Proposal Masterthesis

Evaluation of possible improvements to increase sample efficiency of the AlphaZero algorithm.

COLIN CLAUSEN

28.11.2019

Dozent: PROF. DR.-ING. SVEN TOMFORDE
Betreut von: SIMON REICHHUBER

Contents

1 Basics AlphaZero

Beschreibe die Entwicklung von AlphaGo zu AlphaGoZero zu AlphaZero zu MuZero. Gehe auf die Funktionsweise von AlphaZero genauer ein.

2 Previous work

[1] schlägt vor einen dritten Ausgabekopf zum Netzwerk hinzuzufügen, der die

3 Motivation and possible improvemennts

The AlphaZero algorithm is a way to attain super human playing strength in arbitrary board games, even ones that before were dominated by human players such as Go. At least for more complex games like Go this is however still only possible by using very substantial amounts of processing power, thousands of machines clustered together, making it impossible to leverage the algorithm on hard problems for anyone but the largest organizations.

This motivates the search for improvements that allow to achieve results with lower requirements on computational power, preferably without losing the generality of AlphaZero. Most time during AlphaZero training is spent on generating example games, thus it follows that to improve learning speed either more learning has to happen per game or games have to be played out faster.

Previous work has established that large efficiency gains are possible and further improvements shall be systematically investigated and compared to previous improvements as well as a baseline.

3.1 Means of evaluation

To be able to experiment fast without massive usage of computing resources simpler games are typically used when evaluating AlphaZero. Connect 4 is suggested to be used as a primary evaluation game, because classical methods can achieve perfect play and thus evaluation of training progress can be done by judging moves made by AlphaZero in absolute terms. In similar fashion the game of Hex on a 9x9 board can also be used, for example this is done in [1], as there are datasets of near-perfect play available, this however seems to be a slightly inferior approach, as the network might be penalized in situations that have multiple optimal moves.

If time and computational resources permit another idea for evaluation on a more complex game would be to implement Chess and judge training progress by the average centipawn loss as measured by a strong Chess engine such as Stockfish, as can be done to judge human players [4]. Since Chess is not a solved game this is less accurate than Connect 4, but has the advantage of giving results on a much more challenging domain.

3.2 Understanding the learning process

An interesting property of the AlphaZero algorithm is the perceived stability of the training. Other learning systems often struggle to learn two players games in a pure self-play regime, because they get trapped in an endless cycle of learning and unlearning the same mistakes.

A typical way to prevent this normally would be to include earlier iterations of a learner in the learning process, to prevent forgetting previous knowledge. This was originally done in AlphaGo and AlphaGoZero, however with further work it became clear that even in the AlphaZero regime, with no measures taken at all to prevent forgetting learning works. This raises the question of how stable the training truly is. It might be that some amount of forgetting still occurs, just not enough to stop progress.

To research this a dataset of specific game situations could be used, checking which positions are played correctly after every learning iteration. This way it can be compared how every learning iterations adds or removes more understanding of the game to the used neural network. A better understanding of the learning progress could be valuable to suggest additional ways to speed up training.

3.3 Suggested improvements to be researched

4 Introduction

Example introduction section thingy.

4.1 Related work

Example subsection about related work, read all of them: [3], [2]

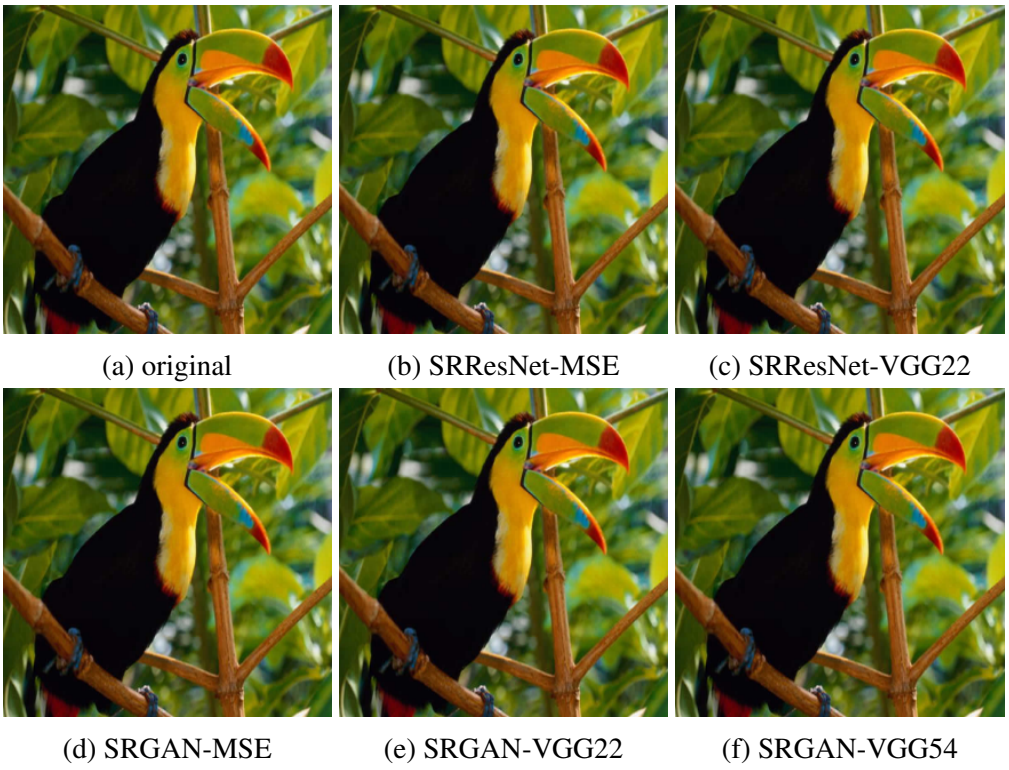
4.1.1 Example formula, also this subsubsection has a label

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data(x)}}[logD(x)] + \mathbb{E}_{z\sim p_{noise}(z)}[log(1-D(G(z)))].$$

4.2 Subsection with a ref to a label

Here is the ref 4.1.1

5 Images!



(a) original (b) SRResNet-MSE (c) SRResNet-VGG22

(d) SRGAN-MSE (e) SRGAN-VGG22 (f) SRGAN-VGG54

Figure 1: Results on one image from the BSD100 set, best viewn with zoom

5.1 Itemization

Itemization is a thing.

- SRGAN-MSE: l^{SR}_{MSE} uses the standard mean squared error.
- SRGAN-VGG22: $l^{SR}_{VGG/2.2}$ uses lower-level features of VGG for the VGG loss
- SRGAN-VGG54: $l^{SR}_{VGG/5.4}$ uses higher-level features of VGG for the VGG loss. This is the network that is named simply SRGAN, using higher level features offers the best chance for the network to focus on realistic content of the images, not just by local pixel relationships, but in a more global context.

5.2 Tables

Set5	nearest	bicubic	SRCNN	SelfExSR	DRCN	ESPCN	SRResNet	SRGAN	HR
PSNR	26.26	28.43	30.07	30.33	31.52	30.76	**32.05**	29.40	∞
SSIM	0.7552	0.8211	0.8627	0.872	0.8938	0.8784	**0.9019**	0.8472	1
MOS	1.28	1.97	2.57	2.65	3.26	2.89	3.37	**3.58**	4.32
Set14									
PSNR	24.64	25.99	27.18	27.45	28.02	27.66	**28.49**	26.02	∞
SSIM	0.7100	0.7486	0.7861	0.7972	0.8074	0.8004	**0.8184**	0.7397	1
MOS	1.20	1.80	2.26	2.34	2.84	2.52	2.98	**3.72**	4.32
BSD100									
PSNR	25.02	25.94	26.68	26.83	27.21	27.02	**27.58**	25.16	∞
SSIM	0.6606	0.6935	0.7291	0.7387	0.7493	0.7442	**0.7620**	0.6688	1
MOS	1.11	1.47	1.87	1.89	2.12	2.01	2.29	**3.56**	4.46

Table 1: Comparison of the proposed methods compared to previous methods. SRResNet represents the improvements gained from simply using a deeper network, while SRGAN represents combining the deeper network with the proposed perceptual loss.

6 Text with footnote

Since the paper was published an independent implementation of SRGAN has been created [1].

References

[1] Anonymous. Three-head neural network architecture for alphazero learning. In *Submitted to International Conference on Learning Representations*, 2020. under review.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Matej Guid and Ivan Bratko. Using heuristic-search based engines for estimating human skill at chess. *ICGA journal*, 34(2):71–81, 2011.

[1] https://github.com/aitorzip/PyTorch-SRGAN