Switchable-Shake Normalization

Sajad Norouzi
Department of Electrical
and Computer Engineering
University of Toronto
`s.norouzi@mail.utoronto.edu`

Rasa Hosseinzadeh
Department of Computer Science
University of Toronto
`rasa@cs.toronto.edu`

Saina Asani
Department of Computer Science
University of Toronto
`saina@cs.toronto.edu`

Abstract

In this paper we proposed a variant of Shake-Shake regularization called Switchable-Shake. In this method we assign a weight to each branch and use Shake-Shake to avoid overfitting. The scope of this method is not limited to multi-branch architectures and we used this method to assign weights to different normalization techniques which was inspired by Switchable-Normalization. Our experimental results show that combination of normalizations using Switchable-Shake surpasses the vanilla ResNet and Switchable-Normalization on CIFAR-10 dataset. The implementation is available at `https://github.com/csc421/shake_and_switch`

1 Introduction

Deep neural networks (DNNs) outperformed all previously proposed methods in different fields like computer vision and natural language processing. However, due to the huge number of parameters several problems might occur. Overfitting, massive computational requirements, and vanishing or exploding gradients are among challenging problems to address. Lot of methods has been proposed to mitigate these issues.

By introducing skip-connections ResNet [1] managed to avoid the vanishing and exploding gradients problem in convolutional neural networks(CNNs) which in turn made it possible to increase the number of layers. Since the introduction of ResNet, some variants have been proposed [2, 3] to improve the expressiveness power of the model.

Traditionally, L2 regularization has been used to increase the generalization power of models. Artificially generated data by augmentation tools also helps decreasing the overfitting. By introducing randomness into the training procedure of machine learning models, we can reduce overfitting. Variants of this include Stochastic Gradient Descent, Dropout [4], Batch-Normalization[5], and some types of data augmentations.

Shake-Shake is one of the recently proposed regularization techniques [6] which is designed specifically for ResNetXt[3] or any other multi-branch architecture. This method achieves stochasticity by generating random coefficients and computing noisy linear combination of different branches.

Normalization techniques are among the most widely used approaches to reduce the time and resources necessary for training DNNs. Batch normalization[5], layer normalization[7] and instance Normalization[8] are the most well-known variants of this category. Considering different limitations, each of these methods has its own set of advantages and disadvantages. By combining all these

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

techniques and allowing the network to choose among them, Switchable Normalization [9] can adapt to various situations like having small batch-size or different tasks such as classification and translation.

In this paper we have two contributions: (1) We propose a variant of Shake-Shake which allows the network to weight branches differently. (2) We apply this method to regularize switchable normalization.

2 Related Work

2.1 Normalizations

The exact merit behind normalization is still an active area of research, but based on the original batch normalization paper which was the first normalization technique applied to DNNs, normalization can address the problem of internal covariate shift. Covariate shift refers to the problem of change in the distribution of inputs of a machine learning model. In neural networks we can consider each layer as a separate learning system. So changes in the parameters of each layer cause covariate shift for all the subsequent layers. This phenomena is known as a Internal Covariate Shift.

Batch normalization computes mean and variance of every channel over each batch in the training time. For test time a moving average of batch statistics which has been computed during training is used. Batch normalization is sensitive to the batch-size and for small batch-sizes the variances of estimated values are high. Therefore, in situations that batch size is limited due to different reasons , like having high resolution input images, batch normalization performs poorly. Besides, in recurrent neural networks in order to gain improvements out of batch normalization we need to keep statistics of each time step which can be overwhelming. Additionally, due to difference of sequence lengths in the test and train phase the method might become ineffective. In order to overcome these problems other normalization methods has been proposed.

Layer normalization computes the same statistics as before but over all channels instead of the batch so it omits the dependence on the batch size. Experimental results show that it works better on RNNs[7]; however, batch normalization outperforms layer normalization on CNNs.

Instance norms [8] computes the statistics per each instance's channel which means it is neither dependent on number of channels in the layer nor on the batch size. Experimental results show that this method suits on the task of style transfer better than the approaches mentioned previously. This way of computing statistics increases the flexibility of normalization methods in a sense that each channel can be subtracted and multiplied by its own coefficients.

As it can be observed in figure 1 various normalization methods correspond to estimating data statistics over various subset of hidden layer activations, which makes each of them dependant on different parameters of the model and this makes each of them suitable for particular tasks.

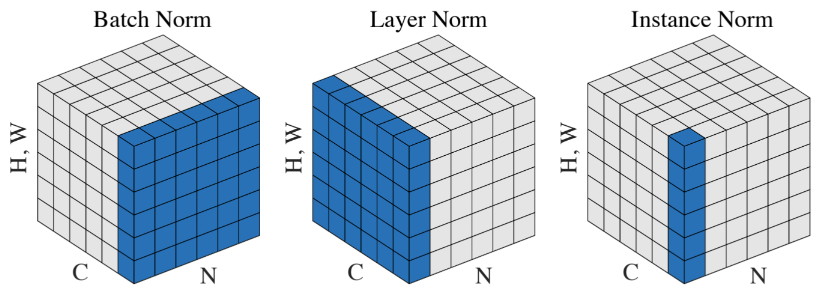

Figure 1: Visualization of various normalization methods on activation maps of a CNN [10]

2.2 Switchable Normalization

All of the discussed normalization schema can be represented as follows:

$$\hat{h} = \gamma \frac{h - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{1}$$

In which μ is the mean and σ is a variance computed over the chosen subset of activation map for each method and γ, β are learnable parameters.

As it is stated in the previous section each normalization has its own strengths and weaknesses. The aim of Switchable Normalization[9] is to come up with an adaptive normalization block which can tune itself in order to fit the current task.

Switch normalization uses the general formula 1 and computes the mean and variance as follows:

$$\begin{aligned}
&\mu_{switch} = w_1^{\mu}\mu_{batch} + w_2^{\mu}\mu_{layer} + w_3^{\mu}\mu_{instance}\\
&\sigma_{switch} = w_1^{\sigma}\sigma_{batch} + w_2^{\sigma}\sigma_{layer} + w_3^{\sigma}\sigma_{instance}\\
&w_k^p = \frac{e^{\lambda_k^p}}{\Sigma_{z \in \{in,bn,ln\}} e^{\lambda_z^p}}\\
&k \in \{in, bn, ln\}\\
&p \in \{\mu, \sigma\}
\end{aligned} \tag{2}$$

The statistics of Switchable-Normalization is a weighted combination of previous methods statistics and the weights are the outputs of applying softmax to learnable parameters. This is the same way that DARTS [11] which is an architecture search algorithm chooses different module in its search space. It is worth to mention both of these algorithms initialize their learnable parameters to equal values.

2.3 Shake-Shake Regularization

One of the recently proposed regularization techniques used for multi-branch architectures like ResNetXt [3] is called Shake-Shake. The main idea is to multiply each branch with a uniformly chosen random number such that sum of coefficients over all branches is one. For example, for a two branch architecture, it takes the following form in the forward pass:

$$\begin{aligned}
&x_{i+1} = x_i + \alpha \times \mathcal{F}_1(x_i) + (1 - \alpha) \times \mathcal{F}_2(x_i)\\
&\alpha \sim \mathcal{U}(0, 1)
\end{aligned} \tag{3}$$

A unique characteristics of this method is that it also change the backward computational graph. In the original paper three various backward graph has been used. The first one is to use the same coefficients that had been used for the forward pass. This is the correct way of computing back-propagation. The paper calls this variant as *Shake-Keep*. The backward formula for this variant is as follows:

$$\begin{aligned}
&\overline{\mathcal{F}_1(x_i)} = \overline{x_{i+1}} \times \alpha\\
&\overline{\mathcal{F}_2(x_i)} = \overline{x_{i+1}} \times (1 - \alpha)
\end{aligned} \tag{4}$$

In this notation $\overline{\mathcal{F}}$ means $\frac{\partial}{\partial \mathcal{F}} Loss$.
Another variant is to multiply each branch with a new random variable sampled from a uniform distribution similar to the one used for the forward pass. The paper refers to this variant as *Shake-Shake*. The backward formula for this variant is as follows:

$$\begin{aligned}
&\overline{\mathcal{F}_1(x_i)} = \overline{x_{i+1}} \times \beta\\
&\overline{\mathcal{F}_2(x_i)} = \overline{x_{i+1}} \times (1 - \beta)\\
&\beta \sim \mathcal{U}(0, 1)
\end{aligned} \tag{5}$$

The last variant is *Shake-Even* and it is multiplying backward pass with the expected value of the random variable of the forward pass which in this case is 0.5.

$$\begin{aligned} \overline{\mathcal{F}_1(x_i)} &= \overline{x_{i+1}} \times 0.5 \\ \overline{\mathcal{F}_2(x_i)} &= \overline{x_{i+1}} \times 0.5 \end{aligned} \tag{6}$$

Shake-Drop [12] is a generalization of Shake-Shake to single-branch ResNet-like architectures. It is basically a combination of Drop-Path [13] and Shake-Shake. It samples a Bernoulli random variable and decides whether to add noise or keep the original branch values.

3 Switchable-Shake

3.1 Proposed method

In the Shake-Shake regularization there is an assumption that the importance of different branches is the same. Therefore the expected value of random coefficients are equal. In this paper, we would like to introduce a variant of Shake-Shake which can also be applicable to scenarios that one of the branches is more suitable to the problem like what happens in Switchable normalization. In other words, we want to make the means of random coefficient distributions trainable. Following Shake-Shake we keep our distribution uniform and use the reparameterization trick similar to equation 7.

$$\alpha \sim \sigma \times \mathcal{U}(0,1) + (\mu - \sigma/2) \tag{7}$$

Although adding randomness to the network can increase the generalization power of the model, it definitely makes the training harder. So we believe in this case if we let the variance of the noise be trainable then the network makes it as small as possible. Hence, we reduce the number of parameters and use only one learnable parameter like equation 8. We call this version of Shake-Shake *Switchable-Shake*.

$$\alpha \sim 2\mu \times \mathcal{U}(0,1) \tag{8}$$

We first apply this new formulation of the noise to ResNetXt and compare it with the original Shake Shake. Then we apply this method on varied normalization layers to imitate the behavior of Switchable-Normalization. In figure 2 you can find a visualization of Switchable-Shake on normalization layer.

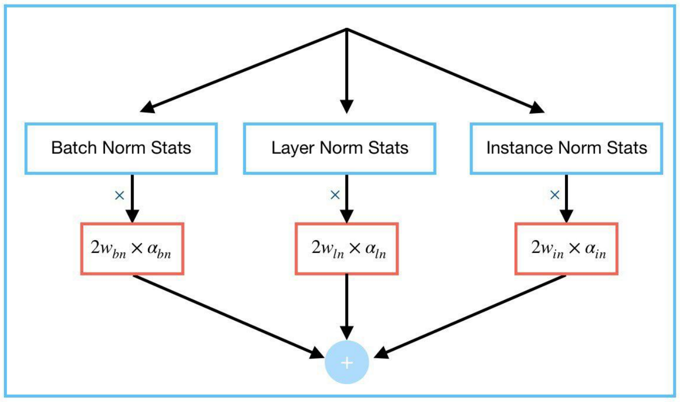
Batch Norm Stats
Layer Norm Stats
Instance Norm Stats
×
×
×
$2w_{bn} \times \alpha_{bn}$
$2w_{ln} \times \alpha_{ln}$
$2w_{in} \times \alpha_{in}$
+

Figure 2: The forward pass of Switchable-Shake

We also believe that the softmax function can decrease the effects of the noise, so we replaced the softmax in original Switchable-Normalization by normalized version of absolute values.

3.2 Lower-bound

Due to the adaptive characteristics of the parameters, the optimal normalization can be different during training. When initializing weights to random numbers, it is more likely that the normalization with higher weight becomes dominant and it causes the network to adapt to a particular normalization. By introducing a scheduled lower bound to the weights of branches, we prevent the network from getting stuck into a local optimum. Both Switch-Normalization and DARTS[11] used equal initialization of parameters to address this problem. We will evaluate the effectiveness of these methods in experimental section.

3.3 Differences of Backward Phase

The original Shake-Shake regularization utilized different parameters in forward and backward phase (Shake-Shake and Shake-Even variants). The effect of these changes is tolerable on multi branch networks since the branches in the same layer usually learn similar features and in the case of summation of branches the error signal is the same for all of them. But the normalizations used by switchable-norm (instance, layer, batch) are inherently different. This causes the Shake-Shake and Shake-Even variants to fail on this particular task and only Shake-Keep works well since the noise that was used to produce the results is kept the same in the backward phase. Our experiments show that Shake-Keep has a considerable advantage over other two and in the case of Shake-Shake the training procedure might diverge as well.

4 Experiments

First, we compared the effectiveness of Switchable-Shake on ResNetXt to that of original Shake-Shake. We have run the models with several different configurations. All the implementations in this section are based on [14]. We have changed the original code to implement our algorithms.

As discussed in section 3.2 we have used a lower-bound for weights of each branch. For testing the proposed method in this paper, experiments with and without weight lowerbound were conducted. All models are trained on CIFAR-10 for 300 epochs with an initial learning rate of 0.3, which is decreased by 0.3x after epochs 40, 90, 140, 190, and 240

When using lower-bound, both linear and exponential lower-bound decay were used. As it can be observed in table 1 the effect of using lower-bound is considerable, but the choice between linear and exponential has a marginal impact. In both lower-bound decay schema, the lower-bound were set to zero during the last 50 epochs.

Model	Top One Accuracy
Swtichable-Shake (No Lower-bound)	0.9404
Switchable-Shake + Exponential Lowerbound	0.9474
Swtichable-Shake + Linear Lower-bound with Half Variance	0.9455
Swtichable-Shake + Linear Lower-bound	**0.9498**
Original Shake-Shake	0.9497

Table 1: Comparison of various lower-bound scheduling schema on Switchable-Shake

We also reduced the variance of random noise added to the network by a factor of two, more precisely we used $w = 0.5\mu + \mu \times \mathcal{U}(0, 1)$ as the weights. The final accuracy of this model was 0.9455 which is less than original version.

Table 1 summarize accuracies of all the models. The benefits of our model is pretty marginal and it can be due to stochasticity of the results. Our explanation for not observing any significant improvement is that adding these weights to a multi-branch architectures does not increase the expressiveness of the network simply because the original networks could have imitated the new behavior by scaling weights of different branches. Our first intuition was that it is easier for the Switchable-Shake model to achieve this objective. However, experimental results showed this is not the case although with training network for longer time it might happen. It is worth to mention that Shake-Shake also shows its advantage when the network is trained for a huge number of epochs like 1800 but due to resource limitations we were unable to train longer, so we have to run both models for 1800 epochs to have a fair comparison. We leave this as a future work.

One of our main ideas was to apply Switchable-Shake to normalization layers and compare it with Switchable-Normalization. The main difference of this part with the previous one is that in a multi branch architecture, branches in the same layer are doing the same operation which is convolution with similar hyper-parameters also since we sum outputs of branches the error signals in the backward phase are the same with different coefficients, so doing shake in both the forward and backward path does not cause serious issues. When choosing among various normalization schema if the random coefficients during backward phase differ from forward the training becomes unstable. This was established by experiments as well. By running Shake-Even and Shake-Shake for a a small number of epochs it has been observed that these methods do not work on normalization layers. So all of the following experiments were done using Shake-Keep schema.

Based on the results of Switchable-Shake we concluded that lower-bound can have an observable benefits. So, in this section we have experimented with both lower-bound and equal initialization. Similar to previous part, all models are trained on CIFAR-10 for 300 epochs with an initial learning rate of 0.3, which is decayed by cosine learning rate scheduler[15]. The architecture in this section is ResNet32.

Unfortunately, the results of applying Switchable-Normalization on CIFAR-10 dataset are not available. For the sake of comparison, we took the original implementation of Switchable-Normalization in PyTorch and trained it on CIFAR-10. Besides, to increase the certainty we have implemented the algorithm in Tensorflow. Both of the implementations have substantially less scores than our models. The summarization of all results are available in table 2. All the results reported in the table are based on using multiple normalization branches rather than multi-branch architectures considered in the previous section.

Model	Top One Accuracy
ResNet32	0.9430
ResNet32 + Switchable-Normalization (Tensorflow)	0.9254
ResNet32 + Switchable-Normalization (PyTorch)	0.9297
ResNet32 + Shake-Keep	0.9395
ResNet32 + SS + Lower-bound	0.9447
ResNet32 + SS + Equal-Initialization + Variance-Weighting	**0.9491**

Table 2: Comparison of various methods on ResNet. SS stands for Switchable-Shake

4.1 Analysis

Qualitatively most of the time batch normalization has significantly higher weight, but as we go deeper in to the network, sometimes other normalization techniques get high rates as well. Especially in the last layer the dominant normalization becomes instance normalization or layer normalization. It is also observed that in some layers non of the normalization methods becomes dominant which shows none of them has advantage over the others and their combination works better.

One of the goals of normalization is to make training faster. Our hybrid model has a training curve that reaches the maximum accuracy faster than its unmixed counterparts. Most of the time during training the accuracy of hybrid model is higher than other ones 3.

5 Conclusion

By combining ideas from Shake-Shake regularization and Switchable-Normalization, we proposed a new normalization block. Empirically, we showed that this normalization technique can improve the state of the art performance on Cifar-10 dataset.

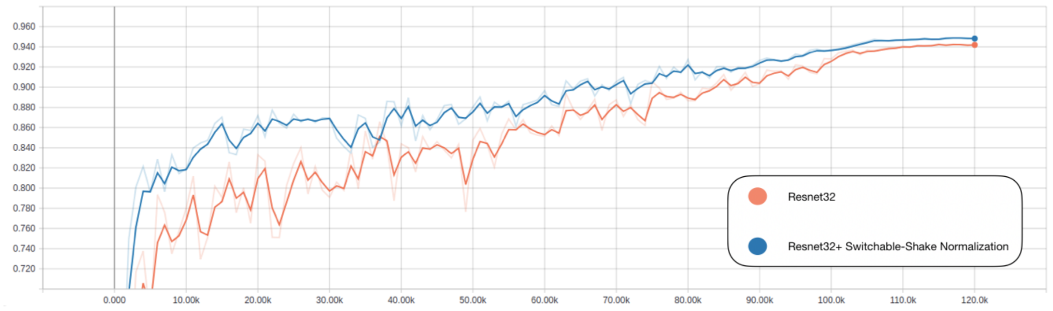

Figure 3: Learning curve of our model in comparison to learning curve of ResNet

6 Future Works

State of the art machine translation architectures are from Transformer Family. Right now, they only use layer normalization. Our method can be extended to those architectures. Our proposed regularization can also be used in differentiable architecture search algorithms like DARTS. Our method computed the normalization statistics based on previously hard-coded subsets corresponding to layer, instance and batch. One way to extend our work is to make the subset selection learnable.

7 Contributions

Task	Main Contributor
Implementation of Switchable-Shake	Sajad
Implementation of Switchable-Shake on normalizations	Rasa
Debugging Switchable-Shake implementation	Saina
Experiments on ResNetXt	Rasa
Experiments on ResNet	Sajad
Reproducing the results of original switchable using PyTorch and Tensorflow	Saina
The Main Idea	Sajad and Rasa
Lower-bound on Switchable-Shake	Saina
finding the effectiveness of shake-keep over shake-shake and shake-even on the normalizations	Rasa
Writeup	All Group Members

Table 3: List of Conributions

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[3] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

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

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

[9] Ping Luo, Jiamin Ren, and Zhanglin Peng. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018.

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

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[12] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *arXiv preprint arXiv:1802.02375*, 2018.

[13] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[14] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.