Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 175 (2020) 135–140

Procedia Computer Science

www.elsevier.com/locate/procedia

The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 9-12, 2020, Leuven, Belgium

Method for classifying images in databases through deep convolutional networks

Noel Varela[a,*], Comas-González Zoe[b], Ternera-Muñoz Yesith R[c], Esmeral-Romero Ernesto F[d], Nelson Alberto Lizardo Zelaya[e]

[a,b,c,d] *Universidad de la Costa, Barranquilla, Colombia*
[e] *Universidad Tecnológica Centroamericana (UNITEC), San Pedro Sula, Honduras*

Abstract

Since 2006, deep structured learning, or more commonly called deep learning or hierarchical learning, has become a new area of research in machine learning. In recent years, techniques developed from deep learning research have impacted on a wide range of information and particularly image processing studies, within traditional and new fields, including key aspects of machine learning and artificial intelligence. This paper proposes an alternative scheme for training data management in CNNs, consisting of selective-adaptive data sampling. By means of experiments with the CIFAR10 database for image classification.

Keywords: Dynamic training; Deep convolutional networks; Image classification.

1. Introduction

Convolutional neural networks (CNN), have shown a high performance for classification and object detection tasks in digital images [1] and also for video and real time object detection, where two well-known algorithms have stood out due to their processing speed, these are Yolo [2] and SSD [3]. In terms of image classification or object detection, augmentation consists in applying transformations to training images, which are: rotation, mirror,

* Corresponding author. Tel.:+57-3235810446.
E-mail address: nvarela2@cuc.edu.co

10.1016/j.procs.2020.07.022

cropping, Gaussian noise, among others [4][5]. The training of a CNN is time consuming and computationally expensive as it uses large sets of training images (typically several tens of thousands of images are required). This makes difficult the use of other techniques that seek to improve the performance of CNNs or to achieve automatic network design. For example, the papers of [6] and [7] propose to carry out the optimization of the CNNs architecture for image classification tasks (see Figure 1), by using evolutionary algorithms. This is due to the fact that, initially, the number of convolutional layers, as well as the number of activation maps per layer (linked to the number of convolutional filters), are not previously defined, but must be adjusted according to the problem to be solved, as well as other user specifications, adding to the fact that it is difficult for inexperienced researchers to design and propose new architectures that will be successful.

Although the strategy of optimizing CNN's architecture through an evolutionary algorithm has been successful, its computational complexity is very high and can be prohibitive for many researchers and practitioners who do not have a sufficiently powerful computing infrastructure. For example, in [8], the author reports that the time required for the evolution of a CNN using a variant of the genetic algorithm and applied to the CIFAR101 dataset was 35 days of computation on a GPU. In the same way, [9] reports that the method described there consumes 2,750 days of computation on a GPU to evolve a network for the same CIFAR problem10 (for more data on state-of-the-art systems, see [10]).

Therefore, this paper proposes a methodology for the intelligent use of the images that are used for network training in order to reduce the training time and facilitate the use of other techniques of higher computational cost. The proposal is inspired by metaheuristic optimization or distribution estimation algorithms, but instead of sampling the solutions (in this case network architectures), a selection and a directed sampling are applied to the training data.

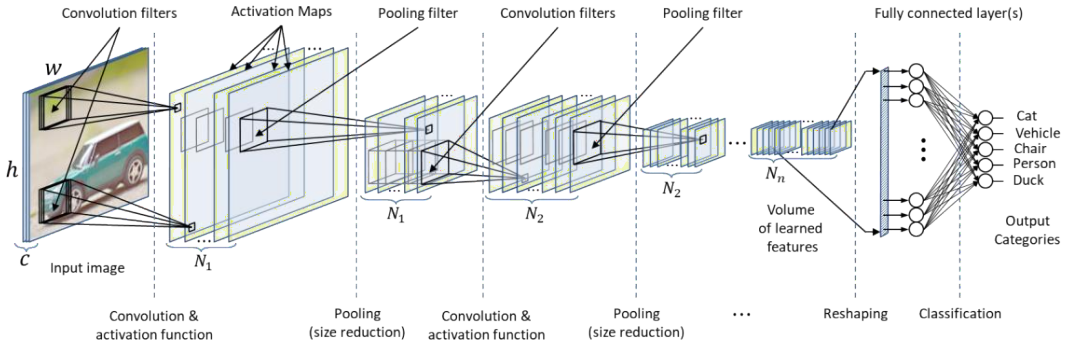

Fig. 1. Structure of a Convolutional Neuronal Network for Image Classification

2. Method

The task selected in the present study, on which the proposed method will be applied, is that of image classification on the CIFAR10 data set. This task seeks to correctly classify each image in a set composed of 50,000 training images and 10,000 test images (also called validation) distributed in 10 categories.

Figure 2 presents images belonging to the CIFAR10 repository. The choice of this dataset was made under criteria such as that it is a relatively small and well-known set of images on which many successful results have been reported in the literature [11] [12]. In other words, CIFAR10 was selected because it is a diagnostic problem, not with the aim of overcoming the performance of existing deep networks, but to show the effectiveness of the proposal in terms of achieving more efficient training of these networks.

Fig. 2. Images belonging to the CIFAR 10 repository, each column shows a selection of images corresponding to each tag.

3. Proposal

This section describes each of the steps in the proposal. It works on the assumption that using selective-adaptive image sampling at the training stage can decrease training time without significantly decreasing network performance. The parameters used by the method are the selection values (α), the resampling percentage (β) and the performance threshold (δ) [13].

1. The first step is to create a subset A of the original training image set, based on a selection value α. An immediate consequence of this selection is the creation of the subset B, made up of the images that have not been selected. In other words, a partition of the original image set is made. It is worth mentioning that the performance of the network will always be evaluated based on a validation set that is independent of any set used in the training. That is, the latter set is never used for the adjustment of the network weights.
2. The next step is done within the iterative process of training the networks. Each iteration or 'epoch' is denoted as n and a network is trained during a set of iterations $n \in [1,2,\ldots, N]$. For each epoch, the CNN is trained with the set A, and then evaluated with the validation set to find a value for network performance (correct percentage in the image classification).
3. The third step is to calculate the difference between two performance values from consecutive iterations to guide the decision to resample or not the training set. First, an evaluation is made to determine if the current performance is higher than the previous one. If not, the set A is resampled according to Equation (1), called I-type resampling, using the current set. If the current performance is greater than the previous one, it is determined if the difference exceeds the performance threshold δ, otherwise, the set A is resampled according to Equation (2), called type-II resampling, based on the best training set $A*$. Type-I resampling of the A set consists of replacing a percentage β of its elements (in this case images), selected at random, with elements from the B set, also selected at random. This can be defined as the union of $\alpha(1 - \beta) = \alpha - \alpha\beta$ elements selected from A under uniform distribution, with $\alpha\beta$ elements selected from the set B. The result of this operation will be the new set A. Equation (1) describes this process [14].

$$A \leftarrow \{A(U(a - a\beta))\} \cup \{B(U(a\beta))\}, \tag{1}$$

Type-II resampling is very similar to Type-I resampling, except that instead of selecting elements from the current A set, they are selected from the best $A*$ training set, i.e. the one that has resulted in the best performance up to the current iteration. Logically, with this method the best set matches the training set from the previous iteration. This process is described by Equation (2). In this case, the value of the percentage β is governed by Equation (3), where γ is a multiplication factor and its value is set close to 1.0, while n represents the training time and $\beta 0$ is the initial value of β. In this way, the value of β potentially decreases [15]:

$$A \leftarrow \{A * (U(a - a\beta))\} \cup \{B(U(a\beta))\}, \tag{2}$$

$$\beta \leftarrow (\gamma^2)\beta_0, \tag{3}$$

4. Experiment Design

The implementation of the proposal was done using the Python language together with the Pytorch library [16] for the creation of CNN. In order to test the effect of the different parameters (α, β and δ), a series of experiments were carried out, varying the values given to these parameters.

The experiment included a selection of $\alpha \in \{15000, 20000, 25000\}$ images to form the set A and with values of $\beta \in \{0.1, 0.2, \ldots, 0.7\}$ for the fraction or percentage of replacement and the value of γ was set at 0.97. The value of the performance threshold was kept fixed at $\delta = 0.01$ because this learning rate is a typical value observed during regular network training. From the combination of these parameter values, 21 experiments were generated, run with a Nvidia GeForce GTX 1070 GPU, and the times are reported in the results section of this device.

The CNN architecture used in the study is based on the one described in [12], where the CIFAR10 image set is also used for experimentation and good results are reported. This architecture is shown in Table 1. The hyperparameters are: Training batch size: 512, initial learning rate equal to 0.001, the chosen optimizer is Adam because it compares favorably with other optimizers [13]. The metric used for the image classification task (also used in studies like [12]) is the following:

$$Accuracy = \frac{\#Correct\ predictions}{\#total\ predictions} \tag{4}$$

Table 1. CNN Architecture.

Quantity	Layer	Size	Channels inbound	Channels outgoing	Activation Function
1	3×3 conv.	32 x 16 x 32	3	3	ReLu
3	3×3 conv.	32 x 16 x 32	3	32	ReLu
1	Max. Pooling	16 x 32 x32	32	3	-
2	3×3 conv.	16 x 32x 64	32	32	ReLu
3	3×3 conv.	16 x 16 x 64	32	32	ReLu
1	Max. Pooling	8 x 16 x 64	32	64	-
2	3×3 conv.	4 x 48x 128	64	144	ReLu
3	3×3 conv.	4 x 4 x 144	128	144	ReLu
2	Max Pooling	2 x 4 x 144	144	144	-
3	3×3 conv.	2 x 2 x144	144	144	ReLu
1	Avg. Pooling	1 x 2 x 144	144	144	-
2	**Fully Connected**	**1266**	**144**	**10**	**Softmax**

5. Results

Given the set of parameters described in the previous section, the CNN deep network training was carried out multiple times with the architecture reported in Table 1 and taking a different combination of parameters each time, for a total of 21 experiments. For each of these experiments, the maximum percentage of classification obtained by the corresponding network was recorded. The time it took the network to execute the complete training was also recorded, considering a maximum of 200 epochs. The results are reported in Table 2. Since all the nets reach 100%

classification over their respective training set, and since this set is dynamic, only the percentage of classification over the validation set is reported, since this is the real indicator of the performance of the nets.

Table 2. Classification (shown in %) obtained with different combinations of parameters

Selection	**Replacement fraction, β**						
α	0.1	0.2	0.3	0.4	0.5	0.6	0.7
25,000	81.2	82.3	83.4	83.5	83.2	81.7	84.7
20,000	81.8	81.1	83.1	82.2	81.4	81.2	82.3
15,000	**80.3**	**80.2**	**80.2**	**81.6**	**81.2**	**83.0**	**83.3**

As shown in Table 2, the highest-ranking percentage is achieved by using the largest number of images in the selection (α = 25,000) and with the largest fraction of replacement (β = 0.7). This result (highlighted using a bold font type) indicates that the performance of the network is favored by having a greater number of images for training, but also by including the greatest diversity in that set (achieved through resampling).

On the other hand, analyzing Table 3, it shows that the greater the number of images selected, the greater the time required for training. This observation makes sense and is in fact one of the main assumptions for which this proposal was designed. What is more interesting is that the parameter of the replacement fraction has virtually no effect on training time, since all the times are kept approximately constant for all the values in Table 3 on the same line.

Table 3. Training time required with different parameter combinations.

Selection	**Replacement fraction, β**						
α	0.1	0.2	0.3	0.4	0.5	0.6	0.7
25,000	33m:30s	33m:15s	34m:14s	38m:54s	33m:14s	33m:25s	33m:52s
20,000	28m:15s	28m:16s	29m:15s	28m:12s	28m:15s	28m:14s	29m:22s
15,000	**22m:21s**	**22m:14s**	**22m:33s**	**22m:31s**	**21m:35s**	**22m:22s**	**22m:02s**

Once the combination of parameters that generates the best performance has been determined, it is necessary to compare those results against the default alternatives, that is, against networks trained without the dynamic training scheme. The corresponding results are reported in Table 4.

Table 4. Summary of comparative results, regular vs. dynamic training

Training images	**Dynamic training**	**Classification accuracy**	**Time spent**
100%	No	86.47 %	51m:14s
50%	No	82.45 %	30m:58s
50%	**YES**	**83.99 %**	**31m:01s**

Finally, to check that Dynamic Training complements other techniques and maintains its relevance in the classification results, an experiment was carried out (with 20 replicates) where Dynamic Training and the data enhancement techniques are applied together. This is compared against using only data augmentation in a typical training. The results are reported in Table 5.

Table 5. Comparative results, dynamic training and data augmentation.

Pictures from Training	Enlargement of data	Dynamic Training	Accuracy of classification	Time taken
100%	Yes	No	91.88 %	24m:15s
50%	Yes	No	87.14 %	14m:45s
50%	**Yes**	**Yes**	**90.13 %**	**14m:14s**

6. Conclusions

It was observed that through the dynamic training proposal, which consists of selective-adaptive sampling of the training data, the computation time needed to train a CNN applied to the image classification task can be reduced by more than half. The results so far show that this saving in training time is achieved without negatively affecting network performance. In addition, Dynamic Training can be used together with other performance improvement techniques, such as Data Augmentation, with gains of nearly half the training time saved.

References

[1] Singh, R., Khurana, R., Kushwaha, A. K. S., & Srivastava, R. (2020). Combining CNN streams of dynamic image and depth data for action recognition. Multimedia Systems, 1-10.

[2] Mostafa, H., & Wang, X. (2019). Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. arXiv preprint arXiv:1902.05967.

[3] Viloria, A., Angulo, M. G., Kamatkar, S. J., de la Hoz – Hernandez, J., Guiliany, J. G., Bilbao, O. R., & Hernandez-P, H. (2020). Prediction Rules in E-Learning Systems Using Genetic Programming. In Smart Innovation, Systems and Technologies (Vol. 164, pp. 55–63). Springer. https://doi.org/10.1007/978-981-32-9889-7_5.

[4] Zheng, Q., Yang, M., Tian, X., Jiang, N., & Wang, D. (2020). A Full Stage Data Augmentation Method in Deep Convolutional Neural Network for Natural Image Classification. Discrete Dynamics in Nature and Society, 2020.

[5] Haque, N., Reddy, N. D., & Krishna, K. M. (2017). Joint semantic and motion segmentation for dynamic scenes using deep convolutional networks. arXiv preprint arXiv:1704.08331.

[6] Wang, H. M. X. (2019). Parameter Efficient Training of Deep Convolutional Neural Networks by Dynamic Sparse Reparameterization. arXiv preprint arXiv:1902.05967.

[7] Tang, M., Liu, Y., & Durlofsky, L. J. (2020). A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. Journal of Computational Physics, 109456.

[8] Yang, J., Liang, J., Shen, H., Wang, K., Rosin, P. L., & Yang, M. H. (2018). Dynamic match kernel with deep convolutional features for image retrieval. IEEE Transactions on Image Processing, 27(11), 5288-5302.

[9] Popov, V., Shakev, N., Ahmed, S., & Toplaov, A. (2018, September). Recognition of Dynamic Targets using a Deep Convolutional Neural Network. In ANNA'18; Advances in Neural Networks and Applications 2018 (pp. 1-6). VDE.

[10] Aimone, J. B., & Severa, W. M. (2017). Context-modulation of hippocampal dynamics and deep convolutional networks. arXiv preprint arXiv:1711.09876.

[11] Nah, S., Hyun Kim, T., & Mu Lee, K. (2017). Deep multi-scale convolutional neural network for dynamic scene deblurring. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3883-3891).

[12] Manessi, F., Rozza, A., & Manzo, M. (2020). Dynamic graph convolutional networks. Pattern Recognition, 97, 107000.

[13] Mo, S., Zhu, Y., Zabaras, N., Shi, X., & Wu, J. (2019). Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. Water Resources Research, 55(1), 703-728.

[14] Viloria, A., Varela, N., Lezama, O. B. P., Llinás, N. O., Flores, Y., Palma, H. H., … Marín-González, F. (2020). Classification of Digitized Documents Applying Neural Networks. In Lecture Notes in Electrical Engineering (Vol. 637, pp. 213–220). Springer. https://doi.org/10.1007/978-981-15-2612-1_20.

[15] Shao, R., Lan, X., & Yuen, P. C. (2017, October). Deep convolutional dynamic texture learning with adaptive channel-discriminability for 3D mask face anti-spoofing. In 2017 IEEE International Joint Conference on Biometrics (IJCB) (pp. 748-755). IEEE.

[16] Bak, C., Erdem, A., & Erdem, E. (2016). Two-stream convolutional networks for dynamic saliency prediction. arXiv preprint arXiv:1607.04730, 2(3), 6.