Imagination Augmented Agent for Deep Reinforcement Learning

Angela Denninger

Munich, 03.09.2018

Dynamic Vision and Learning Group

Supervisor: Prof. Laura Leal-Taix'e
Advisor: Tim Meinhardt

Abstract

bla

Contents

1 Reinforcement Learning

Reinforcement Learning refers to a kind of Machine Learning method in which an agent learns to solve a given task by maximizing the received reward signal. Where the agent represents the reinforcement learning algorithm.

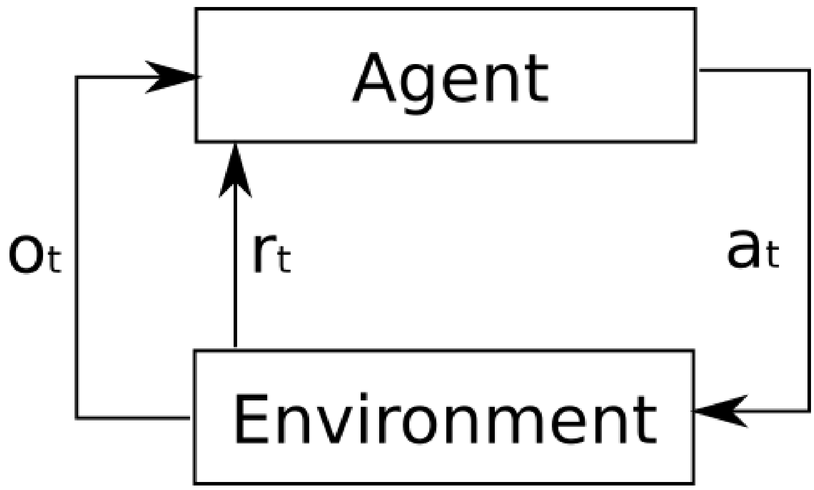


Figure 1: Agent: reinforcement learning algorithm; Environment: object the agent is acting on; a_t action the agent performce in the environment; o_t: observation (current state) of the envirionment the agent receives; r_t reward signal the agent receives from the environment

The Figure 1 shows the interaction of the agent with the environment where the environment refers to the object the agent is acting on. As initial state the agent receives the observation o_t of the environment at time $t = 0$. The observation o_t can be the complete state of the environment or just a subset of it. The agent then, based on the obervation o_t, performce an action a_t in the environment, choosen from a set of possible actions (e.g. moving to the right or moving to the left). After performing the action a_t the agent receives the new observation o_{t+1} of the environment and also the reward signal r_{t+1} which evaluates how good the choosen action was. Based on the two new information the agent again choose an action a_t. The loop continuouse until the environment sent a termination state.

The goal of the agent is to maximize the received reward, which is done by learning an optimal action choosing function called policy using the received reward signal. There exists different reinforcement learning algorithm which all follow the above described iterative learning algorithm, but differ in the update strategy for learning an optimal policy. In the follwing different types of reinforcement learning algorithm will described in more details.

1.1 Q-Learning

To receive our goal to maximize the expected reward, we want to find a function which calculates the maximum expected future reward, for each action at each state. So when we in a certain state we can simple ask this function about the expected reward for each possible action we can take and choose the best one.
Value based methods: where we learn a value function that will map each state action pair to a value. The action with the biggest value is the best action to take for each state.
The base idear behind reinforcement learning is the Bellman Equation. By iteratively update the Q-value function the Q-value function will converge to the optimal Q-value function and will learn to choose the optimal actions.
Q-Learning is an off-policy, model-free RL algorithm based on the Bellman Equation:

$$v(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1})|S_t = s] \tag{1}$$

$\mathbb{E}$ in the above equation refers to the expectation, while λ refers to the discount factor. We can re-write it in the form of Q-value:

$$Q_\pi(s,a) = \mathbb{E}_{s'}[r + \lambda Q_\pi(s',a')|s,a] \tag{2}$$

The optimal Q-value, denoted as Q* can be expressed as:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \lambda \max_{a'} Q^*(s',a')|s,a] \tag{3}$$

The goal is to maximize the Q-value.

1.2 Value based vs Policy based reinforcement learning methods

Value based methods (Q-learning, Deep Q-learning): where we learn a value function that will map each state action pair to a value. Thanks to these methods, we find the best action to take for each state-the action with the biggest value. This works well when you have a finite set of actions.

Policy based methods (REINFORCE with Policy Gradients): where we directly optimize the policy without using a value function. This is useful when the action space is continuous or stochastic. The main problem is finding a good score function to compute how good a policy is. We use total rewards of the episode.

1.3 Model-free versus Model-based Reinforcement Learning

Model-free reinforcement learning maps observation of the environment directly to values or actions. In contrast to this model-based reinforcement learning algorithm are using a model of the environment to simulate the dynamics of the environment. The model knows the transition probability $T(s_{t+1}|s_t,a_t))$ to the next state s_{t+1} given the current state s_t and the current action a_t. By taking this model into account adverse consequences of trial-and error can be avoid, also the performance of the agent can be increased by increasing the amount of internal simulations. But there are some drawbacks. If the model is imperfect the performance of model-based agents suffers. Also it is not always possible to get an exact transition model or to get an transition model at all. In real world application it is often impossible to get a good enough transition model.

For more information about reinforcement learning see TODO

1.4 Deep Q Network

$$r_j + \gamma \max_{a'} Q(st+1, a'; \theta^-) \tag{4}$$

θ parameters in the neural network
TODO link DQN
remember that value function calculates what is the maximum expected future reward given a state and an action

1.5 Advantage-Actor-Critic (A2C)

Advantage-Actor-Critic is a deep reinforcement learning algorithm. Deep reinforcement learning means that the algorithm is using a neuronal network to learn the decission making function. A2C combines value based and policy based reinforcement learning and consits of two parts. A critic that measures how good the action taken is (value-based) and an actor that controls how our agent behaves (policy-based).

The **actor** learns the policy function $\pi(a|s,\theta)$ (probability of choosing action a given state s), which is used to decide the best action a given a specific state s. The actor controls how the agent behaves. θ are the learnable weights of the neural network.

The **critic** learns the value function $V(s,w)$, which messures how good a certain state s is to be in. The value function V is used to calculate the expected cumulative reward $Q(s,a)$ from following the policy π from state s.

$$Q(s,a) = r_{t+1} + \gamma V^{\pi}(s_{t+1}) \tag{5}$$

To update the policy function, we use the **Advantage function** which tells us the improvement of a certain action compaired to the average action taken at state s. In other words, it estimate the improvement of the true reward compared to the expected reward of the current state s by using the temporal difference error:

$$A(s,a) = Q(s,a) - V(s) \tag{6}$$

The advantage function push up the probability of an action from a state s if this action was better than the expected value.

Actor Critic policy $\pi(a|s,\theta)$ **update**:

$$policyloss = -log(\pi_\theta(a|s)) * A \tag{7}$$

$$\nabla\theta = A\nabla_\theta log\pi_\theta(a|s) \tag{8}$$

Actor Critic value update: TODO

$$loss = \sum(R - V(s))^2 \tag{9}$$

gradient

$$\nabla w = 2 * \sum(R - V(s))^2 \tag{10}$$

2 Imagination Augmented Agent (I2A)

In the paper "Imagination-augmented agents for deep reinforcement learning" Weber et al. [1] combines the advantages of model free and model based reinforcement learning to get an agent which is robost against model imperfections but is able to use the advantages of model based agents.

To do this they train a model of the environment for internal simulations. Imagine the future and learn to do better actions based on the imagined future.

In figure 2 network architecture contains 3 parts. A Imagination core which imaginate the future

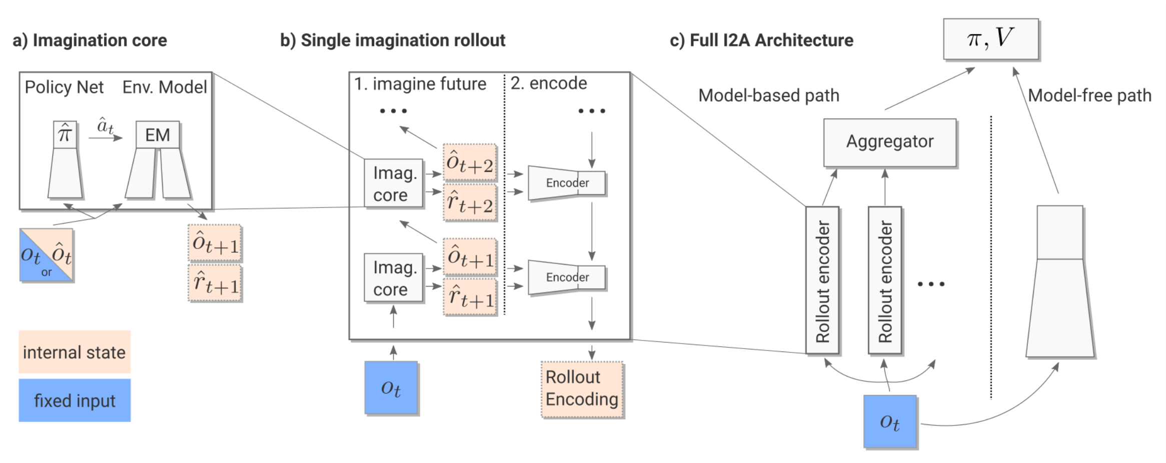
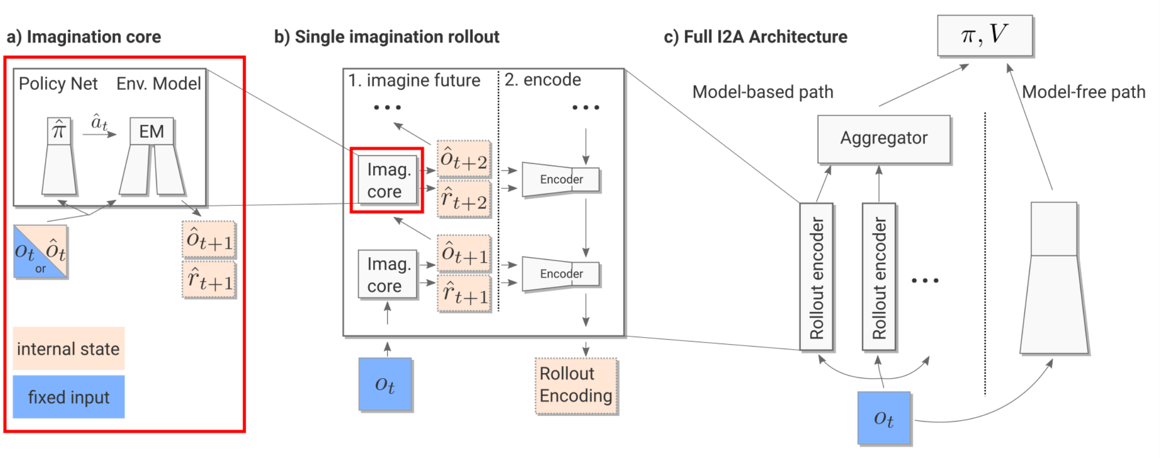


Figure 2: Network architecture for deep reinforcement learning which combines model free and model based reinforcement learning

Imagination Core



Imagine the next observation $\hat{o}_{t+i+1}$ and next reward $\hat{r}_{t+i+1}$ given observation $\hat{o}_{t+i}$ for n rollouts where $i = 0, ..., n$

Rollout Policy Network $\hat{\pi}$

Rollout policy network $\hat{\pi}$ decides the next action $\hat{a}_t$ Distillation loss

Make $\hat{\pi}$ (rollout policy) and π (i2a policy) similar

Cross Entropy between π and $\hat{\pi}$

$$l_{dist}(\pi, \hat{\pi})(o_t) = \lambda_{dist} \sum_a \pi(a|o_t) log\hat{\pi}(a|o_t) \tag{11}$$

Environment Model (EM)

o_t: initial observation $\hat{o}_t$: predicted observation $\hat{r}_t$: predicted reward Given observation o_t or $\hat{o}_t$ and action $\hat{a}_t$ predict (imagine) the next observation $\hat{o}_{t+1}$ and next reward $\hat{r}_{t+1}$

Environment Model Architecture

Input:

– Stack of last 3 observations

– Action as one hot vector

Trained with paris of $(o_t, a_t) \rightarrow (o_{t+1}, r_{t+1})$ generated from a pretrained model-free advantage-actor-critic policy

Environment Model Training

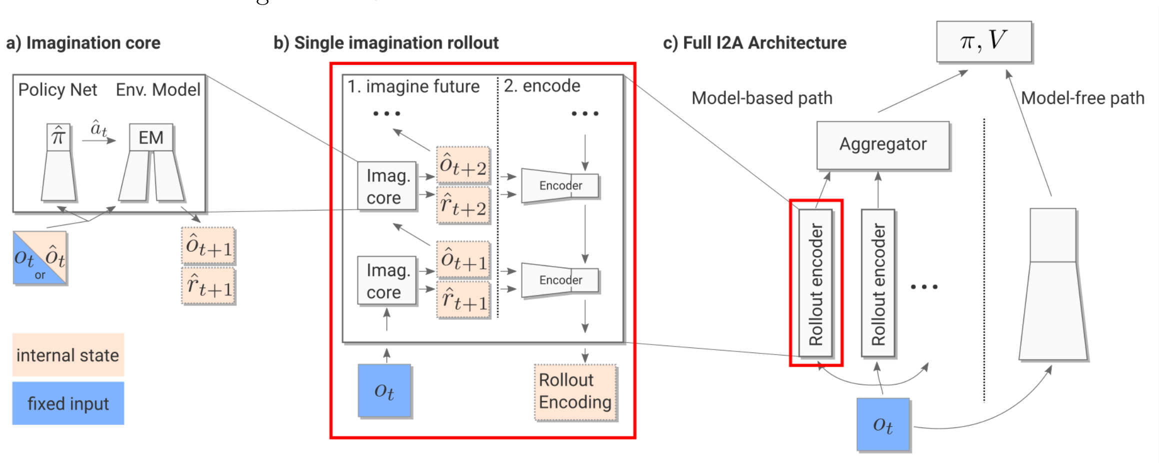
Maximize the log likelihood of the probability $p(o_t|a_{t-1}, o_{t-1})$ Image can be seen as Bernoulli distribution

$$p(o_t|a_{t-1}, o_{t-1}) = x^y(1-x)^{1-y} \tag{12}$$

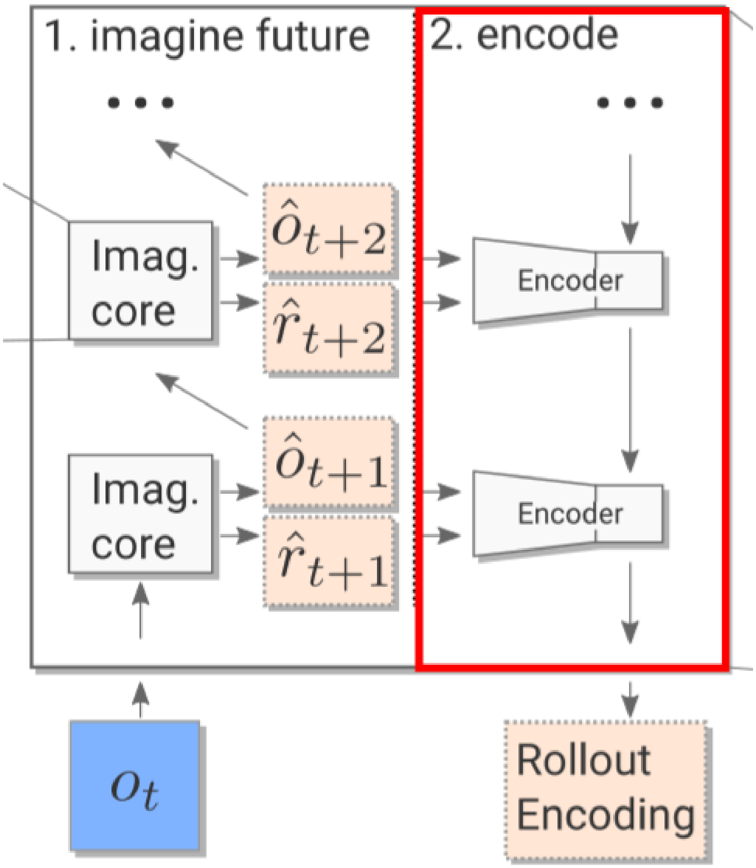
$logp(o_t|a_{t-1}, o_{t-1})$ equals to Binary Cross Entropy loss

$$env_{loss}(x, y) = \frac{1}{N} \sum y_n log x_n + (1 - y_n) log(1 - x_n) \tag{13}$$

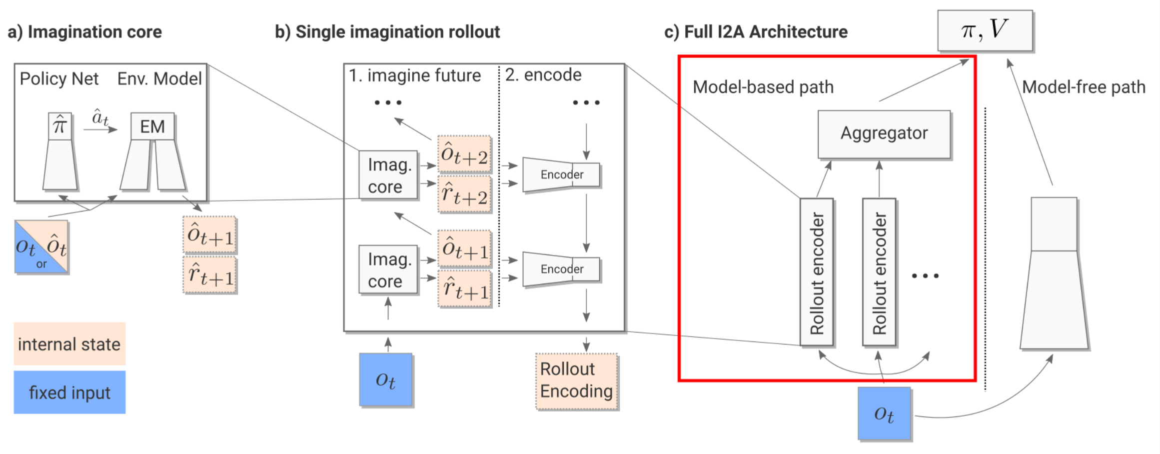
I2A Architecture - Imagination Rollout



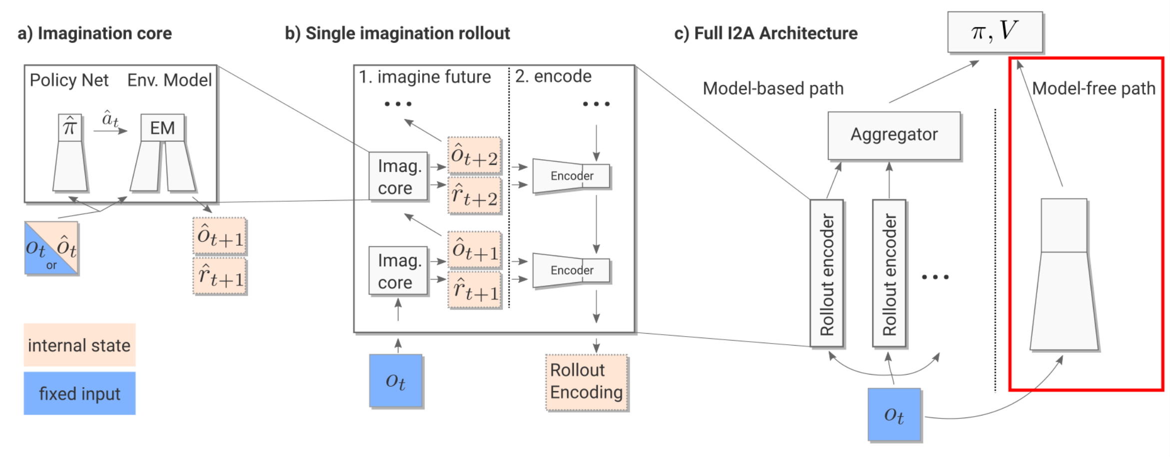
The imagination core imagines trajectories of features $f = (\hat{o}, \hat{r})$
The rollout encoder encode these trajectories
I2A Architecture - Imagination Future
Input:
observation o_t
(1 MiniPacman frame)
start action a Output:
n imagined trajectories $(\hat{o}_{t+i}, \hat{r}_{t+i}$ for $i = 0, ..., n)\hat{}$ $\rightarrow$ internal state
I2A Architecture - Encoder
CNN Network followed by an LSTM Network CNN Network:
Encode observation and reward $\hat{o}_{t+i}, \hat{r}_{t+i}$ LSTM Network:
Learns long-term dependencies



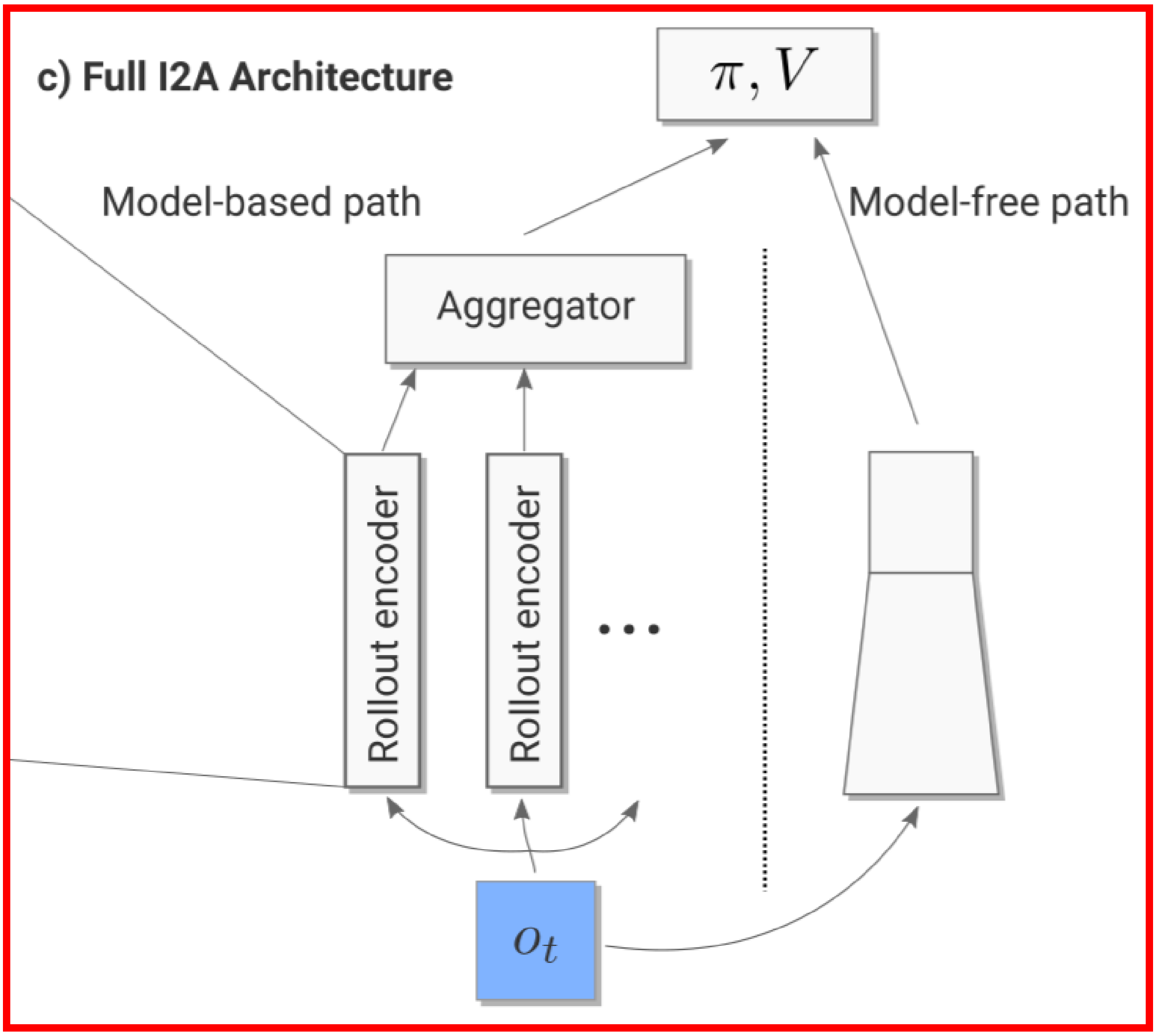
I2A Architecture - Model Based Path



I2A Architecture - Model Based Path
For each action a the agent can take, do a imagination rollout Aggregator:
Concatinate all action rollouts
I2A Architecture - Model Free Path



CNN Layers followed by Fully Connected Layers
I2A Training Input:
Observation o_t Output:
Policy π and value function V Train with Advantage-Actor-Critic (A2C)



3 Environment Model

b

4 Mini Pacman

In diesem Kapitel wird die Implementierung der Approximation von Flächen mit gekrümmten Dreiecken

5 Results

6 Future Work

Um noch besser

Bei d

References

[1] Theophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. *CoRR*, abs/1707.06203, 2017.