Improved Policy Networks for Computer Go

Tristan Cazenave

Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, PARIS, FRANCE

Abstract. Golois uses residual policy networks to play Go. Two improvements to these residual policy networks are proposed and tested. The first one is to use three output planes. The second one is to add Spatial Batch Normalization.

1 Introduction

Deep Learning for the game of Go with convolutional neural networks has been addressed by [2]. It has been further improved using larger networks [7, 10]. AlphaGo [9] combines Monte Carlo Tree Search with a policy and a value network.

Residual Networks improve the training of very deep networks [4]. These networks can gain accuracy from considerably increased depth. On the ImageNet dataset a 152 layers networks achieves 3.57% error. It won the 1st place on the ILSVRC 2015 classification task. The principle of residual nets is to add the input of the layer to the output of each layer. With this simple modification training is faster and enables deeper networks.

Residual networks were recently successfully adapted to computer Go [1]. As a follow up to this paper, we propose improvements to residual networks for computer Go.

The second section details different proposed improvements to policy networks for computer Go, the third section gives experimental results, and the last section concludes.

2 Proposed Improvements

We propose two improvements for policy networks. The first improvement is to use multiple output planes as in DarkForest. The second improvement is to use Spatial Batch Normalization.

2.1 Multiple Output Planes

In DarkForest [10] training with multiple output planes containing the next three moves to play has been shown to improve the level of play of a usual policy network with 13 layers.

We propose to test this improvement for residual networks and for deeper networks. So instead of having only one output plane we will test multiple output planes for our architectures.

2.2 Spatial Batch Normalization

The usual layer used in computer Go program such as AlphaGo [7] and DarkForest [10] is composed of a convolutional layer and of a rectified linear unit (ReLU) layer [8] as shown in figure 1. A ReLU layer simply takes the maximum between 0 and the input of the layer. It enables to train deeper networks by reinforcing the signal through the network. Convolutional layers are composed of small filters (usually 3×3 filters) that are passed all over the input plane to compute the output plane.

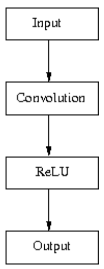

Fig. 1. A usual layer.

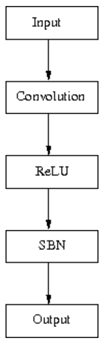

Fig. 2. A layer of DarkForest.

In [1] we proposed to use residual layers as used for image classification. A residual layer adds the input of the layer to the output of the layer using addition and identity as shown in figure 5.

In the code of the open source DarkForest Go program, Spatial Batch Normalization [6] is used after the ReLU layer as shown in figure 2.

Batch Normalisation uses the mean and the variance of the training examples in minibatches to normalize the activations of the network [6]. The principle of Spatial Batch Normalization is to use Batch Normalization for convolutional networks that deal with two dimensional inputs and outputs.

Batch normalization approximates the statistics of the training set with sample statistics drawn from a mini-batch. Given a batch of examples $x_1, \ldots x_m$, the sample mean and sample standard deviation are

$$\bar{x} = \frac{1}{m} \sum_i^m x_i \tag{1}$$

$$\sigma^2 = \frac{1}{m} \sum (x_i - \bar{x})^2. \tag{2}$$

They can be used to standardize the data

$$\hat{x}_i = \frac{x_i - \bar{x}}{\sigma}. \tag{3}$$

To account for the change in the representational capacity of a layer, batch normalization uses additional learnable parameters γ and β, which respectively scale and shift the data, leading to a layer of the form

$$BN(x_i) = \gamma \times \hat{x}_i + \beta. \tag{4}$$

By setting γ to the standard deviation and β to the expectation, we can recover the original layer representation.

The usual residual layers described in [4] already use Spatial Batch Normalization to improve training on images. The architecture commonly used is given in figure 3.

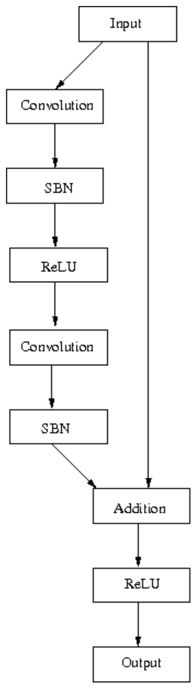

Fig. 3. The original residual layer.

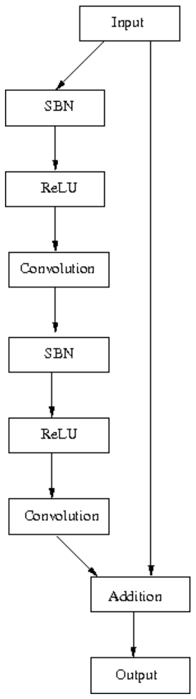

Fig. 4. A residual layer with identity mapping.

Identity mappings were proposed in [5] as an improvement to the original residual layer of [4]. The proposed improved architecture is given in figure 4. The original residual layers also use identity mappings but in a different way.

We propose a new residual layer architecture. It is given in figure 6. It adds a Spatial Batch Normalization after the ReLU layer and outside of the residual block. This is a new architecture that we propose and test in this paper. We call it the Golois layer.

The input layer of our network is also residual. It uses a 5×5 convolutional layer in parallel to a 1×1 convolutional layer and adds the outputs of the two layers before the ReLU layer. It is depicted in figure 7.

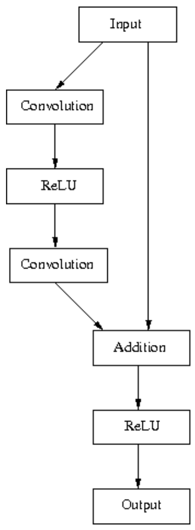

Fig. 5. A residual layer.

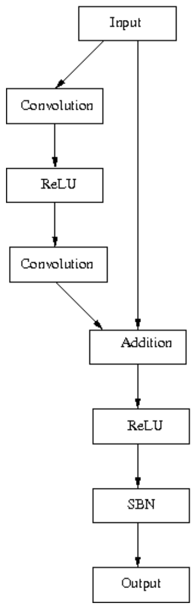

Fig. 6. A Golois layer with Spatial Batch Normalization.

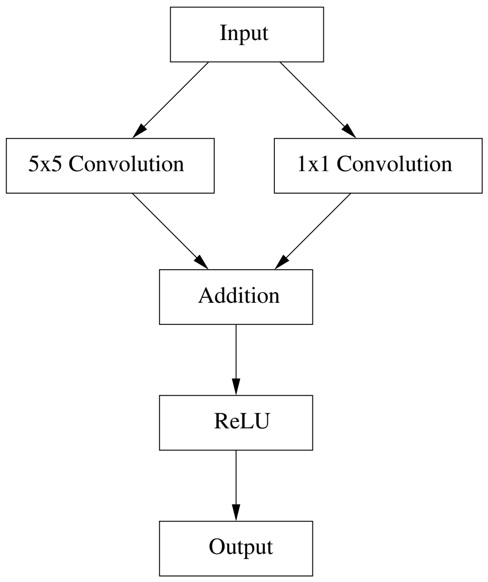

Fig. 7. The first residual layer of the network for computer Go.

The output layer of the network is a 3×3 convolutional layer with one to three output planes followed by a SoftMax.

3 Experimental Results

In this section we will explain the experiments evaluating policy networks. We first present the data that was used for training and testing. We then describe the input planes of the networks and the training and testing phases with results given as percentages on the test set. We give experimental results comparing networks with one and three output planes. We also compare the Golois layer to other residual layers. We finish the section describing our Go playing program Golois.

3.1 The Data

We use the GoGoD dataset [3]. It is composed of many professional games played until today. We used the games from 1900 to 2014 for the training set and the games from 2015 and 2016 as the test set. In our experiments we use the first 500 000 positions of the test set to evaluate the error and the accuracy of the networks.

3.2 Input and Output Planes

The networks use 45 19×19 input planes: three planes for the colors of the intersections, one plane filled with ones, one plane filled with zeros, one plane for the third line, one plane filled with one if there is a ko, one plane with a one for the ko move, ten planes for the liberties of the friend and of the enemy colors (1, 2, 3, 4, ≥ 5 liberties), fourteen planes for the liberties of the friend and of the enemy colors if a move of the color is played on the intersection (0, 1, 2, 3, 4, 5, ≥ 6 liberties), one plane to tell if a friend move on the intersection is captured in a ladder, one plane to tell if a string can be captured in a ladder, one plane to tell if a string is captured in a ladder, one plane to tell if an opponent move is captured in a ladder, one plane to tell if a friend move captures in a ladder, one plane to tell if friend move escapes a ladder, one plane to tell if a friend move threatens a ladder, one plane to tell is an opponent move threatens a ladder, and five planes for each of the last five moves.

The output of a network is a 19×19 plane and the target is also a 19×19 plane with a one for the move played and zeros elsewhere.

The choice of these input planes is similar to other Deep Learning programs such as DarkForest [10] and AlphaGo [7, 9] with a little more focus on tactical ladders calculations.

3.3 Training

In order to train the network we build minibatches of size 50 composed of 50 states chosen randomly in the training set, each state is randomly mirrored to one of its eight symmetric states. The accuracy and the error on the test set are computed every 5 000 000 training examples. We define an epoch as 5 000 000 training examples.

We do not use an epoch as the total number of examples as there are many examples and as the training procedures chooses examples randomly in the training set to build minibatches.

The algorithm for updating the learning rate is the same as in [1]. The principle is to divide the learning rate by two each time the training error stalls, i.e. is greater than the previous average training error over the last 5 000 000 training examples.

3.4 Multiple Output Planes

We compare training a 28 convolutional layers residual network with one and three output planes. In DarkForest it was found that it improves the level of play but does not change the training and testing phases. We found that training was slightly more difficult with three output planes. Figure 8 gives the evolution of the accuracy for networks with one and three output planes. We can see that even if the network with three output planes is initially worse, it eventually reaches a greater accuracy. Training was stopped when the learning rate became too small to induce changes in the performance of the network. This is why the three output planes network was trained longer than the other network, its learning rate stayed greater for a longer time. We also found that using three output planes enables the network to better generalize. We can see in figure 9 that the training error stays close to the test error for the three output planes network. In figure 10 the difference between the test error and the training error becomes greater for the one output plane network.

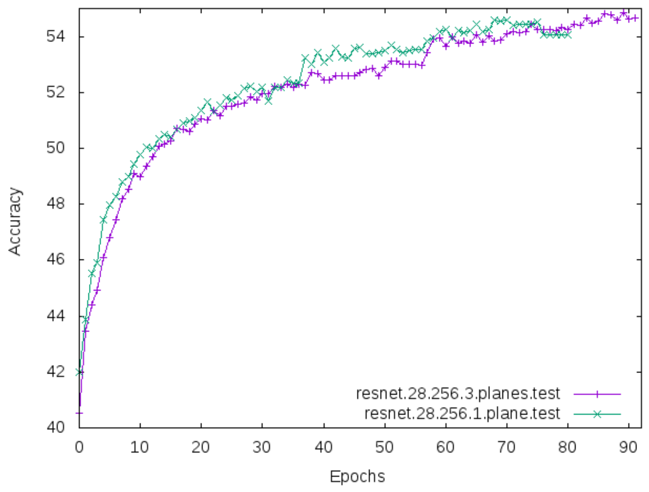

Fig. 8. Evolution of the accuracy of 28 convolutional layers residual networks with 1 and 3 output planes. The learning rate is initially set to 0.2 and divided by two each time the training error stalls. The accuracy is taken only over the next move to play.

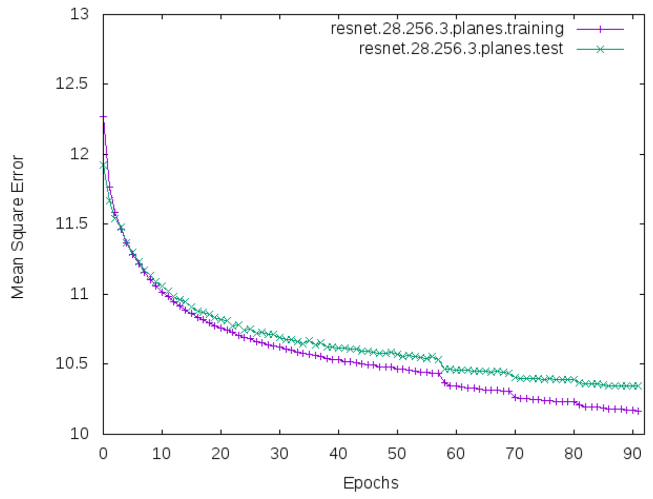

Fig. 9. Evolution of the training and test errors of a 28 convolutional layers residual network with three output planes. The errors are calculated over the three output planes.

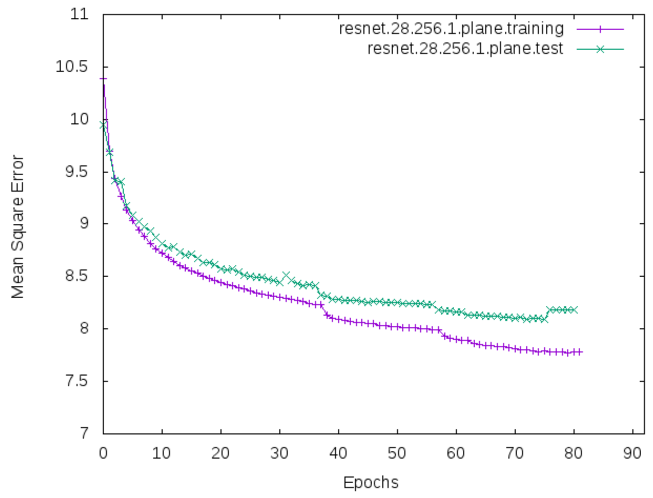

Fig. 10. Evolution of the training and test errors of a 28 convolutional layers residual network with 1 output plane.

3.5 Spatial Batch Normalization

In order to test Spatial Batch Normalization we trained two 14 layers residual networks with 128 feature planes on the GoGoD data set. The only difference between the two networks is that the second adds Spatial Batch Normalization after the residual layer.

The evolution of the mean square error on the test set is given in figure 11. We can observe that the error of the network with Spatial Batch Normalization is consistently smaller than the one without Spatial Batch Normalization.

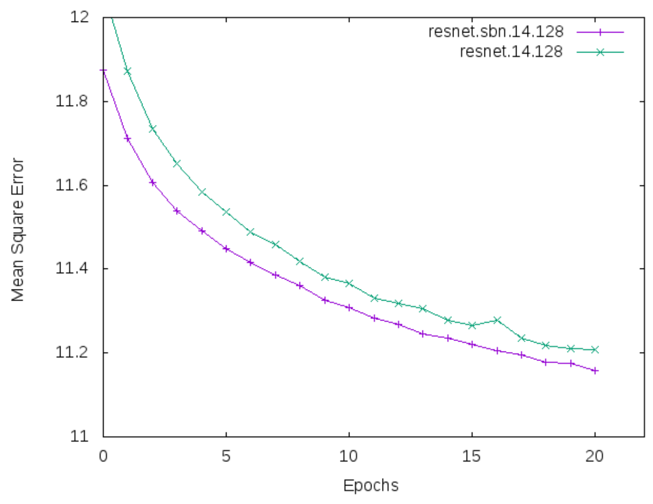

Fig. 11. Evolution of the error of a 14 convolutional layers network with 128 feature planes on the GoGoD test set with and without Spatial Batch Normalization. The learning rate is 0.2 for both networks.

Figure 12 gives the evolution of the accuracy for both networks on the GoGoD test set.

Figure 13 give the evolution of the training error on the GoGoD training set for 4 residual networks composed of 14 layers and 256 feature planes. The resnet.original.14.256 network is composed of the original residual layers of figure 3 [4]. The resnet.mapping.14.256 network is composed of the residual layers with identity mappings of figure 4 [5]. The resnet.14.256 network is composed of the residual layers without Spatial Batch Normalization of figure 5. The resnet.golois.14.256 network is composed of the residual layers of figure 6.

The networks are trained on 20 000 000 examples with a minibatch of size 50 and a learning rate of 2.0. An epoch is defined as 1 000 000 training examples. At every epoch the average training error over the last 500 000 training examples is plotted.

We can observe that the original resnet performs worse than the other networks. The identity mapping network starts close to the residual network without Spatial Batch

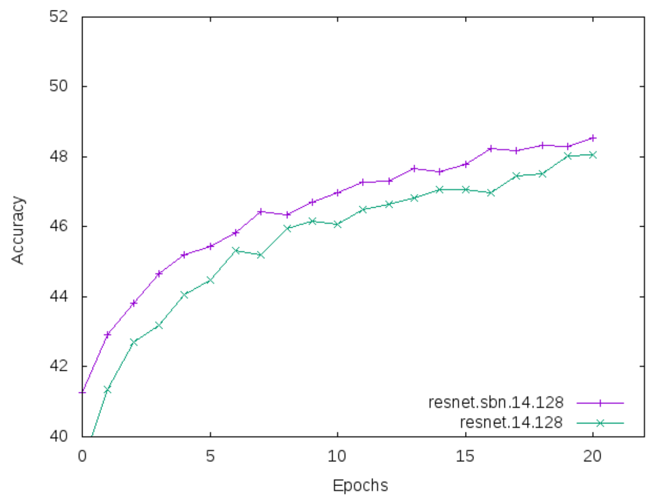

Fig. 12. Evolution of the accuracy of a 14 convolutional layers network with 128 feature planes on the GoGoD test set with and without Spatial Batch Normalization. The learning rate is 0.2 for both networks.

Normalization but performs worse with more examples. The best network is the network with Golois layers.

3.6 Golois

A network with 3 output planes, residual layers and 28 convolutional layers plays on KGS as Golois7. It plays instantly the best move of the policy network. It has played more than 6 000 games. It is ranked 4 Dan. In comparison AlphaGo and DarkForest policy networks reached 3 Dan.

4 Conclusion

We evaluated two improvements to deep residual networks for computer Go. Using three output planes enables the networks to generalize better and reach a greater accuracy. A new residual layer with Spatial Batch Normalization has been shown to perform better than existing residual layers.

In future work we plan to train a 28 layers network with Spatial Batch Normalization and to train a residual value network.

References

1. Tristan Cazenave. Residual networks for computer Go. *IEEE TCIAIG, available online*, 2017.

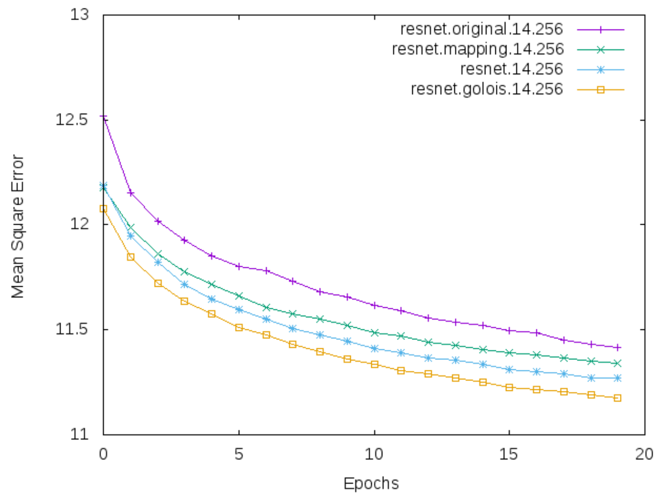

Fig. 13. Evolution of the training error of 14 convolutional layers residual networks with 256 feature planes on the GoGoD test set for different architectures. The learning rate is 2.0 for all networks.

2. Christopher Clark and Amos Storkey. Training deep convolutional neural networks to play go. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1766–1774, 2015.
3. T. Mark Hall and John Fairbairn. Games of go on download. *http://gogodonline.co.uk/*, 2016.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.
6. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.
7. Chris J Maddison, Aja Huang, Ilya Sutskever, and David Silver. Move evaluation in go using deep convolutional neural networks. *arXiv preprint arXiv:1412.6564*, 2014.
8. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
9. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
10. Yuandong Tian and Yan Zhu. Better computer go player with neural network and long-term prediction. *arXiv preprint arXiv:1511.06410*, 2015.