SAND2019-13625R

Optimizing Dynamic Timing Analysis with Reinforcement Learning

James Obert*, Angie Shia
Sandia National Labs
Albuquerque, NM, USA
{jobert, ashia }@sandia.gov

Abstract— **There are multiple factors involved in successfully manufacturing ASIC/VLSI chips, and ensuring operational specifications are maintained throughout the design and manufacturing process is often challenging. Dynamic timing analysis (DTA) is the principal method used to validate that a manufactured chip complies to its design specifications. In DTA functionality of both synchronous and asynchronous designs are verified by applying input signals and checking for correct output signals. In complex designs where the number of input signal permutations is extremely large, the computing resources required to properly verify the functionality of a chip is prohibitive. In this paper, a strategy using reinforcement learning (RL) for reducing DTA time and resources in such cases is discussed. RL assisted DTA holds much promise in ensuring that VLSI chip design and functionality are fully and optimally verified.**

Keywords— ASIC Design Verification; Reinforcment Learning; Dynamic Timing Analysis; ASIC Design.

James Obert, Angie Shia are actively inolved in ASIC verification and design anaylsis at Sandia National Labs. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

I. INTRODUCTION

A. Overview

Speed, area and power are the three primary VLSI/ASIC design properties that a designer must consider. In terms of functionality, meeting timing constraints is the most important design property [1]. A chip's timing constraints are tested at a set clock rate using either static (STA) [2] or dynamic timing analysis (DTA) [3]. In DTA a set of signals is input to the chip and functionality of the chip's design is verified and predicted output signals are observed. STA tests the static delays within the ASIC chip's circuits without the injection of logic-dependent input signals. DTA's goal is to verify that an ASIC design can operate without errors at a specified clock rate and is generally accomplished using the simulated manufacturing design synthesis files [4,5]. The goals of STA are similar to DTA verification but does not test the full functionality of the design and does not require that the functionality of the design be simulated. DTA and STA are often used in a complimentary fashion with logic functionality first verified with DTA prior to using STA to refine circuit layouts based on simplified timing models which mostly disregard circuit logic. STA is not able to verify asynchronous designs while DTA can. For this reason, DTA is an essential verification method for asynchronous designs. Additionally, DTA is also the best choice when designs have clocks crossing into many domains.

In verifying all logical paths within an ASIC, the quality of DTA increases as the number of input signal permutations increases. However, as the number of input signal combinations increases, the simulation time also increases. Since the more efficient STA cannot effectively test the functionality of many designs, the only viable alternative is to efficiently speed-up DTA design verification of complex ASIC designs. In this paper, an efficient method using reinforcement learning (RL) for generating an optimized set of DTA input signals is illustrated. It is shown through simulated RL generated input signals, that it is possible to efficiently refine an ASIC design using DTA.

The remainder of this paper is structured as follows. Section II discusses an RL optimized DTA (ODTA)

methodology and covers the key concepts involved in generating an optimal set of ASIC input signals using RL. Section III describes the experimental ASIC design dataset, the simulations used and the analysis results. Finally, in Section IV, conclusions and future work are discussed.

II. METHODOLOGY

A. ASIC Logic Paths

Figure 1 for DTA conceptual illustration purposes shows a very simple ASIC design which includes an AND gate, NOT gate and OR gate [6].

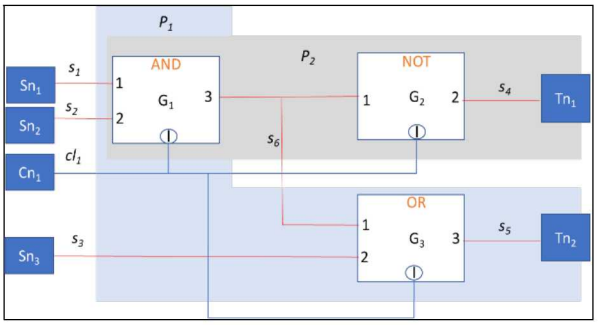


Figure 1: Simple ASIC Design.

To illustrate how DTA is performed, consider digital input signals S_1, S_2, S_3 in Figure 1. As shown in Figure 2 below, signals S_1, S_2, S_3 change state according to the specific logic gates they are processed by on the leading-edge of each Cl_1 clock cycle. Signals S_1 and S_2 propagate through gates G_1, G_2 and G_3 and contributes to the output signal S_5 on logic path P_1 and output signal S_4 on logic path P_2. Signal S_3 propagates through gate G_1 and contribute the output signal S_5 on logic path P_1. The input signals S_1, S_2 and S_3 in period t_1 prior to the first clock cycle, are initialized by the DTA simulator in states {0, 0, 1} respectively, and the predicted output signal states of S_4 and S_5 are observed at a target clocking rate.

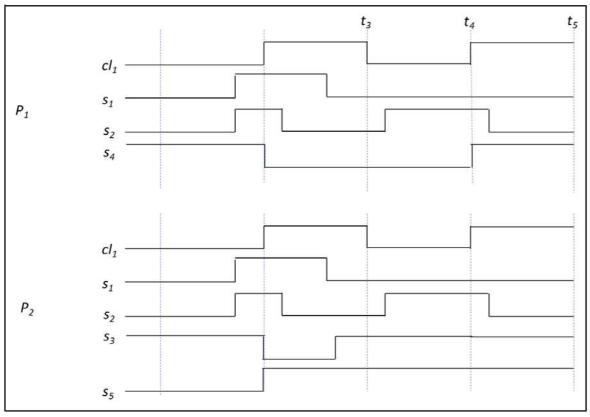


Figure 2: Logic Timing Diagram.

In this simple example, if each input signal in its intended application fully varies between 1 and 0 in all permutations, then the number of possible input signal S_1, S_2 and S_3 state combinations is 2^3 or 8. Real-world System-on-Chip (SoC) ASICs logic gate numbers often reach one million, and the number of input signal state permutations required to fully verify the ASIC using DTA often range in the hundreds of thousands. In many cases, specific input signal permutations may not be valid for the intended ASIC application; thus, testing with such combinations wastes time and resources. It is possible to speed-up DTA verification by reducing the number of input signal state permutations introduced to the ASIC design. If ground-truth output signal distributions for a minimal set of logic testing input signals are available, RL can form a policy which selects a valid subset of input signal state permutations by sampling the output signal distributions while strategically introducing input signal state permutations to the ASIC during DTA. The following sections discuss the methods for achieving DTA speedup in this manner.

B. RL Policy Metric

During the ASIC design process, a set of baseline input signals are used to characterize the basic logic operations of the ASIC. In the characterization process, the baseline input signals are injected, and the output signals are analyzed. Injecting the set of baseline input signals into the ASIC design will exercise known and intended logic states at a target clocking rate but is not intended to fully exercise or test all possible input and output signal states. In support of future DTA simulations, the statistical distribution of each output signal's state transitions is captured. The goal of RL guided DTA is to reduce DTA related processing time and resources by strategically selecting a subset of all possible input signal permutations. The effect of selecting a closely related subset of input signal permutations in RL guided DTA is that only those ASIC logic states most similar to the intended baseline logic states are analyzed. This strategic selection is achieved by optimally selecting those input signal permutations that produce output signal distributions closely matching the output baseline signal distributions. The primary metric used to guide RL in finding the optimal set of input signal permutations is the relative entropy between each DTA synthesized output signal's distribution and each baseline output signal's distribution. In relation to RL, each of the input signals is considered a *feature* which the RL algorithm will process. Feature Centric Entropy Analysis (FCEA) is used to compare individual feature distribution relative entropies. Using Equation 1, individual feature distribution K-L Divergence values are calculated for baseline and DTA generated signal distributions. In Equation 1, K-L Divergence measures the relative entropy change between feature f_q and f'_q distributions. Where F_q is a baseline feature probability distribution and F_q' is a target or DTA probability distribution.

$$D_{KLF}(F_q{}'||F_q) = \sum_{f_q, f'_q} Prob(f'_q) \ln \frac{Prob(f'_q)}{Prob(f_q)} \qquad (1)$$

C. *Reinforcement Learning*

There two principal methods used to implement RL. The first method called the *policy gradients* method [8, 9] is implemented as an artificial neural network which learns a policy for picking actions by adjusting it's weights through gradient descent [10] using feedback from the environment. (ie. Function approximating neural network [11])

Using *value functions* is the second principal implementation method. In the policy gradient method the optimal actions are learned for each given state. Using value functions, the agent predicts the reward for a given state, given specific sequences of actions (ie. *Q*-Learning [12]). *Q*-learning (The *Q* function returns the reward of an action while in a specific state) can iteratively find an optimal actions decision policy given that the environment follows a finite Markov Decision Process[16]. Starting at the current state, the optimal actions policy found maximizes the expected total reward when all successive possible steps are considered. If the environment follows a finite Markov Decision Process, *Q*-learning coupled with an *e-Greedy* [9] actions process over an infinite exploration time, can find an optimal action-selection policy.

D. *DTA and Reinforcement Learning*

Figure 3 below illustrates how RL is used to select an optimal subset of input signals in RL-guided DTA.

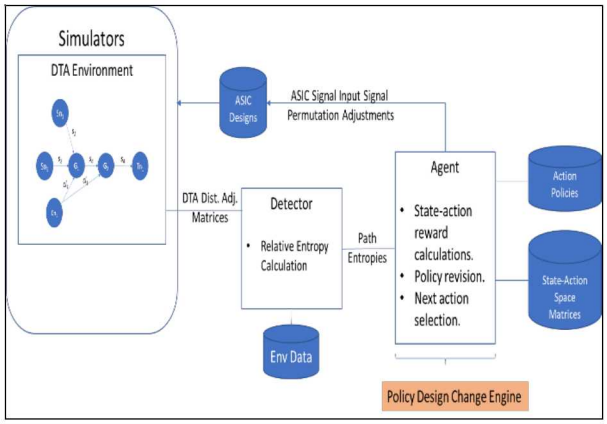


Figure 3: DTA and RL used for optimal input signal selection.

In Figure 3, KL-Divergence (KLD) values are calculated over multiple policy exploration steps of an RL episode. On each RL episode step, an e-Greedy actions strategy is used by the DTA agent to guide an optimal action based on either a value function or policy gradient component (contained within the *Policy Design Change Engine* - PDCE). On each step the PDCE using the optimal policy derived at that point in the episode, presents a specific input signal permutation to the DTA simulation environment. The KLD values between the individual baseline output signal distributions and the current step's individual output signal distributions are calculated on each step. An action-state-reward (q_m, ϱ_l, Kld_u) tuple is stored in an action-state-reward matrix on each step to be referenced in determining future rewards. On successive episode steps, using the *e*-Greedy based strategy, the action-state matrix is referenced to calculate the reward value for that current state and action taken. Iteratively over time using the *e*-Greedy strategy, a function approximator (See Figure 4) is trained to recognize an optimal policy (π) for selecting future actions based on past action-state-reward tuples.

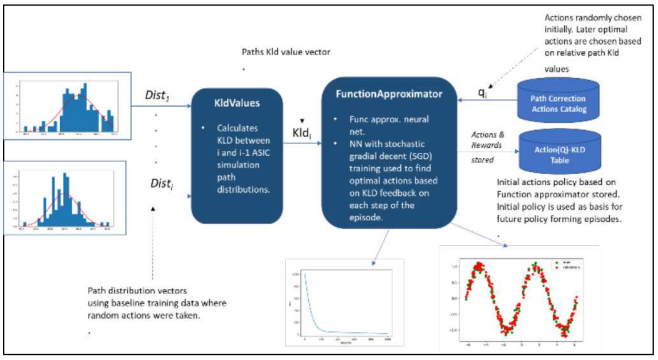


Figure 4: DTA and RL used for optimal input signal selection.

Figure 5 below illustrates the related rewards (Kld_u) predicted by the PDCE in successive episode steps.

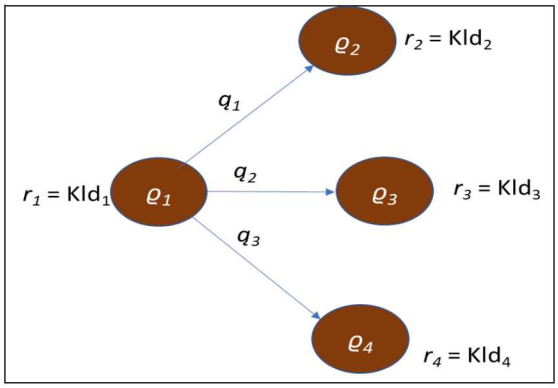


Figure 5: Q-learning in RL-guided DTA.

When using a *Q*-Learning value function, the $Q_{\varrho,q}$ value for each successive episode step is determined using Equation 2.

$$Q_{\varrho,q} \leftarrow \sum_{q_m} (Q_{\varrho,q} + \gamma max_{q'} Q_{\varrho',q'}) \qquad (2)$$

Where:

γ : discount factor (look ahead factor)
q_m : Previous policy recommended action.
q'_m : Current policy recommended action.
$Q_{\varrho,q}$: Cumulative look ahead state.
r_w : Reward Kld_u .
ϱ_l : Previous individual episode step state .
ϱ'_l : Current Individual episode step state.

When using the policy gradient method a function approximator neural network is trained iteratively by adjusting the policy weights θ based on the gradient of some utility/performance measure $\nabla_\theta \mathbb{E}_\tau[R(\tau)]$ with respect to the policy weights. Where τ is a trajectory and R(τ) is the return for path τ. These methods seek to maximize performance, so their updates approximate gradient ascent in $\nabla_\theta \mathbb{E}_\tau[R(\tau)]$. For gradient decent,

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \mathbb{E}_\tau[R(\tau)] \quad (3)$$

$\nabla_\theta \mathbb{E}_\tau[R(\tau)]$ is a stochastic estimate whose expectation approximates the gradient of the performance measure with respect to θ_t. τ denotes a state-action sequence with horizon H, $s_0, u_0, \cdots, s_{H-1}, u_{H-1}$ with the total reward from trajectory τ being:

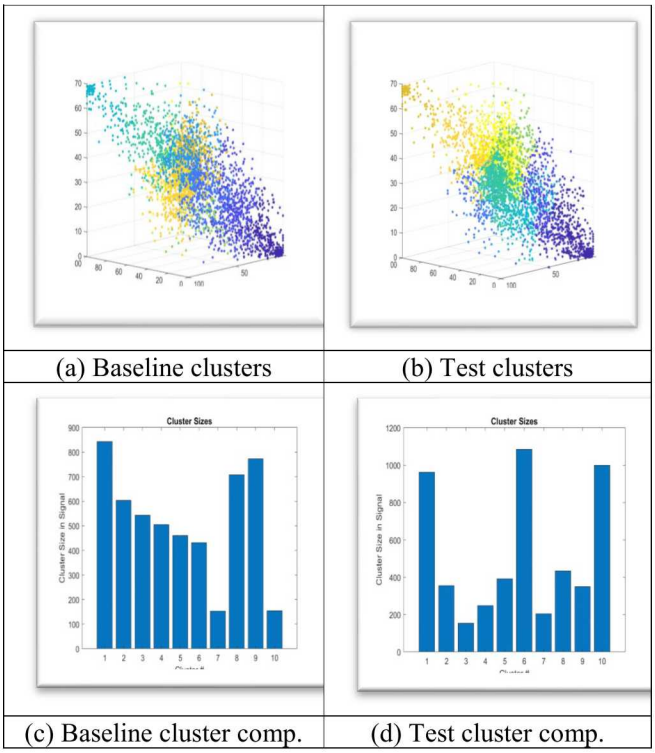
$$R(\tau) = \sum_{t=0}^{H-1} R(s_t, u_t) \quad (4)$$

I. Experimental Results

Experimental Results

A. Experiments

To illustrate the ODTA methodology an ASIC with a total of 5900 signals was analyzed. To test the timing of target logic states in the ASIC, a subset of all input signal permutations was injected into the ASIC, and all intermediate and output signals were captured. Subsequently, randomly generated input signal permutations for each step of the RL policy derivation episode were injected into the ASIC. Figure 6 variations in agglomerative hierachial clustering [13] patterns and compostion shows the degree to which statistical and signal distribution variations exist between the baseline and a single test case signal set. The Kld_u (relative reward values) were calculated between baseline and intermediate and output signal test case signal ditributions. The Kld_u was stored within the action-state-reward (q_m, ϱ_l, Kld_u) tuple matrix row during each episode step. Figure 7, shows the Kld_u values for a single episode step. As indicated by the arrows in Figure 7, several signals were divergent from the baseline in during this step. The degree of divergence indicated by the Kld_u value is directly proportional to the degree which the ASIC's logical behavior varies from the baseline test case. The RL optimal DTA policy was then derived using an *e*-Greedy strategy with policy gradient methodology.



(a) Baseline clusters (b) Test clusters

(c) Baseline cluster comp. (d) Test cluster comp.

Figure 6: Clustering patterns and cluster composition variations.

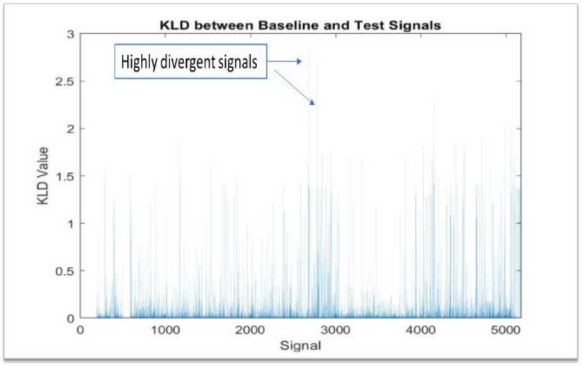


Figure 7: KL-Divergence values between baseline and an input signal permutation test case.

Given a large enough number of RL policy discovery episodes and steps, the value function or policy gradient method lead to finding an optimal action policy. However, when using value functions alone, large changes in the policy can occur while searching for the best actions at a specific state. Such abrupt changes in policy are less likely using the policy gradient method where the action-state space is continuous. In application such as robotics, abrupt changes in policy can result in actions that can damage equipment.

Because of the simplicity and directness, the policy gradient method was utilized in our experiments.

B. Results

Summarizing, the ODTA methodology was implemented as follows below.

The purpose of the *initial discovery phase* is to randomly gather input signal permutation vectors to boot-strap the training of the function approximator artificial neural network. The function approximating neural network iteratively learns how to select an optimal set of input signal permutations. The initial discovery phase implemented is depicted in Figure 8 and follows the below stated process:

- Randomly synthesized test input signal permutations were generated and on each step of an episode.
- The respective Kld_u values were calculated between each episode step's output signal and baseline output signal distributions.
- Individual Kld_u values, corresponding input signal vectors, and state were recorded in the actions table.
- The above actions were repeated for the number of steps in the initial discovery episode

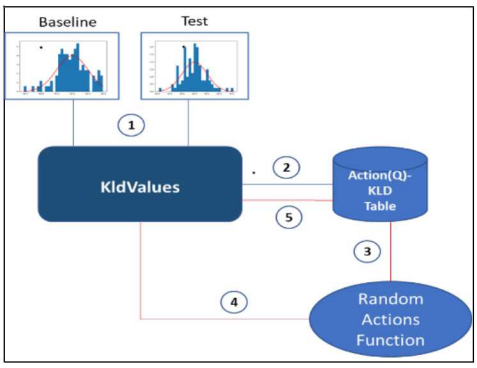


Figure 8: Randomly injected input signals.

In finding *the optimal DTA policy* the function approximator in Figure 9 is trained using only those randomly generated input signal permutation vectors stored in the actions table that have a low Kld_u value.

Figure 10 below shows those output signals permutations that were least conforming and those output signals that were most conforming. Those input signal permutations that resulted in the set of most conforming output signal vectors were used on completion of the initial discovery phase to boot strap train the function approximator.

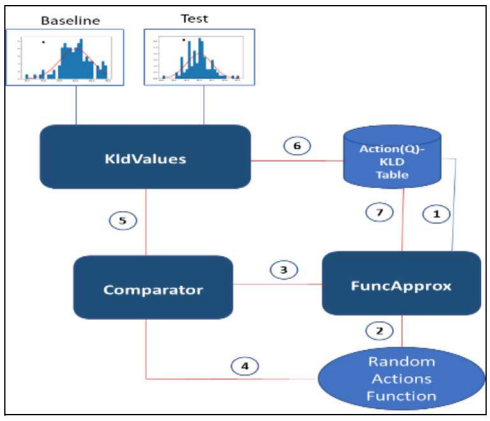


Figure 9: Optimal policy convergence.

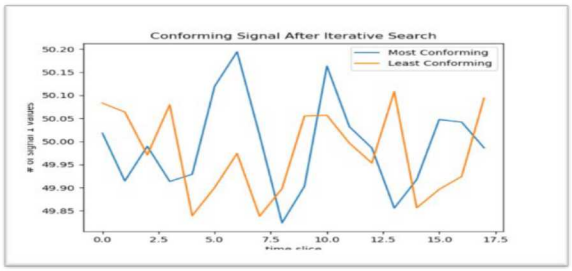


Figure 10: After initial random search, the most and least conforming signals are identified.

The function approximator artificial feed-forward neural network in our experiments was composed of 4 hidden layers with rectified linear unit (ReLU) activation function [14] and Adaptive Moment Estimation (ADAM) optimization learning algorithm [15]. The partially trained function approximator was used to guide future policy decisions using an *e*-Greedy strategy. On a selected number of episode steps a random input signal permutation was input to the ASIC and on all other steps the function approximator was used to select a policy guided input signal permutation for input based on what it has learned previously. As the number of discovery episode steps proceeded, the function approximator continued to learn an optimal policy for selecting sets of input signal permutations that resulted in low Kld_u values as depicted in Figure 11 convergence graph.

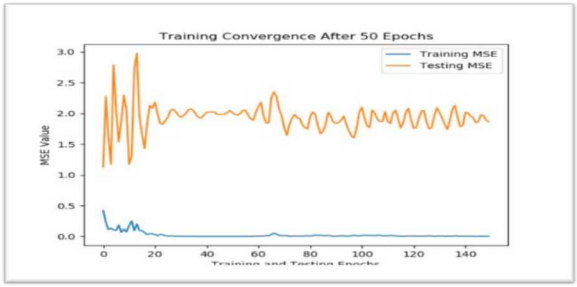


Figure 11: The deep feed-forward NN convergence.

As shown in Figure 11, as training and testing progressed using the observed optimal set of input signal vectors, the mean square error (MSE) between target and predicted Kld_u values fell below the standard 5% accuracy threshold. It is expected using the *e*-Greedy strategy with additional RL training episodes, that the MSE would drop to even lower values resulting in a global optimal ODTA policy convergence.

II. Conclusions and Future Work

In this paper it was shown that it's possible to use RL to strategically select ASIC design input signals during DTA. Use of RL-aided input signal selection during DTA promises to optimally extend ASIC logic functionality testing by introducing input signal permutations to an ASIC that are not in the heuristically derived test set. Introducing RL-derived input signal permutations to the ASIC effectively increases the accuracy of logic functionality testing in both synchronous and asynchronous logic designs. Additionally, RL-aided DTA minimizes processing time and computing resources by only introducing the set of input signal permutations that most closely produce known target output signals.

Using additional ASIC DTA datasets, future analysis will include quantifying aggregate logic testing accuracy, computing resource and processing time efficiency gains yielded when ODTA is utilized.

References

[1] F. Vahid, Register-transfer level (RTL) design, in Digital Design with RTL Design,V erilog and VHDL, 2nd edn. (John Wiley & Sons, 2010), p. 247.A.

[2] Lavagno, Martin, and Scheffer *Electronic Design Automation For Integrated Circuits Handbook*, ISBN 0-8493-3096-3.

[3] Deepak Kumar Tala, "Verilog Tutorial", www.asic-world.com, 2001.

[4] GE Calma, GDSII™ Stream Format Manual, Release 6.0, Document No. B97E060, Calma Co., Sunnyvale, CA (1987).

[5] J. R. Buchanan, The GDSII stream format (1996), http://jupiter.math.nctu.edu.tw/_weng/courses/IC 2007/PROJECT_NCTU_MATH_CELL_LAYOUT/The_GDSII Stream_Format.htm.

[6] Null, Linda; Lobur, Julia (2006). *The essentials of computer organization and architecture*. Jones & Bartlett Publishers. p. 121. ISBN 978-0-7637-3769-6.

[7] Barr, Keith (2007). *ASIC Design in the Silicon Sandbox: A Complete Guide to Building Mixed-signal Integrated Circuits*. New York: McGraw-Hill. ISBN 978-0-07-148161-8. OCLC 76935560.

[8] J. Peters and S. Schaal, \Policy gradient methods for robotics," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (Beijing, China), 2006.

[9] R. S. Sutton and A. G. Barto., Reinforcement learning: An Introduction. MIT Press, 1998.

[10] Barzilai, Jonathan; Borwein, Jonathan M. (1988). "Two-Point Step Size Gradient Methods". *IMA Journal of Numerical Analysis*. **8** (1): 141–148.

[11] Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", *Neural Networks*, 4(2), 251–257.

[12] Russell, Stuart J.; Norvig, Peter (2010). *Artificial Intelligence: A Modern Approach* (Third ed.). Prentice Hall. p. 649. ISBN 978-0136042594.

[13] Kaufman, L.; Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis* (1 ed.). New York: John Wiley. ISBN 0-471-87876-6.

[14] Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). *Deep sparse rectifier neural networks*, AISTATS.

[15] DiederikP.Kingma, Jimmy Le Ba, *ADAM: A Method for Stochastic Optimization, ICLR 2015.*

[16] Bellman, R. (1957). ″A Markovian Decision Process″. Journal of Mathematics and Mechanics.