DropAll: Generalization of Two Convolutional Neural Network Regularization Methods

Xavier Frazão and Luís A. Alexandre

Dept. of Informatics, Univ. Beira Interior
and Instituto de Telecomunicações *
Covilhã, Portugal
xavierfrazao@gmail.com
lfbaa@ubi.pt
http://www.ubi.pt

Abstract. We introduce DropAll, a generalization of DropOut [1] and DropConnect [2], for regularization of fully-connected layers within convolutional neural networks. Applying these methods amounts to subsampling a neural network by dropping units. Training with DropOut, a randomly selected subset of activations are dropped, when training with DropConnect we drop a randomly subsets of weights. With DropAll we can perform both methods. We show the validity of our proposal by improving the classification error of networks trained with DropOut and DropConnect, on a common image classification dataset. To improve the classification, we also used a new method for combining networks, which was proposed in [3].

1 Introduction

Convolutional neural networks (CNNs) are hierarchical neural networks whose convolutional layers alternate with subsampling layers, reminiscent of simple and complex cells in the primary visual cortex [4]. Although these networks are efficient when performing classification, they have the disadvantage of being computationally heavy, which makes their training slow and cumbersome.

With the emergence of parallel programming and taking advantage of the processing power of Graphics Processing Units (GPUs), training these networks takes significantly less time, making it possible to train large networks [5, 6] and also making it possible to train multiple networks for the same problem and combine their results [2, 1], an approach that can significantly increase the classification accuracy.

Besides the training time, the major problem of these networks is the overfitting. Overfitting still remains a challenge to overcome when it comes to training extremely large neural networks or working in domains which offer very small amounts of data. Many regularization methods have been proposed to prevent

* We acknowledge the support given by Instituto de Telecomunicações through project PEst-OE/EEI/LA0008/2013

this problem. These methods combined with large datasets have made it possible to apply large neural networks for solving machine learning problems in several domains. Two new approachs have been recently proposed: DropOut [1] and DropConnect [2], which is a generalization of the previous. When training with DropOut, a randomly selected subset of activations is droped. With DropConnect, we randomly drop the weights. Both techniques are only possible for fully connected layers.

In this paper, we propose a generalization of both methods named DropAll. With this approach we were able to train a network with DropOut, DropConnect or both and taking advantage of each method.

2 Convolutional Neural Networks

A classical convolutional network is composed of alternating layers of convolution and pooling. The purpose of the first convolutional layer is to extract patterns found within local regions of the input images. This is done by convolving filters over the input image, computing the inner product of the filter at every location in the image and outputting the result as feature maps c. A non-linear function $f()$ is then applied to each feature map $c : a = f(c)$. The resulting activations a are passed to the pooling/subsampling layers. These layers aggregate the information within a set of small local regions, $\{R_j\}_{j=1}^{n}$, producing a pooled feature map s of smaller size as output.

Representing the aggregation function as $pool()$, then for each feature map c, we have: $s_j = pool(f(c_i)) \ \forall_i \in R_j$.

The two common choices to perform $pool()$ are average and max-pooling. The first takes the arithmetic mean of the elements in each pooling region, while max-pooling selects the largest element of the pooling region.

A range of functions $f()$ can be used as a non-linearity – $tanh$, $logistic$, $softmax$ and $relu$ are the most common choices.

In a convolutional network model, the convolutional layers, which take the pooled maps as input, can thus extract features that are increasingly invariant to local transformations of the input image.

The last layer is always a fully connected layer with one output unit per class in the recognition task. The activation function $softmax$, is the most common choice for the last layer such that each neuron output can be interpreted as the probability of a particular input image belonging to that class.

3 Related Work

3.1 Ensembles of CNNs

Model combination improves the performance of machine learning models. Averaging the predictions of several models is most helpful when the individual models are different from each other, in other words, to make them different they must have different hyperparameters or be trained on different data.

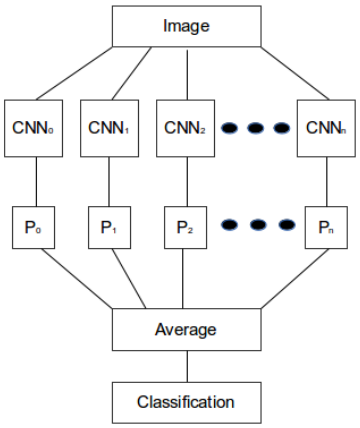

Fig. 1. The output probabilities are averaged to make the final prediction.

The standard model architecture to combine networks can be seen in figure 1. Given some input pattern, the output probabilities from all CNN are averaged before making a prediction. For output i, the average output S_i is given by:

$$S_i = \frac{1}{n} \sum_{j=1}^{n} r_j(i) \tag{1}$$

where $r_j(i)$ is the output i of network j for a given input patern.

We recently proposed a new approach to combining neural networks called Weighted Convolutional Neural Network Ensemble (WCNNE)[3] that presented better results than doing just the simple average of the predictions. This method consists in applying a different weight for each network. Networks that had a lower classification error in the validation set, will have a larger weight when combining the results. The model architecture can be seen in figure 2. Given some input pattern, the output probabilities from all CNNs are multiplied by a weight before the prediction:

$$S_i = \sum_{j=1}^{n} W_j r_j(i) \tag{2}$$

The weights W_k is choosen by rank and are based on the order of accuracy in the validation set. This means that the weights are fixed, independently on the value of the error:

$$W_k = \frac{R(A_k)}{\sum_{i=1}^{n} R(A_i)} \tag{3}$$

where $R()$ is a function that gives the position of the network based on the validation accuracy sorted in increasing order. For example, the network with

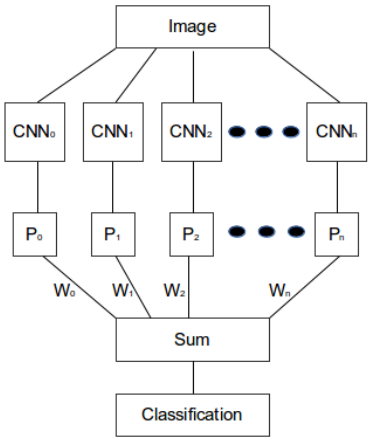

Fig. 2. The output probabilities are weighted based on the accuracy of the network evalueted on the validation set.

largest accuracy will have an $R()$ value of n, the network with the second largest accuracy an $R()$ value of $n-1$ and so on until the network with lowest accuracy gets an $R() = 1$.

This method has the particularity of not looking only at the value of the validation error, but also for the network positions in terms of the ranked error list. Even though the difference in error between the two networks might be minimal, the weight value remains fixed, attributing a significantly greater importance to the network that achieved better results in the validation set.

3.2 Regularization

Two approachs for regularizing CNNs have been recently proposed, DropOut [1] and DropConnect [2]. Applying DropOut and DropConnect amounts to subsampling a neural network by dropping units. Since each of these processes acts differently as a way to control overfitting, the combination of several of these networks can bring gains, as will be shown below.

DropOut is applied to the outputs of a fully connected layer where each element of an output layer is kept with probability p, otherwise being set to 0 with probability $(1-p)$. If we further assume a neural activation function with $a(0) = 0$, such as $tanh$ and $relu$, the output of a layer can be written as:

$$r = m * a(Wv) \tag{4}$$

where m is a binary mask vector of size d with each element j coming independently from a Bernoulli distribution $m_j \sim Bernoulli(p)$, W is a matrix with weights of a fully-connected layer and v are the fully-connected layer inputs [2].

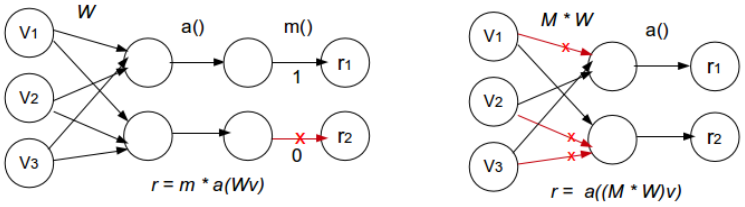

Fig. 3. The left figure is an example of DropOut. Right figure is an example of DropConnect.

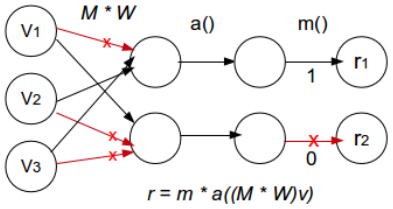

Fig. 4. Example of DropAll model.

DropConnect is similar to DropOut, but applied to the weights W. The connections are choosen randomly during the training. For a DropConnect layer, the output is given as:

$$r = a((M * W)v) \tag{5}$$

where M is weight binary mask, and $M_{ij} \sim Bernoulli(p)$. Each element of the mask M is drawn independently for each example during training [2]. Figure 3 illustrates the differences between the two methods.

4 DropAll

DropAll is a generalization of DropOut [1] and DropConnect [2], for regularizing fully-connected layers within neural deep networks. In the previous section we saw that DropOut is described by equation 4 and DropConnect is described by equation 5. For a DropAll layer, the output is given as:

$$r = m * a((M * W)v) \tag{6}$$

The DropAll model is presented graphically in figure 4. This approach has the particularity of being easily adaptable to one of the previous methods. In these two methods, we had only one variable where we choose the percentage of drops, with DropAll we have 2 variables. One variable controls the drop rate of the activation while the other variable controls the drop rate of the weight. If we

Table 1. CIFAR-10 average classification error in percentage and standard deviation using 4 types of networks and 2 types of combiners, using 64 feature maps.

Model	DropAll	DropConnect	DropOut	NoDrop
5 networks	11.20 ±0.10	11.18 ±0.15	11.28 ±0.17	10.92 ±0.15
WCNNE	10.01	9.81	10.31	10.03
Simple Average	10.03	9.84	10.48	10.06

set one of these variables to one, the drop rate value will be zero and we obtain either DropOut or DropConnect.

In both methods the value of the drop rate used is usually 0.5, however if we train with DropAll with 0.5 in both rates, network discards a lot of information, which is reflected in the results. To solve this problem, the drop rate must be smaller for both variables. When testing the network with different drop rates, we concluded that 0.25 is a good compromise.

In the following section we compare DropAll with DropConnect, DropAll and NoDrop (trained network without dropping units). All of these methods used in conjunction, provide a greater randomness when tested and combining the results from different networks trained by these techniques significantly improves the classification rates.

5 Experiments

Our experiments use a fast GPU based convolutional network library called Cuda-convnet [7] in conjunction with Li's code [1] that allows training networks with DropOut, Dropconnet and DropAll. We use a NVIDIA TESLA C2075 GPU to run the experiments. For each dataset we train five networks with DropAll, DropConnect, DropOut and NoDrop (five of each).

Once the networks are trained we save the mean and standard deviation of the classification errors produced individually by each network and the classification error produced by these networks when combined with our proposed method [3] and simple average. These results are shown in Tables 1-3. We used the CIFAR-10 dataset [8] to evaluate our approach.

5.1 CIFAR-10

The CIFAR-10 dataset [8] consists of 32 x 32 color images drawn from 10 classes split into 50 000 train and 10 000 test images.

Before feeding these images to our network, we subtract the per-pixel mean computed over the training set from each image as was done in [1]. The images are cropped to 24x24 with horizontal flips.

We use two feature extractors to perform the experiment. The first, consists in 2 convolutional layers, with 64 feature maps in each layer, 2 maxpooling layers, 2 locally connected layers, a fully connected layer which has 128 *relu*

Table 2. CIFAR-10 average classification error in percentage and standard deviation using 4 types of networks and 2 types of combiners, using, using 128 feature maps.

Model	DropAll	DropConnect	DropOut	NoDrop
5 networks	10.67 ±0.11	10.53 ±0.14	10.53 ±0.13	10.53 ±0.17
WCNNE	9.57	9.68	9.55	9.61
Simple Average	9.62	9.81	9.71	9.64

Table 3. CIFAR-10 average classification error combining our 12 best networks using 2 types of combiners, using 128 feature maps. Previous state-of-the-art using the same architecture is 9.32% [2]. Current state-of-the art of CIFAR-10 is 8.81% [9].

Model	WCNNE	Simple Average
12 networks	9.09	9.22

units on which NoDrop, DropOut, DropConnect or DropAll are applied and a output layer with $softmax$ units. We train for three stages of epochs, 500-100-100 with an initial learning rate of 0.001, that its reduced by factor 10 between each stage. We chose this fixed number of epochs because it is when the validation error stops improving. Training a network takes around 4 hours. The second feature extractor is similar but with 128 feature maps in each layer and the number of epochs is smaller, 350-100-50. Training a network with 128 maps takes around 20 hours. In these experiments we compared the results using our approach (WCNNE) [3] for combining networks and simple average, both described in this paper.

The first experiment used a feature extractor with 64 feature maps (summarized in Table 1) and combined networks that were trained with DropAll, DropConnect, DropOut and NoDrop. NoDrop individually obtained better results and networks with DropOut were the ones with the worst individual results. By combining the nets, DropConnect achieved better results.

The second experiment used a feature extractor with 128 feature maps (summarized in Table 2), we also combine networks that were trained with DropAll, DropConnect, DropOut and NoDrop. DropOut individually achieved better results, and networks trained with DropAll were the ones with worst result. By combining the nets, DropOut achieved better results.

In addition we join all models and combine our 12 best networks with lowest validation error, and the results were significantly better (see Table 3). All of these methods used in conjunction provide a greater randomness and significantly improve the classification rate. If we combine our 12 best networks without DropAll networks the error is slightly worse, 9.12%.

6 Conclusions

In this paper, we propose a new method named DropAll that is a generalization of two well-known methods for regularization of convolutional neural networks, used to avoid overfitting. These problem still remains a challenge to overcome when it comes to training extremely large neural networks or working in domains which offer very small amounts of data.

DropAll by itself, did not increase performance when we evaluate a network, however, the flexibility of this method makes it possible to train a network using the potential of DropOut and DropConnect. In general, networks trained with these forms of regularization benefit from an increase randomness, which is a plus when we wish to combine the results of multiple networks. As shown, the combination of all methods significantly improves the classification rate of the problem used in the experiments section to validate our proposal.

References

1. Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.
2. Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Sanjoy Dasgupta and David Mcallester, Eds. May 2013, vol. 28, pp. 1058–1066, JMLR Workshop and Conference Proceedings.
3. Xavier Frazao and Luís A. Alexandre, "Weighted convolutional neural network ensemble," in *submitted*, 2014.
4. Kunihiko Fukushima, "A neural network model for selective attention in visual pattern recognition," *Biol. Cybern.*, vol. 55, no. 1, pp. 5–16, Oct. 1986.
5. K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 2146–2153.
6. Kumar Chellapilla, Sidd Puri, and Patrice Simard, "High Performance Convolutional Neural Networks for Document Processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, Guy Lorette, Ed., La Baule (France), Oct. 2006, Université de Rennes 1, Suvisoft, http://www.suvisoft.com Université de Rennes 1.
7. Alex Krizhevsky, "Cuda-convnet," `http://code.google.com/p/cuda-convnet/`, 2012.
8. Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
9. Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.