Influence of Dropout and Dynamic Receptive Field Operations on Convolutional Networks

Roman Nemkov
nemkov.roman@yandex.ru

Viktoria Berezina
berezinava@yandex.ru

Dmitry Mezentsev
terrakoto@me.com

Oksana Mezentseva
omezentceva@ncfu.ru

Department of Information Systems & Technologies, North-Caucasus Federal University.
2, Kulakov Prospect, Stavropol, Russian Federation

Abstract

The method and the experiments which have been performed in order to struggle with coadaptation and to improve generalization abilities of networks with the help of two techniques: dynamic receptive fields and dropout have been presented of the article. It is an effective approach for networks training. The use of the method, combining the dropout technique and dynamic receptive fields, allows to reduce the generalization error and prevents the co-adaptation of neurons.

1 The problem of coadaptation in convolutional neural networks and the proposed method for the struggle with it

The main algorithm of convolutional neural networks (CNN) training is the backpropagation (BPA) one.

As we known, weights changing happens according to the formula (1).

$$\Delta w_{ij} = -\eta \delta_j o_i \tag{1}$$

where η is learning rate, usually it is a constant, δ_j is a local gradient for j neuron, o_i is an input signal for j neuron. A weight is changed to the value which received from a multiplication of local gradient to an input value (an output value from previous layer) for this weight. In this formulation the rule is similar to Hebb's rule (the empirical regularity was found in neural networks of living things) [1].

But, the formula (1) has analytical view which doesn't take into account a network architecture absolutely. Any neural network is a particular form of a graph and therefore different optimizing techniques appeared to arise very soon for training improvement. The techniques have been based on (2).

$$\Delta w_{i_j} = -\eta \delta_j(network\_architecture) o_i(network\_architecture) \tag{2}$$

We consider $\delta_j(network\_architecture)$ firstly. Today one of the key technique for backpropagation of local gradient taking into account of network architecture is a dropout technique [2], i.e. a technique of neurons dropout during the training. The technique has come a long way since 2012. And today it is the main way for

In: S. Hölldobler, A. Malikov (eds.): Proceedings of the YSIP-3 Workshop, Stavropol and Arkhyz, Russian Federation, 17-09-2019–20-09-2019, published at http://ceur-ws.org

the struggle with overtraining for deep neural networks. There are different variants of this technique with wide range of modifications of each (dropconnects [3], dropblocks[4]).

Neural networks and especially deep ones tend to the overtraining. The dropout technique allows to obstruct this process. When we delete a part of neurons during training process we have another neural network. If we have n neurons then we can obtain 2^n networks from the original network with custom weights with a total of $O(n^2)$. From the mathematical viewpoint we can consider such training as training of 2^n sparse (partially related) networks with common weights [5].

We can imagine functioning of usual neural network as (3), (4) during forward propagation.

$$z_i^{l+1} = w_i^{l+1} y^l + b_i^{l+1} \tag{3}$$

$$y_i^{l+1} = f(z_i^{l+1}) \tag{4}$$

where z_i^{l+1} is weighted sum for i neuron and l+1 layer, w is custom weights, b is bias for neuron, y_i^{l+1} is a neuron output, $f(\cdot)$ is activation function of neuron (usually it's sigmoid function or ReLU for modern models).

The forward propagation for dropout technique changes for each input pattern according to the formulas (5) – (8):

$$r_j^l \sim Bernoulli(p) \tag{5}$$

$$\tilde{y}^l = r^l y^l \tag{6}$$

$$z_i^{l+1} = w_i^{l+1} \tilde{y}^l + b_i^{l+1} \tag{7}$$

$$y_i^{l+1} = f(z_i^{l+1}) \tag{8}$$

where r_j^l is Bernoulli random distribution for neurons of the same layer if to include them in forward propagation or not.

In the simplest case a delete technique means the neuron deletion with probability of 0.5. During the test values of weights are multiplied by vector of probability of neurons participate in the training (9):

$$W_{test}^l = pW^l \tag{9}$$

The use of such technique leads to an interesting effect. The derivative which is obtained by each parameter (local gradient) tells it how it should change in order to minimize the function of final losses (taking into account the activity of the other parameters (weights)). Therefore, weights can change correcting errors of other weights. It can lead to an excessive joint adaptation (co-adaptation), which, in turn, leads to the overtraining because these joint adaptations cannot be generalized to data that were not involved in the training.

The dropout prevents joint adaptation for each hidden parameter, making the presence of other hidden parameters unreliable. Therefore, the hidden weight can not rely on other weights correcting their own mistakes.

The trained features for hidden neurons without dropout from autoencoders for MNIST dataset [6] are shown on figure 1.a (The picture was taken from [2]). The same features which were obtained by dropout with probability of 0.5 are shown on figure 1.b.

As we can see the features on the right part of figure 1 are clear and not similar to each other. This increases the ability for invariant pattern recognition.

However, as seen from (2), the architecture change can be applied not only to a local gradient but and to an input as well. A receptive field (RF) works with an input. If we use RFs with nonstandard forms we will increase quantity of information which influences the neuron-detector setup [7].

Therefore the proposed method in that work consists in combination of these two techniques (dependent on network architectures) for two purposes: the struggle with overtraining and improving the quality of the invariant recognition. In this approach the training rule (1) will completely depend on the architectural properties of the network. And this new information embedded in the rule (2) not clearly will decrease network entropy (if an entropy is used in the view of cross-entropy in output layer) with more rapid speed than usual during the training process.

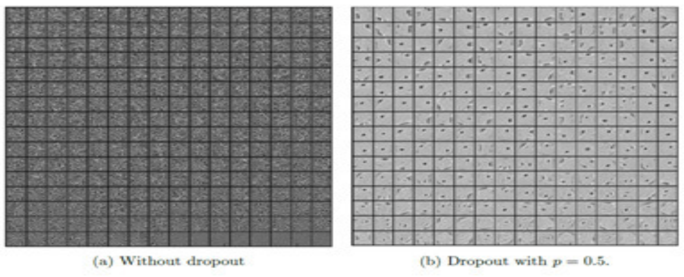

Figure 1: The trained features for MNIST for hidden layer of the autoencoder (with dropout (b) and without dropout (a))

2 Dynamic receptive fields

The idea of CNN using with dynamic RFs consists in the fact that if we change the set of RFs for some layers then the same pattern can be perceived in different ways by the network. With the help of this we can increase a training dataset. It is known that a classical form of RF is a square. We offer to use the template for obtaining RFs with a nonstandard form. The template consists of indexes which identify their neighbors within two discrete steps from the element of the index on the pixel matrix. If we change all RFs for the card (a layer consists of cards) the additional information will influence custom features and it will lead to obtaining better invariants, as we can see on figure 2.

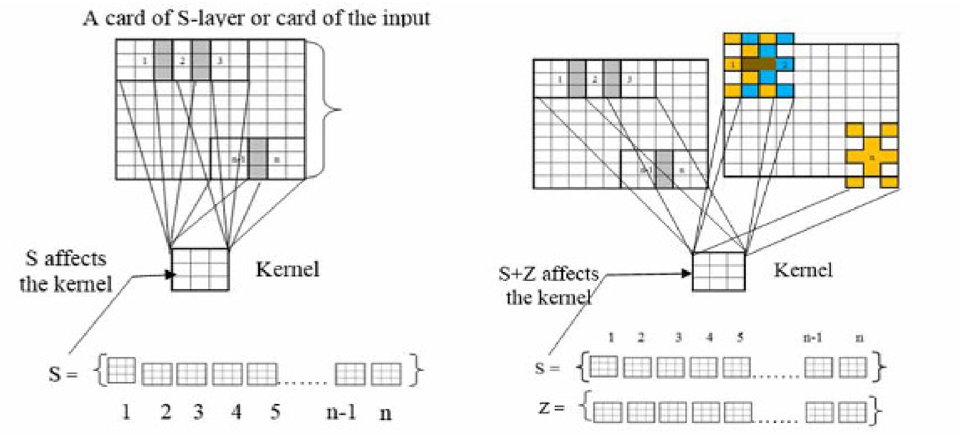

Figure 2: a) standard RFs, b) standard and nonstandard RFs

The proposed method [8, 9] adapts BPA and for forward propagation this method has the following steps:

1. During the training process before feeding the pattern we must change the standard set of RFs on the set of non-standard RFs for the required combination of the convolutional layer (C-layer) by special algorithm.
2. During forward process the output values of C-layer neurons can be obtained according to the formula 10.

$$\begin{cases} C^i_{m,n} = \varphi(b + \sum_{q\in Q_i}\sum_{k=0}^{K_C-1}\sum_{l=0}^{K_C-1}([X^q_{m+k+F_i(\cdot),n+l+F_j(\cdot)} * W^q_{k,l}] + [X^q_{m+k+F_i(\cdot),n+l+F_j(\cdot)} * A^q_{k,l}])) \\ S^i_{m,n} = \varphi(b + u\sum_{k=0}^{K_S-1}\sum_{l=0}^{K_S-1} C^i_{m*K_S+k,n*K_S+l}) \end{cases} \tag{10}$$

where $C^i_{m,n}$ is a neuron output for i Card of a C-layer in the position m and n, $\varphi(\cdot) = A * tanh(B * p)$ when $A = 1.7159$, $B = 2/3$, b is a bias, Q_i is a set of indexes of cards of a previous layer which are linked with the C^i card, K_C is a size of square RF for $C^i_{m,n}$ neuron, $X^q_{m+k,n+l}$ is input value for $C^i_{m,n}$ neuron, W and A vectors are custom weights for neurons of C-layer, $S^i_{m,n}$ is an output value of neuron for pooling; $F_i(\cdot)$ and $F_j(\cdot)$ are $F_i(RF_{m,n}; k; l)$, $F_j(RF_{m,n}; k; l)$, i.e. functions which returning the row and column offsets for the

RF template belonging to the neuron m, n at the position k, l within this template. $index_{k,l}$ is an element of template of $RF_{m,n}$ in the position k, l, $index_{k,l} = 0..24$. The functions are determined by the following formulas:

$$F_i(\cdot) = \begin{cases} 0; index_{k,l} \in \{0,4,5,16,17\} \\ 1; index_{k,l} \in \{6,7,8,18,19\} \\ 2; index_{k,l} \in \{20,21,22,23,24\} \\ -1; index_{k,l} \in \{1,2,3,14,15\} \\ -2; index_{k,l} \in \{9,10,11,12,13\} \end{cases}$$
$$F_j(\cdot) = \begin{cases} 0; index_{k,l} \in \{0,2,7,11,22\} \\ 1; index_{k,l} \in \{3,5,8,12,23\} \\ 2; index_{k,l} \in \{13,15,17,19,24\} \\ -1; index_{k,l} \in \{1,4,6,10,21\} \\ -2; index_{k,l} \in \{9,14,16,18,20\} \end{cases}, \quad (11)$$

There are no problems in combining the two techniques. After the feeding the next pattern, it is necessary to select the corresponding RFs for the neurons, and also to decide which neurons will be skipped. Details of the implementation of dynamic RFs are given in [7, 8, 9]. The details of the dropout are given in [2, 5].

3 Experiments

The experiments with the proposed method are carried out on MNIST [6]. It consists of 784 patterns, each of them is 28x28 in grayscale. The test dataset is 10 Kb, the training dataset is 60 Kb.

We have used the classical LeNet-5 architecture. The type of regularization is L2.

We have got the result of 0.95 within the test dataset without any techniques.

The results of work with the help of dropout and the proposed method of dropout and the dynamic RFs combination are shown on figure 3. The parameters of the network were taken from the similar work [8].

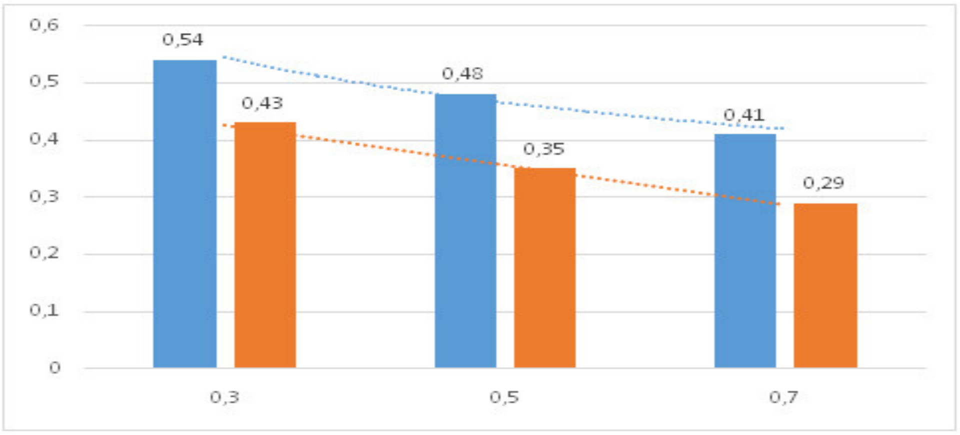

Figure 3: The results of the network performance using only dropout and dropout combined with dynamic RFs

Blue color is the result of LeNet-5 work with the help of dropout and red is the result of the combined method. A horizontal axis means the probability of neuron dropout. It is seen that the more increase of the probability of a neuron dropout the more decrease of the generalization error and the generalizing abilities of the network grow (or, equivalently, the networks committee).

4 Conclusion

The use of the method, combining the dropout technique and dynamic receptive fields, allows to reduce the generalization error and prevents the co-adaptation of neurons. In general, the architectural changes occurring in the graph of convolutional neural networks have a positive effect on the quality of invariant recognition and, in fact, correspond to the committee of networks being trained with common weights.

References

[1] Hebb D.O. The Organization of Behavior. John Wiley & Sons, New York, 1949.

[2] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580, 2012.

[3] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus. Regularization of Neural Network using DropConnect, International Conference on Machine Learning, 2013.

[4] Golnaz Ghiasi, Tsung-Yi Lin, Quoc V. Le DropBlock. A regularization method for convolutional networks, 30 Oct 2018, NIPS 2018, https://arxiv.org/pdf/1810.12890.

[5] P. Baldi, Peter Sadowski. Understanding Dropout, Advances in neural information processing systems, January 2013

[6] https://en.wikipedia.org/wiki/MNIST_database

[7] Nemkov R., Mezentsev D., Mezentseva O., Brodnikov M. Image Recognition by a Second-Order Convolutional Neural Network with Dynamic Receptive Fields, Young Scientists International Workshop on Trends in Information Processing (YSIP2). Dombai, Russian Federation, May 16-20, 2017. 212 . http://ceur-ws.org/Vol-1837/paper21.pdf.

[8] Nemkov, R. M., Mezentseva O. S. Dynamical change of the perceiving properties of convolutional neural networks and its impact on generalization. Neurocomputers: development and application, 2015, no. 2, pp.12-18.

[9] Nemkov, R. The method of a mathematical model parameters synthesis for a convolution neural network with an expanded training set. The modern problems science and education, 2015. 1. URL: http://www.science-education.ru/125-19867.