Performing Regression on Complex Data using a Squeeze-and-Excitation Residual Neural Network, Chess as a Model System

Gaëtan Serré
gaetan.serre@universite-paris-saclay.fr

université PARIS-SACLAY

Abstract

As neural networks for image recognition become more and more powerful and AI becomes more and more present in all fields, it would be very helpful to be able to use these models with non-image data and get good performance. This paper aims to show that this is possible through an model system very related to artificial intelligence: chess.

We can simplify a chess engine in two main components: an evaluation function, which is a function that takes a chess position as input and returns a score (usually, > 0 if Whites are winning and < 0 if Blacks are winning) and a search algorithm which will uses the evaluation function to find the best move to play in a given position. Generally, we ask the search algorithm to search at a certain depth d meaning that our engine will return the best move to play by searching d moves in advance.

I introduce GAiA, a chess engine which uses a Squeeze-and-Excitation[3] residual network as an evaluation function and the Stockfish[5] search algorithm which is known for its efficiency. GAiA's neural network try to reproduce the evaluation function of Stockfish 14. With only few hours of training, GAiA's neural network has an accuracy of 0.93 (using the coefficient of determination).

These results suggest that Squeeze-and-Excitation residual networks, which are the state-of-the-art neural networks for image recognition, can be used with data that are not images.

1 Introduction

Artificial intelligence for image recognition is becoming increasingly powerful. Since 2015, thanks to the ResNet[2] architecture, We achieve astounding performance on the ImageNet database. Furthermore, in 2017 is introduced Squeeze-and-Excitation[3] network architecture which significantly improve the performance of ResNet with almost no computation costs.

The question is: can we use this architecture on data that are not originally images. If so, we just have to encode our data in 3-dimensional tensors and look for the number of Squeeze-and-Excitation residual blocks (Figure 2) that optimizes our results to get excellent performance. The challenge is to know if Squeeze-and-Excitation networks are generalizable to any kind of machine learning problem.

As a chess fan, I have chosen to show that it was possible to create a high-performance chess engine using this type of neural network. I have called this chess engine GAiA.

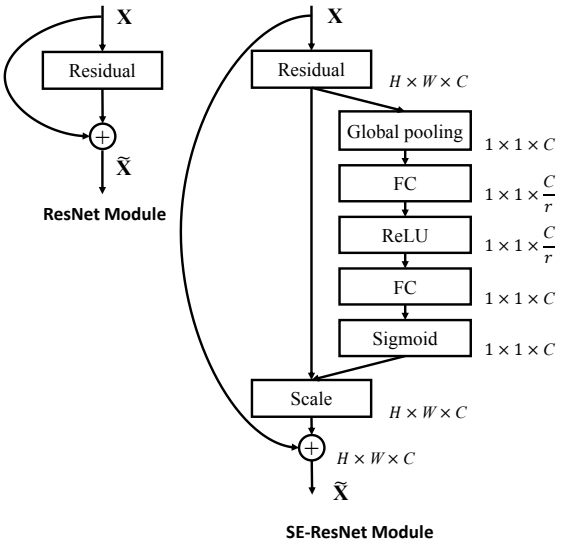
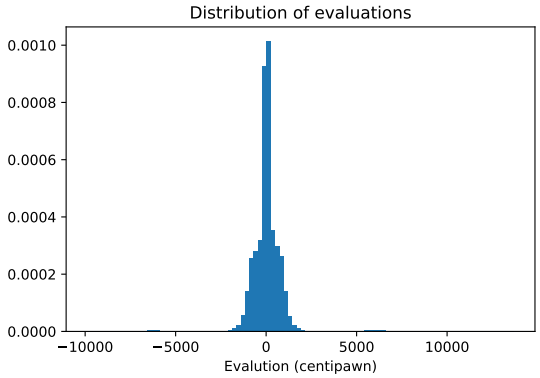


Figure 2: The schema of the original Residual block (left) and the SE-ResNet block (right)

2 Data

GAiA's neural network tries to recreate the evaluation function of Stockfish 14, a open-source world reknown chess engine. It uses heuristics function written by human experts to evaluate position. Stockfish also uses a classical alpha-beta game tree search with plenty optimizations to find the best move. GAiA is a combination of the Stockfish game tree search algorithm and a Squeeze-and-Excitation residual neural network as evaluation function.
In order to train GAiA's neural network, I needed tons of different chess position. Lichess.org[1] is a popular, free and open-source chess platform which provides millions of games played by humans every month. From these games, I extracted millions of different positions along with their Stockfish 14 evaluation. If we look at the distribution of the evaluations (Figure 3), we can see that the overwhelming majority is close to 0 i.e draw game. Each position is encoded as a 8×8 image with 15 channels: 12 representing each chess pieces, 1 for the actual player, 1 for the en-passant square and 1 for the castling rights.



count	1,389,333
mean	30.14
std	942.35
min	9873
25%	-254.0
50%	28.0
75%	320.0
max	13,678.0

Figure 3: Distribution of the evaluations

3 Neural Network Structure

GAiA's model architecture is composed of 4 SE-ResNet blocks of 2 convolutional layers each with 64 filters and a kernel shape of 1×1 using $ReLU$ activation function. To find these hyperparameters, I tried several values and these ones seemed to be the best compromise between computation costs and accuracy. (Figure 4 & 5) The output is a fully connected layer using *Linear* readout function(Figure 6). This model architecture is designed based on Maia[4]. The loss is the Mean Absolute Error and the accuracy the coefficient of determination. I used the framework Tensorflow 2.7 to create and train the neural network. The model is trained on $\sim 400k$ positions and tested on $\sim 150k$ during 30 epochs (Figure 7).

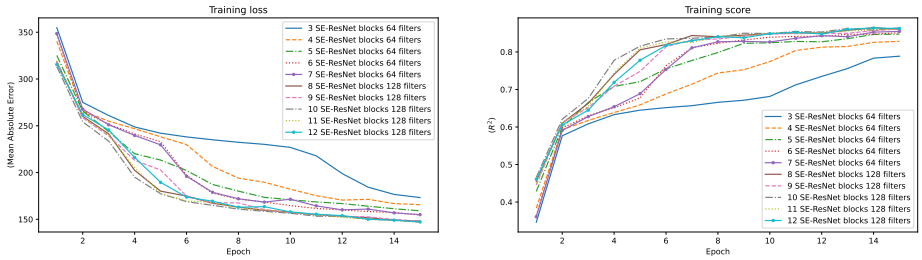


Figure 4: Loss and accuracy on train sets

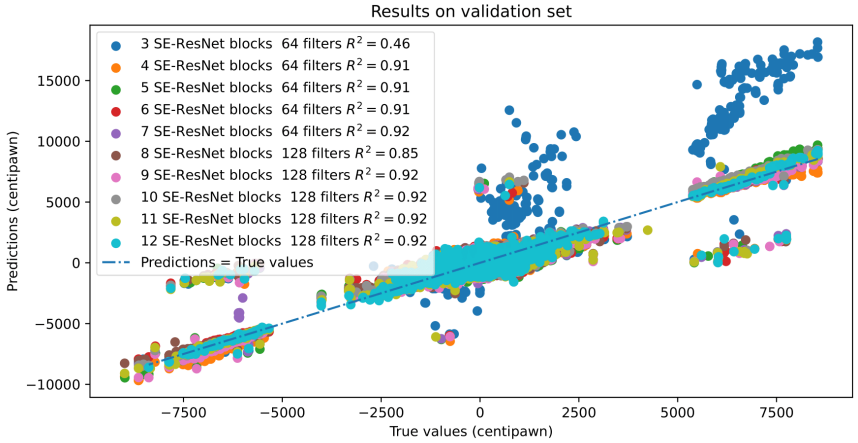


Figure 5: Score on validation sets

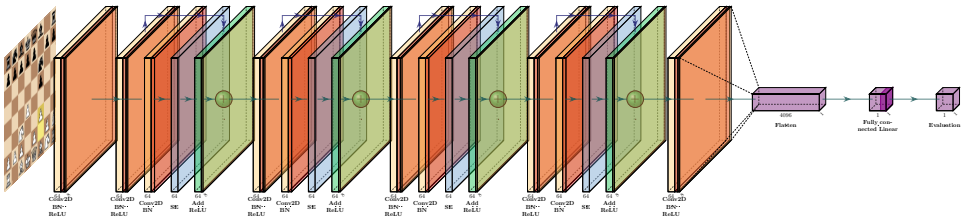


Figure 6: GAiA's neural network architecture

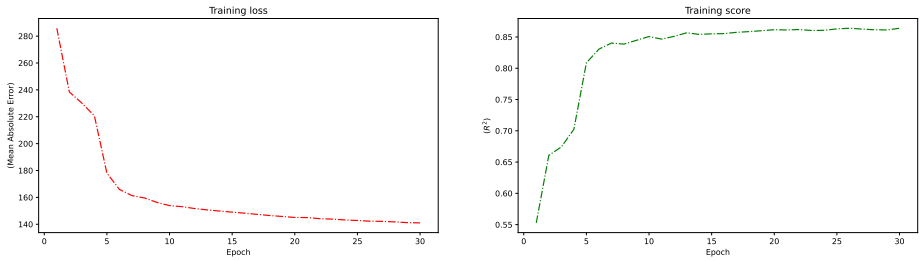


Figure 7: Loss and accuracy during training

4 Results

As we can see on the Figure 8, GAiA's neural network has great performance and I only trained it on $\sim 400k$ positions. However, some points are misplaced. These are positions where one of the kings is in check. This is due to the fact that Stockfish cannot statically evaluate position where a king is in check and so I had to search at depth 1. The resulting evaluation takes into account the next move

and therefore is much more difficult to predict. However, the evaluations of such positions produced by the network are not meaningless, and are even more accurate than those of Stockfish (Figure 9) Even if GAiA (the complete chess engine) can see much less positions per second than "classic" chess engine (inferring an evaluation of a chessboard from such a complex neural network is much more time consuming than use heuristics functions), it has been able to defeat many engines around 2000 elo on Lichess.org.

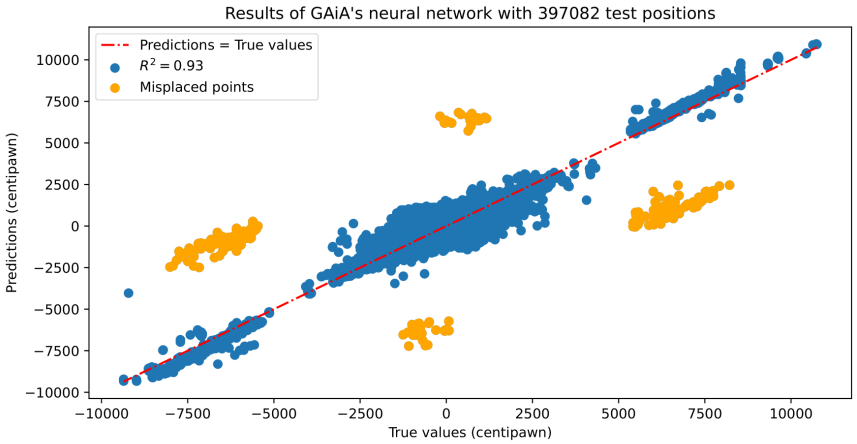
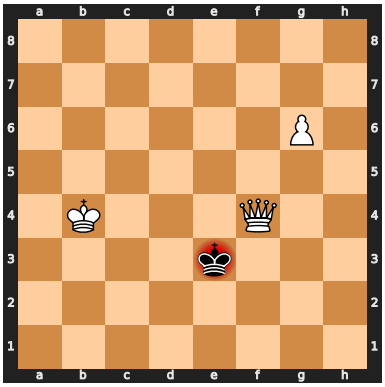
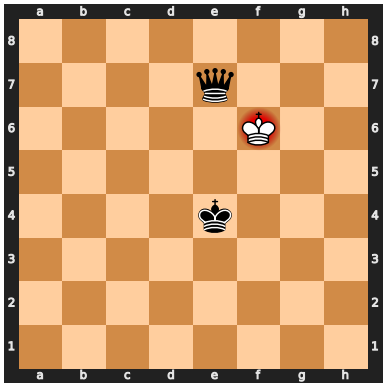


Figure 8: GAiA's neural network results



(a) Stockfish score: 863, GAiA score: 6348



(b) Stockfish score: -55, GAiA score: -6287

Figure 9: Example of misplaced positions

5 Conclusion

Artificial intelligence is being used in more and more fields. It is becoming very complex to design models capable of answering very precise problems. We could see through this article that it was possible (for some problems) to transform our data into images in order to use a neural network specialized in image recognition. It may not be the most efficient model but it allows to have very good performances easily. In my chess example, I was able to design a chess engine able to beat any non-professional human player in only a few hours of training. It would be interesting to find other problems that can be solved in this way to corroborate these results.

References

[1] T. Duplessis. Lichess. `https://lichess.org/`, 2021. [accessed 23-November-2021].

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[3] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks, 2019.

[4] R. McIlroy-Young, S. Sen, J. Kleinberg, and A. Anderson. Aligning superhuman ai with human behavior, Jul 2020.

[5] J. K. Tord Romstad, Marco Costalba. Stockfish chess engine. `https://stockfishchess.org/`, 2021. [accessed 23-November-2021].