[AI502] Midterm Project Survey Paper

Victor Cappa[*]
KAIST School of Computing
victor.cappa@kaist.ac.kr

Abstract

This survey paper subsumes and analyzes different papers regarding deep learning related topics such as a new alternative training methodology based on increasing the batch size instead of the most common technique based on decaying the learning rate and we analyze the implications of this approach and the benefits that we can obtain by applying this principle [1], we address the problem of the variance of the adaptive learning rate of adaptive stochastic optimization algorithms such as Adam and we propose a new alternative algorithm better able to tackle this problem called RAdam (Rectified Adam) [2], we introduce an alternative neural network architecture called MobileNetV2 specifically tailored for mobile and resource constrained environments based on a novel layer module called linear bottleneck [3], and finally, we analyze four different techniques to improve Batch Normalization such as Inference Example Weighing, Ghost BN, Weight Decay in BN, and generalizing batch and group normalization for small batch sizes[4].

1 Don't decay the Learning Rate, Increase the Batch Size

It is common practice to use learning rate dacay schedulers to decay the learning rate while training. However, we can achieve similar results by increasing the batch size instead of decaying the learning rate, assuming we are using for stochastic gradient descent (SGD), SGD with momentum, Nesterov momentum, and Adam. We are interested in increasing the batch size while maintaining the same number of epochs because this approach allows us to successfully reduce the number of parameter updates required to train the model and because large batches can be parallelized across many machines, thus allowing to speed up training. In particular we are interested in three different techniques to achieve this goal.

Decay Rate Scheduler We observe that we can obtain the same learning curve on both training and test sets by increasing the batch size during training while maintaining a constant learning rate instead of adopting a decay rate scheduler to decay the learning rate. This strategy achieves near-identical model performance on the test set with the same number of training epochs but significantly fewer parameter updates. To explain this concept we introduce the scale of random fluctuations in the SGD dynamics [5] as $g = \epsilon(\frac{N}{B} - 1)$, where ϵ is the learning rate, N training set size and B batch size. We observe that whenever we decay the learning rate, the noise scale g falls, enabling us to converge to the minimum of the cost function. Adopting a learning rate decay schedulers during training allows to decay the scale of random fluctuations in the SGD dynamics but we can achieve the same result by increasing the batch size and maintaining a constant learning rate. Proving that the noise scale is relevant, and not the learning rate itself. This strategy permits to have significantly fewer parameter updates while maintaining the same test-set performances of the learning rate decay scenario (refer to Section 5.1 of [1]). Concretely, we follow pre-existing training schedules, but when the learning rate is supposed to drop by a factor of α, we instead increase the batch size by α.

[*]Master's program student at KAIST School of Computing. Daejeon, South Korea. ID number: 20206080

Increasing Learning Rate and Batch Size We can further increase the batch size by increasing the learning rate ϵ and scaling B proportionally to ϵ. We prove empitically that it doesn't result in test set performance degradation.

Scaling Batch Size according to Momentum coefficient Finally, another technique consists in increasing the momentum coefficient of SGD with momentum and scaling the batch size B proportionally to $\frac{1}{1-m}$, where m in the momentum coefficient. Although, this slightly reduces the test accuracy. To explain this concept we can extend the previously found noise scale g to include the momentum coefficient for SGD with momentum. The resulting noise scale is given by $g = \epsilon \frac{1}{1-m}(\frac{N}{B} - 1)$.

1.1 Experimental results

We evaluate the experimental results of the different techniques we covered (refer to Section 5.2 of [1]). The original training schedule, based on paper [21], requires approximately 80000 updates and reaches a final test accuracy of 94.3%, the "Increasing batch size" scenario requires approximately 29000 updates, reaching a final accuracy of 94.4%, while "Increased initial learning rate" increases the batch size and the learning rate by a factor of 5 compared to the previous scenario and requires under 6500 updates, reaching a final accuracy of 94.5%. Finally, "Increased momentum coefficient", including all the covered techniques, requires less than 2500 parameter updates, but reaches a lower test accuracy of 93.3%. The median accuracies were 94.3%, 94.2%, 94.2% and 93.5% respectively. By combining all the covered strategies, we are able to train Inception-ResNet-V2 on ImageNet to 77% validation accuracy in under 2500 parameter updates, using batches of 65536 images, and train ResNet-50 on ImageNet to 76.1% validation set accuracy on TPU in under 30 minutes (refer to Section 5.4 of [1]).

2 Rectified Adam, on the variance of the Adaptive Learning Rate

Adaptive learning rate variance reduction techniques, such as the learning rate warm up heuristics, are very important in stabilizing training, accelerating convergence and improving generalization for adaptive stochastic optimization algorithms like RMSprop and Adam. This is due to the problematically large variance of the adaptive learning rate in the early stages of training, due to the limited amount of the training samples being used to compute it, and it can make optimization methods converge to bad/suspicious local minima. We can address this problem with the warm up heuristic, consisting in using a small learning rate in the first few epochs of training, allowing us to reduce the initial variance of the adaptive learning rate, and preventing the gradient distribution from being distorted in the first few updates. This approach has been demonstrated to be effective in many deep learning domains such as image classification, language modelling and neural machine translation, providing further evidence of the importance of controlling the variance of the adaptive learning rate. We focus our analysis on four different variants of Adam such as Adam warm up, Adam-2k, Adam-eps and RAdam (Rectified Adam) [2].

Adam warm up The learning rate warm up strategy sets the step size α_t with smaller values in the first few steps of training. For example, linear warm up sets $\alpha_t = t\,\alpha_0$ when $t < T_w$. This technique successfully avoids the gradients distribution from being distorted in the fist few updates, while not using this approach results in a distorted distribution.

Adam-2k Adam-2k only updates the function to computes the adaptive learning rate in the first 2k iterations, while the computed momentum and network parameters are fixed. Other than this, it follows the original Adam algorithm. It prevents the gradient distribution from being distorted by gathering these additional two thousand samples for estimating the adaptive learning rate.

Adam-eps Adam-eps consists in increasing the value of ϵ in $\hat{\psi}(g_1, \cdots, g_t) = \frac{\sqrt{1-\beta_2^t}}{\epsilon + \sqrt{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} g_i^2}}$, where $\hat{\psi}(.)$ is the modified version (for numerical stability) of the adaptive learning rate function at time t. Assuming it is subject to the uniform distribution, its variance equals to $\frac{1}{12\epsilon^2}$. It prevents the gradient distribution from being distorted, but it produces much worse performance comparing to Adam-2k and Adam warm up, due to the large bias induced by the large values of ϵ.

RAdam (Rectified Adam) In RAdam [2] we assume the square of the adaptive learning rate function $\psi(.)^2 = \frac{1-\beta_2^t}{(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}g_i^2}$ subjects to a scaled inverse chi-square distribution with ρ degrees of freedom. Moreover we assume $\min_{\rho_t}[\psi(.)] = [\psi(.)]|_{\rho_t=\rho_\infty}$ and mark this minimal value as C_{var} and we know from Theorem 1 of Section 3.2 of [2] that $[\psi(.)]$ monotonically decreases as ρ increases. Thus we rectify the variance of the adaptive learning rate such that $Var[r_t\,\psi(g_1,\cdots,g_t)] = C_{\text{var}}$ where the rectification term r_t is approximated as $r_t = \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$.

2.1 Experimental results of RAdam

In this section we evaluate RAdam on One Billion Word for language modeling, Cifar10 and ImageNet for image classification. Using RAdam results in significant improvement in training behaviour, faster convergence and test performances over the vanilla Adam in all three datasets, and also demonstrates a higher robustness to wider range of learning rates compared to Adam and SGD. RAdam fails to outperform SGD in both ImageNet and CIFAR10 in terms of test accuracy, but it results in a better training performance with the final training accuracy of SGD, Adam, and RAdam on ImageNet being 69.57, 69.12 and 70.30 respectively.

3 MobileNetV2: inverted residuals and linear bottlenecks

MobileNetV2 is a neural network architecture based on depth-wise separable convolutions, linear bottlenecks and inverted residual connections, specifically tailored for resource constrained environments. Each layer module of MobileNetV2, called bottleneck residual block [3], takes as an input a linear low-dimensional subspace which is first expanded to a higher dimensional inner space by means of a point-wise convolution followed by a non-linear transformation such as ReLu6, this is successively filtered with a depth-wise convolution followed by a linear transformation followed by a non-linear transformation such as ReLu6, achieving spatial filtering of the higher dimensional tensor, and finally, the spatially filtered feature map is projected back to a low dimensional subspace of the same size of the former input with a point-wise convolution followed by a linear activation. To conclude, the inverted residual connection is introduced, summing the input feature map and the output feature map. The ratio between the size of the linear bottleneck and the inner size is called expansion ratio and is denoted by t, with typical values of t being between 5 and 10. Depth-wise separable convolutions are used instead of standard convolutions because they are computationally efficient. Assuming the set of layer activation forms a manifold of interest, and assuming this manifold of interest is a low dimensional representation of an high-dimensional space (Manifold hypothesis) we justify the use linear bottlenecks in the architecture, where linear activation are necessary in order to preserve the information. This concept have been already successfully exploited in MobileNetV1 [7].

MobileNetV2 architecture MobileNetV2 contains an initial convolution layer with 32 filters and stride 2, followed by 19 bottleneck residual blocks. The number of output channels of each bottleneck residual block increases from 16 to 320 and all spatial convolutions use 3x3 kernels and stride equal to 1 or 2, while the expansion ratio t is equal to 6 except for the first layer where it is equal to 1. Finally, we have a point-wise convolution layer, a 7x7 average pooling layer and a final point-wise convolution layer. ReLU6, dropout and batch normalization are used too. Complete details about the model architecture and model implementation can be found in Table 2 of [3].

3.1 Experimental results of MobileNetV2

In image classification on Imagenet, MobileNetV2 has similar performances and number of parameters compared to MobileNetV1 and ShuffleNet, but requires around half of the MAdds. In object detection on COCO dataset, MobileNetV2 coupled with SSDLite is highly efficient and accurate especially compared to YOLOv2, where it is around 20 times more efficient and has around 10 times fewer parameters while still having better performances. SSD300 and SSD512 have higher mAP precision but have around 8 times more parameters and the latter requires 120 times more MAdds. In semantic segmentation on VOC dataset, MobileNetV2 is used with DeepLabv3 [9] and an output stride of 16, resulting in a performance metric mIOU of 75.32, 2.11M parameters and 2.75B MAdds (compared to 80.49, 58.16 and 81.0B for ResNet-101), proving that MobileNetV2 is also a powerful light-weight feature extractor in semantic segmentation tasks.

4 Four relevant Batch Normalization improvements

Normalization techniques in deep learning are a powerful tool allowing to make neural networks more amenable to optimization, improve generalization [16], and allowing the training of very deep networks without the use of careful initialization schemes [13][14], custom non-linearities [15], or other more complicated techniques. Typical normalization techniques include Batch Normalization [10], Group Normalization [11], Layer Normalization [12], Instance Normalization [17], Weight Normalization, Decorrelated Batch Normalization, Iterative Normalization, Batch Renormalization, Switchable Normalization and Differentiable Normalization. In this section we focus on four different strategies that can be used to improve Batch Normalization:

Inference Example Weighting By default BN doesn't include the current example's value in the layer statistics during inference. As a result the output range of BN is wider during inference than during training. We fix this issue by modifying the inference behaviour of Batch Normalization layers by incorporating example statistics during inference. We compute the moving average as $\mu_i = \alpha E[x_i] + (1-\alpha)m_x$, the moving average over x^2 as $\sigma_i^2 = (\alpha E[x_i^2] + (1-\alpha)m_{x^2}) - \mu_i^2$ and $\hat{x}_i = \gamma \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta$, where α is the contribution of x_i to the normalization statistics and is an hyper parameter to tune on the validation set. This simple technique can exploit previously computed statistics, as a result we can apply it retroactively to any model trained with BN without the need of any re-training. This technique is proven to be uniformly beneficial across all models, is easy to tune and can be used retroactively to any model which uses BN. ResNet-152 on ImageNet, for example, with a proper setting of α can increase accuracy by up to 0.6%, top-5 accuracy by 0.16%, and loss by a relative 4.7% [4].

Ghost Batch Normalization for medium-batch sizes Ghost Batch Normalization can be used to reduce the generalization gap between large-batch and small-batch models and is also a powerful regularization technique in the medium-batch sizes. It consists in calculating normalization statistics on disjoint subsets of each training batch. With an overall batch size of B and a ghost batch size of B' such that B' evenly divides B, the normalization statistics for example i are calculated as $\mu_i = \frac{1}{B'} \sum_{j=1}^{B} x_j [\left\lfloor \frac{jB'}{B} \right\rfloor = \left\lfloor \frac{iB'}{B} \right\rfloor]$ $\sigma_i^2 = \frac{1}{B'} \sum_{j=1}^{B} x_j^2 [\left\lfloor \frac{jB'}{B} \right\rfloor = \left\lfloor \frac{iB'}{B} \right\rfloor] - \mu_i^2$ where $[\cdot]$ is the Iverson bracket. We can control the regularization strength of this technique by varying the value of B', with large values of B' leading to a weaker regularization effect. This approach can be combined with Inference Example Weighting, and has no additional computational costs during training compared to vanilla BN. Concretely, this approach alone was able of improving performance by 5.8% on Caltech-256 and 0.84% on CIFAR-100 [4]. On SVHN, where the baseline ResNet-18 already had a performance of 98.79%, introducing this regularization produced no change.

Weight Decay regularization of scaling and shifting parameters γ and β Weight decay [18] is a regularization technique that scales the weight of a neural network after each update step by a factor of $1 - \delta$. While applied on parameters γ and β it is proven it has an inverse effect of the effective learning rate, as shown in [19] and [20]. To be effective, weight decay on parameters γ and β need to have a connection between the layer in question and the output of the network, such that the weight decay is not undone by any other normalization layer. Residual networks are suitable for this approach, while Inception-style networks are less suitable due to the lack of shortcut connections. Concretely, weight decay on γ and β was evaluated on CIFAR-100, improving the accuracy by a small 0.3%. Moreover, we proved empirically that, even though γ has a multiplicative effect, it doesn't matter whether it is regularized to 0 or 1, but only whether weight decay is applied or not. On Caltech-256 with Inception-v3 and ResNet-50 networks, we found that for Inception-v3, incorporating weight decay on γ and β hurts the performance by 0.13%, while it improved performance for the ResNet-50 network by 0,91%. On SVHN, where the baseline ResNet-18 already had a performance of 98.79%, introducing this regularization produced no change.

Combining Batch and Group Normalization for very small-batch sizes BN is very effective in the medium to large-batch setting, but it still suffers in small-batch settings, due to the lack of enough examples to calculate reliable normalization statistics. To solve this problem we calculate normalization statistics both within groups of channels of each example and across examples in groups within each batch, proposing a variant that generalizes both Group and Batch Normalization assuming the batch size B is greater than 1. Concretely, this technique was evaluated with B = 2 and incorporating inference example weighting, and for CIFAR-100, this approach improves validation set performance over a tuned Group Normalization by 0.69% (from 73.91% to 74.60%), and on Caltech-256 performance dramatically improved by 5,0% (from 48.2% to 53.2%).

4.1 Additional experimental results

For medium to large batch sizes (B > 3), we have consistent improvements by combining Ghost Batch Normalization, Inference Example Weighing, and weight decay. For B = 2 the generalization of Batch and Group Normalization leads to most improvement, while for B = 1 weight decay is the best option. Moreover, these four techniques are shown to be applicable in the context of transfer learning, leading to consistent improvements over both group and batch normalization (see Section 4.3 of [4]).

References

[1] Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." International Conference on Learning Representations (2018).

[2] Liu, Liyuan, et al. "On the variance of the adaptive learning rate and beyond." International Conference on Learning Representations (2020).

[3] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition (2018).

[4]Summers, Cecilia, and Michael J. Dinneen. "Four Things Everyone Should Know to Improve Batch Normalization." International Conference on Learning Representations (2020).

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

[6] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In ICLR, 2018.

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017

[8] Mobilenetv2 source code. Available from https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet.

[9] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. CoRR, abs/1706.05587, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In In Proceedings of The 32nd International Conference on Machine Learning, pp. 448–456, 2015.

[11] Yuxin Wu and Kaiming He. Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19, 2018.

[12] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015.

[14] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In International Conference on Learning Representations, 2019.

[15] Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In Advances in neural information processing systems, pp. 971–980, 2017.

[16] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In International Conference on Learning Representations, 2019.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.

[18] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In Advances in neural information processing systems, pp. 950–957, 1992.

[19] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In Advances in Neural Information Processing Systems, pp. 2164–2174, 2018.

[20] Twan van Laarhoven. L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350, 2017.

[21] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.