Improving 'Learning to Act by predicting the future' with semantic segmentation of the visual stream

Antoine Guillot
ENS Paris-Saclay
Cachan, France
antoine.guil@outlook.fr

Abstract

We present an approach to improve the paper 'Learning to act by predicting the future' by Dosovitskiy and Koltun (ICLR, 2017) [2]. Their paper (referred as DFP from now on) yielded state-of-the-art results in reinforcement learning using raw sensory-motor stream. In this paper, we sensibly improve the performances of DFP by adding a new input which segment the raw visual stream and detect the different objects in this visual stream. This improved the performances in the most challenging task presented in the original paper.

1. Introduction

Reinforcement learning is a subfield of Machine Learning in which the agent learn to act to maximize some rewards from the environment. Recently, reinforcement learning has met impressive success in complex tasks. In 2012, Deep Mind manages to reach human and more than human performances in Atari Video games [4]. Last year, AlphaGo managed to beat Lee Sedol (18 times world champion of Go) at the game of Go[5].
These successes relied on on deep neural networks and classical reinforcement learning approaches which learn how to maximize the reward. DFP changes the approach and sees reinforcement learning as a specific supervised learning task. They assume that the environment provides measurements feedbacks and that the reward is a function of these measurements. Hence, instead of directly predicting the reward function, they forecast the measurements over several times periods. From this forecast, the can compute the best action a-priori and improve the behavior of the agent.
They implemented this approach in Doom, a first person shooter which provides the stream of measurement and the visual stream through its engine.

2. A formal description of DFP

This part is a short summary of the part 3) of the paper [2]. We denote $o_t =< s_t, m_t >$ the observation from the environment at a time t. s_t is the raw sensory inputs (for instance the visual stream) and m_t the measurements stream (here, the ammo, the life and the number of frags of the agent). The agent aims to predict m_t at different time offsets f=(u_{τ_1},u_{τ_2}, ...u_{τ_n}) and the reward function u can be written $u(f, g) = g^T f$ where g is the vector encoding the goal of the agent. This formulation of the goal allows extensive modularity, and an agent trained with specific goals can use other goals or a different reward decay just by changing the vector g.
To predict the future measurement p_t^{α}, we assume to have a parametric mapping F:

$$p_t^{\alpha} = F(o_t, a, g, \Theta)$$

where o_t is the input stream, a is an action, g is the goal vector and Θ are the learned parameters. From these forecasts, the best action for the agent can easily be selected:

$$a_t^* = arg\ max_a F(o_t, a, g, \Theta)$$

To ensure some exploration the agent follows an ϵ-greedy policy during the training. Given a set $D = ([o_i, a_i, g_i, f_i])_{i \leq N}$ of training examples, the learning problem can be formulated:

$$arg\ min\ L(\Theta) = \sum_{i \leq N} ||F(o_i, a_i, g_i, \Theta) - f_i||_2 \quad (1)$$

The parameters Θ are updated after every k steps and the set D is sampled from a set of the M most recent experiences. In the original paper F is a deep neural network which takes as input the measurements, the goal vector and the raw visual stream from the game (Figure 1,taken from the original paper). Our work consisted in the addition of a new input.

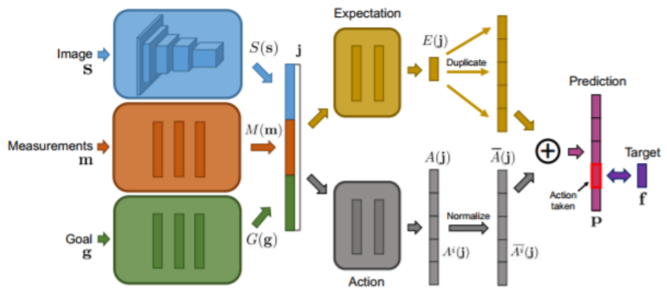

Figure 1. Network structure. The image s, measurements m, and goal g are first processed separately by three input modules. The outputs of these modules are concatenated into a joint representation j. This joint representation is processed by two parallel streams that predict the expected measurements $E(j)$ and the normalized action-conditional differences $Ai(j)$, which are then combined to produce the final prediction for each action.

Instead of only using the raw visual input, we perform a semantic segmentation on the frames from the game. The segmented images are fed to a new input and processed by a new CNN. In the next section, we detail how the segmentation is performed.

3. Semantic segmentation

Given a picture X of width W and height H, its pixels x_{ij} and a set of classes $(C_1, ..., C_N)$, the goal of semantic segmentation is to assign each pixel to one of the N classes. Typically these classes represent a type of objects and the goal is to build a mapping from the pictures space $\mathbb{R}^{W\times H}$ to the classes space $(0,1)^{W\times H\times N}$. The channel K of the normalized output can be seen as a heat-map of the probability that a pixel (i,j) belongs to the class K. The state-of-the arts methods for segmentation relies on CNN.

3.1. Fully convolutionnal network (FCN)

This was the first method to use end-to-end learning and CNN for segmentation [3].An FCN is an encoder-decoder architecture where the output is the ground-truth segmentation of the picture. To perform the segmentation, we used a FCN which worked at different scales by directly connecting the intermediate pooling layer to the output. These shortcuts allow the network to have access to global and local information at the same time. The implementation was done by adapting the code from [1]. Our architecture is shown on Fig.2

3.2. Training and performances

The training was performed on 300,000 pictures extracted by running the pre-trained scenario provided in DFP. The input image is a 84×84 gray-scale picture. The game engine provides the ground truth segmentation and the different classes.The dataset was split between the training set (70%) and the test set (30%) The neural network was trained for 10 epochs and reached an mean intersection over union score of 0.54 over the 14 classes.

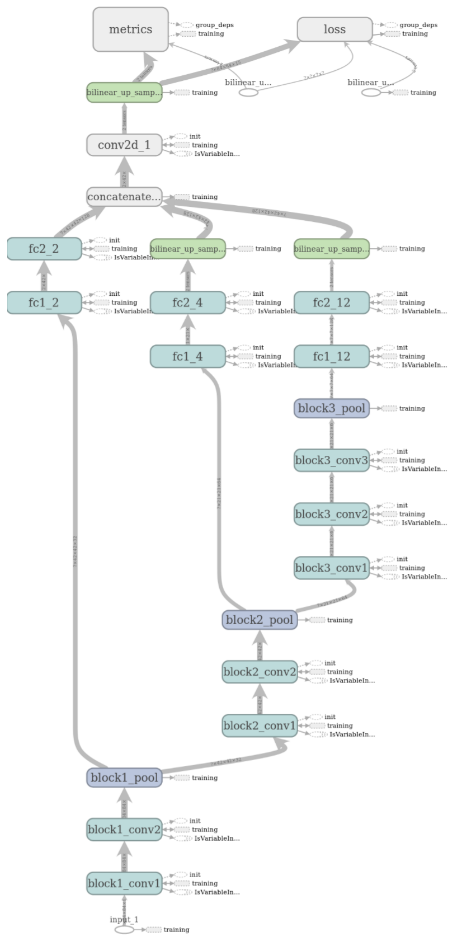

Figure 2. Architecture of the Neural Network for segmentation.

3.3. Discussion

The network is small compared to recent state-of-the-art architectures:

- when the training of the agents is performed, the segmentation network is called several millions of times and larger network led to intractable computation time.
- the input image has a low-resolution and only one channel, a larger network would not lead to sensible IoU improvements. Furthermore, the network performance are sufficient to significantly improve the agent performances.

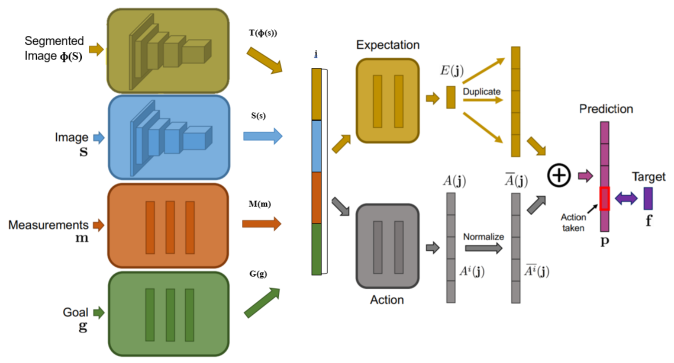

Figure 3. Network architecture with the segmentation input. Φ denote the segmentation function after the training of the segmentation neural network. T denotes the processing of the segmented picture by the agent. Other notations are the one from Fig.1

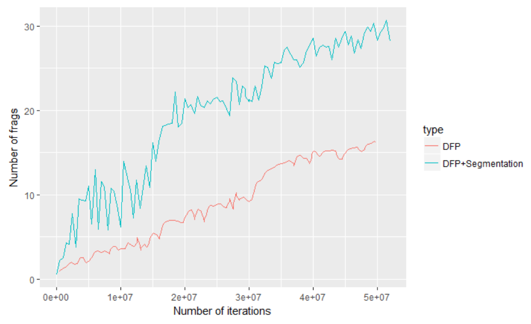

Figure 4. Evolution of the number of frags with the number of iterations

4. Final model and results

The architecture is presented on Fig.3. The segmented image is added as an input to the network. The segmented image is a tensor of dimension $84 \times 84 \times 14$.The segmented visual stream is processed through a network with the same architecture that the network which process the raw visual stream. The segmentation neural network weights are frozen during the training of the agent.

4.1. Training

The training was done on 800,000 batch similarly what was done in the original paper. We only trained the model on the environment ”D4: Battle 2” which is the most complex environment in the original paper. We used the ”basic” architecture (as mentioned in the original paper) for the different neural networks in order to ensure fast convergence.The new net converges faster and reach better performances than the original net(Fig.4).

4.2. Results

The trained agent was tested on D4 Battle and compared to the pretrained agents from the paper. The results slightly differs from the paper since they used a specific version of Doom. Our approach sensibly outperform the original paper and yield state-of-the-art results.

	Ours	DFP (our)	DFP (paper)
Mean frags	24.77	11.35	16.5

4.3. Generalization of the agent to new environments

In the original paper, the agents were also tested in different environment from the one they were trained in. In our case, we assess the performance of an agent trained in D4 Battle in the other environments. The environments D3-tx and D4-tx use different textures from D4-Battle and D3-Battle.

	Ours	DFP (our)	DFP (paper)
D3-Battle	23.6	14.85	17.8
D3-tx	7.77	12.52	8.1
D4-tx	2.1	6.2	5.1

This table can lead us to two important conclusions:

- The addition of a segmentation improves the performance in new environment with the same visual setting (D3-battle).
- When the visual environment changes, the performances fell drastically.Actually the mean IoU score of the segmentation net in the new environment is only around 0.27. Hence, the segmentation network feeds wrong information to the general network. The general network has been trained in an environment where the segmentation was working well and interprets the output of the segmentation net as being correct (when they are often wrong).

5. Possible extensions of the project

The possible extensions of the project directly raise from the limitation we've shown before. The segmentation net needs to generalize better to new environment:

1. The segmentation net could be retrained in new and more general environments. Replacing the original segmentation net by one that has been trained in the new environment would increase the generalization abilities of the agent.
2. In our case, we froze the weights of the segmentation net before the training. Unfreezing these weights in a new environment would lead the segmentation net to learn the visual features of this environment and to better performances.

References

[1] aurora95. Keras-tensorflow implementation of fully convolutional networks for semantic segmentation (https://github.com/aurora95/keras-fcn).

[2] Dosovitskiy and Koltun. Learning to act by predicting the future. *ICLR*, 2017.

[3] Long and Shelhamer. Fully convolutional networks for semantic segmentation. 2016.

[4] V. Mnih and al. Playing atari with deep reinforcement learning. *NIPS*, 2013.

[5] Silver and al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.