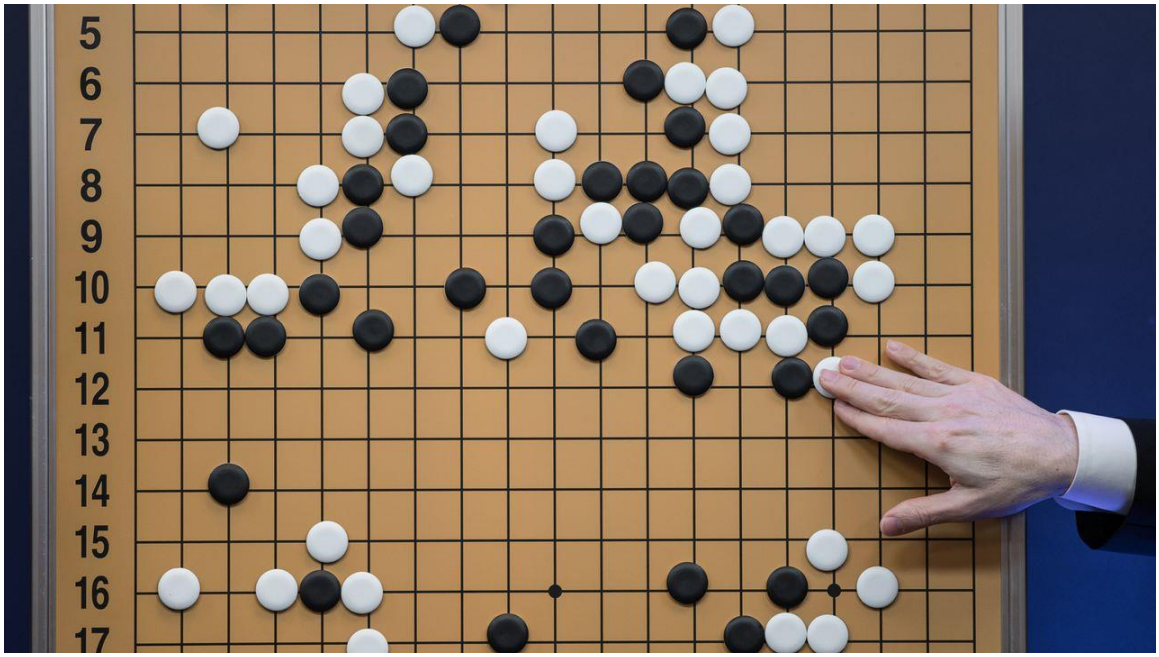


Deep Learning Project

PLAYING GO WITH A NEURAL NETWORK

Maximilien Wemaere | Master IASD | 2022-2023

Dauphine | PSL
UNIVERSITÉ PARIS

I. Introduction

This project aims to restore the knowledge from the *Deep Learning* course with Tristan Cazenave and putting it into practice.

A. CONTEXT

If an AI-based program beat the world chess champion Garry Kasparov in 1987 (Deep blue), it is only in 2015 that the AlphaGo program defeated the world Go champion Lea Sedol. Indeed, the game of Go is much more difficult for a computer as it requires lot of instinct because the number of possible moves is very high.

It was necessary to wait for the Deep Learning revolution in the 2010s to be able to address this problem correctly. Indeed, the complexity of the issue: choose the next move from the current state of the game could not be addressed with traditional machine learning such as Bayesian learning or random forest (used for Deep Blue).

This revolution was possible thanks to the convergence of the algorithms of artificial neural networks (from perceptron algorithms), of auto-differentiation, and of stochastic optimization, and the increase of the calculus power of the computers thanks to the reduction of transistors.

B. PROBLEM

Our problem is to build a neural network which as a center of a go-playing program (described in part II) which has the best go level. The algorithms of all the IASD program's students will be assessed by playing against each other.

The constraints are to keep the main structure of the program and to propose a network with less than 100000 parameters.

So, all our work will consist of designing the neural network architecture at the center of the program and train it.

II. Global architecture of the algorithm

Here we will describe the principle of the go-playing program (which is provided by the teacher of the course)

COMPONENTS

The global architecture came from the AlphaGo algorithm: it consists of a neural network which predict two things: a policy and a value. The policy is used to predict the next best move to make, given the current state of the game. It takes as input a representation of the board state and outputs a probability distribution over all possible moves. The value is used to predict the outcome of the game, given the current state of the game. It takes as input a representation of the board state and outputs a predicted score for the current player.

The network is implemented as deep convolutional neural networks, with multiple layers of convolutional and fully connected units. The policy came from a final layer that outputs a probability distribution over all possible

moves, while the value came from a final layer that outputs a single scalar value representing the predicted outcome of the game.

In addition to this network, the AlphaGo architecture also includes a Monte Carlo Tree Search (MCTS) algorithm, which is used to guide the search for the next best move. The MCTS algorithm uses the outputs of the policy network and the value network to guide its search and combines the predictions of these networks with additional simulations of the game to find the best move.

DATA STRUCTURE

The data comes from Katago, which is a Go-playing program which play against himself to generate games. We have 1 million games in the training set.

Each input data point is a moment of a game, where we must make a choice for the next move depending on the color we are and is shaped as following: 31x19x19. 19x10 is the size of the board in Go. And 31 is the different planes which give different information such as the color to play, the ladders, the current states on two planes, two previous states on four planes ...

The output is the policy: a vector 361 which represent all possible moves and where 1 is the best possible move and 0 for the others, and the value: a single float, close to 1 if White wins and 0 if black wins.

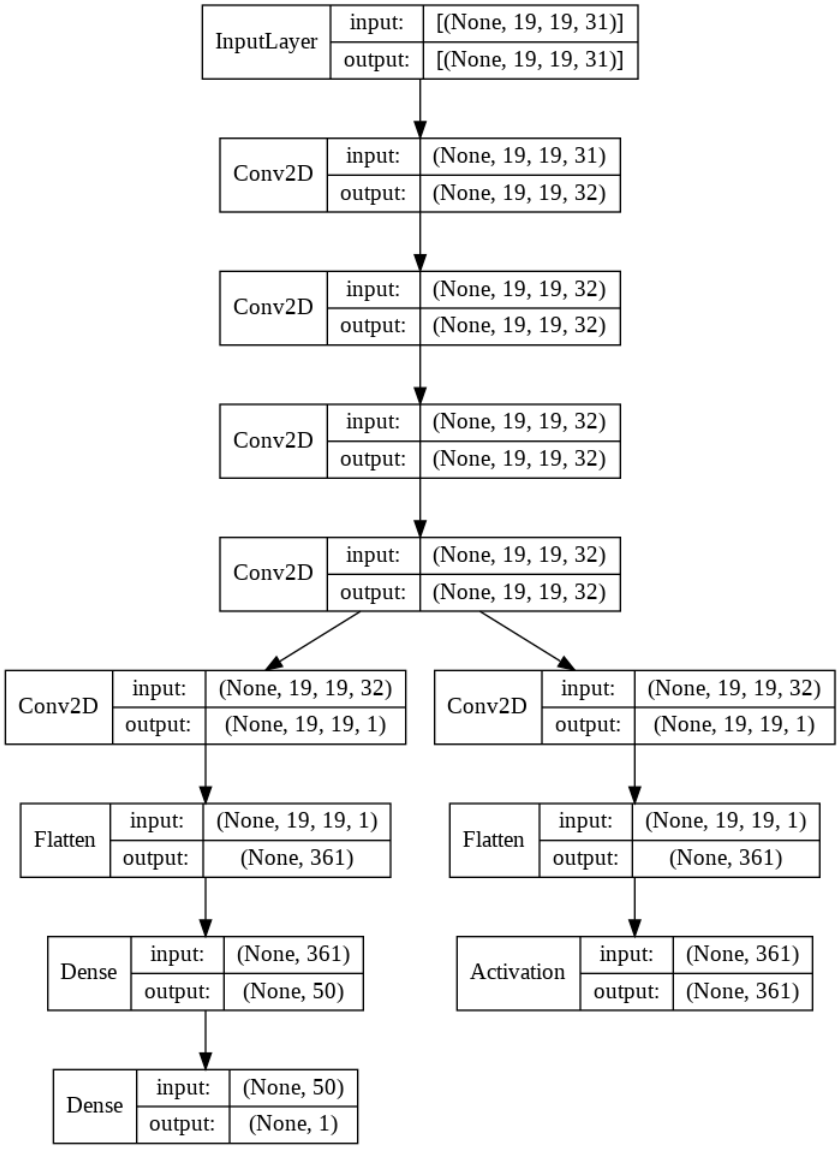
III. Architecture of the neural network

During the project we tested different architectures for this network, we present them here briefly and their results.

The only way we have for assessing our network was the final loss of the policy and of the value, the final categorical accuracy of the policy and the final mean square error of the value. We this four numbers we assessed our different networks in order to choose the better for the competition.

SIMPLE CNN

We first try with a simple CNN, with 3 convolution layers and then we flatten to add some dense layers. without surprise we do not get very good results, so we quickly move on to more complex models.

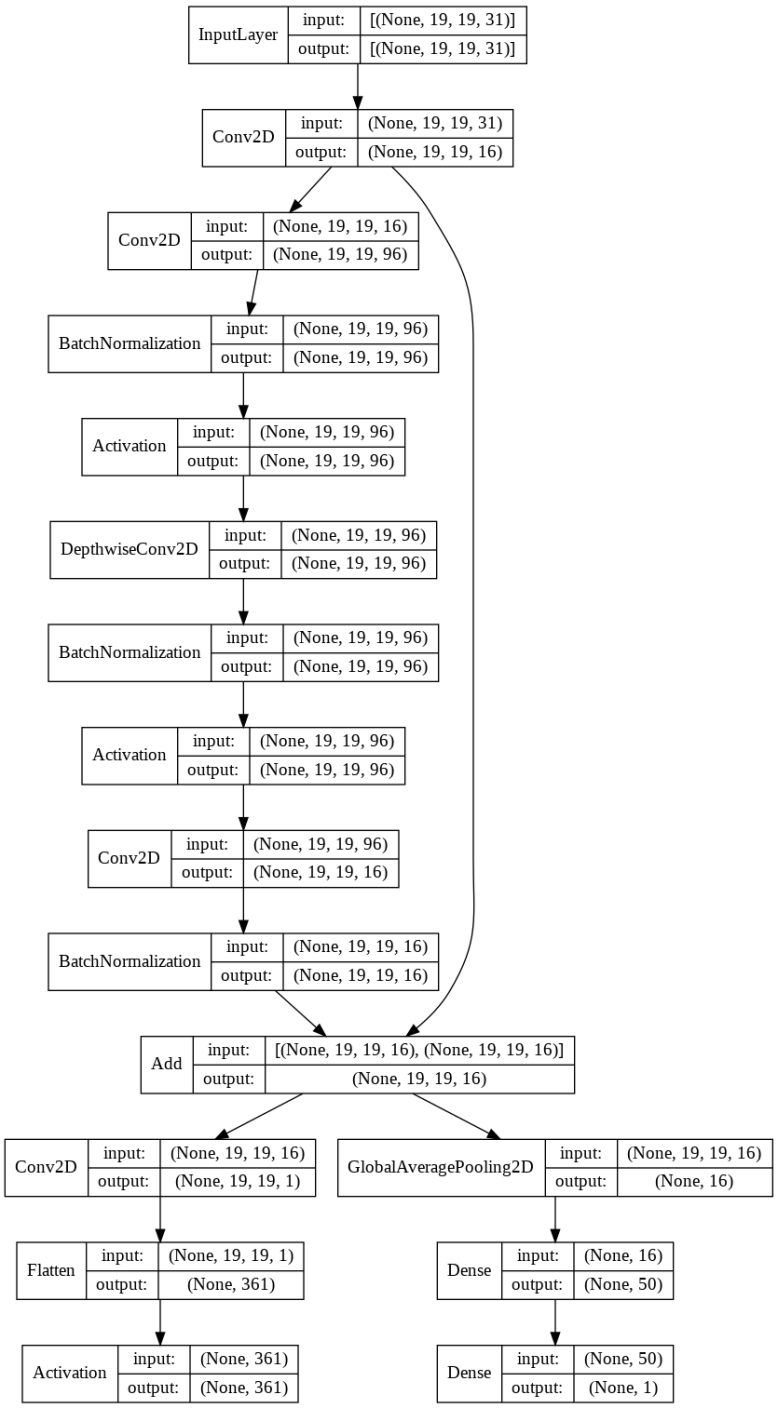


MOBILENET

We tried the model MobileNet, which consist of a big CNN with residual. It means that the inputs are directly added to the outputs. The network is built with a succession of similar blocks containing: a convolution chain with a conv layer, a depthwise conv layer both with batch normalization and activation, and a direct chain to transmit the residues.

With 20 blocks and 500 epochs we had our best results with this Net:

Policy loss	2.509
Value loss	0.637
Policy accuracy	0.378
Value accuracy	0.095



(our MobileNet but with 1 block)

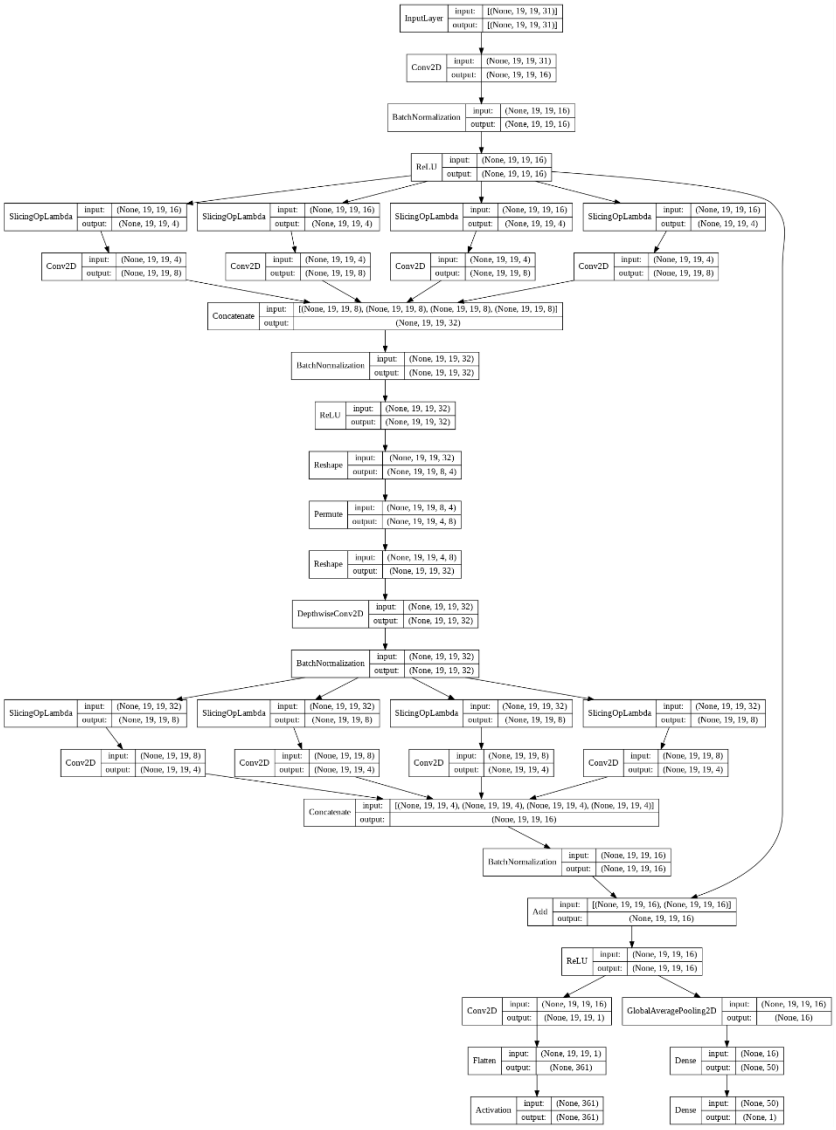
SHUFFLENET

Then we used the ShuffleNet, which is like a MobileNet, but in some blocks, we shuffle the different channel of the images.

We had our best results with 15 blocks and 500 epochs:

Policy loss	2.846
Value loss	0.662
Policy accuracy	0.329
Value accuracy	0.105

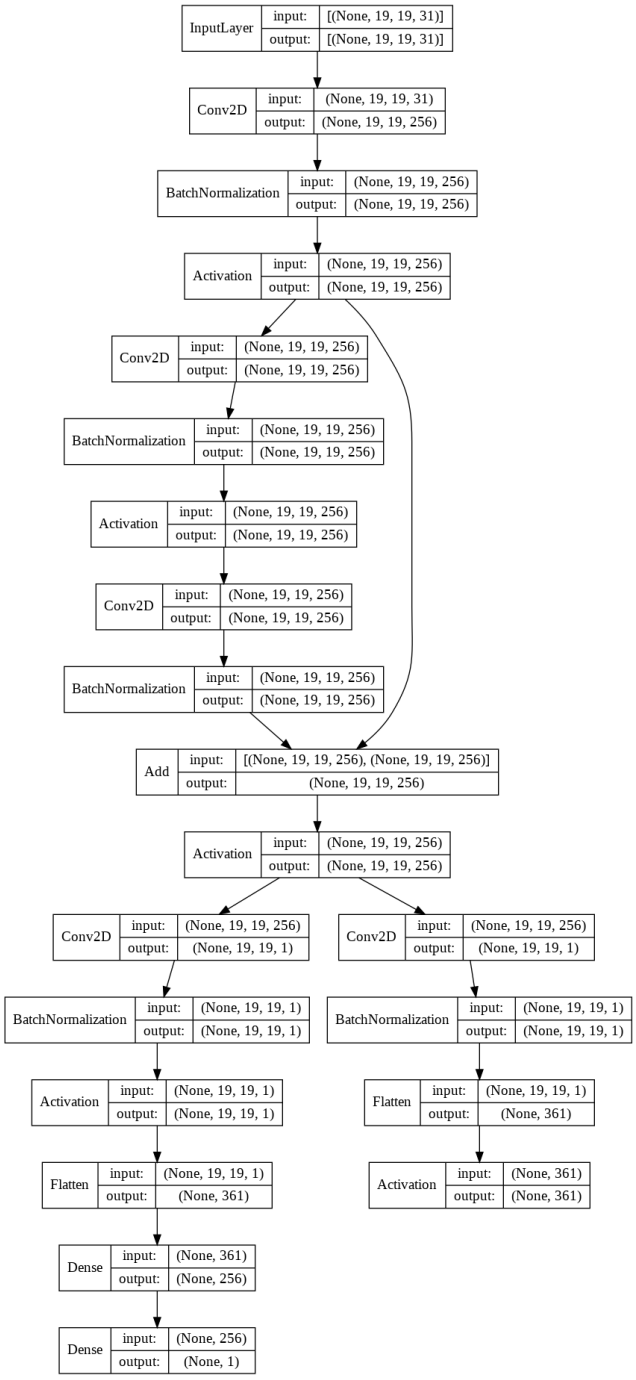
So the results are better with the MobileNet



(our shufflenet but with one block)

LIGHT ALPHAGO

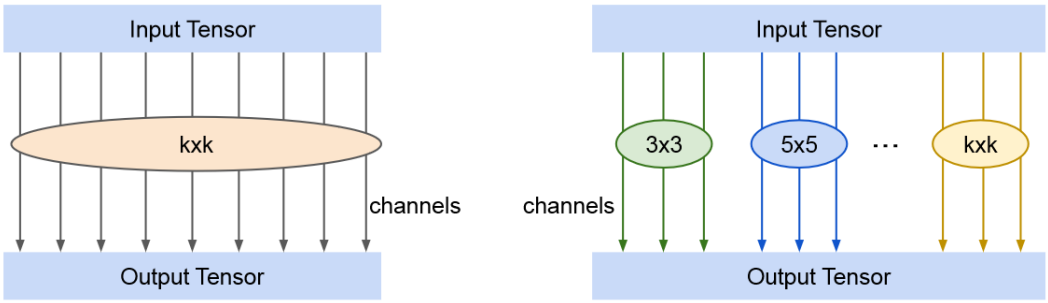
Next, we implement a light version of the AlphaGo networks. It is really similar to MobileNet, with a succession of blocks containing a direct residual chain and a convolutional chain, however the blocks are much larger. We obtain very good results with 20 blocks, however we had much more than 100k parameters so we could not use this algorithm for the competition.



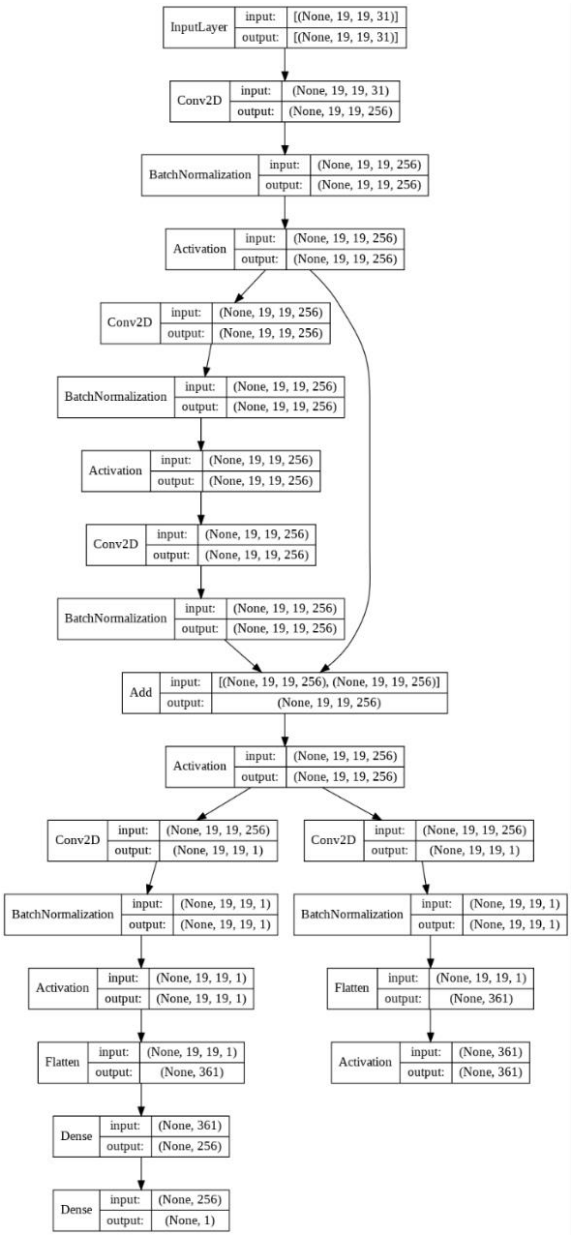
(our Light Alphago but with 1 block)

MOBILENET WITH MIXCONV

The next step to enhance our algorithm is to replace de depthwise convolution layer by a mix convolution layer. It means that different kernels of different size will apply on the different group of channels of the image.



(a) Vanilla Depthwise Convolution (b) Our proposed MixConv

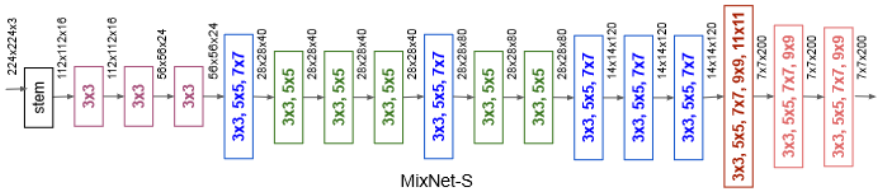


Results are not brilliant, so we pass quickly to the next algorithm (after 500 epochs

Policy loss	2.729
Value loss	0.652
Policy accuracy	0.344
Value accuracy	0.100

MIXNET

MixNet is proposed by [1]. It is roughly a succession of different MixConv layer with different number of kernels. We use the model MixNet-s a bit modified to correspond to answer our problem and have less than 100k parameters.



The results are not so bad not as good as MobileNet. After 400 epochs:

Policy loss	4.010
Value loss	0.668
Policy accuracy	0.170
Value accuracy	0.107

IV. Conclusion

Finally, as MobileNet was the best algorithm we retrain it with 1000 epochs and 20 blocks to have the following performances:

Policy loss	2.379
Value loss	0.643
Policy accuracy	0.400
Value accuracy	0.097

References

[1] MixConv: Mixed Depthwise Convolutional Kernels, Mingxing Tan, Quoc V. Le, Google Brain, Dec' 19