Generalization and Locality in the AlphaZero Algorithm

A study in single player, deterministic, fully-observable environments

Anna Deichler [1], Thomas Moerland [2]

[1] 3ME, Delft University of Technology, [2] Robotics Insitute, Delft University of Technology

Introduction

- AlphaGo has managed to defeat the top level human player in the game of Go.
- The challenging properties of high state-space complexity, long reward horizon and high action branching factor in the game of Go are also shared by many other complex planning problems, such as robotics applications.
- Hypothesis of project: the success of the algorithm can be attributed to the complementary strengths of deep neural networks and Monte Carlo tree search
- Monte Carlo tree search - main strength: locality of information, as each edge stores its own statistics, it is easier to locally separate the effect of actions, main drawback: lack of generalization
- Deep learning - main strength: generalization, drawback: cannot roll out the consequences of decisions
- The project examined the trade-off between geneneralization and locality in single player, deterministic and fully-observable RL environments under time fixed time budgets
- The project also examined a way to allocate search efforts more efficiently through adaptively modifying search efforts based on root return variance in search tree

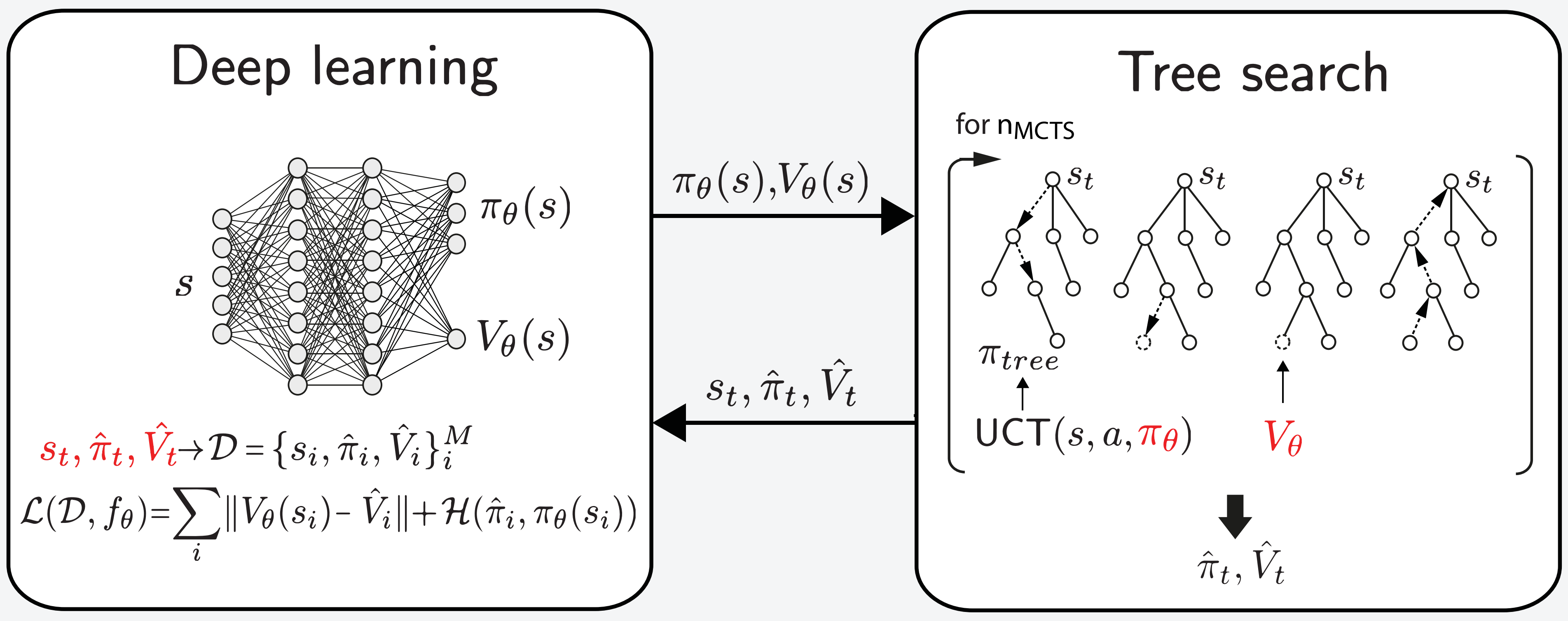
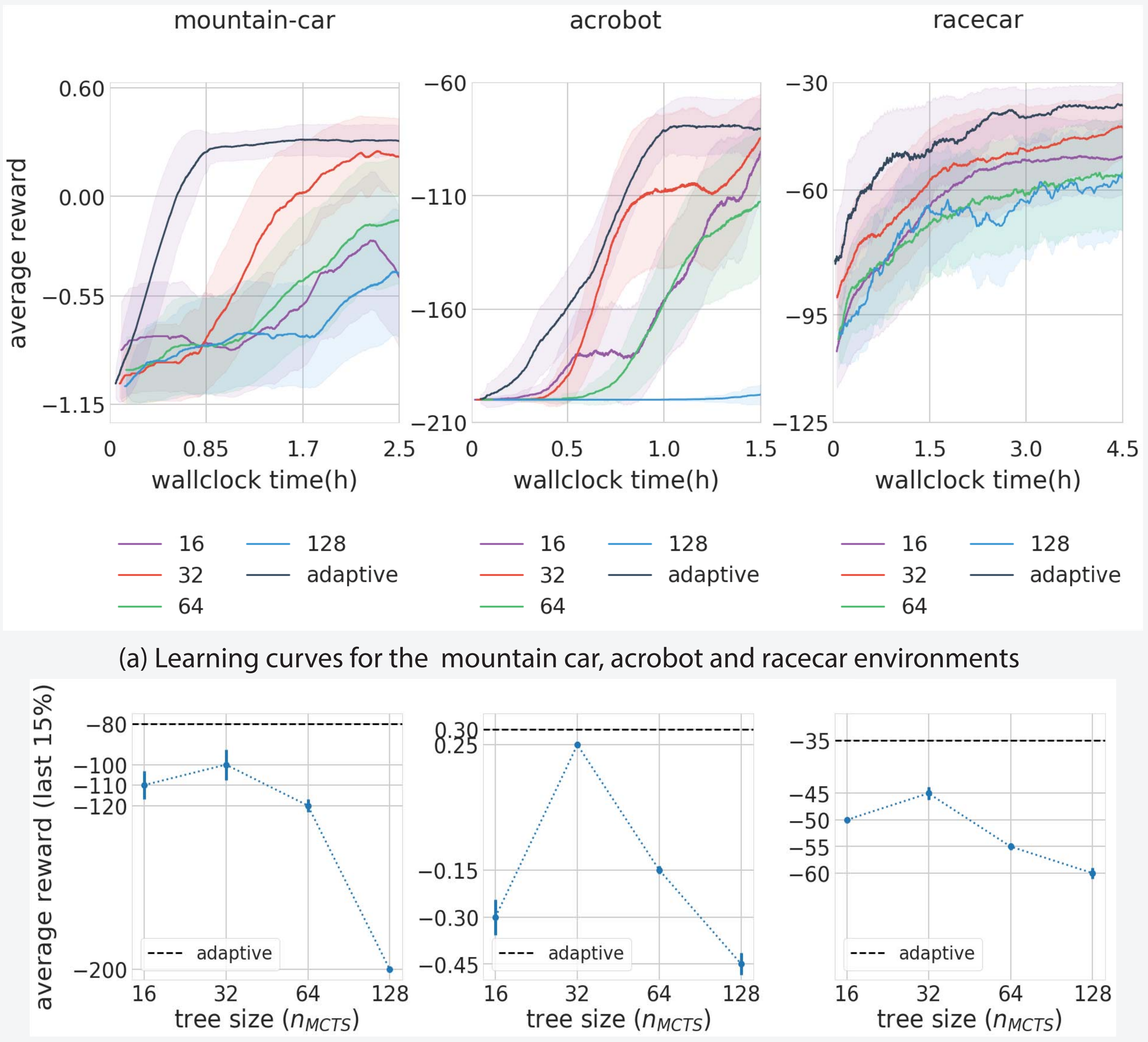

Deep learning
s
$\pi_\theta(s)$
$V_\theta(s)$
$s_t, \hat{\pi}_t, \hat{V}_t \rightarrow \mathcal{D} = \{s_i, \hat{\pi}_i, \hat{V}_i\}_i^M$
$\mathcal{L}(\mathcal{D}, f_\theta) = \sum_i \|V_\theta(s_i) - \hat{V}_i\| + \mathcal{H}(\hat{\pi}_i, \pi_\theta(s_i))$
$\pi_\theta(s), V_\theta(s)$
$s_t, \hat{\pi}_t, \hat{V}_t$
Tree search
for n_{MCTS}
s_t
π_{tree}
UCT(s, a, π_θ)
V_θ
$\hat{\pi}_t, \hat{V}_t$

Figure 1. Schematic view of the algorithm as an iterative learning process from the interplay of a deep learning and tree search system. The neural network provides guidance during tree search. The normalized action counts of the root state and the root value estimate from tree search are used for training the neural network. .



(a) Learning curves for the mountain car, acrobot and racecar environments

(b) Average episodic reward in last 15% of pre-fixed training time

Figure 2. Episodic reward based performance comparison of fixed and variance based adaptive MCTS iteration counts in different RL environments. Each curve is averaged from 3 different seeds. Plots show that mid-sized trees achieve best performance and adaptivelty changing the iteration count improves performance.

Method

Modifying number of iterations in tree search

- Localization versus generalization question was examined through varying the number of MCTS iteration steps n_{MCTS}, while keeping other hyperparameters of the algorithm fixed.
- n_{MCTS} is the number of simulated trajectories performed using the environment emulator before each action selection step in the real environment.
- Under a fixed time budget n_{MCTS} defines how much effort is spent on acquiring more accurate values through building large search trees at each decision step versus improving generalization by updating the network more frequently.
- The examined RL environments have reward distributions with support out of [0,1] range ➡ use adaptive value normalization.

$$UCT = \frac{\bar{Q}_t(s,a) - \mu_t}{\sigma_t} + c \cdot \pi_\theta(a|s) \frac{\sqrt{n(s)}}{n(s,a)+1}$$

Variance based adaptation of number of MCTS iterations

- Instead of a fixed n_{MCTS}, adaptively changing n_{MCTS} based on the uncertainty of the current state's value estimate could increase computational efficiency and performance.
- Focus search efforts on more uncertain parts of the state space by increasing n_{MCTS}.
- Execute additional iterations based on search tree's root return variance:

$$\tilde{R}_{roll} = \frac{t-1}{t}\tilde{R}_{roll} + \frac{1}{t}\tilde{R}_t, \qquad r = \frac{\tilde{R}_t}{\tilde{R}_{roll}}$$

$$n_{adt} = \max(n_{min}, \min(n_{max}, n_{min} \cdot r)) \qquad \tau(r) = \begin{cases} exp(1-r) & \text{if } r \leq 1.0 \\ 1 & \text{if } r > 1.0 \end{cases}$$

Results

- Figures 2-3 show that the AlphaZero algorithm achieves the best performance with middle sized trees in all the examined environments.
- Choosing a high number of n_{MCTS} iterations puts emphasis on local information, which results in more accurate value estimates
- On the other hand, executing a high number of n_{MCTS} is time costly and therefore less time remains for training the neural network and improving it's generalization capacity, which will hurt the overall performance
- Experiment results also show that adaptively changing search efforts based on the root return variance can improve performance.

Discussion

- The results indicate balancing local information and generalization is crucial for the performance of the AlphaZero algorithm
- The algorithm allows exploiting local knowledge through the access to the environment rules in tree search and generalizing past experience through the use of deep neural network
- The combined tree search and deep learning architecture resembles a human decision making model, the dual process theory. According to this view, the local search (MCTS) plays the role of a "slow system", which is conscious and rule-based mode of reasoning. On the other hand, deep learning plays the role of the "fast system", which is unconscious and also called intuition.
- Exploiting the two system structure could be beneficial in other complex sequential decision making problems, such as robotics

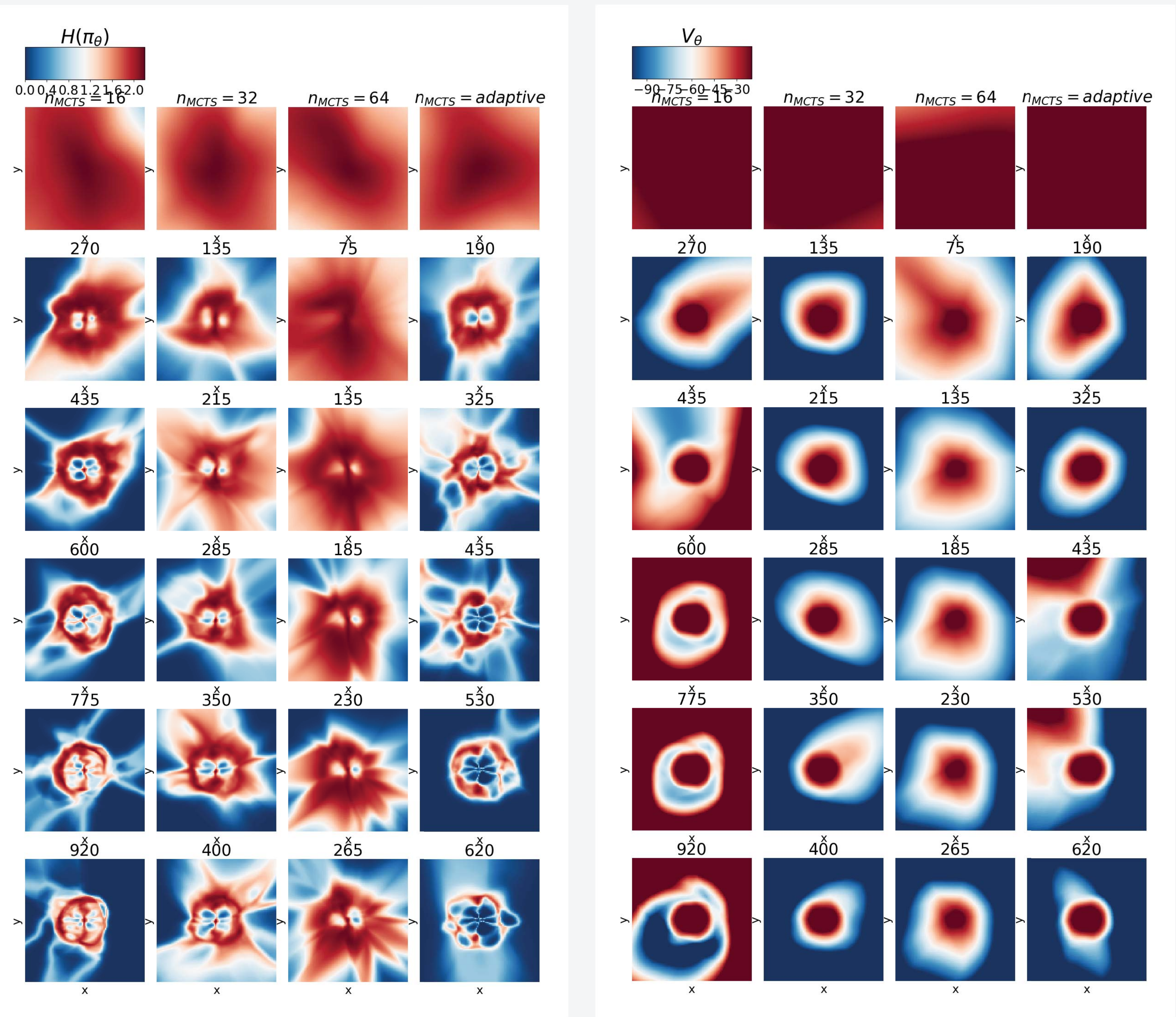


Figure 3. Effects of varying n_{MCTS} on the two-head neural network predictions in the racecar environment (state s=(x,y)), during the learning process . Neural network predictions are evaluated at a fixed state space grid with weights from given episode. Evaluated episodes are taken at equal distances until the pre-fixed training time.