Dynamically Adjusting Transformer Batch Size by Monitoring Gradient Direction Change

Hongfei Xu[1,2]   **Josef van Genabith**[1,2]   **Deyi Xiong**[3]   **Qiuhui Liu**[4*]

[1]Saarland University / Saarland, Germany
[2]German Research Center for Artificial Intelligence / Saarland, Germany
[3]Tianjin University / Tianjin, China
[4]China Mobile Online Services / Henan, China
hfxunlp@foxmail.com, Josef.Van_Genabith@dfki.de,
dyxiong@tju.edu.cn, liuqhano@foxmail.com

Abstract

The choice of hyper-parameters affects the performance of neural models. While much previous research (Sutskever et al., 2013; Duchi et al., 2011; Kingma and Ba, 2015) focuses on accelerating convergence and reducing the effects of the learning rate, comparatively few papers concentrate on the effect of batch size. In this paper, we analyze how increasing batch size affects gradient direction, and propose to evaluate the stability of gradients with their angle change. Based on our observations, the angle change of gradient direction first tends to stabilize (i.e. gradually decrease) while accumulating mini-batches, and then starts to fluctuate. We propose to automatically and dynamically determine batch sizes by accumulating gradients of mini-batches and performing an optimization step at just the time when the direction of gradients starts to fluctuate. To improve the efficiency of our approach for large models, we propose a sampling approach to select gradients of parameters sensitive to the batch size. Our approach dynamically determines proper and efficient batch sizes during training. In our experiments on the WMT 14 English to German and English to French tasks, our approach improves the Transformer with a fixed $25k$ batch size by $+0.73$ and $+0.82$ BLEU respectively.

1 Introduction

The performance of neural models is likely to be affected by the choice of hyper-parameters. While much previous research (Sutskever et al., 2013; Duchi et al., 2011; Kingma and Ba, 2015) focuses on accelerating convergence and reducing the effects of the learning rate, comparatively few papers concentrate on the effect of batch size.

However, batch size is also an important hyper-parameter, and some batch sizes empirically lead to better performance than the others.

Specifically, it has been shown that the performance of the Transformer model (Vaswani et al., 2017) for Neural Machine Translation (NMT) (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) relies heavily on the batch size (Popel and Bojar, 2018; Ott et al., 2018; Abdou et al., 2017; Zhang et al., 2019a).

The influence of batch size on performance raises the question, how to dynamically find proper and efficient batch sizes during training? In this paper, we investigate the relationship between the batch size and gradients, and propose a dynamic batch size approach by monitoring gradient direction changes. Our contributions are as follows:

- We observe the effects on gradients with increasing batch size, and find that a large batch size stabilizes the direction of gradients;

- We propose to automatically determine dynamic batch sizes in training by monitoring the gradient direction change while accumulating gradients of small batches;

- To measure gradient direction change efficiently with large models, we propose an approach to dynamically select those gradients of parameters/layers which are sensitive to the batch size;

- In machine translation experiments, our approach improves the training efficiency and the performance of the Transformer model.

2 Gradient Direction Change and Automated Batch Size

Gradients indicate the direction and size of parameter updates to minimize the loss function in training. To reveal the effects of the batch size in optimization, we evaluate its influence on the direction change of gradients.

---

k	1	2	3	4	5	6	7	8	9	10
Size	4064	8994	12768	17105	21265	25571	29411	33947	38429	43412
$a(g_0^{k-1}, g_0^k)$		51.52	30.37	27.42	22.61	20.87	19.80	19.59	18.92	19.23
$a(g_0^{k-3}, g_0^k)$				59.53	44.20	41.77	35.34	32.19	32.10	34.29

Table 1: The direction change of gradients while accumulating mini-batches.

2.1 Gradient Direction Change with Increasing Batch Size

To investigate the influence of batch size on gradient direction, we gradually accumulate gradients of small mini-batches as the gradients of a large batch that consists of those mini-batches, and observe how the direction of gradients varies.

Let $d_i^j : (x_i^j, y_i^j)$ stands for the large batch concatenated from the ith mini-batch to the jth mini-batch, where x_i^j and y_i^j are inputs and targets. Then the gradients g_i^j of model parameters θ on d_i^j are:

$$g_i^j = \frac{\partial L(\theta, x_i^j, y_i^j)}{\partial \theta} \quad (1)$$

In gradient accumulation, the gradients g_0^k are the sum of g_0^{k-1} and g_k^k:

$$g_0^k = g_0^{k-1} + g_k^k \quad (2)$$

To measure the change of gradient direction during accumulation, we regard the two gradients g_0^{k-1} and g_0^k as 2 vectors, and compute the angle $a(g_0^{k-1}, g_0^k)$ between them:

$$a(g_0^{k-1}, g_0^k) = \arccos(\frac{g_0^{k-1} \bullet g_0^k}{|g_0^{k-1}||g_0^k|}) \quad (3)$$

where "$\bullet$" indicates inner-product of vectors.

We use the angle of 2 vectors rather than cosine similarity because:

- The angle indicates the change between gradient directions;

- When the angle is small, a significant change in the angle only results in a subtle difference in cosine similarity.[1]

We observe the gradient direction varying during accumulating gradients of a Transformer model training on the WMT 14 English-German task following the setting of Vaswani et al. (2017) with a batch size of around $50k$ target tokens. To achieve the gradient of the large batch size, we gradually accumulate gradients of mini-batches with around $4k$ target tokens.

Table 1 shows a typical example: (i) gradient change is high at the beginning, (ii) gradient change reduces with increasing batch size and (iii) eventually it will start fluctuating (here at k=10).[2]

Intuitively, the less the direction of accumulated gradients is moved by the gradients of a new mini-batch, the more certainty there is about the gradient direction. Thus we propose that the magnitude of the angle fluctuation relates to the certainty of the model parameter optimization direction, and may therefore serve as a measure of optimization difficulty.

2.2 Automated Batch Size with Gradient Direction Change

Table 1 shows that the optimization direction is less stable with a small batch than with a large batch. But after the direction of gradients has stabilized, accumulating more mini-batches seems useless as the gradient direction starts to fluctuate.

Thus, we suggest to compute dynamic and efficient batch sizes by accumulating gradients of mini-batches, while evaluating the gradient direction change with each new mini-batch, and stop accumulating more mini-batches and perform an optimization step when the gradient direction fluctuates.

In practice, we only monitor $a(g_0^{k-1}, g_0^k)$ for efficiency. We record the minimum angle change a_{min} while accumulating gradients, and suppose the gradient direction starts to fluctuate, stop accumulating more mini-batches when $a(g_0^{k-1}, g_0^k) > a_{min} * \alpha$. In this way we can achieve a dynamic batch size (the size of d_0^k), where α is a pre-specified hyper-parameter.

---

[1] $cos(5°) \approx 0.9961$, $cos(10°) \approx 0.9848$.

[2] By comparing $\sum_{i=0}^{n} a(g_0^{k-i-1}, g_0^{k-i})$ with $a(g_0^{k-n-1}, g_0^k)$, we can find the direction changes from g_0^{k-i-1} to g_0^k are inconsistent. Otherwise, $\sum_{i=0}^{n} a(g_0^{k-i-1}, g_0^{k-i}) \approx a(g_0^{k-n-1}, g_0^k)$.

2.3 Efficiently Monitoring Gradient Direction Change

In practice, a model may have a large amount of parameters, and the cost of computing the cosine similarity between two corresponding gradient vectors are relatively high. To tackle this issue, we propose to divide model parameters into groups, and monitor gradient direction change only on a selected group in each optimization step. For a multi-layer model, i.e. the Transformer, a group may consist of parameters of 1 layer or several layers.

To select the parameter group which is sensitive to the batch size, we record the angles of gradient direction change $a(g_0^0, g_0^1), ..., a(g_0^{k-1}, g_0^k)$ in the gradient accumulation, and define a_{max} and a_{min} as the maximum and minimum direction change:

$$a_{max} = max(a(g_0^0, g_0^1), ..., a(g_0^{k-1}, g_0^k)) \quad (4)$$

$$a_{min} = min(a(g_0^0, g_0^1), ..., a(g_0^{k-1}, g_0^k)) \quad (5)$$

We then use Δa to measure the uncertainty reduction in the optimization direction:

$$\Delta a = a_{max} - a_{min} \quad (6)$$

Intuitively, the optimization direction of the parameter group which results in a larger Δa profits more from the batch size, and the group with a larger Δa should be more frequently sampled.

We average the recent history of Δa_k of the kth parameter group into $\overline{\Delta a_k}$. Inspired by Gumbel (1954); Maddison et al. (2014); Zhang et al. (2019b), we first add Gumble noise to each $\overline{\Delta a_k}$ to prevent the selection falling into a fixed group:

$$\Delta a_k^* = \overline{\Delta a_k} - \log(-\log u) \quad (7)$$

where $u \in (0, 1)$ is a uniform distribution.

Then we zero negative values[3] in $\Delta a_1^*, ..., \Delta a_n^*$ and normalize them into a probability distribution:

$$p_k = \frac{\Delta a_k^{*\beta}}{\sum\limits_{i=1}^n \Delta a_i^{*\beta}} \quad (8)$$

We use p_k as the probability to sample the kth group, and β is a hyper-parameter to sharpen the probability distribution. We do not use softmax

---

[3] Δa_k is positive, but after adding Gumble noise, there is a small possibility that it turns negative. In our case, negative values only occur very few times.

Batch Size	En-De	En-Fr	Time
25k	27.38	39.34	35h21m
50k	27.93	39.97	60h38m
dyn	**28.11**[†]	**40.16**[†]	**33h37m**

Table 2: Performance. Time is the training time on the WMT 14 En-De task for $100k$ training steps. † indicates $p < 0.01$ in the significance test.

	En-De	En-Fr
min	7069	8025
avg	26264.19	30248.90
max	102165	103352

Table 3: Statistics of Batch Size.

because it would heavily sharpen the distribution when the gap between values is large, and makes it almost impossible to select and evaluate the other groups in addition to the one with highest Δa_k^*.[4]

3 Experiments

We implemented our approaches based on the Neutron implementation (Xu and Liu, 2019) of the Transformer translation model. We applied our approach to the training of the Transformer, and to compare with Vaswani et al. (2017), we conducted our experiments on the WMT 14 English to German and English to French news translation tasks on 2 GTX 1080Ti GPUs. Hyper parameters were tuned on the development set (newstest 2012 and 2013). We followed all settings of Vaswani et al. (2017) except for the batch size. We used a beam size of 4 for decoding, and evaluated case-sensitive tokenized BLEU[5] with significance test (Koehn, 2004).

We used an α of 1.1 to determine the fluctuation of gradient direction by default. We regarded each encoder/decoder layer as a parameter group, and used a β of 3 for the parameter group selection.

3.1 Performance

We compared the results of our dynamic batch size approach to two fixed batch size baselines, the $25k$

---

[4] For example, the result of softmax over [22, 31, 60] is [3.13e-17, 2.54e-13, 1.00], the last element takes almost all possibility mass. But we later find that if Δa is normalized ($\Delta a = (a_{max} - a_{min})/a_{max}$) in Equation 6, the softmax works comparably well, which avoids using the hyper parameter β in Equation 8.

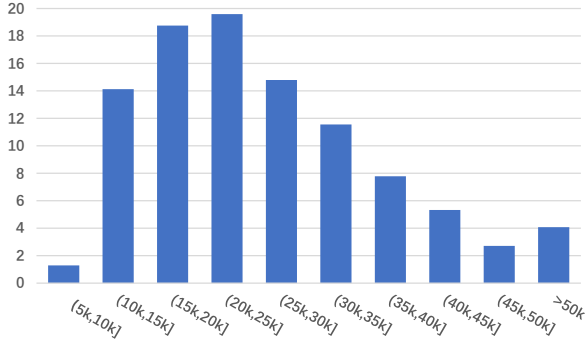[5] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

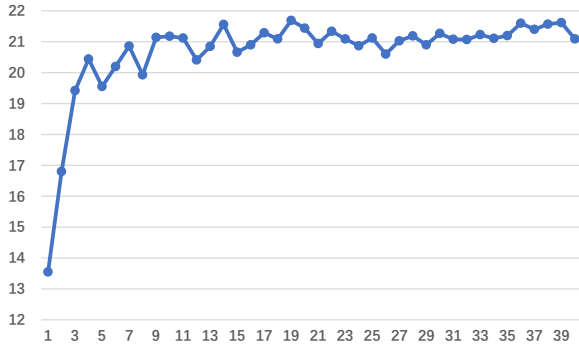Figure 1: Distribution of Dynamic Batch Sizes. Values on y-axis are percentages.

	Batch Size			
α	avg	max	BLEU	Time
1.0	19367.76	60945	27.90	**24h50m**
1.1	26264.19	102165	28.11	33h37m
1.2	36208.47	164908	**28.39**	46h04m
1.3	51470.34	205210	28.37	63h56m

Table 4: Effects of Different α.

Results are shown in Figure 2.

Figure 2 shows that the minimum direction change of gradients was small at the beginning, and gradually increased with training. Given that a small angle change indicates that there is more certainty in the gradient direction, this observation is consistent with the fact that finding the optimization direction is harder and harder with training.

3.3 Effects of α

We studied the effects of different α values on the En-De task, and results are shown in Table 4.[7]

Table 4 shows that with increasing α, the average batch size and the time cost increases along with the performance. A wide range of values works relatively well indicating that its selection is robust, and 1.1 seems to be a good trade off between the cost and the performance in our experiments.[8] It is also worth noting that $\alpha = 1$ outperforms the $25k$ baseline while being 1.42 times faster (Table 2).

4 Related Work

Popel and Bojar (2018) demonstrate that the batch size affects the performance of the Transformer, and a large batch size tends to benefit performance, but they use fixed batch sizes during training. Abdou et al. (2017) propose to use a linearly increasing batch size from 65 to 100 which slightly outperforms their baseline. Smith et al. (2018) show that the same learning curve on both training and test sets can be obtained by increasing the batch size during training instead of decaying the learning rate.

For fast convergence, Balles et al. (2017) propose to approximately estimate the mean value of the batch size for the next batch by maximizing the expected gain with a sample gradient variance ($||g||^2$) computed on the current batch, while our

Figure 2: Minimum Gradient Direction Change during Training. X-axis 2.5k training steps, y averaged a_{min} (Equation 5).

batch size is the empirical value of Vaswani et al. (2017), while Zhang et al. (2019a) investigate $50k$ batch size. Results are shown in Table 2 with the statistics of batch sizes of our approach shown in Table 3 and the detailed distribution of batch sizes for the En-De task shown in Figure 1.

Table 2 and 3 show that our approach outperforms both the fixed $25k$ and $50k$ batch size settings with an average batch size around $26k$, and our approach is slightly faster than the $25k$ setting despite of the additional cost for monitoring gradient direction change.[6]

Figure 1 shows an interesting fact that the most frequently used automated batch sizes were close to the fixed value ($25k$) of Vaswani et al. (2017).

3.2 Analysis of Minimum Gradient Direction Change

In order to observe the varying of minimum gradient direction change during training, we averaged the minimum angle for every $2.5k$ training steps.

---

[6] It is hard to accumulate an accurate $25k$ target tokens in a batch, and in fact, the fixed $25k$ setting results in an average batch size of 26729.79.

[7] We observed that the minimum batch size does not change significantly with increasing α, so we omit it for space.

[8] For $\alpha = 1.2$ on the En-Fr task, the corresponding values are: 44294.16, 185972, **40.35** and 54h12m.

approach compares the gradient direction of change $(a(g_0^{k-1}, g_0^k))$ during accumulation of mini-batches in the assembling of a large batch.

We suggest our approach is complementary to Sutskever et al. (2013); Duchi et al. (2011); Kingma and Ba (2015), as their approaches decide the magnitude of the move in the optimization direction, while our approach provides reliable gradient direction.

5 Conclusion

In this paper, we analyze the effects of accumulated batches on the gradient direction, and propose to achieve efficient automated batch sizes by monitoring change in gradient accumulation and performing an optimization step when the accumulated gradient direction is almost stable. To improve the efficiency of our approach with large models, we propose a sampling approach to select gradients of parameters sensitive to the batch size.

Our approach improves the Transformer with a fixed $25k$ batch size by $+0.73$ and $+0.82$ BLEU on the WMT 14 English to German and English to French tasks respectively while preserving efficiency.

Acknowledgments

References

Mostafa Abdou, Vladan Glončák, and Ondřej Bojar. 2017. Variable mini-batch sizing and pre-trained embeddings. In *Proceedings of the Second Conference on Machine Translation*, pages 680–686, Copenhagen, Denmark. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lukas Balles, Javier Romero, and Philipp Hennig. 2017. Coupling adaptive batch sizes with learning rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Emil Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A∗ sampling. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3086–3094. Curran Associates, Inc.

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

Martin Popel and Ondej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110:43–70.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2018. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Ma-*

chine Learning, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

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

Hongfei Xu and Qiuhui Liu. 2019. Neutron: An Implementation of the Transformer Translation Model and its Variants. *arXiv preprint arXiv:1903.07402*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019a. Improving deep transformer with depth-scaled initialization and merged attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 898–909, Hong Kong, China. Association for Computational Linguistics.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019b. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy. Association for Computational Linguistics.