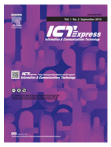
The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 6 (2020) 312–315

www.elsevier.com/locate/icte



The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset

Ibrahem Kandel[*], Mauro Castelli

Nova Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312, Lisbon, Portugal

Received 30 September 2019; received in revised form 28 March 2020; accepted 28 April 2020
Available online 5 May 2020

Abstract

Many hyperparameters have to be tuned to have a robust convolutional neural network that will be able to accurately classify images. One of the most important hyperparameters is the batch size, which is the number of images used to train a single forward and backward pass. In this study, the effect of batch size on the performance of convolutional neural networks and the impact of learning rates will be studied for image classification, specifically for medical images. To train the network faster, a VGG16 network with ImageNet weights was used in this experiment. Our results concluded that a higher batch size does not usually achieve high accuracy, and the learning rate and the optimizer used will have a significant impact as well. Lowering the learning rate and decreasing the batch size will allow the network to train better, especially in the case of fine-tuning.

Keywords: Convolutional neural networks; Deep learning; Image classification; Medical images; Batch size

1. Introduction

Since its introduction nearly two decades ago, convolutional neural networks (*CNNs*) [1] have been used as primary image classification algorithms. The true power of the CNN has been rediscovered by the ImageNet competition [2], where AlexNet architecture [3] succeeded in classifying millions of images with thousands of labels with an accuracy of 85% compared to 74% of the traditional algorithms, and that is when the CNN again became one of the most important algorithms for image classification. One of the main benefits of using a CNN is that it does not need any manual feature extraction to work, which makes it robust against new datasets. CNNs not only succeed in the image classification domain but are also successfully applied in text classification [4], climate change detection [5], and speech recognition [6], among others.

Medical images can be considered very complicated datasets because of the complexity and seriousness, and they require an experienced physician with years of experience to be able to classify the images. Examples of medical images that CNN can be applied to are histopathology images, which are images assessed by pathologists to evaluate whether tissue is cancerous. Histopathology images are very challenging to classify, even for an experienced pathologist, and that is where the CNN can be applied, either in giving a second opinion or giving assistance to the pathologist in classifying these images.

To correctly train the CNN to be able to classify images, many hyperparameters need to be adjusted; these hyperparameters will affect the performance of the network along its time to convergence. One of the main hyperparameters that need to be tuned is the batch size [7], which is the number of images used in every epoch to train the network. Setting this hyperparameter too high can make the network take too long to achieve convergence (no more gain in accuracy); however, if it is too low, it will make the network bounce back and forth without achieving acceptable performance. Also, the nature of the dataset can have an impact on the batch size, especially the medical dataset because of its complexity.

In this study, we investigated the effect of batch size on the performance of CNNs and the impact of learning rates for image classification. Two different optimizers were used to assess the impact of batch size. The CNN architecture used in this experiment was the VGG16 [8]; the network was fine-tuned to suit this dataset and to avoid training the network from

[*] Corresponding author.
E-mail address: D20181143@novaims.unl.pt (I. Kandel).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

scratch. This experimental study aims at providing a better understanding of the batch size value to be considered before addressing a given problem through a CNN. In fact, despite the importance of the batch size value for the learning process of a CNN, scientific literature only provides a few studies on this topic. Additionally, as discussed in Section 2, the results reported in the literature do not report unanimous conclusions, with some authors indicating a preference for large batch size values and other works suggesting the usage of small batch size values. The rest of the paper is organized as follows. In Section 2, previous research done on batch size is presented. In Section 3, our methodology is presented. In Sections 4 and 5, we present our results and then the conclusion.

2. Literature review

Many hyperparameters need to be adjusted before training the CNN to classify images. One of the main hyperparameters that need to be adjusted before beginning the training process is the batch size, where the batch size is the number of images that will be used in the gradient estimation process. Many researchers have studied the effect of batch size on the network performance – either the accuracy of the network or the time that was taken till convergence – to determine which was better: small batches or large batches. On one hand, a small batch size can converge faster than a large batch, but a large batch can reach optimum minima that a small batch size cannot reach. Also, a small batch size can have a significant regularization effect because of its high variance [9], but it will require a small learning rate to prevent it from overshooting the minima [10]. Below are some researches that were done to investigate the pros and cons of using small and large batch sizes.

In 2017, Radiuk [11] investigated the effect of batch size on CNN performance for image classification, the author used two datasets in the experiment, namely, MNIST and CIFAR-10 datasets. Radiuk tested batch sizes with the power of 2, starting from 16 until 1024 and 50, 100, 150, 200, and 250 as well. Radiuk opted for a LeNet architecture for the MNIST dataset and a custom-made network with five convolutional layers for the CIFAR-10 dataset. The optimizer used for both networks was the stochastic gradient descent optimizer with a learning rate of 0.001 for the MNIST and 0.0001 for the CIFAR-10 dataset. For both the datasets, the best accuracy was achieved by the 1024 batch size, and the worst result was with the 16 batch size. The author stated that based on their results, the higher the batch size the higher the network accuracy, meaning that the batch size has a huge impact on the CNN performance.

Bengio [12] stated that a batch size of 32 is a good default value, also he stated that the larger batch size will quicken the computation of the network but will decrease the updates required for the network to reach convergence. The author stated that the batch size likely impacts the convergence time and not network performance. Meanwhile, Masters and Luschi [13] tested the effect of batch sizes between 2^1 and 2^{11}, on AlexNet [3] and ResNet [14] architectures with SGD as an optimizer without momentum to exclude the effect of momentum on the training. The authors studied the effect of batch size on three datasets: CIFAR10, CIFAR100, and ImageNet. The authors stated that the best results were obtained with batch sizes between 2 and 32, and the authors noted the small batch sizes are more robust than the large batch sizes.

In general, the main question regarding the batch size is which is the optimal batch size for training CNNs that will help the network achieve the highest accuracy in the shortest time, especially for complex datasets like a medical image dataset.

3. Methodology

The training of a CNN to classify images can be defined as minimizing a non-convex loss function $L(\theta)$ by using an optimizer like a stochastic gradient descent or Adam optimizer, where $L(\theta)$ is the average cost of training image $L_i(\theta)$ over the dataset, and M is the size of the image dataset.

$$\arg\min_{\theta\in\mathbb{R}} L(\theta)\,;\, L(\theta) = \frac{1}{M}\sum_{i=1}^{M} L_i(\theta)$$

The gradient update has three options to be calculated: using the entire image dataset M, using a single image, or using a number between 1 *and* M. The previous methods are named batch gradient descent, stochastic gradient descent, and mini-batch gradient descent, respectively. Batch size hyperparameter B is the number of images used to update the gradients per time. By using the SGD optimizer, the network weights will be updated using the following equation:

$$w_{t+1} = w_t - \eta\frac{\partial L}{\partial w_t};\ \frac{\partial L}{\partial w_t} = \nabla_W C(w_t; x^{(B)}; y^{(B)})$$

where η is the learning rate, x are the sample images used, y are the image labels, and w are the weights being updated. For the Adam optimizer, the weights will be updated using the following:

$$w_t^i = w_{t-1}^i - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}.\ \hat{m}_t$$

where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t}$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2$ and $\frac{\partial L}{\partial w_t} = \nabla_W C(w_t; x^{(B)}; y^{(B)})$ where $\beta_i \in [0, 1]$ is used to determine how much information is needed from the previous update, m_t is the first momentum where it is the gradients' running average, and v_t is the second momentum where it is the squared gradients' running average. The bias-corrected first and second momentums are $\hat{m}_t$ and $\hat{v}_t$. As is shown from the previous equations, batch size and learning rate have an impact on each other, and they can have a huge impact on the network performance.

To speed up the network training and to increase its robustness, fine-tuning of the VGG16 network was applied. Fine-tuning a network is considered a method of transfer learning, where the knowledge transfer between networks that has been trained on different datasets. Because training CNN weights from scratch requires millions of images and training for days and this amount of images is not available for medical images, usage of transfer learning can be very useful in the medical field [15].

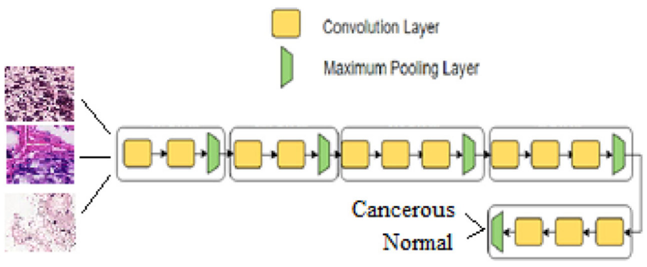


Fig. 1. VGG16 network architecture.

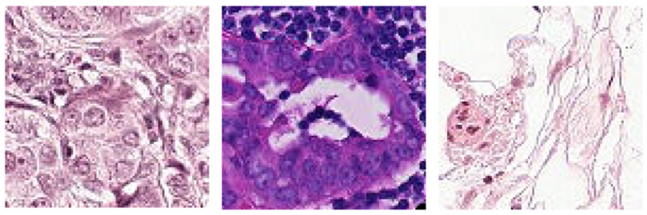


Fig. 2. A sample of the PatchCamelyon dataset.

Table 1
The results of the test AUC of the Adam optimizer.

Test AUC		
Batch size	Adam LR = 0.0001	Adam LR = 0.001
16	**0.9677**	0.9144
32	0.9636	0.9332
64	0.9616	0.9381
128	0.9567	0.9432
256	0.9585	**0.9652**

Table 2
The results of the test AUC of the SGD optimizer.

Test AUC		
Batch size	SGD LR = 0.0001	SGD LR = 0.001
16	0.9555	0.9461
32	**0.9570**	0.9521
64	0.9512	0.9545
128	0.9302	0.9567
256	0.9077	**0.9579**

The VGG16 [8] network is considered one of the most important CNNs for image classification because of its deep yet simple architecture, which gives it a robustness against overfitting while providing good performance; VGG16 is presented in Fig. 1.

The dataset used in this experiment was the PatchCamelyon [16], [17] a public dataset that contains 220,000 binary labeled images to train the CNN. The dataset was balanced, meaning it contained 60% positive to 40% negative images. Another 57,458 images were provided on the Kaggle platform to test the algorithm. All the images were 96 × 96 pixels. A sample of the dataset is presented in Fig. 2.

Image augmentation is usually used to increase the image dataset and also to make the network more robust against translation invariance. Image augmentation is defined as creating duplicates of the original image datasets by flipping, rotating, zooming, and adjusting brightness. In this work, the images were horizontally and vertically flipped, with an image rotation of 180 degrees; some images were zoomed in; and some images were shifted.

To evaluate the CNN classifier performance (i.e. to determine the classifier ability to classify positive images as positive and negative images as negative), the area under the ROC curve was used (AUC), which can be formally defined as [18]:

$$AUC = \frac{1}{2}\left(\frac{TP}{TN+FN} + \frac{TN}{TN+FP}\right)$$

where TP is the true positive metric, which is the positive images classified as positive; TN is the true negative metric, which is the negative images classified as negative; FP is the false positive metric, which is the negative images classified as positive; FN is the false negative metric, which is the positive images classified as negative. The minimum value of the AUC metric was 0.5, which represents that the model had no predictive power, and the maximum was 1, which represents that the model had perfect power in classifying images.

4. Results

The last two blocks of the VGG16 network were fine-tuned using 80% of the dataset and were validated on the remaining 20% of the dataset, after which the best model was saved and used to classify the Kaggle online test set. The batch sizes used in this experiment were $B = [16, 32, 64, 128, 256]$; two optimizers were used, namely SGD and Adam optimizers, and two learning rates were used for each optimizer of 0.001 and 0.0001. For consistency of results and due to the size of the dataset, the number of epochs was fixed to 50 epochs. To overcome overfitting, only the best model was saved, meaning that during the training phase, if the validation accuracy of the epoch was higher than the highest accuracy, then the model was saved. The results of the Kaggle online test set are shown in Tables 1 and 2.

Table 1 shows the results of the Adam optimizer with a learning rate of 0.001 and a learning rate of 0.0001. For a learning rate of 0.001, the lowest batch size (16) achieved the lowest AUC. The highest performance was from using the largest batch size (256); it can be shown that the larger the batch size, the higher the performance. For a learning rate of 0.0001, the difference was mild; however, the highest AUC was achieved by the smallest batch size (16), while the lowest AUC was achieved by the largest batch size (256).

Table 2 shows the result of the SGD optimizer with a learning rate of 0.001 and a learning rate of 0.0001. For a learning rate of 0.001, we can see that the large batch size achieved the highest AUC, while the lowest was by using the smallest batch size (16). For a learning rate of 0.0001, it was the opposite; the largest batch size (256) achieved the lowest AUC, while the 32 batch size achieved the highest followed by the lowest batch size.

The highest overall AUC achieved during the experiments was by the Adam with a learning rate of 0.0001 and batch size of 16.

Our results agree with the ones obtained by Masters and Luschi [13], where the authors stated that smaller batch sizes should be used. According to Radiuk [11], when a large learning rate is used, the higher the batch size, the better the performance of a CNN. While the use of large batch size values is not recommended in our study, the results of Radiuk match our findings on the relation between the batch size and the learning rate. In particular, we highlighted that higher learning rates require larger batch sizes. Finally, Bengio [12] suggested that 32 is a good default value for the batch size. While this is corroborated by our experiments (in which a batch size of 32 provided good results), the best performance was achieved with a batch size of 16.

5. Conclusion

Convolutional neural networks have shown superior accuracy in image classification, but to accurately train a CNN many hyperparameters need to be tuned depending on the dataset being used. The medical field can benefit greatly by using CNN in image classification to increase accuracy. In this paper, we compared the performance of CNN using different batch sizes and different learning rates. According to our results, we can conclude that the learning rate and the batch size have a significant impact on the performance of the network. There is a high correlation between the learning rate and the batch size, when the learning rates are high, the large batch size performs better than with small learning rates. We recommend choosing small batch size with low learning rate. In practical terms, to determine the optimum batch size, we recommend trying smaller batch sizes first(usually 32 or 64), also keeping in mind that small batch sizes require small learning rates. The number of batch sizes should be a power of 2 to take full advantage of the GPUs processing. Subsequently, it is possible to increase the batch size value till satisfactory results are obtained.

CRediT authorship contribution statement

Ibrahem Kandel: Investigation, Visualization, Methodology, Software, Writing - original draft. **Mauro Castelli:** Conceptualization, Supervision, Validation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), Portugal by the project GADgET (DSAIPA/DS/0022/2018) and AICE (DSAIPA/DS/0113/2019). Mauro Castelli acknowledges the financial support from the Slovenian Research Agency (research core funding No.\ P5-0410).

References

[1] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[2] O. Russakovsky, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2014).
[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Vol. 25, 2012.
[4] M. Hughes, I. Li, S. Kotoulas, T. Suzumura, Medical text classification using convolutional neural networks, Stud. Health Technol. Inform. 235 (2017).
[5] Y. Liu, et al., Application of deep convolutional neural networks for detecting extreme weather in climate datasets, 2016.
[6] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, Convolutional neural networks for speech recognition, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (10) (2014) 1533–1545.
[7] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.
[8] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014.
[9] D.R. Wilson, T.R. Martinez, The general inefficiency of batch training for gradient descent learning, Neural Netw. 16 (10) (2003) 1429–1451.
[10] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[11] P. Radiuk, Impact of training set batch size on the performance of convolutional neural networks for diverse datasets, Inf. Technol. Manag. Sci. 20 (2017).
[12] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, 2012, Arxiv.
[13] D. Masters, C. Luschi, Revisiting small batch training for deep neural networks, 2018.
[14] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, Vol. 7, 2015.
[15] N. Tajbakhsh, et al., Convolutional neural networks for medical image analysis: Full training or fine tuning?, IEEE Trans. Med. Imaging 35 (5) (2016) 1299–1312.
[16] B.S. Veeling, J. Linmans, J. Winkens, T. Cohen, M. Welling, Rotation Equivariant CNNs for Digital Pathology BT - Medical Image Computing and Computer Assisted Intervention – MICCAI 2018, 2018, pp. 210–218.
[17] B. Ehteshami Bejnordi, et al., Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast Cancermachine learning detection of breast Cancer lymph node metastasesmachine learning detection of breast Cancer lymph node metastases, JAMA 318 (22) (2017) 2199–2210.
[18] F. Idrees, M. Rajarajan, M. Conti, T.M. Chen, Y. Rahulamathavan, Pindroid: A novel Android malware detection system using ensemble learning methods, Comput. Secur. 68 (2017) 36–46.