Deep Learning Project

Chauvet, Clément
clement.chauvet@dauphine.eu

Cropsal, Télio
telio.cropsal@dauphine.eu

December 8, 2022

The goal of this project is to train a neural network to master the game of Go. The input and output of the network are fixed, with as an input, 31 planes representing the state of the game and as output two heads, a policy, choosing the best move to play, and a value estimating which player has the advantage. Techniques from computer vision are leveraged since the planes can be treated as images. We are facing one critical constraint, the number of parameters should not exceed one million. The whole framework is similar to the one introduced by AlphaGo and extend later on by AlphaZero and the data-set is constituted by games from KataGO.

1 Tried and tested architectures/techniques

1.1 MobileNet

The framework of MobileNet networks is perfect for our case since the objective is to reduce the complexity while maintaining good predictions.

MobileNetV1 introduced depth-wise convolution in order to reduce the number of parameters. MobileNetV2 bring in residual networks for this task, residual blocks add a stronger notion of memory, help against vanishing gradients and can be trained faster and deeper. MobileNetV3 extended this idea with squeeze and excitation layers. The principle is that, each channel is reduced to a single numeric value. And these values are used after some layers to infer weights for scaling each channel.

We tried MobileNet V2 and V3. After multiple tests and reading results in Computer Vision we decided to stay as close as possible to 300 filters on the expand, 80 filters in the trunk and 15 blocks.

1.2 MixConv

MixConv extends the depth-wise convolution. In fact, instead of applying a fixed size kernel to each channel separately, the channels are divided into groups with different kernel sizes. Then a point-wise convolution is applied to increase the number of channels. MixConv allows to have performances similar to a classical depth-wise convolution while lowering the number of parameters.

We tried MixConv layers instead of classical depth-wise layers.

1.3 Coordinate Attention

In the framework of MobileNet, we decided to try and expand the use of attention mechanism. The idea behind this choice is that a squeeze and excite block works similarly to a low parameter attention mechanism on the different planes. But in the case of the game of Go, certain areas of the board are much more important than others. In this setup having an attention mechanism on the other 2 coordinates might lead to better results. Our first idea was to re-implement the Coordinate Attention (CA)[7] block. The CA block is described in 1.3. This model has proven to lead to better results than a classic MobileNetV3 in certain configurations.

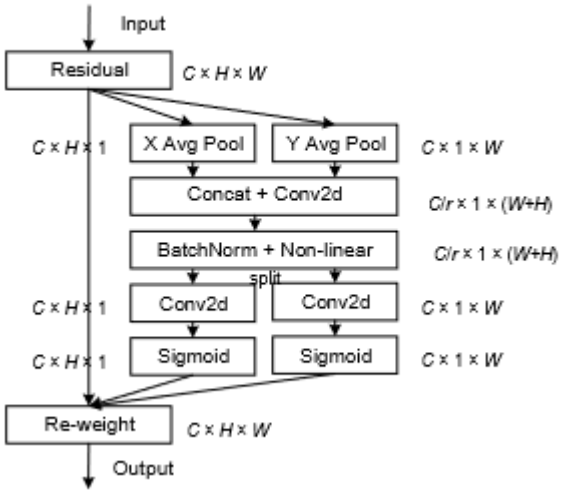

Figure 1: Description of the coordinate attention block from [7]

1.4 CBAM

On the same idea to use more attention mechanisms, Convolutional Block Attention Module (CBAM)[8] uses two attention modules: first a channel attention using both average pooling and max pooling to extract more information from the input while maintaining a constant amount of parameters compared to squeeze and excite, and after re-weighting the image accordingly, it uses a spatial attention block trying to extract regions of interests. This model is described in details in 1.4

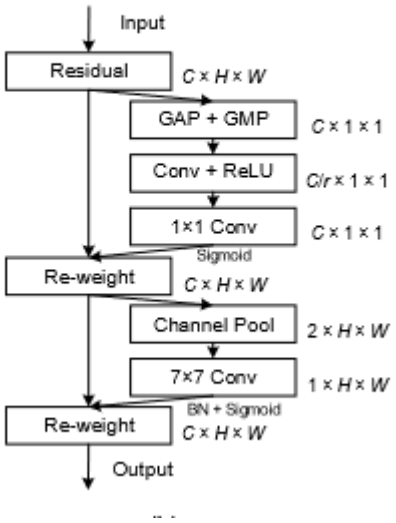

Figure 2: Description of the CBAM from [7]

2 Other tweaks

On every models, we use $N = 10000$ samples for each epochs and a batch size as big as possible depending on the graphic card used. For all our models we use a swish activation function. The loss is defined as the sum of the categorical crossentropy on the policy and the binary crossentropy on the value.

2.1 Learning rate

We tried to implement Cosine Annealing as described in the most recent paper but found it hard to tune because it tend to reduce the learning rate too quickly. For the training we opted for high learning rates in the early phase with a learning rate of 0.05 because we it allows for faster convergence and usually finds better minimums by exploring more. We keep a high learning rate before quickly decreasing as we found out that it converges quickly once the learning rate is below 0.005 We regularly divide our learning rate using a schedule : 0.05 until 700 epochs, 0.005 until 1500, 0.0005 until 2000 and 0.00005 until 2200.

2.2 Optimizer

We tried using different optimizers including Stochastic Gradient Descent (SGD), Adam, Lookahead[9] and found that Adam and Lookahead lead to faster convergence but don't perform well in the case of longer training compared to a classical SGD.

3 Results

At the end of our training, we obtained these results:

	Accuracy	MSE
MobileNetV3	47%	0.064
MobileNetV3 with MixConv	51%	0.057
MobileNetV3 with CBAM	49%	0.060
MobileNetV3 with CA	48%	0.064

4 Our final model

Seeing the results of the different architectures we opted for a MobileNetV3 with MixConv. We chose to use the same parameters as the model we compared in the last section for the last tournament but train it as much as possible on the bigger data-set containing 1 million games. We trained using the following schedule for the learning rate : 0.05 for the first 1500 epochs, 0.005 until epoch 3000, 0.0005 until epoch 4000, and finally 0.00005 for 500 epochs.
It reached 52.5% accuracy and an mse of 0.041

5 Conclusion and possible further improvements

In the end, different attention mechanisms showed promising results but we didn't have the time nor the resources to tune all the parameters and test with longer trainings. We still think that CA or CBAM could beat the MobileNetV3 with MixConv if used optimally.
We also discovered Strip Pooling[10] and Triplet Attention[11] mechanism and think that it could be interesting to try in the context of Go. Particularly Strip Pooling could be interesting as it doesn't work independently on each dimension and could be combined with squeeze and excite.
Finally using hard swish has shown to be more efficient in multiple contexts of Computer Vision and could be tried in our models.

References

[1] David Silver and al. *Mastering the game of Go with deep neural networks and tree search.* 2016.

[2] David Silver and al. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.* 2018.

[3] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig Adam. *Searching for MobileNetV3.* 2019.

[4] Tristan Cazenave. *Mobile Networks for Computer Go.* 2021.

[5] Tristan Cazenave. *Residual Networks for Computer Go.* 2018.

[6] Mingxing Tan, Quoc V. Le. *MixConv: Mixed Depthwise Convolutional Kernels.* 2019.

[7] Qibin Hou, Daquan Zhou, Jiashi Feng. *Coordinate Attention for Efficient Mobile Network Design.* 2021.

[8] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. *CBAM: Convolutional Block Attention Module.* 2021.

[9] Michael R. Zhang, James Lucas, Geoffrey Hinton, Jimmy Ba *Lookahead Optimizer: k steps forward, 1 step back.* 2019.

[10] Qibin Hou, Li Zhang, Ming-Ming Cheng, Jiashi Feng *Strip Pooling: Rethinking Spatial Pooling for Scene Parsing*. 2020.

[11] Diganta Misra, Trikay Nalamada,Ajay Uppili Arasanipalai,Qibin Hou *Rotate to Attend: Convolutional Triplet Attention Module*. 2021.