TTIC 31230 Final Project - Huanqing Wang

1 Introduction

In this report, we aim to to improve the batch normalization method proposed in the paper *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.*

Denoting $\mathbf{x} \in \mathcal{B}$ an input to batch normalization (BN) that is from a minibatch $\mathcal{B}$, batch normalization transforms $\mathbf{x}$ according to the following expression:

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}} + \boldsymbol{\beta}$$

where $\hat{\boldsymbol{\mu}}_{\mathcal{B}}$ is the sample mean and $\hat{\boldsymbol{\sigma}}_{\mathcal{B}}$ is the sample standard deviation of the mini-batch $\mathcal{B}$. We also include element-wise *scale parameter* $\boldsymbol{\gamma}$ and *shift parameter* $\boldsymbol{\beta}$ that need to be learned jointly with the other model parameters.

Throughout the experiment, we will use the data set CIFAR-10 and adopt ResNet in training and ELU as the activation function. More specifically, the structure of our network would be:

$$output = ELU(BatchNorm(Conv(input))) + input.$$

Moreover, the network consists of three stages, and each stage has number of output channels 3, 6 and 12 respectively.

2 Improvement 1: A Different Initialization

2.1 Motivation and Implementation

We first consider simplifying batch normalization without losing trainability a lot. Since each layer in the network contains numerous features, computing its standard deviation would be a lot of calculation and thus time-consuming. On the other hand, if we do not divide the standard deviation, batch normalization would lose its power of correcting extreme scaling in the feature space and possibly lose trainability. Consequently, one option could be initializing the parameter $\boldsymbol{\gamma}$ as

$$\boldsymbol{\gamma} = \frac{1}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}_{(1)}}}$$

where $\hat{\boldsymbol{\sigma}}_{\mathcal{B}_{(1)}}$ is the standard deviation of the first batch. In this way, we could transform the input $\mathbf{x}$ to

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot (\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}) + \boldsymbol{\beta}.$$

Adopting this new transformation, at least the scaling would be controlled, i.e. $\hat{var}(\mathrm{BN}(\mathbf{x})) = 1$, for the first mini-batch. More importantly, we noticed that the parameter $\boldsymbol{\gamma}$ does not change a lot across mini-batches in the original batch normalization implementation. Therefore, the value of $\boldsymbol{\gamma}$ would strongly depend on initialization and may not deviate a lot from the original paper's definition throughout training.

2.2 Empirical Results

Batch normalization does not perform well on CIFAR-10 and it could not acheive 20% or lower test error even with 3 convolutional layers. Hence, the number of layers we experimented on are chosen to be small. We conducted the experiment testing each combination of convolutional layers k and learning rate (lr) 10 times, and 3 epoches were used each time. In the summary table below, we averaged the total 30 errors from 3 epoches and 10 experiments.

First of all, we explore the effect of initialization: we replace $\boldsymbol{\gamma} = 1$ by

$$\boldsymbol{\gamma} = \frac{1}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}_{(1)}}}$$

without changing the batch normalization transformation

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}} + \boldsymbol{\beta}.$$

Although this is not a natural initialization, we could explore the effect of initialization and transformation separately and see their individual effects on trainability. Since different learning rates have different trainability, we adjusted the number of layers being experimented for each learning rate.

	k=3	k=6	k=12	k=18	k=36	k=60	k=72
Old BN Train Error	59.72	55.32	53.37	52.98	53.62	57.08	64.6
Old BN Test Error	57.11	52.83	50.57	50.9	51.16	53.15	58.92
New BN Train Error	58.97	57.01	53.24	51.64	51.99	54.83	59.2
New BN Test Error	57.24	53.31	49	49.42	48.95	53.28	55.65

Table 1: Errors at lr = 0.005 comparing Original BN and New Initialization

	k=3	k=6	k=12	k=18	k=24	k=36	k=60
Old BN Train Error	55.88	53.04	50.25	50.5	51.99	51.61	64.17
Old BN Test Error	54.38	50.7	46.64	48.19	49.65	50.44	58.73
New BN Train Error	56.82	52.99	49.65	49.46	49.33	48.49	59.86
New BN Test Error	54.92	49.02	45.58	45.42	46.93	46.46	56.89

Table 2: Errors at lr = 0.01 comparing Original BN and New Initialization

	k=3	k=12	k=18	k=24	k=30	k=42	k=51
Old BN Train Error	52.18	47.04	49.07	52.91	49.93	63.58	76.78
Old BN Test Error	52.82	44.17	45.85	49.82	47.47	59.21	74.13
New BN Train Error	52.42	48	45.48	46.24	50.39	56.99	71.67
New BN Test Error	51.25	45.23	43.26	44.08	47.31	55.76	70.7

Table 3: Errors at lr = 0.05 comparing Original BN and New Initialization

The results show that both of the train and test errors of our new initialization are comparable with the original batch normalization method. Most of the error differences are around 2%, and our new initialization is better most of the time. Admittedly, given the errors are around 50%, this improvement might not be too significant. However, we could also notice that as the number of layers, i.e. k, increases, adopting the new initialization still has stable errors and performs increasingly better, compared with the original BN.

As discussed in the motivation section, this comparable result is because $\boldsymbol{\gamma}$ does not change a lot across batches, and thus error differences between the methods become larger with bigger k's. Then we change the transformation to

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot (\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}) + \boldsymbol{\beta}.$$

	k=3	k=6	k=12	k=18	k=36
Old BN Train Error	52.18	49.47	47.04	49.07	53.62
Old BN Test Error	52.82	47.99	44.17	45.85	51.16
New BN Train Error	66.41	63.89	57.86	61.95	90
New BN Test Error	60.82	59.85	54.03	57.2	90

Table 4: Errors at lr = 0.005 comparing Original and New BN

	k=3	k=6	k=12	k=18	k=24
Old BN Train Error	52.18	49.47	47.04	49.07	51.99
Old BN Test Error	52.82	47.99	44.17	45.85	49.65
New BN Train Error	53.43	50.57	48.43	51.02	59.97
New BN Test Error	50.68	47.63	47.05	45.68	54.16

Table 5: Errors at lr = 0.01 comparing Original and New BN

	k=3	k=6	k=12	k=18	k=24
Old BN Train Error	52.18	49.47	47.04	49.07	52.91
Old BN Test Error	52.82	47.99	44.17	45.85	49.82
New BN Train Error	53.43	50.57	48.43	51.02	90
New BN Test Error	50.68	47.63	47.05	45.68	90

Table 6: Errors at lr = 0.05 comparing Original and New BN

From the above results we can see that our new transformation does not perform as well as batch normalization in general. Only when $lr = 0.1$ and k small does our new method have the same performance with batch normalization. Since the output of our new transformation still has variance proportional to the input variance, we were actually not normalizing the layer. Hence, when k is large this method would not be reliable.

3 Improvement 2: A Different Affine Transformation

3.1 Motivation and Implementation

In this section, we focus on improving the performance of batch normalization. One of the reasons of using affine transformation after the normalization is to "undo" the normalization if needed. Hence, we are actually assuming that each batch would potentially have different means and standard deviations. However, by definition $\boldsymbol{\gamma}$ do not change across batches as a parameter, and the affine transformation would imply that the output would have roughly the same mean and standard deviation. To resolve this issue, we consider another transformation: for $c_1, c_2 \in (0, 1)$,

$$\mathrm{BN}(\mathbf{x}) = \tilde{\boldsymbol{\gamma}}_{\mathcal{B}} \odot \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}} + \tilde{\boldsymbol{\beta}}_{\mathcal{B}}$$

where

$$\tilde{\boldsymbol{\gamma}} = c_1 \cdot \boldsymbol{\gamma} + (1 - c_1) \cdot \hat{\boldsymbol{\sigma}}_{\mathcal{B}}$$
$$\tilde{\boldsymbol{\beta}} = c_2 \cdot \boldsymbol{\beta} + (1 - c_2) \cdot \hat{\boldsymbol{\mu}}_{\mathcal{B}}.$$

The tuning parameters c_1 and c_2 are expected to be closed to 1, and they determine the weight of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ we put on the affine transformation. This new transformation takes each layer's own mean and standard deviation into account, but the output is mainly determined by the parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$.

3.2 Empirical Results

The experiments in this section have the same structure with experiments in section 2.2. We also experimented on different choices of c_1 and c_2.

	k=3	k=6	k=12	k=18	k=24	k=36	k=60
Old BN Train Error	55.88	53.04	50.25	50.5	51.99	51.61	64.17
Old BN Test Error	54.38	50.7	46.64	48.19	49.65	50.44	58.73
New BN Train Error	56.47	54.27	51.2	49.72	51.29	53.26	63.04
New BN Test Error	54.69	52.36	48.58	47.39	48.04	50.55	59.91

Table 7: Errors at lr = 0.01, $c_1 = c_2 = 0.8$

	k=3	k=6	k=12	k=18	k=24	k=36	k=60
Old BN Train Error	55.88	53.04	50.25	50.5	51.99	51.61	64.17
Old BN Test Error	54.38	50.7	46.64	48.19	49.65	50.44	58.73
New BN Train Error	54.93	52.76	50.72	49.08	51.1	48.74	55.56
New BN Test Error	54.37	49.8	47.59	46.83	53.51	47.17	52.95

Table 8: Errors at lr = 0.01, $c_1 = c_2 = 0.9$

	k=3	k=6	k=12	k=18	k=24	k=36	k=60
Old BN Train Error	55.88	53.04	50.25	50.5	51.99	51.61	64.17
Old BN Test Error	54.38	50.7	46.64	48.19	49.65	50.44	58.73
New BN Train Error	56.55	54.23	50.53	49.32	50.01	50.22	60.05
New BN Test Error	54.62	51.16	47.64	46.78	48.33	51	57.12

Table 9: Errors at lr = 0.01, $c_1 = c_2 = 0.95$

	k=3	k=6	k=12	k=18	k=36	k=60	k=72
Old BN Train Error	59.72	55.32	53.37	52.98	53.62	57.08	64.6
Old BN Test Error	57.11	52.83	50.57	50.9	51.16	53.15	58.92
New BN Train Error	59.09	57.4	55.3	52.67	53.03	57.61	65.73
New BN Test Error	55.11	54.41	52.17	50.75	51.26	55.18	62.62

Table 10: Errors at lr = 0.005, $c_1 = c_2 = 0.8$

	k=3	k=6	k=12	k=18	k=36	k=60	k=72
Old BN Train Error	59.72	55.32	53.37	52.98	53.62	57.08	64.6
Old BN Test Error	57.11	52.83	50.57	50.9	51.16	53.15	58.92
New BN Train Error	58.4	54.54	53.13	51.59	52.62	57.78	57.84
New BN Test Error	57.33	51.32	50.91	49.54	49.28	53.43	52.91

Table 11: Errors at lr = 0.005, $c_1 = c_2 = 0.9$

	k=3	k=6	k=12	k=18	k=36	k=60	k=72
Old BN Train Error	59.72	55.32	53.37	52.98	53.62	57.08	64.6
Old BN Test Error	57.11	52.83	50.57	50.9	51.16	53.15	58.92
New BN Train Error	60.25	55.12	53.77	53.84	52.83	56.24	60.28
New BN Test Error	56.58	52.91	50.4	50.42	49.86	53.82	55.63

Table 12: Errors at lr = 0.005, $c_1 = c_2 = 0.95$

The results above show that our new affine transformation has train and test errors similar to the original batch normalization. When $c_1 = c_2 = 0.95$, our transformation is basically batch normalization because we are not putting much weight on each layer's statistics. When c_1 and c_2 get smaller, we can see that the errors are smaller compared to the original batch normalization, but do not differ a lot.

On the other hand, we are complicating the model aiming to achieve better performance by introducing tuning parameters c_1 and c_2. Our experiments therefore may not be treated as successful. One possible reason is that we are changing the transformation universally for all the layers, but determining if the normalization should be undone might depend on each layer.

4 Conclusion and Further Work

In the previous sections, we discussed two possible improvements on batch normalization. In the first improvement, we tried to use a different initialization of $\boldsymbol{\gamma}$. Initializing a different $\boldsymbol{\gamma}$ itself yields slightly better trainability, although the initialization itself is not intuitive. This is probably because a large variance of features may cause issues, not small variance. The transformation does not work if we stop updating $\boldsymbol{\gamma}$ and use the initialized $\boldsymbol{\gamma}$ throughout training. This is probably because the output of our new transformation still has variance proportional to the input variance, so we are basically not standardizing layers.

In the second improvement, we aimed to vary the means and variances of the output layers based on their own means and variances, instead of using theoretically the same $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ across batches. This is done by transforming $\boldsymbol{\gamma}$ to a linear combination of $\boldsymbol{\gamma}$ and the layer's own standard deviation, and similarly for $\boldsymbol{\beta}$. This method does improve trainability, in particular when the tuning parameters are around 0.8 and 0.9. However, the improvement is not significant.

In terms of the improvement on initialization, we did not change the initialization of $\boldsymbol{\beta}$. Not using biases with batch normalization is often adopted in practice, and thus there would be no need bothering its initialization. In terms of using a different affine transformation, we could find some other transformation function such that it will "undo" normalization if necessary. However, determining if normalization is necessary might be difficult and sometimes data-dependent.