure 13(a) and (b), where although the adaptive nature of the learning rate in Adam helps to move quickly across flatter region, the effect is much less pronounced without the cyclical learning rate. Figure 13 not only gives credence to the hypothesis that the cyclical learning rate helps to escape saddle point plateaus but it also demonstrates that the near-optimal learning rate is likely included in the cyclical learning rate policy.

Some explanation about Figure 13 is in order here. Following Li *et al.* (2018), we first assemble the network weight matrix by concatenating columns of network weights at each epoch. We then perform a principal component analysis and use the first two components for plotting the loss landscape. Even though all three plots in Figure 13 seem to converge to the local minimum, bear in mind that this is only for the first two components, with the first two components contributing to 84.84%, 88.89%, and 89.5% of the variance, respectively. With the first two components accounting for a large portion of the variance, it is thus reasonable to use Figure 13 as a qualitative guide.

7. Conclusion

In this paper, we perform an empirical study exploring how cyclical learning rate can be applied to train transformer-based neural networks for NMT. From the various experiment results, we have explored the use of CLR and demonstrated the benefits of CLR for transformer-based networks unequivocally. Not only does CLR help to improve the generalization capability in terms of test set results but it also allows using smaller batch size for training without adversely affecting the generalization capability. Instead of just using default optimizers and learning rate policies, we hope to bring insights to the NMT community on the importance and the way of choosing a useful optimizer and an associated learning rate policy.

Acknowledgements. We would like to appreciate Peng Yang for his support relating to loss surface visualization.

References

Aarts E.H.L. and **Korst J.H.M.** (2003). Simulated annealing and boltzmann machines. In Michael A. Arbib (ed), *Handbook of Brain Theory and Neural Networks* (2nd ed). Cambridge, Massachusetts: MIT Press, pp. 1039–1044.

Bahdanau D., **Cho K.** and **Bengio Y.** (2015). Neural machine translation by jointly learning to align and translate. In Bengio Y. and LeCun Y. (eds), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Bengio Y., **Louradour J.**, **Collobert R.** and **Weston J.** (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009*, ACM International Conference Proceeding Series, vol. 382, pp. 41–48.

Bottou L. (2010). Large-scale machine learning with stochastic gradient descent. In *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22–27, 2010 - Keynote, Invited and Contributed Papers*, pp. 177–186.

Cettolo M., **Girardi C.** and **Federico M.** (2012). Wit ^{3}: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy, pp. 261–268.

Chung J., **Gülçehre Ç.**, **Cho K.** and **Bengio Y.** (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. ArXiv, abs/1412.3555.

Dauphin Y., **de Vries H.** and **Bengio Y.** (2015). Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada*, pp. 1504–1512.

Devlin J., **Chang M.-W.**, **Lee K.** and **Toutanova K.** (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Dinh L., **Pascanu R.**, **Bengio S.** and **Bengio Y.** (2017). Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017, Volume 70*, pp. 1019–1028.

Duchi J.C., **Hazan E.** and **Singer Y.** (2010). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **12**, 2121–2159.

He K., **Zhang X.**, **Ren S.** and **Sun J.** (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hoang C.D.V., **Haffari G.** and **Cohn T.** (2017). Decoding as continuous optimization in neural machine translation. arXiv preprint arXiv:1701.02854.

Hochreiter S. and **Schmidhuber J.** (1997a). Flat minima. *Neural Computation* **9**, 1–42.

Hochreiter S. and **Schmidhuber J.** (1997b). Long short-term memory. *Neural Computation* **9**, 1735–1780.

Huang G., **Liu Z.** and **Weinberger K.Q.** (2016). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269.

Keskar N.S., **Mudigere D.**, **Nocedal J.**, **Smelyanskiy M.** and **Tang P.T.P.** (2016). On large-batch training for deep learning: Generalization gap and sharp minima. ArXiv, abs/1609.04836.

Kingma D.P. and **Ba J.** (2014). Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

Koehn P., **Hoang H.**, **Birch A.**, **Callison-Burch C.**, **Federico M.**, **Bertoldi N.**, **Cowan B.**, **Shen W.**, **Moran C.**, **Zens R.**, **Dyer C.**, **Bojar O.**, **Constantin A.** and **Herbst E.** (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic. Association for Computational Linguistics, pp. 177–180.

Krizhevsky A., **Sutskever I.** and **Hinton G.E.** (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM* **60**, 84–90.

Li H., **Xu Z.**, **Taylor G.** and **Goldstein T.** (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3–8 December 2018, MontrÉal, Canada*, pp. 6391–6401.

Liu L., **Jiang H.**, **He P.**, **Chen W.**, **Liu X.**, **Gao J.** and **Han J.** (2019). On the variance of the adaptive learning rate and beyond. *ArXiv*, abs/1908.03265.

Loshchilov I. and **Hutter F.** (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.

Luo L., **Xiong Y.**, **Liu Y.** and **Sun X.** (2019). Adaptive gradient methods with dynamic bound of learning rate. ArXiv, abs/1902.09843.

McCandlish S., **Kaplan J.**, **Amodei D.** and **Team O.D.** (2018). An empirical model of large-batch training. ArXiv, abs/1812.06162.

Ott M., **Edunov S.**, **Baevski A.**, **Fan A.**, **Gross S.**, **Ng N.**, **Grangier D.** and **Auli M.** (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Demonstrations*, pp. 48–53.

Popel M. and **Bojar O.** (2018). Training tips for the transformer model. *Prague Bulletin of Mathematical Linguistics* **110**, 43–70.

Reddi S.J., **Kale S.** and **Kumar S.** (2018). On the convergence of adam and beyond. ArXiv, abs/1904.09237.

Ruder S. (2016). An overview of gradient descent optimization algorithms. ArXiv, abs/1609.04747.

Sennrich R., **Haddow B.** and **Birch A.** (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)*, Berlin, Germany. Association for Computational Linguistics, pp. 1715–1725.

Simonyan K. and **Zisserman A.** (2014). Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556.

Smith L.N. (2017). Cyclical learning rates for training neural networks. In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472.

Smith L.N. and **Topin N.** (2017). Super-convergence: Very fast training of residual networks using large learning rates. ArXiv, abs/1708.07120.

Smith S.L., **Kindermans P.**, **Ying C.** and **Le Q.V.** (2018). Don't decay the learning rate, increase the batch size. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Vaswani A., **Shazeer N.**, **Parmar N.**, **Uszkoreit J.**, **Jones L.**, **Gomez A.N.**, **Kaiser L.U.** and **Polosukhin I.** (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 5998–6008.

Wiseman S. and **Rush A.M.** (2016). Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1–4, 2016*, pp. 1296–1306.

Zhang M.R., **Lucas J.**, **Hinton G.E.** and **Ba J.** (2019). Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8–14 December 2019, Vancouver, BC, Canada*, pp. 9593–9604.

A. Appendix

Figures A1–A3, Table A1 are included in this Appendix.

Figure A1. Range test curve of SGD for the IWSLT2014-de-en dataset.

https://doi.org/10.1017/S135132492200002X Published online by Cambridge University Press

Table A1. The comparison results of one GPU and four GPUs with a batch size of 4096

	one GPU		four GPUs	
Learning rate policy	Best BLEU	Epoch	Best BLEU	Epoch
adam_cyc_nshrink_5e-4	32.65	18	31.72	18
adam_cyc_yshrink_5e-4	31.29	18	31.72	18
adam_inv_5e-4	30.88	16	30.66	16
sgd_cyc_nshrink_6.9	30.16	45	27.10	49
adam_inv_3e-4	30.46	34	30.16	37
sgd_inv_30	30.78	42	29.71	49

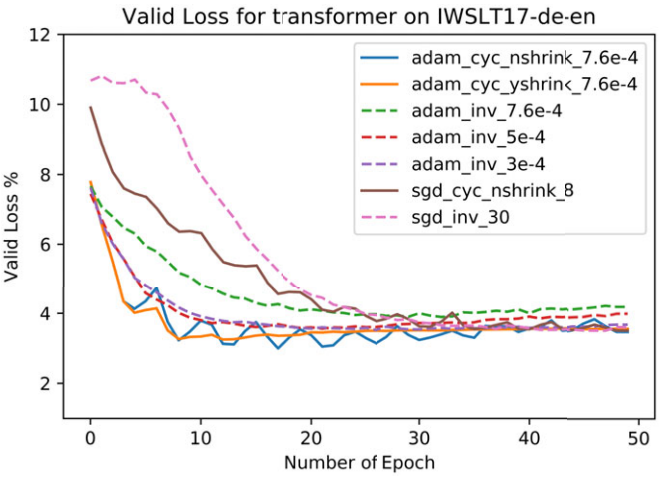


Figure A2. Effects of applying CLR to training NMT on IWSLT2017-de-en.

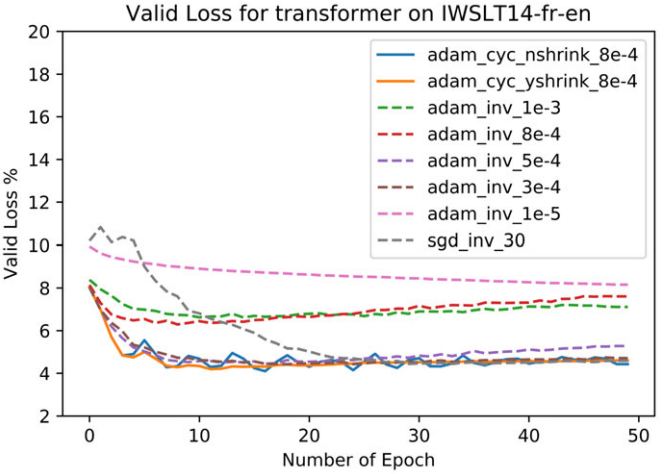


Figure A3. Effects of applying CLR to training NMT on IWSLT2014-fr-en.

Cite this article: Wang W, Lee CM, Liu J, Colakoglu T and Peng W (2023). An empirical study of cyclical learning rate on neural machine translation. *Natural Language Engineering* **29**, 316–336. https://doi.org/10.1017/S135132492200002X