Neural Networks 71 (2015) 1–10

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Towards dropout training for convolutional neural networks

Haibing Wu, Xiaodong Gu *

Department of Electronic Engineering, Fudan University, Shanghai 200433, China

ARTICLE INFO

Article history:
Received 14 January 2015
Received in revised form 19 June 2015
Accepted 16 July 2015
Available online 29 July 2015

Keywords:
Deep learning
Convolutional neural networks
Max-pooling dropout

ABSTRACT

Recently, dropout has seen increasing use in deep learning. For deep convolutional neural networks, dropout is known to work well in fully-connected layers. However, its effect in convolutional and pooling layers is still not clear. This paper demonstrates that *max-pooling dropout* is equivalent to randomly picking activation based on a multinomial distribution at training time. In light of this insight, we advocate employing our proposed *probabilistic weighted pooling*, instead of commonly used max-pooling, to act as model averaging at test time. Empirical evidence validates the superiority of probabilistic weighted pooling. We also empirically show that the effect of convolutional dropout is not trivial, despite the dramatically reduced possibility of over-fitting due to the convolutional architecture. Elaborately designing dropout training simultaneously in max-pooling and fully-connected layers, we achieve state-of-the-art performance on MNIST, and very competitive results on CIFAR-10 and CIFAR-100, relative to other approaches without data augmentation. Finally, we compare max-pooling dropout and stochastic pooling, both of which introduce stochasticity based on multinomial distributions at pooling stage.

1. Introduction

Deep convolutional neural networks (CNNs) have recently told many success stories in visual recognition tasks and are now record holders on many challenging datasets. A standard CNN consists of alternating convolutional and pooling layers, with fully-connected layers on top. Compared to regular feed-forward networks with similarly-sized layers, CNNs have much fewer connections and parameters due to the local-connectivity and shared-filter architecture in convolutional layers, so they are far less prone to over-fitting. Another nice property of CNNs is that pooling operation provides a form of translation invariance and thus benefits generalization. Despite these attractive qualities and despite the fact that CNNs are much easier to train than other regular, deep, feed-forward neural networks, big CNNs with millions or billions of parameters still easily overfit relatively small training data.

Dropout (Hinton, Srivastave, Krizhevsky, Sutskever, & Salakhutdinov, 2012) is a recently proposed regularizer to fight against over-fitting. It is a regularization method that stochastically sets to zero the activations of hidden units for each training case at training time. This breaks up co-adaption of feature detectors since the dropped-out units cannot influence other retained units. Another way to interpret dropout is that it yields a very efficient form of model averaging where the number of trained models is exponential in that of units, and these models share the same parameters. Dropout has also inspired other stochastic model averaging methods such as stochastic pooling (Zeiler & Fergus, 2013) and DropConnect (Wan, Zeiler, Zhang, LeCun, & Fergus, 2013).

Although dropout is known to work well in fully-connected layers of convolutional neural nets (Hinton et al., 2012; Krizhevsky, Sutskever, & Hinton, 2012; Wan et al., 2013), its effect in convolutional and pooling layers is, however, not well studied. This paper shows that using *max-pooling dropout* at training time is equivalent to sampling activation based on a multinomial distribution, and the distribution has a tunable parameter p (the retaining probability). In light of this, *probabilistic weighted pooling* is proposed and employed at test time to efficiently average all possibly max-pooling dropout trained networks. Our empirical evidence confirms the superiority of probabilistic weighted pooling over max-pooling. Like *fully-connected dropout*, the number of possible max-pooling dropout models also grows exponentially with the increase of the number of hidden units that are fed into pooling layers, but decreases with the increase of pooling region's size. We also empirically show that the effect of *convolutional dropout* is not trivial, despite the dramatically reduced possibility of over-fitting due to the convolutional architecture. Carefully designing dropout training simultaneously in max-pooling and fully-connected layers, we report state-of-the-art results on MNIST, and very competitive results on CIFAR-10 and CIFAR-100, in comparisons with other approaches without data augmentation.

* Corresponding author.
E-mail addresses: haibingwu13@fudan.edu.cn (H. Wu), xdgu@fudan.edu.cn (X. Gu).

http://dx.doi.org/10.1016/j.neunet.2015.07.007
0893-6080/

As both stochastic pooling (Zeiler & Fergus, 2013) and max-pooling dropout randomly sample activation based on multinomial distributions at pooling stage, it becomes interesting to compare their performance. Experimental results show that stochastic pooling performs between max-pooling dropout with different retaining probabilities, yet max-pooling dropout with typical retaining probabilities often outperforms stochastic pooling by a large margin.

In this paper, dropout on the input to max-pooling layers is also called max-pooling dropout for brevity. Similarly, dropout on the input to convolutional (or fully-connected) layers is called convolutional (or fully-connected) dropout.

2. Review of dropout training for convolutional neural networks

CNNs have far been known to produce remarkable performance on MNIST (LeCun, Bottou, Bengio, & Haffner, 1998), but they, together with other neural network models, fell out of favor in practical machine learning as simpler models such as SVMs became the popular choices in the 1990s and 2000s. With deep learning renaissance (Bengio, Courville, & Vincent, 2013; Ciresan, Meier, & Schmidhuber, 2012; Hinton & Salakhutdinov, 2006), CNNs regained attentions from machine learning and computer vision community. Like other deep models, many issues can arise with deep CNNs if they are naively trained. Two main issues are computation time and over-fitting. Regarding the former problem, GPUs help a lot by speeding up computation significantly.

To combat over-fitting, a wide range of regularization techniques have been developed. A simple but effective method is adding l_2 penalty to the network weights. Other common forms of regularization include early stopping, Bayesian fitting (Mackay, 1995), weight elimination (Ledoux & Talagrand, 1991) and data augmentation. In practice, employing these techniques when training big neural networks provides better test performances than smaller networks trained without any regularization.

Dropout is a new regularization technique that has been more recently employed in deep learning. It is similar to bagging (Breiman, 1996), in which a set of models are trained on different subsets of the same training data. At test time, different models' predictions are averaged together. In traditional bagging, each model has independent parameters, and all members would be trained explicitly. In the case of dropout training, there are exponentially many possibly trained models, and these models share the same parameters, but not all of them are explicitly trained. Actually, the number of explicitly trained models is not larger than $m \times e$, where m is the number of training examples, and e is the training epochs. This is much smaller than the number of possibly trained models, 2^n (n is the number of hidden units in a feed-forward neural network). Therefore, a vast majority of models are not explicitly trained at training time.

At test time, bagging makes a prediction by averaging together all the sub-models' predictions with the arithmetic mean, but it is not obvious how to do so with the exponentially many models trained by dropout. Fortunately, the average prediction of exponentially many sub-models can be approximately computed simply by running the whole network with the weights scaled by retaining probability. The approximation has been mathematically characterized for linear and sigmoidal networks (Baldi & Sadowski, 2014; Wager, Wang, & Liang, 2013); for piecewise linear networks such as rectified linear networks, Warde, Goodfellow, Courville, and Bengio (2014) empirically showed that weight-scaling approximation is a remarkable and accurate surrogate for the true geometric mean, by comparing against the true average in small enough networks that the exact computation is tractable.

Since dropout was thought to be far less advantageous in convolutional layers, pioneering work by Hinton et al. (2012) only applied it to fully-connected layers. It was the reason they provided that the convolutional shared-filter architecture was a drastic reduction in the number of parameters and thus reduced its possibility to overfit in convolutional layers. Wonderful work by Krizhevsky et al. (2012) trained a very big convolutional neural net, which had 60 million parameters, to classify 1.2 million high-resolution images of ImageNet into the 1000 different categories. Two primary methods were used to reduce over-fitting in their experiments. The first one was data augmentation, an easiest and most commonly used approach to reduce over-fitting for image data. Dropout was exactly the second one. Also, it was only used in fully-connected layers. In the ILSVRC-2012 competition, their deep convolutional neural net yielded top-5 test error rate of 15.3%, far better than the second-best entry, 26.2%, achieved by shallow learning with hand-craft feature engineering. This was considered as a breakthrough in computer vision. From then on, the community believes that deep convolutional nets not only perform best on simple hand-written digits, but also really work on complex natural images.

Compared to original work on dropout, (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) provided more exhaustive experimental results. In their experiments on CIFAR-10, using dropout in fully-connected layers reduced the test error from 15.60% to 14.32%. Adding dropout to convolutional layers further reduced the error to 12.61%, revealing that applying dropout to convolutional layers aided generalization. Similar performance gains can be observed on CIFAR-100 and SVHN. Still, they did not explore max-pooling dropout.

Stochastic pooling (Zeiler & Fergus, 2013) is a dropout-inspired regularization method. The authors replaced the conventional deterministic pooling operations with a stochastic procedure. Instead of always capturing the strongest activity within each pooling region as max-pooling does, stochastic pooling randomly picks the activations according to a multinomial distribution. At test time, probability weighting is used as an estimate to the average over all possible models. Interestingly, stochastic pooling resembles the case of using dropout in max-pooling layers, so it is worth comparing them.

DropConnect (Wan et al., 2013) is a natural generalization of dropout for regularizing large feed-forward nets. Instead of setting to zero the activations, it sets a randomly picked subset of weights within the network to zero with probability $1 - p$. In other words, the fully-connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen stochastically during training. Each unit thus only receives input from a random subset of units in the previous layer. DropConnect resembles dropout as it involves stochasticity within the model, but differs in that the stochasticity is on the weights, rather than the output vectors of a layer. Results on several visual recognition datasets showed that DropConnect often outperformed dropout.

Maxout network (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013) is another model inspired by dropout. The maxout unit picks the maximum value within a group of linear pieces as its activation. This type of nonlinearity is a generalization of the rectified activation function and is capable of approximating the arbitrary convex function. Combining with dropout, maxout networks have been shown to achieve best results on MNIST, CIFAR-10, CIFAR-100 and SVHN. However, the authors did not train maxout networks without dropout. Besides, they did not train the rectified counterparts with dropout and directly compare it with maxout networks. Therefore, it was not clear that which factor contributed to such remarkable results.