Cosine Annealing, Mixnet and Swish Activation for Computer Go

Tristan Cazenave, Julien Sentuc, and Mathurin Videau

LAMSADE, Université Paris-Dauphine, PSL, CNRS, France
Tristan.Cazenave@dauphine.psl.eu

Abstract. The architecture of neural networks in neural based computer game programs influences greatly the strength of the game playing programs. We present developments on the recently tested Mobile Network architecture that has good results for the game of Go. The three proposed improvements deal with the optimization process, the activation function and the convolution layers. These three modifications improve the accuracy of the policy and the error of the evaluation, as well as the playing strength of a computer Go program using the resulting networks.

1 Introduction

Important breakthroughs in Computer Go have been achieved in the past years. These advances were made possible by the advent of Convolutional Neural Network (CNN) and development of Monte-Carlo Tree Search. Because of their versatility, CNN architectures are constantly evolving. Thus, the purpose of this article is to use these recent changes to improve supervised learning in Computer Go. We also hope that these improvements will transfer to the Reinforcement Learning setup.

Classically, CNN for Go have more than one head. At least, these networks use a policy head, to prescribe moves, and a value head, to evaluate the board quality in terms of future incomes. This output configuration has been popularized by the groundbreaking AlphaZero [7]. In 2017, AlphaGoZero reached a superhuman level without initial knowledge except the game rules. Thereafter, DeepMind's algorithm AlphaZero achieved comparable results for Chess and Shogi. This achievement has been made possible by the use of deep reinforcement learning from self-play.

Closer to our work, various architecture has been evaluated for learning to play Go in a supervised way. Typically, the dataset used for the supervised learning is constituted of superhuman games produced by Deep Reinforcement Learning agent like AlphaZero or Katago. In our study, we used Katago to constitute our dataset.

KataGo [9] like Alpha Zero only learns from neural-net-guided Monte Carlo Tree Search self-play. KataGo improves learning compared to AlphaGo Zero. Mainly, it converges to superhuman level much faster than comparable methods such as Alpha Zero, ELF/OpenGo or Leela Zero. It uses different optimizations strategies like using a low number of playouts for most of the moves in a game to gather more data about the value in a shorter time, or using additional training targets to regularize the network. An innovation in the Katago program is to use GlobalAverage Pooling in some layers of the network in conjunction with residual layers.

Architectures of the Neural Network used in Deep Reinforcement Learning has been shown to have a great impact on the performances of the resulting playing engines. For example, the use of residual networks increased Alpha GO's ELO by 600. Residual Networks used in Alpha Zero were compared to Mobile Networks [4] with policy and value heads different from the Alpha Zero ones, for instance a fully convolutional policy head and a global average pooling value head. These mobile networks are more efficient in terms of computation and parameters than their classic CNN counterparts. Also, further improvement of mobile networks have been tested [3]. The main architecture change is the introduction of the Squeeze and Excitation block, adding channel attention to the network. Mobile Networks presented better results than Residual Networks, both for small and large networks on the Leela dataset composed of games played at a superhuman level [4]. They had a better accuracy and value error.

2 Improving Supervised Learning

Here, we present different improvements made to increase performance. They rely on three different aspects of the training : optimization, activation function and architecture.

2.1 Cosine Annealing

Better optimization schema can lead to better results. Indeed, by using a different optimization strategy, a neural net can end in a better optimum. In this paper, this is achieved by using Stochastic Gradient Descent with warms Restart (SGDR) [5]. In particular, the learning rate is restarted multiple times. This way, the objective landscape is explored further and the best solution of all restart is kept. Furthermore, using a peculiarly aggressive learning rate strategies like cosine annealing (Equation 1) can lead to better convergence.

$$\eta_t = \eta^i_{\min} + \frac{1}{2}(\eta^i_{\max} - \eta^i_{\min})(1 + \cos(\frac{T_{cur}}{T_i}\pi)) \quad (1)$$

with η_t the learning rate at time t, T_{cur} the number of step since the last restart and i the current number of cycles done. Thus, T_i indicates the number of steps allowed for the cycle i and $\eta^i_{\min}, \eta^i_{\max}$ the range of values the learning rate can take during the cycle i.

We compare this cosine annealing with what we call *division annealing*. Division annealing, simply divide the learning rate by a constant at predefined epochs.

2.2 MixNet

Traditional depthwise convolution suffers from the limitations of single kernel size.*Tan et al.* [8] proposed to replace the vanilla depth-wise convolution with *MixConv*. Their module takes advantage of bigger kernel size in the convolution. The idea is to mix up multiple kernels of different sizes in a single depthwise convolution operator in order to capture different types of patterns at different scales from input images. They achieved significant performance gain in image classification compared to mobilenet-v3 on both ImageNet classification and COCO object detection. Even better, they showed that mixing kernel size allows using bigger kernels.

2.3 Swish Activation

Non-linearity plays an important role in neural network. Without them, they lose their expressiveness power. It also has an important impact on the neural net training. In particular, the activation shape the derivatives of the network. These important properties motivated *Ramachandran et al.* [6] to search for good activation functions. From their research, they discovered the Swish activation function :

$$x \cdot \text{sigmoid}(x)$$

This activation, used as drop down replacement for ReLU, gives significant improvement on diverse tasks and networks.

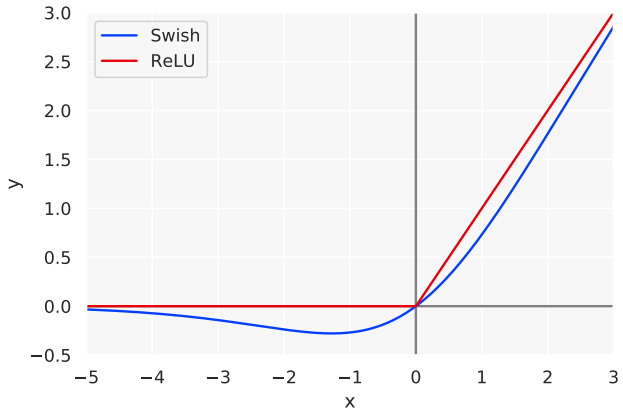

Fig. 1: Swish function plot

3 Experimental Results

In all experiments, instead of learning the final result of a game, the value head is labeled with the Q value coming from MCTS. We use a simple mixed convolution composed of half of 3×3 kernels and half of 5×5 kernels. The dataset is composed of self played games from Katago [9]. The label, for the policy head, is a one for the move played by Katago and zeros for other moves. The label, for the value head, is the evaluation of the position given by the Monte Carlo Tree Search of Katago. A value between 0 and 1 giving the probability of winning for White.

3.1 Training with cosine annealing

In early experiments, we tested multiple learning rate parameters using learning rate restart leads to no improvement. Figure 2 makes the comparison of cosine annealing

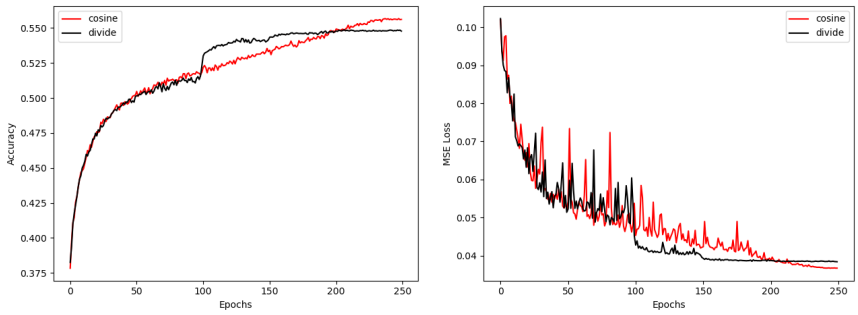

Fig. 2: Cosine annealing versus division annealing for the accuracy and the MSE of the 16 blocks mobile network. The learning rate of division annealing is divided by 10 at epoch 100, 150 and 200.

with division annealing for the two best run. Cosine annealing ends up with better accuracy and MSE. Moreover, the learning curve for cosine annealing is smoother, for instance there are no bumps on the learning curve because of learning rate changes. So in the following experiments, we use cosine annealing without restarts, there is only one cycle.

3.2 Training small networks with Mixnet and Swish activation

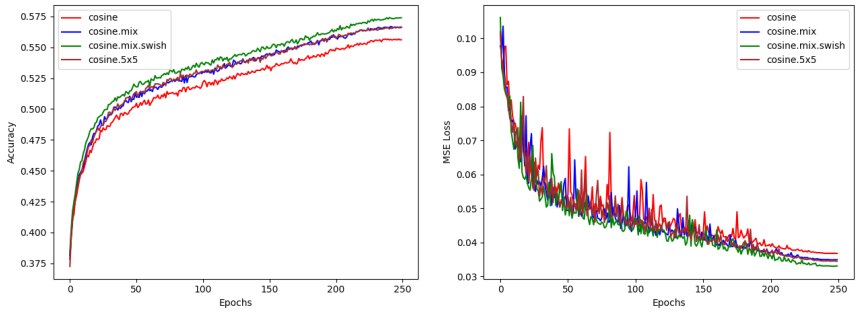

Fig. 3: Mobile network with kernels 3×3 and 5×5, Mixnet and Mixnet with Swish activation for the accuracy and the MSE of the 16 blocks mobile network.

Figure 3 compares 16 blocks mobile networks, Mixnet and Mixnet with Swish activation. We see that Mixnet with Swish has better results than Mixnet alone, and that Mixnet alone has better result than using only 3×3 kernels. Notice that Mixnet also has similar results to a mobile network with 5×5 kernels, despite having less parameters.

3.3 Training big networks with Mixnet and Swish activation

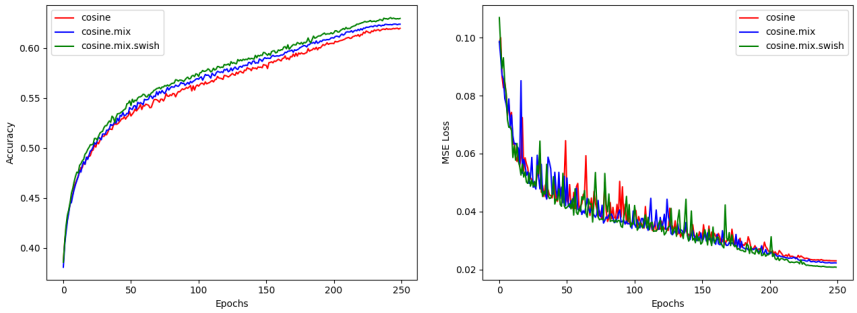

Fig. 4: Mobile network, Mixnet and Mixnet with Swish activation for the accuracy and the MSE of the 48 blocks mobile network.

We have tested the Swish activation instead of the Rectified Linear Unit activation (ReLU) in the inverted residual blocks. Figure 4 compares 48 blocks mobile network, with 3×3 kernels only, Mixnet, and Mixnet with Swish activation. Mixnet with Swish activation is better than Mixnet which is better than 3×3 kernels only.

3.4 Playing

Table 1 gives the experiments done for comparing the different networks. Each line is the result of 400 games between two networks using given constants and numbers of playouts at each move. Each player plays 200 games as Black and 200 games as White. Both players use the Batch MCTS search algorithm [2]. In order to randomize the starting position of each game, the first 20 moves are played randomly according to the probabilities given by the policy. The properties of the Max players are given in columns 2 to 7. *Blocks* is the number of blocks of the mobile network, *Planes* is the number of planes in the trunk, *M* is the use of Mixnets, *S* is the use of the Swish activation, *C* is the PUCT constant and *Playouts* is the number of playouts at each move. Columns 8-13 are the properties of the Min player.The last two columns are the winrate of the Max player and the standard deviation of the winrate.

Lines 1-5 give the experiments used to find the best PUCT constant for the 16 blocks mobile network. Each constant is played against the upper constant. We observe that every constant between 0.05 and 0.30 is worse than the upper constant while 0.40 beat 0.50. We assume the best constant for 100 playouts and the 16 blocks network is 0.40.

Lines 6-9 compare the 16 blocks mobile network with Mixnet and Swish activation with a constant 0.40 to the 16 blocks mobile network with Mixnet only. Various constants are tested for the Mixnet only network and the best constant for this network is 0.20 yielding a winrate of 0.5850 for the Mixnet with Swish activation network. We can conclude that the Swish activation makes the 16 blocks mobile network stronger.

Lines 10-14 compare the 16 blocks mobile network with Mixnet and Swish activation with a constant 0.40 to the 16 blocks mobile network with no Mixnet and no Swish activation. The best constant for the second network is 0.20 yielding a winrate of 0.6950 for the first network. It is higher than the 0.5850 winrate obtained with Mixnet. We can conclude, Mixnet improves the strength of the 16 blocks mobile network.

Lines 15-16 show that 0.40 is also a good constant for the 48 blocks mobile network. We will use it in the next experiments.

Lines 17-22 compare the 48 blocks mobile network with Mixnet and Swish activation with a constant 0.40 to the 48 blocks mobile network with Mixnet only. The best performance for the second network is obtained with the 0.20 constant which gives a 0.5600 winrate for the first network. Swish activation is also beneficial to the 48 blocks mobile network.

Lines 23-27 compare the 48 blocks mobile network with Mixnet and Swish activation with a constant 0.40 to the 48 blocks mobile network without Mixnet and without Swish activation. The best performance for the second network is obtained with the 0.30 constant which gives a 0.6450 winrate for the first network. This is a worse result for the second network than with Mixnet, so Mixnet improves the strength of the 48 blocks mobile network.

Table 2 gives the accuracy, the MSE, the GPU and the CPU speed in terms of batches of size 32 processed per seconds for different networks. We did not train the residual networks on our new dataset, so we only give the speeds for residual networks [1]. In previous experiments, both the accuracy and MSE of residual networks were largely behind those of mobile networks [3]. The mobile.16.64 line is a mobile network with 16 inverted residual blocks and 64 planes in the trunk. The mobile.mix.swish.48.128 network is a mobile network with mixed convolutions, Swish activation, 48 inverted residual blocks and 128 planes in the trunk.

We can observe that the GPU speed of the networks with mixed convolutions and Swish activation is a little smaller than the speed of the original mobile networks. The accuracy and the MSE are better.

Table 1: Making the networks play.

	Blocks	Planes	M	S	C	Playouts	Blocks	Planes	M	S	C	Playouts	Winrate	σ
1	16	64	y	y	0.05	100	16	64	y	y	0.10	100	0.4250	0.025
2	16	64	y	y	0.10	100	16	64	y	y	0.20	100	0.4775	0.025
3	16	64	y	y	0.20	100	16	64	y	y	0.30	100	0.4700	0.025
4	16	64	y	y	0.30	100	16	64	y	y	0.40	100	0.4925	0.025
5	16	64	y	y	0.40	100	16	64	y	y	0.50	100	0.5025	0.025
6	16	64	y	y	0.40	100	16	64	y	n	0.40	100	0.6375	0.024
7	16	64	y	y	0.40	100	16	64	y	n	0.30	100	0.6125	0.024
8	16	64	y	y	0.40	100	16	64	y	n	0.20	100	**0.5850**	0.025
9	16	64	y	y	0.40	100	16	64	y	n	0.10	100	0.6425	0.024
10	16	64	y	y	0.40	100	16	64	n	n	0.50	100	0.8250	0.019
11	16	64	y	y	0.40	100	16	64	n	n	0.40	100	0.7475	0.022
12	16	64	y	y	0.40	100	16	64	n	n	0.30	100	0.7125	0.023
13	16	64	y	y	0.40	100	16	64	n	n	0.20	100	**0.6950**	0.023
14	16	64	y	y	0.40	100	16	64	n	n	0.10	100	0.7500	0.022
15	48	128	y	y	0.30	100	48	128	y	y	0.40	100	0.4900	0.025
16	48	128	y	y	0.40	100	48	128	y	y	0.50	100	0.5200	0.018
17	48	128	y	y	0.40	100	48	128	y	n	0.50	100	0.7125	0.023
18	48	128	y	y	0.40	100	48	128	y	n	0.40	100	0.6300	0.024
19	48	128	y	y	0.40	100	48	128	y	n	0.30	100	0.5775	0.025
20	48	128	y	y	0.40	100	48	128	y	n	0.20	100	**0.5600**	0.025
21	48	128	y	y	0.40	100	48	128	y	n	0.10	100	0.5850	0.025
22	48	128	y	y	0.40	100	48	128	y	n	0.05	100	0.6475	0.024
23	48	128	y	y	0.40	100	48	128	n	n	0.50	100	0.7050	0.023
24	48	128	y	y	0.40	100	48	128	n	n	0.40	100	0.6700	0.024
25	48	128	y	y	0.40	100	48	128	n	n	0.30	100	**0.6450**	0.024
26	48	128	y	y	0.40	100	48	128	n	n	0.20	100	0.6525	0.024
27	48	128	y	y	0.40	100	48	128	n	n	0.10	100	0.6925	0.023

Table 2: Accuracy and speed.

Network	Accuracy	MSE	GPU Speed	CPU Speed
residual.20.256			20.78	1.30
residual.40.256			12.62	0.68
mobile.16.64	55.61	0.0367	30.70	6.11
mobile.mix.16.64	56.64	0.0349	26.58	4.11
mobile.mix.swish.16.64	57.40	0.0331	21.60	2.75
mobile.48.128	61.97	0.0230	13.06	0.75
mobile.mix.48.128	62.37	0.0223	10.35	0.54
mobile.mix.swish.48.128	62.95	0.0208	7.73	0.42

4 Conclusion

We proposed three improvements to Mobile Networks for Computer Go. They improve both the supervised training and the architecture of the network by using Swish activation and mixed convolutions. The large network using mixed convolutions and the Swish activation has a winrate of 0.6450 against a similar network not using them. This brings a 104 Elo improvement. Also, the network trained with cosine annealing has better accuracy and evaluation error than the network trained dividing by 10 the learning rate.

It would be interesting to experiment these improvements in other games and also in the deep reinforcement learning framework.

Acknowledgment

This work was granted access to the HPC resources of IDRIS under the allocation 2021-AD011012539 made by GENCI.

This work was supported in part by the French government under management of "Agence Nationale de la Recherche" as part of the “Investissements d’avenir” program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

References

1. Cazenave, T.: Residual networks for computer go. IEEE Transactions on Games **10**(1), 107–110 (2018)
2. Cazenave, T.: Batch monte carlo tree search. Arxiv (2021)
3. Cazenave, T.: Improving model and search for computer Go. In: IEEE Conference on Games (2021)
4. Cazenave, T.: Mobile networks for computer Go. IEEE Transactions on Games (2021)
5. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
6. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
7. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
8. Tan, M., Le, Q.V.: Mixconv: Mixed depthwise convolutional kernels. arXiv preprint arXiv:1907.09595 (2019)
9. Wu, D.J.: Accelerating self-play learning in go. CoRR **abs/1902.10565** (2019)