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

Gaëtan Serré[a]

[a]gaetan.serre@universite-paris-saclay.fr

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 and a search algorithm which will uses the evaluation function to find the best move to play in a given position.
I introduce GAiA, a chess engine which uses a Squeeze-and-Excitation residual network as an evaluation function. GAiA's neural network try to reproduce the evaluation function of a world reknown chess engine. 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.

Key words: Artificial Intelligence, Deep Learning, Neural Networks, Chess

Introduction

Artificial intelligence for image recognition is becoming increasingly powerful. Since 2015, thanks to the ResNet[4] architecture, we achieve astounding performance on the ImageNet database. Furthermore, in 2017 is introduced Squeeze-and-Excitation[5] network architecture which significantly improve the performance of ResNet with almost no additional 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 1) and other hyperparameters 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.

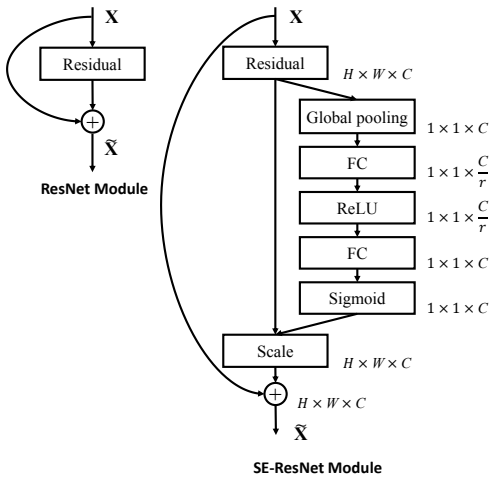


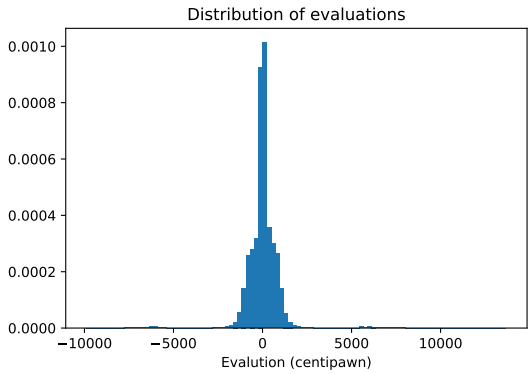
Fig. 1. The schema of the original Residual block (left) and the SE-ResNet block (right)

Data

GAiA's neural network tries to recreate the evaluation function of Stockfish 14[7], 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 network as evaluation function.

In order to train GAiA's neural network, I needed tons of different chess position. Lichess.org[3] 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 2), 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

Fig. 2. Distribution of the evaluations

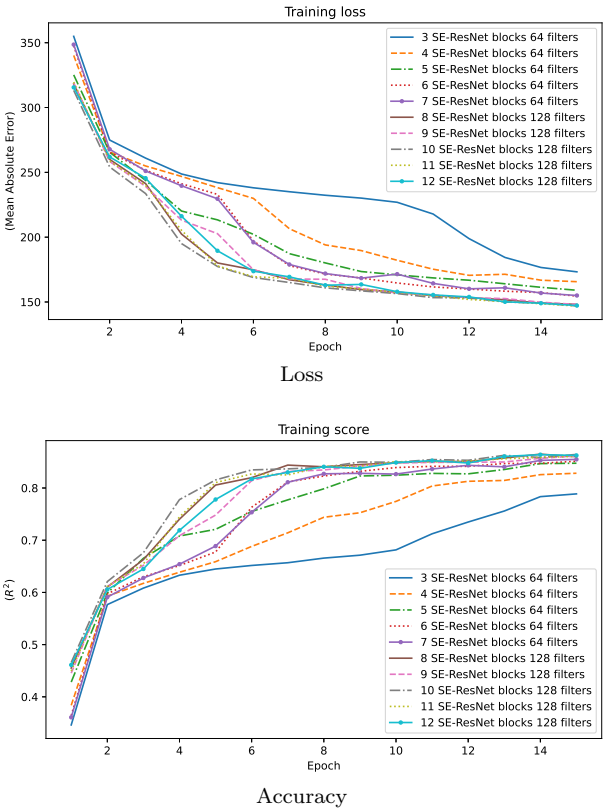


Fig. 3. Loss and accuracy on train sets

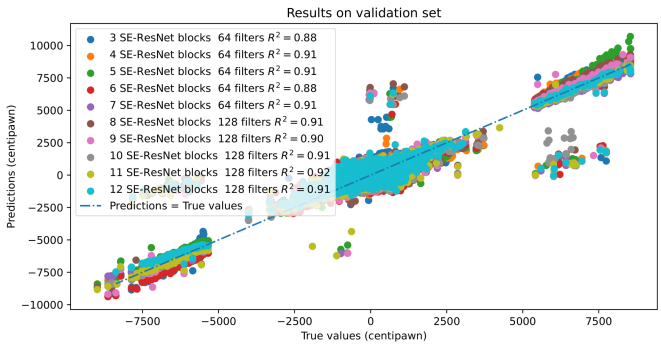


Fig. 4. Score on validation sets

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 3 & 4) The output is a fully connected layer using *Linear* readout function (Figure 5). This model architecture is designed based on Maia[6]. The loss is the Mean Absolute Error[9] and the accuracy is the coefficient of determination[8]. I used the framework Tensorflow 2.7[1] to create and train the neural network. You can see an overview of the model in Figure 5 and Table 1.

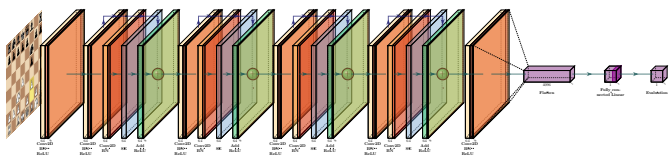


Fig. 5. GAiA's neural network architecture

Name	GAiA's network
Model type	Squeeze-and-Excitation residual network
SE-ResNet blocks	4
Convolutional filters	64
Loss function	Mean Absolute Error
Accuracy function	Coefficient of determination
Batch size	32
Epochs	30
Optimizer	ADAM

Table 1. SEResNet configuration

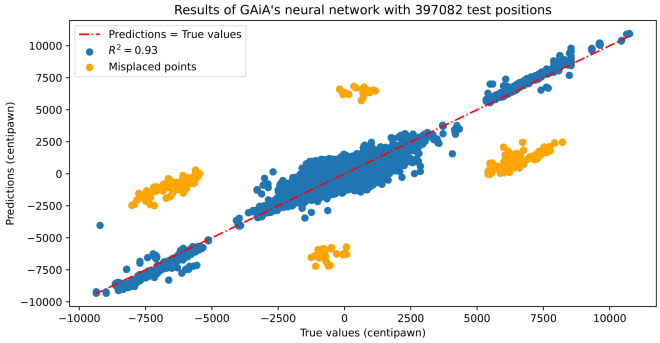
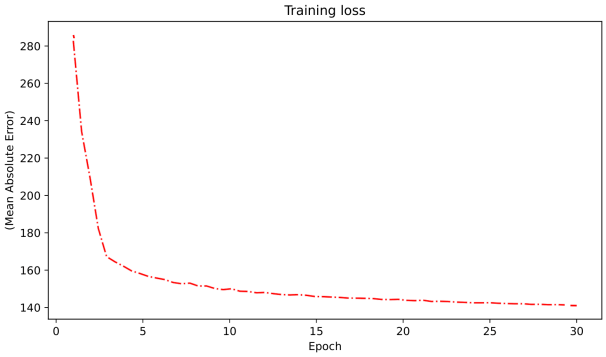


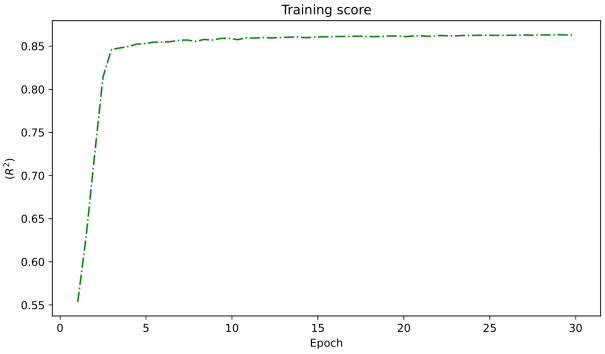
Fig. 7. GAiA's neural network results

Results

The network is trained on $\sim 1,400k$ positions during 30 epochs (Figure 6) and tested on $\sim 400k$ positions As we can see on the Figure 7, GAiA's neural network has great performance. 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 (one move ahead). 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 quite close to a static evaluation of the position (Figure 8).

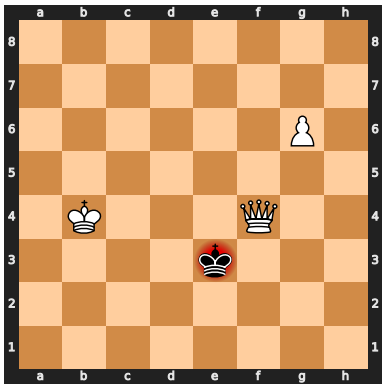


Loss

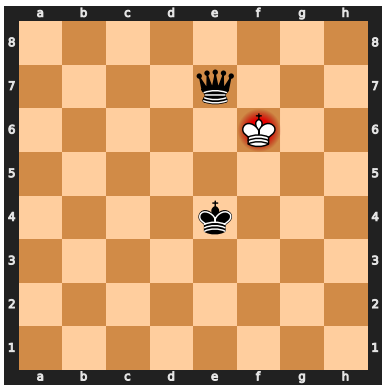


Accuracy

Fig. 6. Loss and accuracy during training



Stockfish score: 863, GAiA score: 6348



Stockfish score: -55, GAiA score: -6287

Fig. 8. Examples of misplaced positions

Engine

Now that I have the evaluation function of GAiA, I still need the program which will search in the game tree and use this function in order to have a working chess engine. Moreover, a chess engine is not just about the search algorithm and the evaluation function (even if these are the most important parts) but also the time management (how much time should the engine use to play a move in a real game) and many other things. I chose to use the Stockfish source code as a basis for GAiA because their algorithms are efficient and the code is really well written. In addition, I also needed a way to infer a score from a position

using GAiA's neural network. Since, Stockfish (and therefore GAiA) is written in C++, I chose to use ONNX[2], which is an awesome library created in 2018 for inferring and even training artificial intelligence model. ONNX is very optimized and supports many backends such as CUDA or TensorRT. However, even though ONNX is fast, GAiA can see much less positions per second than "classical" chess engine (inferring an evaluation of a chessboard from such a complex neural network is much more time consuming than use heuristics functions). Still, GAiA has been able to defeat many engines around 2300 elo on Lichess.org.

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. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow, Large-scale machine learning on heterogeneous systems, 11 2015.
2. ONNX Runtime developers. ONNX Runtime, 11 2018.
3. Thibault Duplessis. Lichess. `https://lichess.org/`, 2021. [accessed 23-November-2021].
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
5. Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.
6. Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior, Jul 2020.
7. Joona Kiiski Tord Romstad, Marco Costalba. Stockfish chess engine. `https://stockfishchess.org/`, 2021. [accessed 23-November-2021].
8. Wikipedia. Coefficient of determination — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Coefficient%20of%20determination&oldid=1060800537`, 2022. [Online; accessed 11-January-2022].
9. Wikipedia. Mean absolute error — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Mean%20absolute%20error&oldid=1053388699`, 2022. [Online; accessed 11-January-2022].