Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Object-oriented dynamics prediction

Seminar Paper Outline

Neural Networks

Malte Korn
Matr.Nr. 6549067
malte.korn@studium.uni-hamburg.de

17.07.2020

Abstract

The Object oriented dynamic predictor (OODP) created by Zhu et al. detects different game object and their interaction to improve reinforcement learning. In this paper a reproduction of the OODP is described and the results are compared to the original OODP.

Contents

1 Introduction

Deep reinforcement learning produces outstanding moves by learning game environments. In the last decade Deepmind's computer programs gained human-like or even better performance in video games like Atari games [1], the board game Go [2] and even more complex games like "Starcraft 2"[3]. Despite these impressive results the computer programs fail when it comes to a new game environment. The common deep reinforcement learning method is based on specific pixel formations. That is why they are often failing in new game environments, even when they are selected from the same game genre.
The model-based approach is one method to overcome these problems. In difference to model-free methods, model-based methods try to construct a model of the game environment and simulate the next step based on this model[4].
The object-oriented dynamic predictor is a way to introduce model-based methods in the field of deep reinforcement learning on game environments. The goal is to detect various objects (as well dynamic as static) and their interactions with each other to predict the next frame based on the current frame and chosen action.

2 Background

2.1 Model-based method

In the field of deep reinforcement learning there is the model-free and model-based approach. In the model-free method there are no restrictions for the deep neural network. The policies are learned directly from the inputs.
Apart from that, there is the model-based approach. Here the deep neuronal network predicts a model of the environment first. At model-based reinforcement learning, the policies are based on this model. When it comes to model-based methods of reinforcement learning on game environments, detecting dynamic and static objects and predicting their interactions is a promising approach[4]. The Object-Oriented Dynamic Predictor is based on this model-based approach.

2.2 Object-Oriented Dynamic Predictor

A novel unsupervised end-to-end neuronal network framework was developed by the group of Zhu et al.[5] called Object-Oriented Dynamic Predictor. The framework is based on three main parts; the Object Detector to detect dynamic as well as static objects in the input image, the Dynamic Net to learn the motion and interaction between those objects, and the Background Extractor to extract the parts of the game environment which do not change over time.
Based on these three models and the Spatial Transformer Network [6], the Object-Oriented Dynamic Predictor is able to predict the next frame based on the current frame and action.

2.2.1 Object Detector

The most important task is to detect the objects of the game environment and differentiate between static and dynamic objects. The objects are represented by a mask $M_{O_i} \in [0,1]^{H \times W}$, where H and W denote the height and width of the input image[5]. Each object mask is created by an individual trained model with the same model architecture. It is necessary to define whether a object mask is static or dynamic for the Dynamic Net. Usually, the number of dynamic and static objects is unknown. So a slightly higher number of object detection models is defined. Unneeded detection models will consist of zeros by the end of successful training.
To avoid overlays in the object masks, they are adjusted by a pixel-wise softmax function.

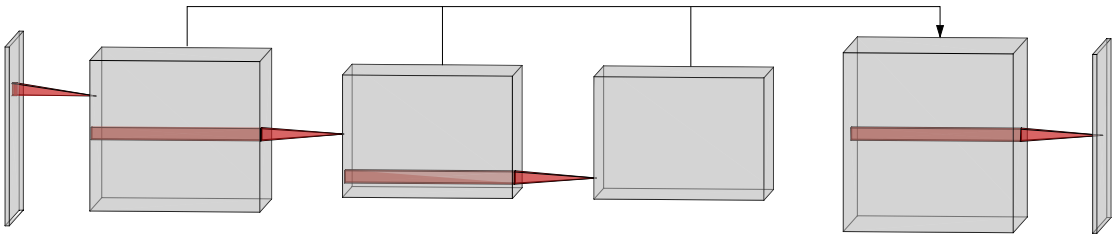

Figure 1: Object Detection Model

2.2.2 Dynamics Net

The prediction of the movements of the dynamic object and the interactions with the static objects is done by the Dynamic Net. First the pixel coordinates $((x_{D_i}, y_{D_i})$, where D_i is the i-th dynamic object mask) of the center of the detected dynamic object are calculated. An area of a defined size is cropped at this coordinate on each other object mask. In addition for each pixel of the cropped object masks, there is stored the x and y coordinates of the pixel position (figure 2). The

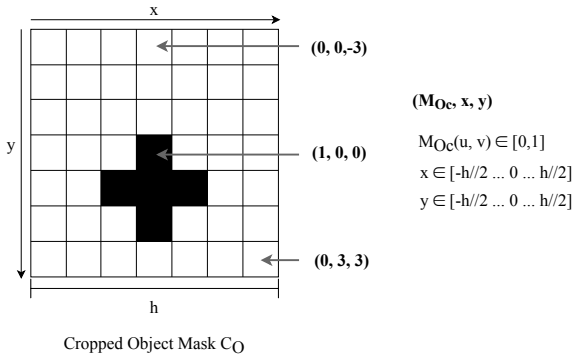

Figure 2: Example input matrix of the Dynamic Net.

Dynamic Net consists of a convolutional network for each of those cropped and modified objectmasks. The output of each model is a vector of the size $2 \times n_a$ where n_a is the number of actions. The vector represents the predicted x and y movements of the dynamic objectmask for each action.

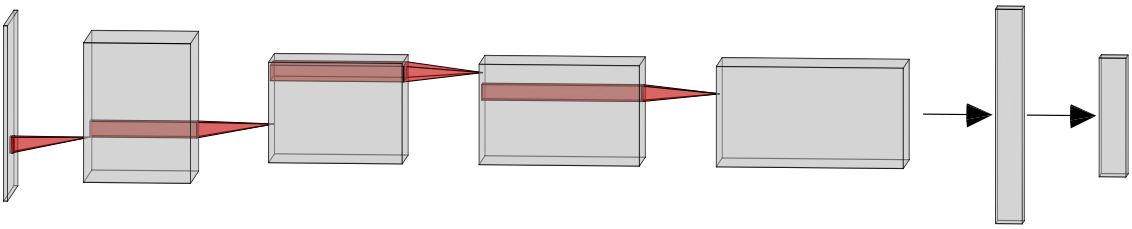

Figure 3: Dynamic Prediction Model

2.2.3 Background Extractor

The background is extracted by a basic autoencoder. The encoding is done by convolutional layers with an increased stride followed by a fully connected layer to produce the latent vector. The decoding is done by transposed convolutional layers.

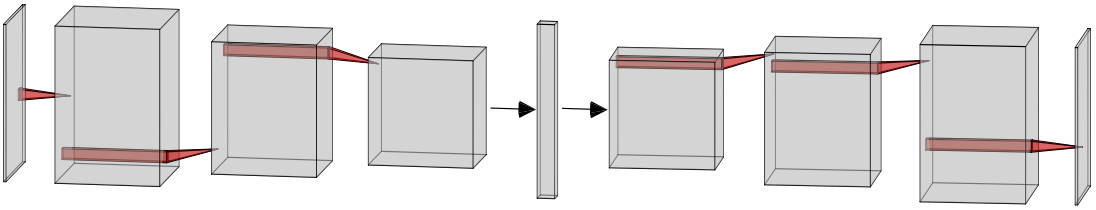

Figure 4: Background Extractor

2.2.4 Spatial Transformer Network

The Spatial Transformer Network (STM) is able to spatially transform images [6]. In the case of the OODP-Model, it is used to shift images by the predicted movements $V_{D_i}^{(t)}$ of the Dynamics Net. The shifted image is needed to rebuild the next frame $\hat{I}^{(t+1)}$, which is also the output from the OODP-Model. The output image is calculated by:

$$\hat{I}^{(t+1)} = \sum_{i}^{n_D} \mathrm{STM}(M_{D_i}^{(t)} \cdot I^{(t)}, V_{D_i}^{(t)}) + (1 - \sum_{i}^{n_D} \mathrm{STM}(M_{D_i}^{(t)}, V_{D_i}^{(t)}) \cdot I_{bg}^{(t)}.$$

2.3 Training

The interaction between the components mentioned, as well as the stream of the input image, is shown in Figure 5. The input image is processed by the Background Extractor as well as the Object Detector. Based on the detected object

mask the motion of the dynamic object mask is predicted by the Dynamic net. The output image is composed of the Background Extraction Stream and the Dynamics Inference Stream. In the Background Extraction Stream, the inverse of the spatial transformed object mask is applied on the background image, given by the Background Extractor. In the Dynamics Inference Stream the dynamic object mask is applied on the input image to receive the dynamic object, first. Then it is spatial transformed and merged with the Background Extraction Stream.
The models are fit by various of loss functions. Because of the collaboration of the different models it is necessary to train them collectively.

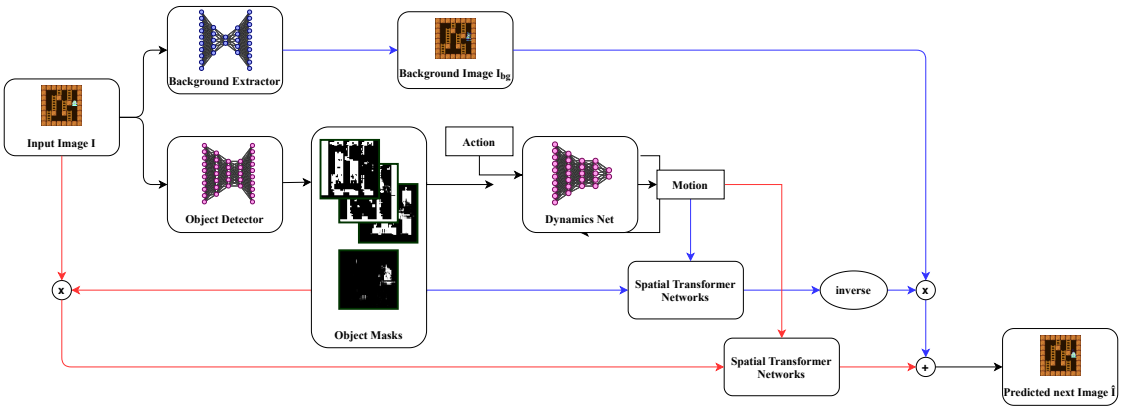

Figure 5: OODP Model. The output is composed of the Background Extraction Stream (blue) and the Dynamics Inference Stream (red).

2.3.1 Losses

The losses are taken from Zhu et al.[5]. Some are slightly modified.

Background loss:
The loss for the background detection is calculated by,

$$\mathcal{L}_{background} = \|I_{bg}^{(t+1)} - I_{bg}^{(t)}\|.$$

where $I_{bg}^{(t)}$ is the output of the background detector for the current frame and $I_{bg}^{(t+1)}$ for the next frame.

Entropy loss:
The object masks created by the Object Detector Model supposed to consist of either 0 or 1. A entropy loss function is used for determination,

$$\mathcal{L}_{entropy} = \|\sum_{c}^{n_O} -M_{O_c} \log M_{O_c}\|.$$

where M_{O_c} is the c-th object mask detected from the Object Detector Model.

Highway loss:
The prediction of the movement of the dynamic mask is calculated by the calculated coordinates of the dynamic mask form the current an the next image. This loss is only meaningful, when the dynamic mask and so the calculated coordinates are close to the truth. However bad predicted coordinates of the object masks do not impair the training. The highway loss is predicted by:

$$\mathcal{L}_{highway} = \sum_{i}^{n_D} \|(x_{D_i}, y_{D_i})^{(t)} + V_{D_i}^{(t)} - (x_{D_i}, y_{D_i})^{(t+1)}\|^2.$$

where V_{D_i} is the predicted movement of the dynamic object mask D_i based on the given action.

Prediction loss:
The prediction loss is calculated by the squared error of the next image output of the OODP-Model $\hat{I}^{(t+1)}$ and the true next image $I^{(t+1)}$.

$$\mathcal{L}_{prediction} = \|\hat{I}^{(t+1)} - I^{(t+1)}\|^2.$$

Reconstruction loss:
The reconstruction loss is calculated by the squared error of the current true image $I^{(t)}$ and the reconstruction by the sum oft the dynamic object mask $M_{D_i}^{(t)}$ applied current image $I^{(t)}$ to cut out the players agent and the inverted object mask applied on the predicted background $I_{bg}^{(t)}$,

$$\mathcal{L}_{reconstruction} = \|\sum_{i}^{n_D} M_{D_i}^{(t)} \cdot I^{(t)} + (1 - \sum_{i}^{n_D} M_{D_i}^{(t)}) \cdot I_{bg}^{(t)} - I^{(t)}\|^2.$$

Consistency loss:
The consistency loss is calculated by the squared error of the dynamic object mask of the next image and the spatial transformed dynamic object mask of the current image.

$$\mathcal{L}_{consistency} = \sum_{i}^{n_D} \|M_{D_i}^{(t+1)} - \mathrm{STN}(M_{D_i}^{(t)}, V_{D_i}^{(t)})\|^2.$$

Overall loss:
The loss for fitting the models is created by the sum of the calculated losses. Some of the losses are need to adjust by a factor λ.

$$\mathcal{L}_{all} = \mathcal{L}_{background} + \lambda_{ent}\mathcal{L}_{entropy} + \mathcal{L}_{highway} + \lambda_{pre}\mathcal{L}_{prediction} + \lambda_{rec}\mathcal{L}_{reconstruction} + \mathcal{L}_{consistency}$$

3 Implementation

The implementation is based on the Github project from Zhu et al.[1]. Different from the Github project, in this implementation the models were rebuilt with Keras and were trained by a precast data set. The calculations were adjusted to the slightly different model.

3.1 Reproduction of the given Github project.

3.1.1 Game environment

The game of choice is monsterkong from the openAI gym toolkit[2]. It is a 2D game with a controllable agent. The actions include up, down, left, right and jump. For the sake of simplicity some elements were removed from the game. The remaining elements are the player's agent as a dynamic object. Furthermore there are blocks and ladders left as static objects. There are 15 different levels with different arrangements of the static objects. At every game start the players agent is placed randomly in the game environment.

3.1.2 Data set

The data set was created by running random actions and saving the current and next state, and the action performed.

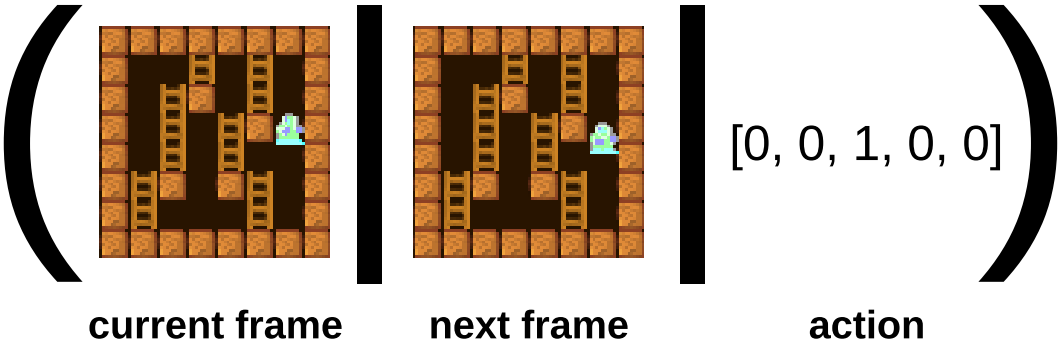

Figure 6: A example entry of the data set. The RGB pixel values of the current frame and the next frame are mapped between 0 and 1. The action is represented by a one-hot vector.

3.1.3 Training

The models where trained by a data set build out of five different levels. For each level 32768 data entries where sampled. The models where trained by a batch size

[1]https://github.com/mig-zh/OODP
[2]https://gym.openai.com/envs/MonsterKong-v0

of 16 and the data set where processed 175 times. The validation was run on ten different levels.

4 Analysis

After 1,700,000 training steps reasonable results were received. In comparison, the related paper write about a maximum steps of 1,000,000 with the same batch size. The performance of the model was defined by the predicted movement of the dynamic object[5]. For the valuation of the performance the true movement difference between the current and the next image has to be determined manually. This time consumption part has not done jet and will be evaluate soon.
Further it is important to evaluate the object detection. In figure 8 the detected object masks of the input image (figure 7 (1)) are shown. The detection of the dynamic object and the dark brown background parts are reasonably successful. Furthermore the detection of the blocks and ladders is not that successful. These parts are split up in the object masks 3 and 4.

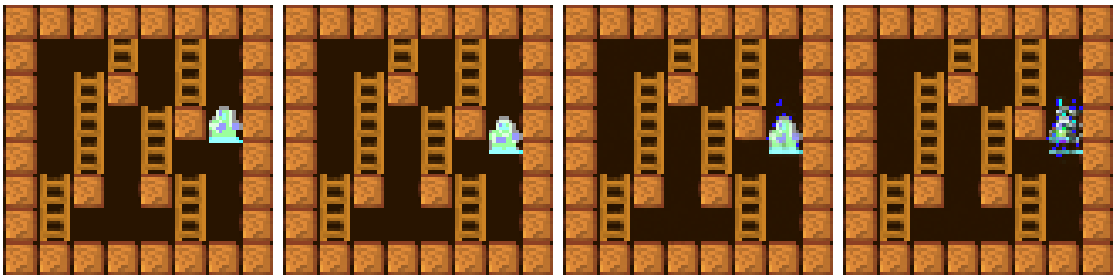

Figure 7: Input and outputs of the OODP. (1) current frame, also input of the Background Detector and the Object Detector; (2) next frame, necessary for training, for evaluation of the predicted next frame for example; (3) predicted next image, also output of the OODP; (4) extracted background, output of the Background Extractor.

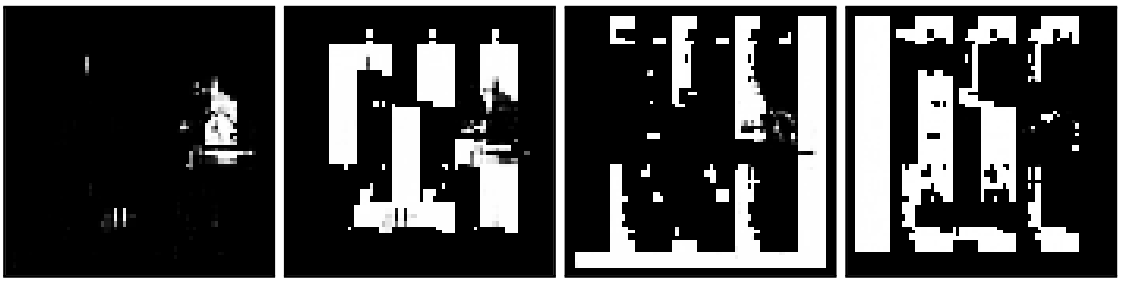

Figure 8: Detected object masks. (1) dynamic object mask; (2-4) static object masks.

5 Conclusion

The implementation of the OODP does not reach the performance of the original OODP from Zhu et al.[5]. Both the number of training steps needed and the performance of the model does not reach the results of the original OODP. The performance of the OODP may be improved by increasing the number of training steps. The total necessary number of training steps could be decreased by higher learning rate. Furthermore a different initialisation of the layers used for the models may decrease the number of training steps, too.
Also the different sampling of the trainings data may cause the weaker results. The trainings data used may have a lot of redundant movements. By train with these data over and over again (175 times in this case) the model may forced to learn the redundant data. This may cause some kind of overfit. In the original model, the trainings data is sampled new for each batch. This may cause a better coverage of the possible actions and frames for the game enviroment. Redundant data can also occur, but repetition is less likely than in the pre sampled data set.

6 Outlook

The accurate detection of the dynamic object mask is the foundation of the further success of training. That is because of the interaction of the dynamic object with every other object and the resulting movement predicted by the Dynamic Predictor. So an improvement of the detection of the dynamic object may decrease the number of training steps. Because of the static field of view of the game environment, integrating a model for background subtraction may improve the detection of dynamic objects[7]. By subtracting the next frame from the current frame, the area of the dynamic object could be predetermined.

References

[1] V. Mnih, K. Kavukcuoglu, and Silver, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. A. Nature, and U. 2017, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2016.

[3] O. Vinyals, I. Babuschkin, and Czarnecki, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[4] C.-N. Fiechter, "Efficient reinforcement learning: model-based Acrobot control," *COLT '94: Proceedings of the seventh annual conference on Computational learning theory*, no. April, pp. 88–97, 1994.

[5] G. Zhu, Z. Huang, and C. Zhang, "Object-oriented dynamics predictor," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. Nips, pp. 9804–9815, 2018.

[6] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 2017–2025, 2015.

[7] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," pp. 751–767, 2000.