An Adaptive Batch Normalization in Deep Learning

Wael Alsobhi
computer science department
Umm Al-Qura University
Jamoum, Makkah, Saudi Arabia
s44280061@st.uqu.edu.sa

Tarik Alafif
computer science department
Umm Al-Qura University
Jamoum, Makkah, Saudi Arabia
tkafif@uqu.edu.sa

Alaa Abdel-Hakim
computer science department
Umm Al-Qura University
Jamoum, Makkah, Saudi Arabia
adali@uqu.edu.sa
alaa.aly@eng.au.edu.eg

Weiwei Zong
WeCare.WeTeach
Tory, Michigan, USA
wecare.weteach@gmail.com

Abstract—Batch Normalization (BN) is a way to accelerate and stabilize training in deep convolutional neural networks. However, the BN works continuously within the network structure, although some training data may not always require it. In this research work, we propose a threshold-based adaptive BN approach that separates the data that requires the BN and data that does not require it. The experimental evaluation demonstrates that proposed approach achieves better performance mostly in small batch sizes than the traditional BN using MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. It also reduces the occurrence of internal variable transformation to increase network stability.

Index Terms—Batch Normalization, Convolutional neural networks.

I. INTRODUCTION

The batch normalization (BN) is the process of normalizing data in a neural network to make it homogeneous with each other. The BN layer is introduced by Loffe and Szegedy [3]. Which, improves the classification capabilities of the deep convolutional learning model. It standardizes the data batches with the aim of reducing the internal covariate shift. This process is typically performed by adding the batch normalization layer inside the classification model in front of the input layer. Therefore, it plays its role before entering the convolutional layers of the model.

The traditional BN works continuously within the network architecture, which is applied to each input data. However, some training data may not always require the BN. The need for the BN comes from the diversity of data from the same class. Every data object in the same class must have a finite difference in its structure. The BN removes these differences in the data by smoothing out the features. Therefore, the traditional BN does not consider the relations of the data with their related classes when entering the training data.

The BN is not required for all training data entering the network architecture. An adaptivity of the BN is required to enhance the traditional BN. The adaptive method allows improving the BN performance using small batch sizes.

II. PRELIMINARIES

Loffe and Szegedy [3] suggested normalizing the input data of all constructed sub-networks. Since the normalization of the data within the network in each layer has to be aligned with the covariance matrix. This matrix finds the similarities about the correlation value between the training parameters values, and once there is a negative standardization value here, the normalization is required. Since training is often mostly performed with mini-batches, the covariance and mean value can be determined and used to normalize the activation in the network. Since it is possible that the mini-batch size is smaller than the number of parameters in the layer whose activation are to be normalized, a singular covariance matrix is generated. It is therefore proposed to use the variance $\sigma^2$ to assign the activation vectors parameter for normalization. The mean of the batch $\mu$, normalization value $\bar{x}$, and the variance $\sigma^2$ are computed as follows [3]:

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (1)$$

$$\bar{x} = x - \mu \quad (2)$$
\[ \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (\bar{x}_i)^2 \] (3)

This gives the normalized input data \( \hat{x}_i \):

\[ \hat{x}_i = \frac{\bar{x}_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \] (4)

During training, the \( \mu \) and the \( \sigma^2 \) are computed from the mini batch parts, which enter the training process to determine the best values during testing. Since the normalization is carried out before the linearity or non-linearity of the activation functions, it can lead to the fact that the input variables are only in the linear part of the function. To prevent this behavior, there are two additional learnable parameters, \( \gamma \) and \( \beta \), which are introduced to ensure numerical stability. In the last step of the batch normalization algorithm, the computed value of \( \hat{x}_i \) is shifted and scaled with the \( \gamma \) [3].

\[ BN_{\gamma_B, \beta}(\hat{x}_i) = \gamma_B \hat{x}_i + \beta \] (5)

The adjustment of the parameters is seamlessly integrated into the back-propagation algorithm:

\[ \Delta \gamma_B = -\gamma \frac{\partial E}{\partial \gamma_B} \] (6)
\[ \Delta \beta = -\gamma \frac{\partial E}{\partial \beta} \] (7)

These are most of the mathematical operations behind normalization. We note the parameters that are added in [3] to increase the stability of the normalization process.

### III. RELATED WORKS

Many research works computed BN differently. To improve the BN work, different BN techniques were proposed such as layer normalization [5], group normalization [6], and instance normalization [7] based on the computing statistics on specific dimensions of the training inputs.

Santurkar et al. [21] presented that BN was a successful technique in optimizing the training process into a stable process. It reduced the loss and enhanced the accuracy of predictions. Also, Bjorck et al. [22] concluded that the set training process parameters with optimal values such as learning rate values, would enhance the training results. On the other hand, they claimed that the primary reason for the success of BN is its ability to enable the use of higher learning rates.

Yanghao et al. [20] proposed a new method for domain adaptation called adaptive BN. They added another task in the batch layer, which added the neuron value with the normalization computation without adding a parameter. The layer standardization ensured that each layer received data from the same domain. The method proved to be effective in cloud detection of remote sensing images.

Yanghao et al. [20] proposed a new method for domain adaptation called adaptive BN. They added another task in the batch layer, which added the neuron value with the normalization computation without adding a parameter. The layer standardization ensured that each layer received data from the same domain. The method proved to be effective in cloud detection of remote sensing images.

Finally, Chai et al. [27] used key concepts from the field of traditional adaptive filtering by using the BN in-target to gain insight into the dynamics and inner workings of BN. They improved the behavior of the BN equations to help stabilize the BN work.

By reviewing the above related works, we note that all the optimized works for the BN layer work continuously without taking into account the actual need of this data in this layer. Since not all training data are required for the
BN, it can affect the work for the BN in the final performance.

IV. AN ADAPTIVE BN METHODOLOGY

Fig. 1 shows the layer of our adaptive BN architecture. The adaptive BN layer aims to increase the accuracy of the popular deep CNN image classification model by modifying the traditional BN layer. The adaptive BN layer operates the original BN functionality on and off. The upper and lower bound requirements are set based on a threshold which determines the need for using the BN.

In the first epoch, the pixel density is computed for each feature $V$. Let us represent each training steps as a vector of data.

$$V = v_1, v_2, ..., v_n$$

(8)

where each vector $v$ is a group of features $f$.

$$v = f_1, f_2, f_3, ..., f_n$$

(9)

The vector average is then recorded as $v$ and assigned to the label category based on this information. An average computation is recorded for each category of the training data.

After recording the average for each category, the upper and lower thresholds for each category are computed by adding or subtracting the pre-assumed percentage of the value of each average of the category. These thresholds represent the range for each category.

After computing the upper and lower thresholds, the second epoch of the model starts. The mean of any feature is computed to the threshold of its class. If it is greater or less than the threshold bound, the BN is run and applied since an object of the batch is out of the category range. All the training data in each subsequent epoch are passed over the computed thresholds in the first epoch.

For example, let us suppose that we have the following average list, which represents class labels for the objects dataset:

Let $Y$ is a class label for different objects such that $Y = \{Y_1, Y_2, Y_3, ..., Y_n\}$. Let $A_v$ be an average pixels colors density such that is $A_v = \{A_{v_1}, A_{v_2}, A_{v_3}, ..., A_{v_n}\}$. Then, we can compute the upper threshold as follows:

$$UpperThresholdValue = A_{v_1} + (A_{v_1} \times Up_v)$$

(10)

This means that if we have a new image from the same class, which enters the training process and obtains an average pixel density, if the average pixels are more than $UpperThresholdValue$, it requires the BN to work on it. We also have the $LowerThresholdValue$ and can be computed as follows:

$$LowerThresholdValue = A_{v_1} - (A_{v_1} \times Lp_v)$$

(11)

If it is less than $LowerThresholdValue$, it also requires the BN to work on it. In this way, a swap-in and swap-out allowance are occurred instead of using the BN continuously on all of the training images.

Algorithm of adaptive BN

Input: Image

Output: A decision for using or not using a batch normalization.

1: UPR-P ← added value from the best class threshold to form the upper BN threshold
2: LOR-P ← deducted value from the best class threshold to form the lower BN threshold
3: bestAverage ← bestAverageClassThreshold
4: upperValue ← upperValueClassThreshold
5: lowerValue ← lowerValueClassThreshold
6: procedure BEGIN PROCESS(Image: int) ▷
7: if EpochCount = 1 then
8: for all learning steps in first epoch do
9: averageFeature ← image[i][j].colorpixels
10: bestAverage = sum(averageFeature)/num(averageFeature)
11: end for
12: else
13: For all learning steps in each epoch
14: averageFeature ← image[i][j].colorpixels
15: upperValue = bestAverage * UPR - P
16: lowerValue = bestAverage * LOR - P
17: IF averageFeature > upperValue Or averageFeature < lowerValue return BN(Image)
18: end if
19: end procedure

Algorithm 1 shows the steps of the proposed methodology. The UPR-P is the added ratio to bestAverage of the class to configure the upper BN threshold. This value is directly related to the accuracy of the model. The LOR-P is the discounted ratio of the class’s bestAverage to set the lowest BN threshold. This value has a direct relationship to the model’s accuracy. The averageFeature is the average feature value of the pixel color. The average feature for the image is the sum of the feature values divided by the numbers of the feature values. The bestAverage is the average value extracted from the sum of the pixel values of the data entered the training process in the first epoch of the same class. The UpperValue is the upper threshold value after adding the UPR-P ratio to the bestAverage. The LowerValue is the lower threshold value after decreasing the LOR-P ratio from the bestAverage.

In our method, a CNN architecture and an adaptive BN layer have been added, as shown in Fig. 1. As shown in Table 1 the hybrid parameters of the CNN architecture are used. In the adaptive BN layer, we apply our main proposed algorithm.

V. EXPERIMENT AND RESULTS EVALUATIONS

In this section, we first present our experiment, the used datasets, and evaluations. Then, we present and discuss the
In the experiment, we use four popular datasets to train the CNN network. The first dataset is MNIST [36]. The MNIST dataset includes a handwritten set of numbers for 10 classes. The second dataset is Fashion-MNIST [35], which contains 70,000 images of different outfits that represent 10 categories. The third dataset is CIFAR-10 [37], which includes images from different objects representing 10 categories. The last dataset is CIFAR-100 [37], which is an expanded collection of CIFAR-10 dataset and has 100 categories.

We perform our experiment on three different scenarios:

- With BN.
- Without BN.
- With Adaptive BN.

The four datasets we used are applied to all three scenarios using four different batch sizes 4, 8, 16, and 32. The metric used to compare performance between the three different models with the four bases of evidence and the four different batch sizes is the measurement of accuracy:

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + F_P + F_N + T_N}$$ (12)

To represent this measurement, a K-fold cross-validation is used, where K is 3. Then, we compute the mean accuracy and the standard deviation for each batch size.

### A. MNIST DATASET

In Table II, the three scenarios are also performed on the MNIST dataset using the upper and lower thresholds and different batch sizes. From Table II, our adaptive based approach is superior in accuracy with the exception of the superiority of without BN scenario at a batch size 32.

| Batch size | BN       | Without BN | Adaptive BN       |
|------------|----------|------------|-------------------|
| 4          | 95.22% (+/- 0.85) | 95.32% (+/- 0.93) | **96.07% (+/- 0.09)** |
| 8          | 95.83% (+/- 0.54) | 96.30% (+/- 0.70) | **96.40% (+/- 0.21)** |
| 16         | 97.32% (+/- 0.14) | 97.37% (+/- 0.16) | **97.40% (+/- 0.24)** |
| 32         | 97.70% (+/- 0.35) | **97.98% (+/- 0.10)** | 97.84% (+/- 0.11) |

In the experiments, the percentage of batches that are required to activate the BN as shown in Fig. 2. We note a large number of data that have been passed through the training process without requiring the BN. We also note a slight increase in the percentage of batches that are required for a BN as the batch size increases.

### B. Fashion-MNIST DATASET

In Table III, the three scenarios are also performed on the Fashion-MNIST dataset with different values of upper and lower thresholds between different batch sizes. We can justify the different threshold values since the increase in the threshold values provides more space in the range of each category, thus introducing very few BN batches. If the threshold value is lowered, the average range is reduced.
Therefore, the BN is applied to a larger number of batches. Based on these experiments, the number of batches entering the BN are controlled as these thresholds are sensitive and may obviously affect the performance of the model. It can be seen that our methodology excels in accuracy, with the exception of the superiority of the BN scenario at a batch size is 32.

| Batch size | BN  | Without BN | Adaptive BN |
|------------|-----|------------|-------------|
| 4          | 80.88% (±/-1.59) | 81.11% (±/-1.16) | 81.30% (±/-0.54) |
| 8          | 83.35% (±/-0.77) | 82.11% (±/-0.31) | 83.37% (±/-0.78) |
| 16         | 83.96% (±/-1.33) | 84.59% (±/-0.93) | 84.75% (±/-0.36) |
| 32         | 86.95% (±/-0.31) | 85.67% (±/-0.37) | 85.40% (±/-0.35) |

The percentage of batches that are required for BN is computed in Fig. 3. We note a large number of data passed to training without requiring the BN. We also note a slight increase in the percentage of batches require the BN activation with increasing batch size.

C. CIFAR-10 DATASETS

In Table IV, the three scenarios are performed on the CIFAR-10 dataset with the upper and lower thresholds using different batch sizes. We can justify the different threshold values since the increase in the threshold values provides more space in the range of each category, thus introducing very few BN batches. If the threshold value is lowered, the average range is reduced. Therefore, the BN is applied to a larger number of batches. Based on these experiments, the number of batches entering the BN are controlled as these thresholds are sensitive and may obviously affect on the accuracy of the model. It can be seen that our method excels in accuracy, except for the BN scenario when the batch size is 32.

| Batch size | BN  | Without BN | Adaptive BN |
|------------|-----|------------|-------------|
| 4          | 38.70% (±/-1.93) | 46.16% (±/-1.34) | 46.72% (±/-0.56) |
| 8          | 49.42% (±/-2.65) | 50.05% (±/-1.83) | 51.15% (±/-0.44) |
| 16         | 47.75% (±/-5.61) | 54.74% (±/-0.76) | 54.94% (±/-1.06) |
| 32         | 57.98% (±/-1.68) | 57.53% (±/-1.59) | 57.77% (±/-0.97) |

The percentage of batches that are required for the BN is computed as shown in Fig. 4. We note a large number of data passed to training process without requiring the BN. We also note a slight increase in the percentage of batches that require the BN with increasing the batch size.

D. CIFAR-100 DATASETS

In Table V, the three scenarios are performed using the CIFAR-100 dataset. Upper and lower thresholds for different batch sizes are applied. It can be seen that our method is superior in accuracy using all batch sizes.

| Batch size | BN  | Without BN | Adaptive BN |
|------------|-----|------------|-------------|
| 4          | 25.50% (±/-0.12) | 26.69% (±/-1.06) | 27.12% (±/-0.27) |
| 8          | 27.18% (±/-2.02) | 28.20% (±/-0.94) | 28.71% (±/-1.33) |
| 16         | 27.68% (±/-1.48) | 28.05% (±/-0.51) | 28.28% (±/-0.02) |
| 32         | 28.89% (±/-0.12) | 30.78% (±/-0.94) | 31.13% (±/-0.16) |

The percentage of batches that are required for the BN activation is computed as shown in Fig. 5. We note a large number of data pass to training process without requiring the BN. We also note a slight increase in the percentage of batches that require the BN activation with increasing batch size.

E. Discussion

Working on our results to strengthen our theory that some, not all, training images require the BN. The BN is extracted since the data are not relevant to their categories. From Fig. 6, we see different training images from the same categories. This is what we assumed, and this is what the methodology is built on, as these images prove it. Only the
distorted images on the left column require to be normalized to make them close to their counterparts in the same category on the right column. We can also see the difference in pixel values between the images on both columns using the different thresholds. One of the main weaknesses in our methodology is determining the upper and lower bounds. The experiments find out the best threshold based on batch size, data quality, and others. Second, if there is one training image in the batch that requires to be normalized, then the entire batch are applied to the BN. Lastly, the methodology is promising and effective, but it results in higher accuracy mostly in small batch sizes.

VI. CONCLUSIONS AND FUTURE WORKS

In this research work, we propose a threshold-based adaptive BN approach that separates the data that require the BN from the data that don’t require it. The proposed approach achieves better performance mostly in small batch sizes than the traditional BN. It also reduces the occurrence of internal variable transformation to increase network stability.

In the future, enhancing the model will be based on finding a mechanism that will enable us to select the best threshold values based on the model’s behavior during execution. The improvement of the model depends on finding a way to normalize the image that requires it to be normalized from the entire batch.

REFERENCES

[1] Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. 86, 2278-2324 (1998) [http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf](http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf)

[2] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going deeper with convolutions. *2015 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*, pp. 1-9 (2015)
[30] Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2015)

[31] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going deeper with convolutions. 2015 IEEE Conference On Computer Vision And Pattern Recognition (CVPR), pp. 1-9 (2015)

[32] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference On Computer Vision And Pattern Recognition (CVPR), pp. 770-778 (2016)

[33] Huang, G., Liu, Z. & Weinberger, K. Densely Connected Convolutional Networks. CoRR. abs/1608.06993 (2016), http://arxiv.org/abs/1608.06993

[34] Yang, S., Wang, X., Yang, Q., Dong, E. & Du, S. Instance Segmentation Based on Improved Self-Adaptive Normalization. Sensors. 22 (2022), https://www.mdpi.com/1424-8220/22/12/4396

[35] Zalando, S. , “The assortment pictures on Zalando’s website. ,”[Online]. Available: http://http://www.zalando.com/

[36] LeCun, Y. and Cortes, C. , “"MNIST" handwritten digit database,” 2010 [Online]. Available: https://yann.lecun.com/exdb/mnist/

[37] Krizhevsky, A. , Nair V. and Hinton, C. , “The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset,” 2010 [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html