# Weighted Convolutional Neural Network Ensemble

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 a new method to combine the output probabilities of convolutional neural networks which we call Weighted Convolutional Neural Network Ensemble. Each network has an associated weight that makes networks with better performance have a greater influence at the time to classify in relation to networks that performed worse. This new approach produces better results than the common method that combines the networks doing just the average of the output probabilities to make the predictions. We show the validity of our proposal by improving the classification rate on a common image classification benchmark.

**Keywords:** Convolutional Neural Networks, Object Recognition, Network Ensemble.

## 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 [1]. 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 [2, 3] and also making it possible to train multiple networks for the same problem and combine their results [4, 5], 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

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

extremely large neural networks or working in domains which offer very small amounts of data. Many regularization methods have also been proposed to prevent this problem. These methods combined with large datasets have made it possible to apply neural large networks for solving machine learning problems in several domains. Two new approachs have been recently proposed, DropOut [5] and DropConnect [4], which is a generalization of the previous. When training with DropOut, a randomly selected subset of activations are droped. With DropConnect, we randomly drop the weights. Both techniques are only possible for fully connected layers. Combining the results from different networks trained by these techniques significantly improves the classification rate. In this paper, we propose a new method to combine the results of networks by applying a different weight for each network, instead of using the common method, that involves averaging the output probabilities of several networks.

## 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 result 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 activation 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

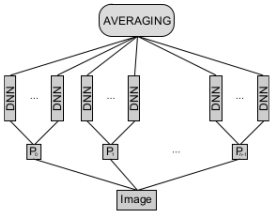


**Fig. 1.** MCDNN architecture. DNNs are trained on inputs preprocessed in different ways. The final prediction is a simple average of all DNNs predictions.

models are different from each other, in other words, to make them different they must have different hyperparameters or be trained on different data.

An example of the combination of multiples CNNs appears in [6], as the "Multi-column Deep Neural Networks for Image Classification"(MCDNN). In that model, the input image is preprocessed by blocks. The dataset is preprocessed before training, then, at the beginning of every epoch, the images are distorted (block). An arbitrary number of CNNs can be trained on inputs preprocessed in different ways. The final predictions are obtained by averaging individual predictions of each CNN. Fig. 1 shows the architecture of a MCDNN.

### 3.2 Regularization

Two new approachs for regularizing CNNs have been recently proposed, DropOut [5] and DropConnect [4]. 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:

$$r = m * a(Wv) \tag{1}$$

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 [4].

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:

$$r = a((M * W)v) \tag{2}$$

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 [4]. Fig. 2 illustrates the differences between the two methods.

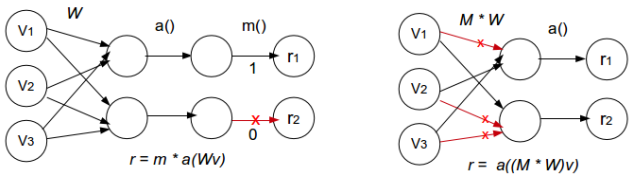


**Fig. 2.** The left figure is an example of DropOut. Right figure is an example of DropConnect.

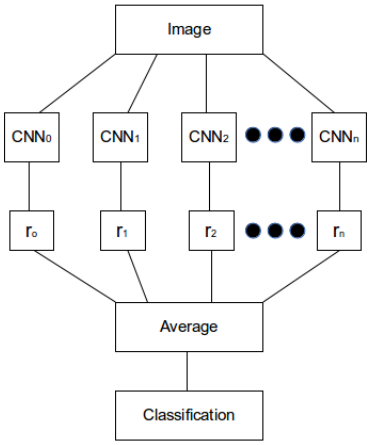


**Fig. 3.** The output probabilities $r$ are averaged to make the final prediction.

## 4 Model Description

The standard model architecture to combine networks can be seen in Fig. 3. 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:

$$S_i = \frac{1}{n} \sum_{j=1}^{n} r_j(i) \tag{3}$$

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

Our approach consists in applying a different weight for each network. In the validation set, networks that had a lower classification error will have a larger weight when combining the results. The model architecture can be seen in Fig. 4. Given some input pattern, the output probabilities from all CNNs are multiplied by a weight $\alpha$ before the prediction:

$$S_i = \sum_{j=1}^{n} \alpha_j r_j(i) \tag{4}$$

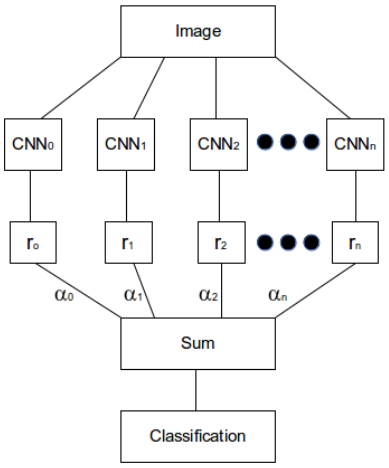


**Fig. 4.** The output probabilities $r$ are weighted based on the accuracy of the network evaluated on the validation set.

We use two different approaches to calculate the weight $\alpha$. The first method consists on a weighted mean:

$$\alpha_k = \frac{A_k}{\sum_{i=1}^{n} A_i} \tag{5}$$

where $A_k$ is the accuracy in the validation set for the network $k$, and $i$ runs over the $n$ networks.

In the second method, the weight $\alpha_k$ is chosen by rank. The weights 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:

$$\alpha_k = \frac{R(A_k)}{\sum_{i=1}^{n} R(A_i)} \tag{6}$$

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 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 $R() = 1$.

The first method has weights that are proportional to the accuracy in validation set. If the errors are close, the weights have values close to each other, and the result will be similar to a simple average.

The second method has the particularity of not looking only at the value of the validation error, but at the network positions. 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.

**Table 1.** CIFAR-10 average classification error in percentage and standard deviation using 3 types of networks and 3 types of combiners, using 64 feature maps.

| Model | DropConnect | DropOut | NoDrop |
|---|---|---|---|
| 5 networks | 11.18 ±0.15 | 11.28 ±0.17 | 10.92 ±0.15 |
| Method 1 | 9.81 | 10.45 | 10.06 |
| Method 2 | 9.81 | 10.31 | 10.03 |
| Simple Average | 9.84 | 10.48 | 10.06 |

**Table 2.** CIFAR-10 average classification error in percentage and standard deviation using 3 types of combiners, using 64 feature maps. Previous state-of-the-art using the same architecture is 9.32% [4].

| Model | Method 1 | Method 2 | Simple Average |
|---|---|---|---|
| 12 networks | 9.50 | 9.37 | 9.47 |

In general, both methods give better results than doing just the simple average of the predictions, as will be shown in the results.

## 5 Experiments

Our experiments use a fast GPU based convolutional network library called Cuda-convnet [7] in conjunction with Li's code [4] that allows training networks with DropOut and Dropconnet. We use a NVIDIA TESLA C2075 GPU to run the experiments. For each dataset we train five networks with 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 two proposed methods methods and simple average. These results are shown in Tables 1-4. We use 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 [5]. 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 $relu$ units on which NoDrop, DropOut or DropConnect are applied and a output layer with $softmax$ units. We train for three stages of epochs, 500-100-100 with

**Table 3.** CIFAR-10 average classification error in percentage and standard deviation using 3 types of networks and 3 types of combiners, using 128 feature maps.

| Model | DropConnect | DropOut | NoDrop |
|---|---|---|---|
| 5 networks | 10.53 ±0.14 | 10.53 ±0.13 | 10.53 ±0.17 |
| Method 1 | 9.77 | 9.68 | 9.63 |
| Method 2 | 9.68 | 9.55 | 9.61 |
| Simple Average | 9.81 | 9.71 | 9.64 |

**Table 4.** CIFAR-10 average classification error in percentage and standard deviation using 3 types of combiners, using 128 feature maps. Previous state-of-the-art using the same architecture is 9.32% [4].

| Model | Method 1 | Method 2 | Simple Average |
|---|---|---|---|
| 12 networks | 9.20 | 9.12 | 9.22 |

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 the three approaches for combining networks 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 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. In addition we combine our 12 best networks (see Table 2) and the results were significantly better than the result shown in Table 1. With method 2 we achieved an error of 9.37%.

The second experiment used a feature extractor with 128 feature maps (summarized in Table 3), we also combine networks that were trained with DropConnect, DropOut and NoDrop. DropOut individually achieved better results, and networks trained with DropConnect were the ones with worst result. When combining our 12 best networks, we obtained better results, as can be seen in Table 4. Using our second method we obtain 9.12%, improving the classification rate of CIFAR-10.

## 6 Conclusions

In this paper, we propose two methods to combine the results of CNNs by applying different weights for each network, instead of simply averaging the output predictions of several networks.

The experiments showed, that applying a weight to each network based on performance ranking (method 2), ensures better results than the other two approaches. We also conclude that when combining a large number of networks, the weighted mean (method 1) performance was similar to the simple average. We also concluded that a feature extractor 128 feature maps achieves better results than a feature extractor with 64 feature maps. However when using 128 maps we have to use less epochs because the network tends to overfit, losing its ability to generalize. We also tested more maps (not presented in results), but the networks overfitted very quick, producing worst results. At the end, with one of our methods, we improved the classification rate of CIFAR-10.

In summary, we demonstrate with our contribution that there are better alternatives to combine the results than doing just the simple average of the networks predictions.

## References

1. Kunihiko Fukushima, "A neural network model for selective attention in visual pattern recognition," *Biol. Cybern.*, vol. 55, no. 1, pp. 5–16, Oct. 1986.
2. 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.
3. 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.
4. 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.
5. 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.
6. Jurgen Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2012, CVPR '12, pp. 3642–3649, IEEE Computer Society.
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.