The Thesis Committee for Suhaib Abdulquddos
certifies that this is the approved version of the following thesis:

Adapting to Unseen Driving Conditions Using Context-Aware Neural Networks

APPROVED BY

SUPERVISING COMMITTEE:

Risto Miikkulainen, Supervisor

Cem C. Tutum

Adapting to Unseen Driving Conditions Using Context-Aware Neural Networks

by

Suhaib Abdulquddos, B.S.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

The University of Texas at Austin

December 2022

Dedicated to advancement of the field of robust artificial intelligence.

Acknowledgments

I want to acknowledge my thesis advisor, Dr. Risto Miikkulainen for his mentorship, advice, and guidance over the course of this project. Undeniably this work would not have yielded the results achieved had it not been for his advising.

I would also like to acknowledge Dr. Cem Tutum for his help and direction in conducting this research.

Lastly, I would like to thank DARPA for funding this research.

Suhaib Abdulquddos

The University of Texas at Austin
December 2022

Abstract

Adapting to Unseen Driving Conditions Using Context-Aware Neural Networks

Suhaib Abdulquddos, M.S.Comp.Sci
The University of Texas at Austin, 2022

Supervisor: Risto Miikkulainen

One of the primary inhibitors to successful deployment of autonomous agents in real-world tasks such as driving is their poor ability to adapt to unseen conditions. Whereas a human might be able to deduce the best course of action when confronted with an unfamiliar set of conditions based on past experiences, artificial agents have difficulty performing in conditions that are significantly different from those in which they were trained.

This thesis explores an approach in which the addition of a context module to a neural network is used to overcome the challenge of adapting to unseen conditions during evaluation. The approach is tested in the CARLA simulator wherein the torque and steering curves of a vehicle are modified during training and evaluation.

Furthermore the agent is trained only on a track with a relatively large radius of curvature but is evaluated on a track with much sharper turns and the agent must learn to adapt its speed and steering during evaluation. Three different neural network architectures are used for these experiments, and their respective performances are compared: Context+Skill, Context-only, Skill-only. It is observed that when both performance and safety of agents behavior are considered, the context+skill network consistently outperforms both the Skill-only and the Context-only architectures.

The results presented in this thesis indicate that the context aware approach is a promising step towards solving the generalization problem in the autonomous vehicle domain. Furthermore, this research presents a framework for comparing the generalization capabilities of various network architectures and approaches. It is posited that the context+skill neural network has the potential to advance the field of machine learning with regards to generalization in domains beyond just autonomous driving; that is, any domain where awareness of changing environment parameters can have a positive impact on performance.

Table of Contents

Chapter 1

Introduction

Improvement in the robustness of artificial intelligence systems remains an open problem, especially when dealing with classes and cases that were not encountered during training (27). It is especially critical in applications with high stakes that autonomous agents be robust to a variety of conditions. Conventional approaches to achieving this kind of robust performance mainly center around introducing variety in the training dataset. In the domain of autonomous vehicles, startups such as Metamoto, Cognata, and rFpro have been formed to tackle this exact problem: simulate environments in which autonomous agents can be trained on a variety of driving conditions (26). However it is not possible to comprehensively train an agent on the infinite set of conditions that it may encounter in the real world.

One particularly important application where such robustness is needed is autonomous driving. The potential economic impact of widespread autonomous vehicle adoption has been comprehensively researched. Studies have shown that the estimates for the potential positive impact of widespread autonomous vehicle adoption range from $936 billion in direct economic impact within the United States alone to indirect societal benefits in excess of $3 trillion globally (2). The significance of autonomous vehicle adoption is difficult to understate, and yet comprehensive analysis of the current state of the art in this field has shown that improvement in areas including perception, decision-making, and adaptation are needed before widespread adoption of autonomous vehicle technology will be possible (6).

The present research aims to present a potential solution to the problem of finite training data by teaching an agent to extrapolate its learned knowledge to

unseen conditions thereby reducing to an extent, the need for an entirely comprehensive set of conditions during training. The ability of an agent to perform in an environment that is vastly different from any kind of environment encountered during training unlocks a large potential set of use cases for artificial intelligence systems in high stakes domains including but not limited to autonomous driving.

As will be described in section 2.4, many approaches have been discussed and demonstrated in the literature with varying degrees of success and applicability for the problems of generalization and overfitting. A novel neural network architecture is proposed which is composed of a distinct skill network and a distinct context network whose generalization abilities are demonstrated to be superior to that of its constituent networks alone.

This thesis is organized as follows:

Chapter 2 discusses the Neuroevolution training algorithm used for all neural networks evaluated in the experiments, the choice of simulation environment for the experiments, and existing approaches in the literature to the generalization problem.

Chapter 3 discusses the novel neural network architecture whose generalization ability in the CARLA domain is the focus of in the present work.

Chapter 4 describes the experimental setup, paradigms, assumptions, and abstractions used in the experiments. The interface between the "brain of the agent" and simulation environment (the agent's perception and controls) is described in detail.

Chapter 5 explains the routines, tasks, and constructs used in training the networks during the experiments. This chapter discusses how agents are evolved to achieve desirable performance and eventually are selected for testing generalization capabilities.

Chapter 6 summarizes the experimental results and presents a case for why the Context+Skill Neural Network generalizes better than either of the two networks it is compared against in the CARLA domain.

Chapter 7 describes the major contributions of this work and limitations encountered. Furthermore, potential avenues of future research are suggested particularly when computation resource limitations are not a limiting factor.

Chapter 8 concludes the work in this thesis and provides a brief summary of the research presented herein and its relevancy to the broader class of generalization problems for which research is ongoing.

The research presented in this thesis paves the way for a promising and novel approach to solve the generalization problem in the autonomous driving domain. There is much room for improvement given the current state-of-the-art, and the addition of context awareness to existing neural network architectures is anticipated to advance progress towards solving the generalization problem. Finally, this approach is likely to yield promising results when applied to other domains and paves the way for advancement in the generalization capabilities of neural networks more broadly.

Chapter 2

Background

Related work and tools to which this research contributes and builds upon are introduced herein. Neuroevolution, the method by which the neural networks in the present experiments are trained is described. The choice of simulation environment CARLA is introduced and a brief rationale for this choice is shared. A preview of the broad problem of generalization and overfitting, and solutions to this problem that have been proposed in the literature is presented. Lastly, prior applications of the neural network architecture of interest in this thesis are summarized.

2.1 Neuroevolution

Evolutionary algorithms are a class of computations that use population based training techniques to evolve populations with the intent that at least one individual with desirable fitness is obtained. Evolutionary algorithms can be used for training anything that can be composed as a genome, or an array of numbers, and whose fitness can be expressed as a tuple of numbers. Neuroevolution is the application of evolutionary algorithms to neural networks (14). The genome of a neural network can be expressed as an array containing its constituent weights and biases, and its fitness can be expressed as the performance of the neural network at a given task.

Backpropagation is one of the most widely used algorithms for training neural networks, and has enjoyed considerable success in the literature (18). Indeed, for many problems, especially supervised learning problems, backpropagation has been shown to yield state of the art performance. In contrast, neuroevolution has been

shown to be a reliable method for training neural networks even when a "ground-truth" for the correct action is not known. This is the precise type of problem that is of interest in the present study, which made neuroevolution an obvious choice for neural network training method.

The generality of neuroevolution makes it a prudent choice for many applications in machine learning. This powerful method for training neural networks also allows for simple meta-learning and architecture search wherein the topology of a neural network during training is also optimized (22). The flexibility and power of neuroevolution is contrasted with the rigidity of traditional machine learning algorithms in which the topology of a neural network is generally fixed over the course of training (techniques such as dropout temporarily remove the effect of neurons but do not remove them entirely from the network). Furthermore, neuroevolution has shown superiority over other methods of training when novel or creative behavior is desired from the neural network (21). In domains where the novel ability to extrapolate learned behavior to unseen conditions is desired (such as a robust agents for autonomous vehicles), neuroevolution becomes a particularly appealing training method.

2.2 CARLA

CARLA is an open-source simulator designed to facilitate autonomous vehicle research (7). CARLA offers a high-fidelity simulation environment wherein a variety of real-world urban driving conditions can be controlled and used for experimentation. The numerous maps built into CARLA were modeled after real-world locations. Variety in driving tracks provides an additional dimension against which adaptation can be measured; simply switching to a track with turns that have larger or smaller radii of curvatures than those encountered during training presents an unseen task which would require the agent to extrapolate learned behavior.

CARLA allows for comprehensive control over vehicle parameters which model various phenomena that one might experience while driving a real car. This control allows for thorough examination of an agent's response to various kinds of changes that it might encounter when driving in the real world. A list of the parameters that can be controlled in CARLA is shown in Table 2.1.

CARLA has a rich history in the academic literature as a testing environ-

Parameter	Description
Torque Curve	Curve that indicates the torque measured in Nm for a specific RPM of the vehicle's engine.
Maximum RPM	The maximum RPM of the vehicle's engine.
Moment of Inertia	The moment of inertia of the vehicle's engine.
Damping Rate @ Full Throttle	Damping ratio when the throttle is maximum.
Damping Rate @ Zero Throttle + Clutch Engaged	Damping ratio when the throttle is zero with clutch engaged.
Damping Rate @ Zero Throttle + Clutch Disengaged	Damping ratio when the throttle is zero with clutch disengaged.
Use Gear Autobox	If True, the vehicle will have an automatic transmission.
Gear Switch Time	Switching time between gears.
Clutch Strength	Clutch strength of the vehicle.
Final Ratio	Fixed ratio from transmission to wheels.
Forward Gears	List of objects defining the vehicle's gears.
Mass	Mass of the vehicle.
Drag Coefficient	Drag coefficient of the vehicle's chassis.
Center Of Mass	Center of mass of the vehicle.
Steering Curve	Curve that indicates the maximum steering for a specific forward speed.
Use Sweep Wheel Collision	Enable the use of sweep for wheel collision. By default, it is disabled and it uses a simple raycast from the axis to the floor for each wheel. This option provides a better collision model in which the full volume of the wheel is checked against collisions.
Wheels	List of wheel physics objects. This list should have 4 elements, where index 0 corresponds to the front left wheel, index 1 corresponds to the front right wheel, index 2 corresponds to the back left wheel and index 3 corresponds to the back right wheel. For 2 wheeled vehicles, set the same values for both front and back wheels.

Table 2.1: List of vehicle parameters that can be controlled in CARLA, taken from CARLA Python API documentation (car).

ment for autonomous vehicle research. It has the advantages of relative simplicity of use, adjustable simulation fidelity, and ease of access by nature of being open source. These advantages make its use appropriate for a variety of research applications. It was shown to be an effective environment for comparison of the relative performances of Deep Q-Networks (DQN) and Deep Deterministic Policy Gradients (DDPG) as autonomous vehicle agents (17). Pérez-Gil et al. indicated that such features as CARLA's powerful Python API, efficient simulation of planning and control, and diverse set of traffic scenarios contributed to their selection of CARLA as a simulation environment.

2.3 Machine Learning for Autonomous Vehicle Applications

A survey of academic literature in the field of autonomous vehicles published in the last decade yields the conclusion that machine learning has been the dominant approach. It has been an instrumental tool in several critical subdomains in the autonomous vehicle technology stack, including road perception (9), vehicle control (1), and even driver fatigue detection (16).

An approach known as end-to-end learning uses a single neural network for both perception and vehicle control (3). Using this approach, the neural network teaches itself efficient internal representations of the environment and uses these internal representations to make control decisions. The input of the neural network is an encoding the the world around it, and the output of the neural network is the agent's response. This approach is contrasted with more complex systems that decompose the autonomous vehicle problem into subproblems such as lane demarcation, path panning, and control, which are in turn tackled as separate problems. The advantages of the end-to-end approach over the decomposition approach include relative simplicity of the architecture and the freedom of the agent to self-optimize to construct the most efficient representation of the intermediate states that would otherwise be designed by a human. The efficiency gain from allowing the agent to construct its own representation of states results in fewer free parameters (and by extension smaller networks) required to solve the problem as compared to more complex systems where the perception network and control networks are treated as separate entities (3).

As mentioned in Chapter 1, the problem of generalization in the autonomous vehicle domain remains unsolved. Numerous approaches for solving the generalization problem have been proposed. Ahmedov et al. proposed a solution inspired by the human neurology (1). The human brain has strong generalization abilities, especially compared to the current state-of-the-art in the autonomous driving domain. They hypothesized that this strong generalization ability of the human brain comes from its functional and structural asymmetry. A brain-inspired deep imitation method comprised of Deep Neural Networks (DNN) with dual Neural Circuit Policies (NCP) was proposed and tested as a solution to the generalization problem. Ahmedov et al. experimented with a novel CNN-DNCP architecture that uses end-to-end learning to convert a pixel map (an image from a dashcam mounted on the vehicle) into steering actions. The DNCP component is comprised of two NCP structures that operate independently, and whose respective outputs are averaged as the final steering action that is applied by the agent. The CNN-NCP and CNN-DNCP architectures were shown to yield superior generalization performance as compared to the convolution neural network alone on the same task.

2.4 Generalization

The problem of generalization is not unique to the autonomous vehicle domain. The gap in performance between humans and computers with respect to dataless classification persists despite the increasing attention paid to this problem in recent years (4). Humans are resilient; even a young child is able to adapt to unseen environments or solve new problems by extrapolating learned behavior. Humans have the ability to draw analogies from learned experiences, and not overfit knowledge from these limited experiences. Despite significant progress in the resiliency of machine learning algorithms, human-like generalization capabilities have yet to be achieved by machine learning agents, even in the current state of the art (8). In deep learning especially, the large number of parameters tends to exacerbate the problem of overfitting and makes generalization very difficult (11). The two most prominent approaches to the problem of generalization are explored in Sections 2.4.2 and 2.4.1.

2.4.1 Weight Decay

Weight decay is proposed as a solution to increase the generalization ability of a deep neural network without compromising performance on a training set (10). The loss function of a neural network is modified to reward neural networks with weights that are small in magnitude and penalize networks that are large in magnitude. This goal is achieved by the addition of the term $\lambda|\mathbf{w}|^2$ to the loss function, where λ is a hyperparameter that controls the extent of weight decay and $\mathbf{w}$ is a vector comprised of all the free parameters of the neural network. This approach has been shown to produce resilient networks that outperform networks of similar architecture and loss functions that lack weight decay (10).

An alternative albeit similar approach to weight decay involves applying a decay term on every free neural network parameter during training (13). During each iteration of the training algorithm, each free parameter is multiplied by a scalar quantity less than 1, thereby decaying the magnitudes of the weights and tending the values of the weights to 0.

Both approaches have the ultimate effect of reducing the magnitudes of the free parameters of the neural networks. This makes the network less sensitive to small perturbations in the input set and forces the network to be generally more resilient to a set of data generated by the same underlying function with minor variations, as opposed to "memorizing" a given training set (10).

2.4.2 Dropout

Dropout has been proposed as a solution to the problem of overfitting in deep learning (20). The idea behind dropout is to make a neural network resilient to unexpected changes during training. Artificial diversity is introduced during training by the deactivation of neurons. The neurons that remain adapt to this uncertainty by compensating for the loss of some neurons from the network. The idea is that this resiliency carries over during inference, thereby giving the neural network as a whole the ability to adapt to unseen inputs.

A dropout probability value p is chosen during training: For each training batch, every neuron has probability p of being temporarily deactivated. When deactivated, a neuron does not feed forward its output to the neurons in the next layer. The set of activated neurons in the resulting network is a subset of the set

of neurons in the entire neural network. Dropout does not occur during evaluation; instead the output value of each neuron is scaled down by a factor of p to compensate for the increased number of neutrons present during training.

Dropout has been shown to be a very effective approach to the generalization problem in supervised learning tasks (20). Across several standard image data sets and many successful deep learning architectures, nearly every deep learning architecture seemed to benefit from dropout.

The purpose of the present work is to demonstrate a novel approach to solving the generalization problem in the autonomous vehicle domain. Dropout is not used in the current set of experiments, however ways of augmenting the current approach with dropout are discussed in Section 7.2.

2.5 Prior Work using Context+Skill Neural Networks

Sections 2.4.2 and 2.4.1 discussed training strategies that can be applied to existing neural network architectures to increase generalizability. However the present study utilizes a novel architecture called Context+Skill to teach agents to extrapolate learned behavior to unseen conditions.

This architecture is inspired by the success of a similar architecture used for Poker (12), wherein two neural networks were co-evolved to determine optimal moves; a "skill" network was used to determine what move to make given the state, and a "context" network was used to account for the opponents playing style. The idea is that while a particular poker move might be generally optimal, a universal skill network alone does not consider that the optimal move for the present game might depend on the playing style of the opponent. An LSTM-network provides the agent with the ability to remember the last several game states, and from this construct an internal representation of the playing style of the opponent. Taking into consideration the playing style of the opponent helps the agent take advantage of the nuanced playing style of the opponent, thereby achieving optimal performance *for the given opponent*. This architecture was shown to generalize very well, even when tested with opponents vastly better, and with vastly different playing styles, than those encountered during training. The combination of the context and skill networks yields superior generalization ability than the generalization abilities of either the context or skill networks alone.

The Context+Skill network has shown effectiveness in other tasks as well. Sensor drift refers to the tendency of a gas sensor to lose accuracy as it is exposed to the gasses it is used to detect over time, due to chemical changes in the sensor itself. The Context+Skill neural network has been shown to be highly successful in the classification of gasses despite the deterioration of the physical sensors used (25). The key in this work was the sequential nature of the data being fed in. As compared to the previous state-of-the-art (which relied on Support Vector Machines), the Context+Skill network maintained higher accuracy. Even as compared to a feed forward network (which also managed to achieve superior performance as compared to the SVM approach), the Context+Skill network performed better.

Lastly, context awareness was shown to benefit the performance of agents playing games as well (24). Flappy Ball was a game developed for the explicit purpose of testing generalization capabilities of autonomous agents. Flappy Ball was motivated by the popular game Flappy Bird, which is not new to the academic literature as an environment for testing Machine Learning algorithms (15). Flappy Ball was designed to be a more difficult version of flappy bird where the effects of gravity and drag change from one simulation to another. The player must adjust their playing strategy based on the current environment parameters, which it must detect on its own. Generalization was tested by training the agents on a fixed range of values for these parameters, and generalization ability was tested by testing on an expanded such with much larger ranges. The generalization performances of the same three neural networks, Context+Skill, Context-only, and Skill-only were compared and the Context+Skill network was shown to outperform its Context-only and Skill-only counterparts (24). The same analysis held true for a modified version of the game Lunar Lander which was also tested. In this game, the mass of the lander and the effects of the side thrusters were made variable so as to make the game more difficult. During generalization the Context+Skill neural network was shown to extrapolate its learned behavior much better than the Context-only and Skill-only networks especially on the set of parameters not shown during training.

Chapter 3

Context+Skill Neural Network

The success of the Context+Skill network in other domains (Section 2.5) provoked the idea that the approach would be well-suited to more tasks that require generalization. The ability to comprehend that the environment is not fixed is critical for generalization performance. Awareness of the current context (for example, the context might be the neural networks own representation of something as simple as "the roads are slippery") is believed to be a prerequisite for generalization in end-to-end learning. Modulation of agent actions with this awareness is the key to generalization ability in the current set up.

The Context+Skill architecture used and studied in the present work is identical to the one used in (24). The focus of the present work is an in-depth study of the generalization performance of the Context+Skill neural network within the autonomous vehicle domain, which is much more complex than the Flappy Ball and Lunar Lander games previously tested.

The architecture of the Context+Skill network is shown in Figure 3.1. The skill module is comprised of a simple feedforward neural network. As discussed in section 2.3, feedforward networks have been successful in many environments. Often such networks also have convolutional modules for perception, however the focus here is on actions (and modulation of actions), not perception. Perception in the present experiments is simplified using rangefinders (Section 4.2.3), thus avoiding confounding the task with imperfect perception).

A neural network that is not aware of context (such as the one shown in figure 3.2) is not able to account for the present conditions or environment. Espe-

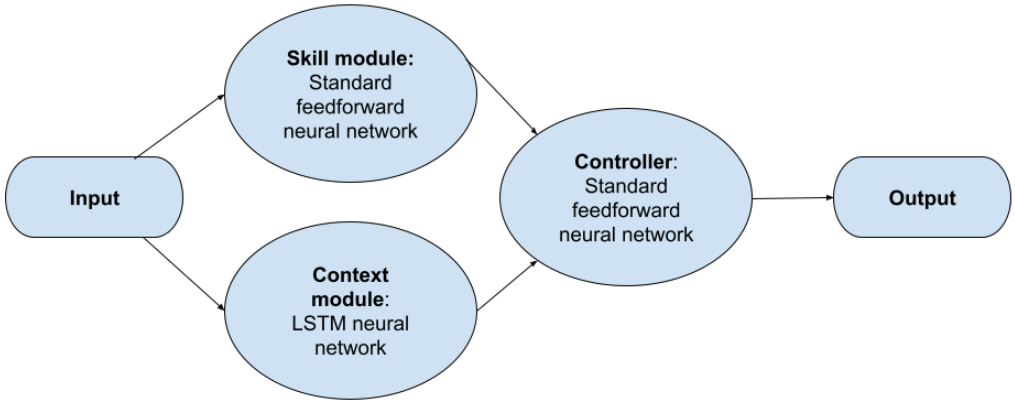


Figure 3.1: Architecture of the Context-Skill neural network.

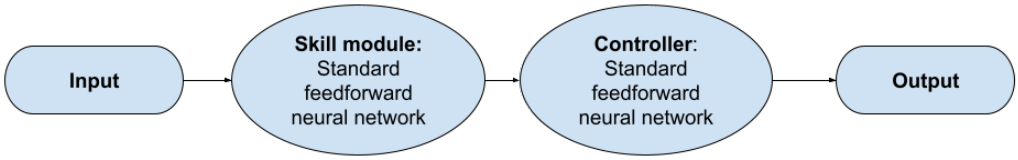


Figure 3.2: Architecture of the Skill-only neural network.

cially when behavior is time dependent (such as steering stiffness or torque curve dampening, see section 4.3.1), perception of the current state does not help the network understand the environment since the state contains no information about past events.

If context is not encoded in the input or the state itself, context awareness can only be achieved by remembering past states. In the context+skill network this functionality is implemented with the context module. This module is comprised of a Long Short-Term Memory (LSTM) cell, which gives the agent the ability to retain information regarding past states (23). The hypothesis is that the LSTM encodes information about the past states, and the output of the context module thus provides an internal representation of the environment. For example, it might provide the knowledge that the agent is currently steering through a curve. The controller takes into account this encoding of environment and combines it with the output of the skill-only network, to yield an appropriate "context aware" action.

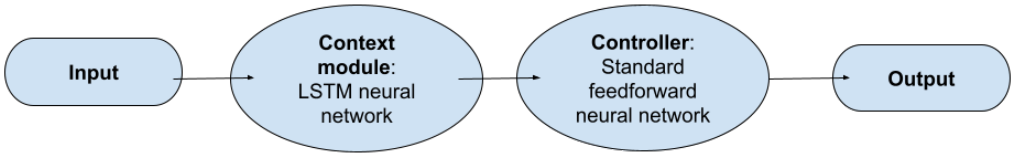


Figure 3.3: Architecture of the Context-only neural network.

The primary focus of the present research is to study the generalization performance in CARLA of the context+skill network architecture. It is compared with two ablations, the Skill-only network (figure 3.2) and the Context-only network (figure 3.3). Each of these two networks contain the controller module and exactly one of the two aforementioned modules. While each of these ablations in principle should be able to do the task, both modules working together are necessary to achieve good generalization. The experiments to test this hypothesis are set up as described in Chapters 4 and 5, and the results are evaluated in Chapter 6.

Chapter 4

Domain Setup

4.1 CARLA Setup

Simulation of an agent in CARLA requires a server-side host for the environment, and a client to command the agent. The host is responsible for simulating all of the physics in the environment, providing a continuous stream of information to the clients regarding the state of the environment. Clients request information from the host and can respond with controls that the host can then apply to agents in the simulation. The nature of this client—server interaction is important for simulation fidelity, and reproducibility of results.

The host operates in one of two modes: variable time-step and fixed time-step. In variable time-step mode, all the computations are performed in real-time. That is to say that the time that passes between frames in the simulation is the exact amount of wall-clock time required to compute the next time-step. It has been shown that simulation fidelity is sensitive to the size of the time-step especially when simulating fast-moving objects (such as vehicles traveling at high speeds; 28), thus the hardware of and computational load on the machine serving as the host for the simulation environment can have a profound impact on the outcome of simulations run in variable time-step mode. In fixed time-step mode, the user is presented with the choice of how large or small to set the time-step. Larger time-steps yield faster results (in terms of wall-clock time required for running an experiment) at the expense of decreased simulation fidelity. All simulations performed for the experiments presented herein used fixed time-step of 1/30 seconds per frame. Decreasing

the time-step beyond this value had virtually no effect on simulation fidelity.

With respect to the interaction of the client and host, a choice must be made regarding whether the client and host should maintain perfect synchrony, or if the host should operate on its own without waiting for input from the client. In the real world the nature of interaction between an autonomous agent and the universe is asynchronous; the universe does not wait for its observers to process input and formulate responses to stimuli. Asynchronous mode makes the task more challenging for the agent as it may miss time-steps or act upon slightly stale information. Furthermore, the agent is left with fewer opportunities to self-correct in response to new information. These issues are exacerbated if the computations performed by the agent are expensive. Yet again, there is the potential for variability across runs: The number of time-steps missed by the agent is subject to computational constraints and factors outside of the agents control. In synchronous mode, the host does not compute the next timestep until the agent has processed the information it received, formulated a response, and instructed the host to proceed to the next timestep. With the goals of maximizing the reproducibility and consistency of results and minimizing the influence of external factors such as computational complexity and hardware speed, all simulations for the experiments presented herein were conducted in synchronous mode.

4.2 Agent Representation

An agent is represented by vehicle simulated in the CARLA environment along with a corresponding neural network that controls how the car responds to stimuli. At each simulation time-step the agent is given a set of information (Section 4.2.3) and formulates a response (Section 4.2.2).

4.2.1 Track

A track is defined by a set of points representing the right and left lane dividers of the trajectory that the vehicle is supposed to traverse, as well as the coordinates of a specified target destination in that track. The tracks are not provided by CARLA itself, but were instead constructed and chosen for this project in order to simulate driving tasks that (1) an autonomous vehicle would encounter in the real world and

(2) are sensitive to perturbations in driving conditions (simulated by changes to evaluation parameters, see Section 5.1).

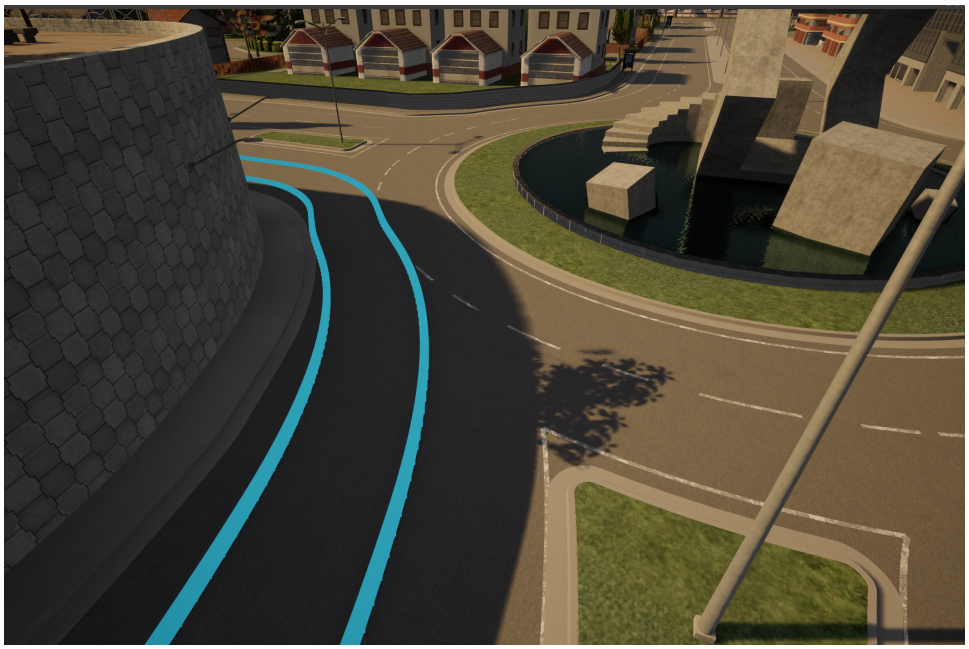


Figure 4.1: A portion of a sample track is shown. A track is comprised of two collections of points, one for each lane boundary, and a pair of coordinates to signify the target destination. This representation facilitates assessment of the agents driving performance (Section 4.3.2) and is required for the rangefinder perception system (Section 4.2.3)

The boundaries of the track are all that the agent "perceives" (see Section 4.2.3) during evaluation. Maintenance of the position of the car within the boundaries of the track affects the score obtained by the agent for the given evaluation. It is not sufficient for the agent to simply arrive at the target; the agent must also stay as close to the center of the track boundaries as possible.

4.2.2 Control

The observation is represented as a simple vector which is fed as input to the network. The input layers and output layers of all three neural networks being compared are identical. The output layer of each neural network consists of two neurons,

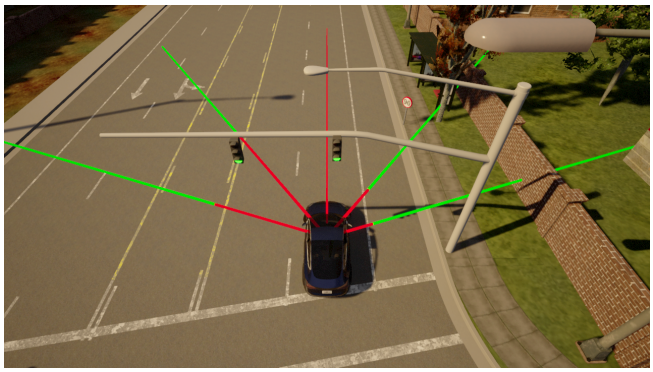
whose range of outputs is $(-1, 1)$ (the tanh activation function is used, see Section 3). The second output neuron is further normalized with the function $f(z_2) = \frac{z_2+1}{2}$ to bring its range of values to $(0, 1)$. The first output value (whose range is $(-1, 1)$) is used as the agents steer response. A value of -1 corresponds to the agent turning the steering wheel all the way to the left, and a steer response of 1 corresponds to the agent turning the steering wheel all the way to the right. Note that there is a relationship between the steer response provided by the agent to the host and the angle that the front wheels of the vehicle actually turn to, as described by the steering curve (see Section 4.3.1). The second output value (whose range is $(0, 1)$) is used as the agents throttle response, with a value of 0 representing no throttle and a value of 1 representing full throttle. Note that there is a relationship between the throttle response provided by the agent and the torque provided by the engine of the vehicle, as described by the torque curve (see Section 4.3.1).

4.2.3 Perception

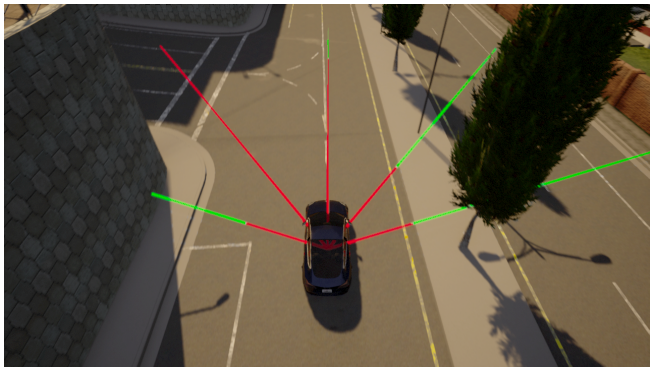
Although CARLA provides a very rich set of sensors including simulated cameras, radar, LIDAR, and semantic LIDAR sensors, none of these were chosen for the experiments. Such sophisticated sensors would necessitate additional modules for each neural network, and the variable accuracy of such computer vision components would introduce confounding variables that would detract from the motivation of the study. In an effort to maintain the focus of the experiments on the novelty of the architecture being studied, a very simple perception mechanism was developed and implemented for the experiments. This perception mechanism, known as rangefinders, provides information to the agent agent that describes how close it is to the lane boundaries of the present track. Visualizations of what the agent is able to "see" are shown in Figure 4.2.

Five finite rays are projected in five different directions as illustrated in Figure 4.2. All rays originate at the same point (the center of the vehicle), have a length of 25 meters, and are oriented as follows:

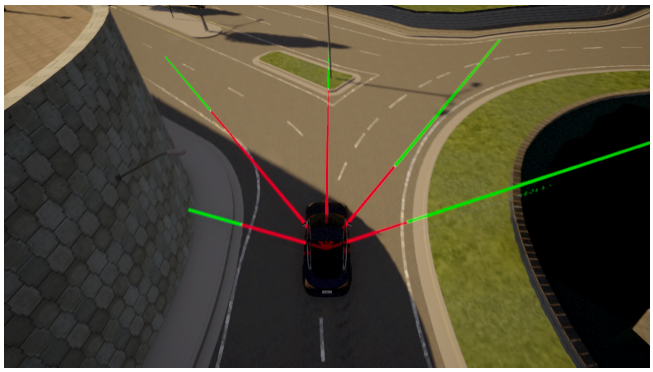
- One center ray oriented parallel to current direction of the vehicle;
- One ray that is offset from the center ray by $-35°$;
- One ray that is offset from the center ray by $35°$;



(a) Rangefinder state when the car is parallel with the direction of the track and inside the lanes



(b) Rangefinder state immediately before a turn



(c) Rangefinder state during a turn

Figure 4.2: A visualization of what the agent "sees" while driving using rangefinders at various points in the experiment. The red portion represents the distance to the track boundary in the direction of the ray, and the green portion shows how much additional "buffer" exists given that the rays are finite length. This enables the agent to perceive the track.

- One ray that is offset from the center ray by $-70°$;
- One ray that is offset from the center ray by $70°$.

The output from each ray is a scalar value representing the absolute distance between the center of the vehicle and the nearest track boundary along the direction of that ray. In the event that there is no intersection between the ray and the track (as is the case of the center ray in Figure 4.2a and the $-35°$ offset ray in 4.2b), the ray simply returns its total length (25 meters).

4.3 Task

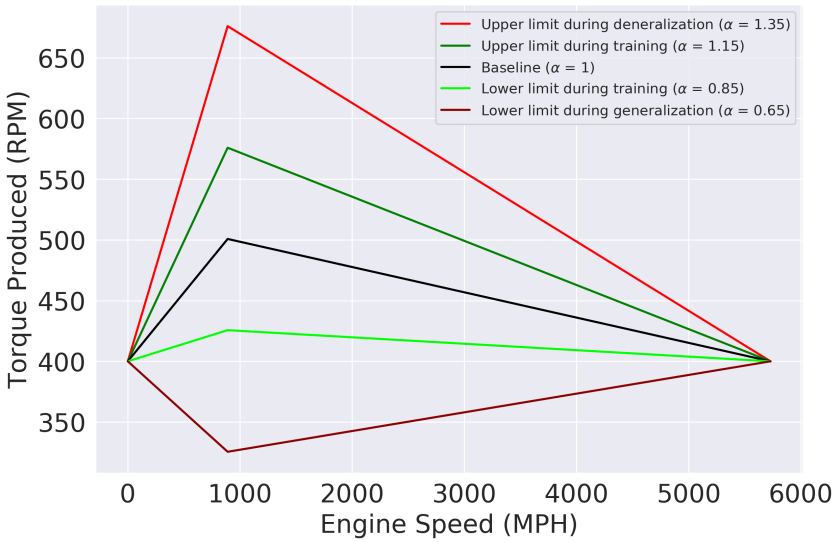
The agent is presented with a number of tasks during training and evaluation. During training the agent learns how to generalize certain behaviors (herein referred to as evaluation parameters, section 4.3.1). During evaluation the agents ability to extrapolate those learned behaviors to new tasks is determined. The tasks described are designed so as to be challenging in a way that a neural network trained on a baseline set of conditions would fail if presented with different evaluation parameters. Generalization ability is needed in order to succeed the tasks described herein.

4.3.1 Evaluation Parameters

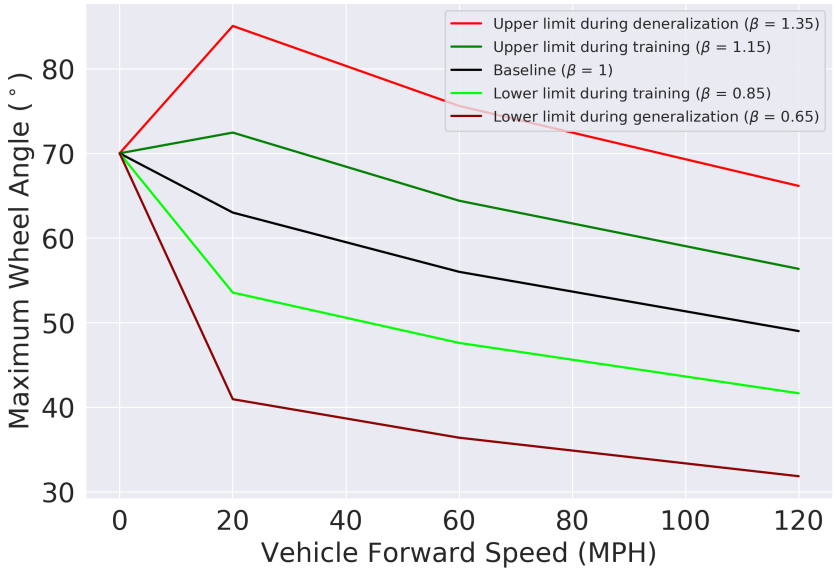
As described in Section 4.1, CARLA provides a number of vehicle properties that may be modified in order to adjust the behavior of the vehicle. For the present study, two of these properties were selected for the training and generalization experiments. The vehicle torque curve and the vehicle steering curve were modified during experimentation in order to provide unseen tasks to which the agent would learn to adapt.

Figure 4.3 shows the extremes of these two properties tested during training and generalization experiments. These curves were obtained by applying two parameters, the torque and steering multipliers, to the respective default curves provided by CARLA for the chosen vehicle (the same vehicle was used for all experiments).

The modified torque curve for a run is obtained by scaling the default value for the torque produced by the engine at an engine speed of 890 RPM by the value of the torque multiplier. For example, applying a torque multiplier value of $\alpha = 1.1$ would yield a curve whose torque produced at an engine speed of 890 RPM is



(a) The torque curve describes the amount of torque produced by the vehicle engine at any given engine speed.



(b) The steering curve describes the maximum angle to which the front two wheels of the vehicle are able to turn (relative to the rest state) at any given speed. Generally the faster a vehicle travels, the less the wheels are able to turn.

Figure 4.3: Vehicle properties altered during training and generalization experiments. Shown here are the baseline values assigned, as well as the upper and lower limits evaluated during training and generalization experiments. These deviations from baseline behavior present the agent with unseen tasks to which it must learn to adapt.

$1.1 * 500.76 = 550.836$ Nm, while the remaining values in the torque curve remain unchanged from the default values.

The modified steering curve for a run is obtained by scaling the default values for the maximum wheel angle of the two front wheels at vehicle speeds of 20, 60, and 120 MPH by the value of the steering multiplier. For example, applying a steering multiplier value of $\beta = 1.1$ would yield a curve whose maximum wheel angle at a vehicle speed of 20 MPH is $1.1 * 63 = 69.3°$, maximum wheel angle at a vehicle speed of 60 MPH is $1.1 * 56 = 61.6°$, and maximum wheel angle at a vehicle speed of 120 MPH is $1.1 * 49 = 53.9°$. The maximum steering angle at rest remains unchanged.

Thus these two properties, torque and steering curves, are reduced to two evaluation parameters, the torque multiplier α and the steering multiplier β. A pair of values for the torque and steering multipliers are required inputs for each run and evaluation as defined in Section 5.1.

Various other parameters such as vehicle mass, wheel friction, and drag coefficient, were found in preliminary experiments to have negligible effect on driving performance. That is, individuals trained only on baseline values performed well in evaluations with significantly different values for these parameters. For example, changing vehicle mass between training and generalization did not present problems for agents even with very rudimentary training. Torque and steering curves were chosen as the dynamic variables for the "unseen" tasks as these properties were observed to have significant effect on vehicle behavior. These two properties directly affect the control of the vehicle (Section 4.2.2); the physical effect of steering wheel input is controlled by the steering curve, and the physical effect of throttle input is controlled by the torque curve.

4.3.2 Metrics

Three metrics were calculated for each evaluation of an agent on a track. The metrics were developed for the purpose of describing various aspects of vehicle performance; they can be used to compute objective functions to be evaluated during optimization. They are:

- **Ending Distance** Each evaluation ends after a specified amount of simulation time, or when the vehicle reaches the target destination on the current track. The ending distance is simply 10 times the Euclidean distance between the

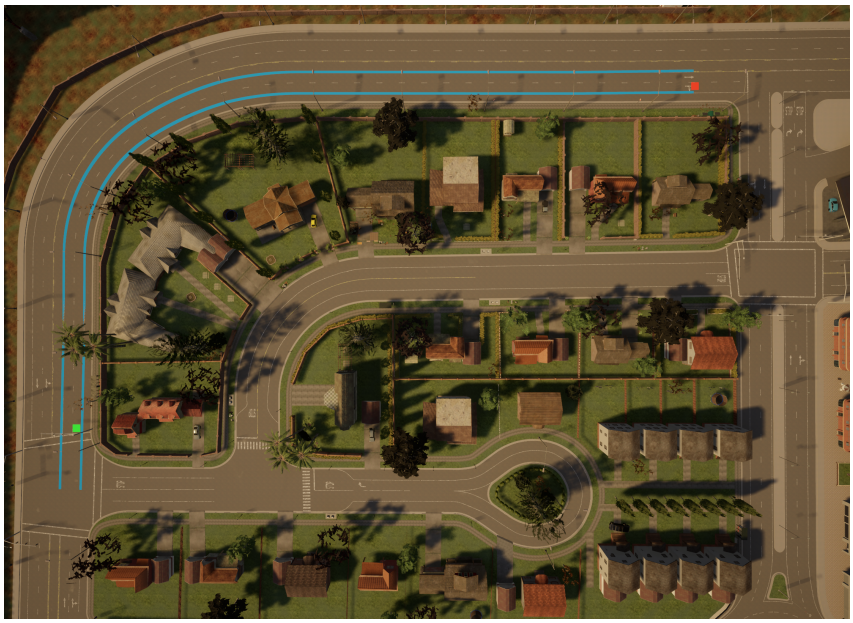
vehicle and the target destination. A vehicle that ends up being very far from the target destination by the end of the evaluation is one that is too slow. Thus the ending distance metric, χ, captures the performance of the agent during the evaluation.

- **Lane-Distance Penalty** At each time-step, the minimum distance between the center of the lane boundaries and the vehicle is computed. The cumulative sum of these distances yields the lane-distance penalty, ψ. An agent that achieves a low lane-distance penalty is one that manages to stay close to the center of the lane at all times. The lane-distance penalty metric captures the safety of the agent during the evaluation.

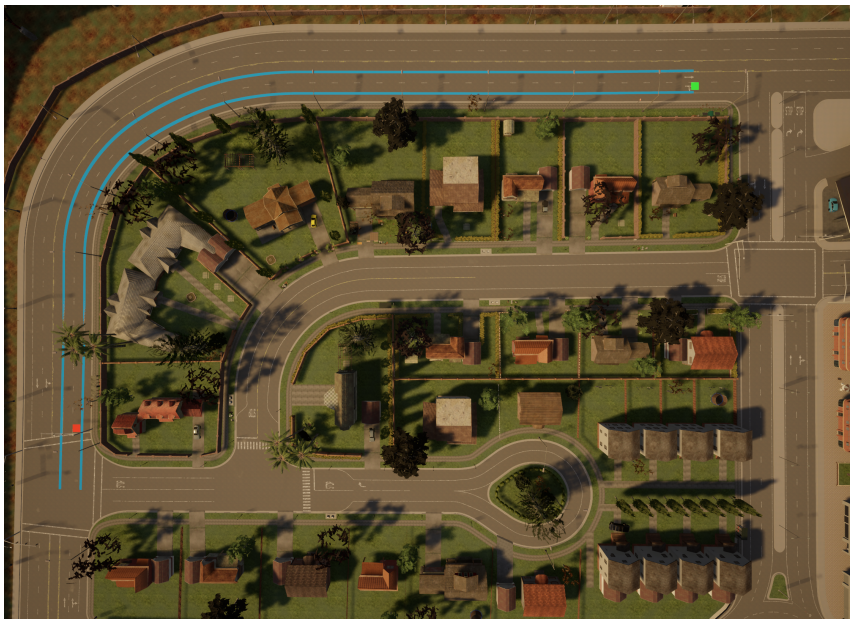
- **Wobble Penalty** At each time-step (except the first), the current position of the steering wheel is compared to the position of the steering wheel in the previous time-step. The absolute value of this difference is computed and the cumulative of these differences yields the wobble penalty. An agent that achieves a low wobble penalty is one whose driving performance is relatively smooth and not jerky. Jerky driving is deemed unsafe and uncomfortable, and as such low values for this metric are desired. Of course adjustments to steering are required to stay on track, however unnecessary readjustments due to over or under steering are highly undesired. The wobble penalty metric, ω, captures the comfort of the agent during the evaluation.

4.3.3 Tracks Used

Three tracks were developed for these experiments. Tracks 1 and 2 were chosen for training due to their relative simplicity and ease of navigation. As shown in 4.4a, Track 1 consists of two straight segments separated by a single right turn with a large radius of curvature, while Track 2 consists of two straight segments separated by a single left turn with the same radius of curvature. Furthermore, both tracks are relatively flat with respect to the z-axis. Although some incline and decline is present, it is relatively small in Tracks 1 and 2. The vehicle naturally loses speed during an incline, and depending on the degree of incline tends to rely more on lower engine speed torque. Successful traversal of Track 1 and 2 teaches agents how to navigate turns and how to control throttle output in response to inclines and declines.



(a) Track 1 is one of two tracks designed for the training experiments.



(b) Track 2 is identical to track 1, except that the starting position and the target destination are reversed.

Figure 4.4: Agents are trained on Tracks 1 and 2. The green and red markings represent the vehicle spawn point and target destination respectively. Training the agent on tracks 1 and 2 allows the agent the opportunity to learn how to turn both right and left, albeit with a wide radius of curvature. During evaluation the agent must learn to adapt to unseen curves with smaller radii of curvature.

Note that inclines and declines are critical features for these agents as the agents do not have access to vehicle braking. Vehicles lose speed while driving up an incline and gain speed while driving down an incline. The effects of steering output are reduced at higher speeds as shown in Figure 4.3a. Successful agents must learn to control their speed in anticipation of impending turns using throttle only.

Figure 4.5: Track 3 is used for generalization experiments. Track 3 presents the agent with an unseen task; containing turns with different radii of curvature than those encountered during training, as well much steeper inclines.

Chapter 5

Training

The experiments described herein were conducted in order to demonstrate the superior generalization ability of the Context+Skill neural network architecture. During training the agent is presented with tasks with varying degrees of deviation from a standard "baseline" driving task. Section 4.3.1 describes the these deviations in detail. During evaluation the extent of these deviations is increased beyond what the agent encounters during training. Thus the agents' performance at these unseen tasks provides a metric for evaluating the extrapolation ability of the Context+Skill neural network.

5.1 Evolutionary Method

A description of the evolutionary method used in the experiments presented in this thesis is provided.

5.1.1 Individual

An individual is represented by a fixed-length list of values i.e. its genome. The values that comprise a genome are subject to change as evolution takes place, by means of mutation and/or recombination. Given a neural network architecture with the appropriate number of parameters, the values within the genome of an individual are deterministically rearranged to give rise to the neural network. The values that comprise the individual correspond to the weights and biases of the neural network.

5.1.2 Population

A population is a fixed-length list of individuals. Individuals within a population may evolve, mate, or die, however the total number of individuals within a population must remain constant. Furthermore the dimensionality of every genome in a population is the same. That is, every individual within a population has the same number of values that comprise its genome.

5.1.3 Evaluation

An evaluation is obtained by representing the values within an individual genome as weights and biases of a neural network, and giving that neural network control of a vehicle in the CARLA environment. An evaluation requires as input the individual to be evaluated, the track on which to be evaluated, and a single set of parameters (torque and steering multipliers α and β) to use for the duration of the evaluation. Evaluations terminate by either timeout (maximum number of simulation steps) or with the car reaching a certain minimum distance from the specified target for the track it is being evaluated on. Each evaluation yields three values: ending distance, lane distance penalty, and wobble penalty (see Section 4.3.2).

5.1.4 Run

A run is obtained by running multiple evaluations of the same individual on the same set of evaluation parameters, and averaging the corresponding values obtained from each evaluation. A run requires as input the individual to be run, the set of tracks upon which the individual is to be evaluated, and a single set of evaluation parameters. Each run yields three values: the average ending distance from the multiple evaluations, the average lane distance penalty from the multiple evaluations, and the average wobble penalty from the multiple evaluations. For instance, a run might consist of evaluating a particular agent on track A and on track B, with evaluation parameters (α_1, β_1). Six values would be obtained in this example (three values for track A and three values for track B), but the run would only yield three values representing the average ac cross all evaluations.

5.1.5 Generalization Round

A generalization round is obtained by combining scores from multiple runs of the same individual on multiple sets of evaluation parameters. A generalization round requires as input: the individual to be tested, a set of evaluation parameters, and a set of tracks. Each generalization round yields three values: the average ending distance from the multiple runs, the average lane distance penalty from the multiple runs, and the average wobble penalty from the multiple runs. As opposed to the score obtained from a single run, the score obtained from a generalization round gives insights into the generalization capabilities of the given individual. For instance, a generalization round might consist running an individual using the following set of evaluation parameters $[(\alpha_1, \beta_1), (\alpha_2, \beta_2), (\alpha_3, \beta_3), (\alpha_4, \beta_4)]$ on tracks A and B. A total of four runs (eight evaluations) would be obtained. The average values of the evaluation parameters from the four runs would be returned as the return value of the generalization round.

5.1.6 Generation Evaluation

A generation evaluation consists of obtaining generalization scores for each individual in a population. Each generation requires as input: a population of individuals to be evaluated, multiple sets of evaluation parameters, and multiple tracks. Each generation evaluation yields the scores (three values per individual) of every individual in the population. For instance a generation evaluation might consist of evaluating a population of individuals $[I_1, I_2, ..., I_N]$ on a list of training parameters $[(\alpha_1, \beta_1), (\alpha_2, \beta_2), (\alpha_3, \beta_3), (\alpha_4, \beta_4)]$ on tracks A and B. The generation evaluation results in a list of scores $[(\chi, \psi, \omega)_1, (\chi, \psi, \omega)_2, ..., (\chi, \psi, \omega)_N]$ corresponding to one generalization score (set of three values) for each individual in the population.

5.1.7 Training Experiment

A training experiment consists of evolving a population of individuals until a specified termination criteria is reached. A training experiment is conducted for each neural network architectures being compared. At the beginning of a training experiment an initial population is populated by randomly generating a specified number of individuals. Each training experiment requires as input: a starting population of individuals, a neural network architecture (whose number of parameters matches

the number of values that comprise each individual in the population), a stopping criterion, a set of tracks, and multiple sets of evaluation parameters. Each training experiment yields a variable number of generation results (run-scores for each individual in each population where the number of populations obtained is equal to the number of generations needed until the termination criterion was reached). The following algorithms and parameters are used for evolution of agents for the training experiments.

- NSGA-II (Non-dominated Sorting Genetic Algorithm II) (5) is used to train and select the agents between generations;
- A population size of 48 is used. This number was chosen in order to maximize genetic diversity with the computational constraints;
- A crossover probability of 0.9 is used;
- A crossover crowding degree of 20 is used;
- An individual mutation probability of 1/48 is used;
- A mutation crowding degree 20 is used.

5.1.8 Generalization Experiment

A generalization experiment is performed on a single individual on a single track over a list of evaluation parameters $[(\alpha_1, \beta_1), (\alpha_2, \beta_2), ..., (\alpha_N, \beta_N)]$. Each generalization experiment yields corresponding evaluation scores $[(\chi, \psi, \omega)_1, (\chi, \psi, \omega)_2, ..., (\chi, \psi, \omega)_N]$ for each set of evaluation parameters.

5.2 Generalization

During generalization, a chosen individual is run using multiple sets of evaluation parameters, and a set of metrics is obtained for each set of evaluation parameters. Results from generalization can be visualized using contour plots or histograms (see Chapter 6). Because two evaluation parameters are varied in the experiments performed, a two-dimensional plot with two axes is sufficient for illustrating the distribution of chosen evaluation parameters for a chosen metric. A contour plot can

be generated for each evaluation metric wherein the color of each point corresponds to the value of the metric for the chosen individual and set of evaluation parameters represented by the coordinates of the point on the contour plot.

Alternatively, histograms can be used to visualize the relative performance of two individuals for the same suite of evaluation parameter sets. For each point on a contour plot (set of evaluation parameters), the difference of the values of a chosen metric for two individuals can be computed. The distribution of these differences can tell us which individual is better for the chosen metric. For instance, if lower values of the chosen metric indicate better performance and the differences are computed by subtracting the corresponding metric values of Individual 1 from Individual 2, a histogram that is skewed to the left would indicate that the Individual 2 is better, and vice-versa.

During generalization training:

- Each run is comprised of two evaluations: an evaluation on Track A and an evaluation on Track B.
- Each generalization round is comprised of 12 runs. Six runs are generated by uniform-randomly sampling the range $[0.85, 1.15]$ six times, and using those values as the torque multiplier (see Section 4.3.1), while holding the steer multiplier at a constant value of 1. The remaining six runs are generated by uniform-randomly sampling the same range of $[0.85, 1.15]$ and using those values as the steer multiplier (see Section 4.3.1) while holding the torque multiplier at a constant value of 1.

Thus generalization during training exposes the agents to various environments. The intent is that the agent learns to become aware of context during training, and this awareness allows the agent to extrapolate learned behavior of dealing with varied environments during evaluation. During evaluation, the range of variation along these two exes is widened significantly beyond anything seen during training.

5.3 Objective Function

During evolution of individuals, multi-objective optimization is used to obtain optimal fitness. The objective functions were chosen in order to optimize performance

(f_0) and safety (f_1). The intent is to have objective functions that adequately describe the behavior of the vehicle in a way that would be meaningful to a human observing the agent. Superior performance means that an agent can drive from a starting point to a desired destination in a time efficient manner. Superior safety means that an agent can stay within its lane, not crash, and not wobble.

As described in (19), multiobjective optimization is ideal for situations wherein multiple traits are desired, however simultaneous performance in these traits poses a conflict. Such is the case for the present problem. For instance, a vehicle may achieve ideal safety by simply remaining in its starting position. The agent will remain in its lane, will not wobble at all nor pose any danger to other vehicles on the road. Obviously such behavior is not desired. Similarly, a vehicle may achieve superior performance by leaving its lane and simply driving towards the target destination using the most direct path. Such behavior would pose serious risks to passengers within the agents vehicle and other vehicles on the road if they are present.

Therefore, the first objective function, performance, is calculated as

$$f_0 = \chi,$$

where χ is the ending distance as described in Section 4.3.2. The second objective function, safety, is calculated as

$$f_1 = \psi + 5.5\omega,$$

where ψ is the lane distance penalty and ω is the wobble penalty as described in Section 4.3.2.

5.4 Termination Criteria

Training terminates when, for a given network, any single agent from the latest population meets both of the following conditions:

- The score of the agent for objective function f_0 is below 55; and
- The score of the agent for objective function f_1 is below 2043.

The reasoning for having two cutoffs is that the desired behavior of the agent must be both safe and performant. Because the population distributions generally ap-

proximate a Pareto front, the objective function values of the best individual in the population is considered a fair metric for how optimal the population is as a whole (especially the best individuals, which are the ones that matter the most).

A consequence of using such termination criteria during training is that the three networks being tested may be trained for a different number of generations. This process is desirable since the various networks have different numbers of free parameters. It is reasonable to assume, for example, that a neural network with a greater number of free parameters may need a greater number of training generations in order to reach the same level of performance and safety as a neural network with fewer parameters. Ultimately, a best-individual-score-based termination criteria normalizes for what might otherwise have been a confounding variable (the number of free parameters in the various neural networks).

Furthermore, it is crucial to remember that the property of interest is *generalizability*. The baseline in the performance and safety of individuals are not the metrics of interest. Ideally, the goal is to have one agent representing each of the neural networks and have each of these agents achieve the same level of performance and safety under baseline conditions (steering and torque multiplier both equal to 1), so that their respective abilities to generalize can be compared. The chosen termination criteria helps to achieve something close to this outcome.

5.5 Individual Selection

As described in Section 5.1, during generalization experiments a single individual is chosen from a population in order to test generalizability, and to determine the metrics for its respective neural network architecture. The individual to use for generalization experiments is chosen from the population by selecting the individual with the lowest Euclidean distance from the origin when objective function values (f_0, f_1) are used as coordinates. Because the population distributions generally approximate a Pareto front, the individual with the lowest Euclidean distance is deemed as the individual with the fairest trade-off between performance and safety. This method of individual selection was chosen after manual inspection of the performance of many individuals in a pareto front and was determined to be a fair and consistent method for obtaining an individual from a population whose performance best reflected the performance of its respective population.

Chapter 6

Results

The results obtained from the aforementioned experiments indicate that the Context+Skill network is able to obtain overall superior generalization performance in the tasks described in Chapters 4 and 5 as compared to the Context-only and Skill-only networks given similar training regimens. Once all three networks had been trained to a sufficient level of safety and performance, the context skill network managed to outperform both Skill-only and Context-only networks while maintaining at least similar (in the case of comparison against the Context-only network) or superior (in the case of comparison against the Skill-only network) standards of safety during the generalization evaluation.

6.1 Evaluation Metric Differences Distributions

Each network is evaluated on an experiment grid wherein the two parameters (torque and steering multipliers) are varied. For each point on the grid and for each objective function the score of two networks are subtracted. Figures 6.5, 6.6, 6.9, 6.10, 6.13, and 6.14 are contour plots that illustrate the relative performance and safety scores of the Context+Skill network as compared to the Context-only and Skill-only networks. For both objective functions, lower scores are desirable because the optimization problems are both minimization problems. Negative values indicate superior performance or safety of the Context+Skill network as compared to the alternative network.

6.2 Safety

As described in Section 5.3, the safety score for a run is computed using a weighted average of two metrics: lane distance penalty and wobbliness penalty. As in the real world, a driver is generally considered a safer and more comfortable driver if they are able to stay as close to the center of the lane as possible without jerking the car (minimal changes in steering). As one might imagine, these two metrics can be difficult to optimize simultaneously, especially when the path to be driven features many bends and turns: A road with many bends and turns necessitates turning the steering wheel and some readjustment. Interestingly, the Context-only network performs worst using this metric. The Context+Skill network roughly matches the wobbliness of the Skill-only network (Figure 6.11), and performs much better than the Context-only network when evaluated using this metric (Figure 6.12).

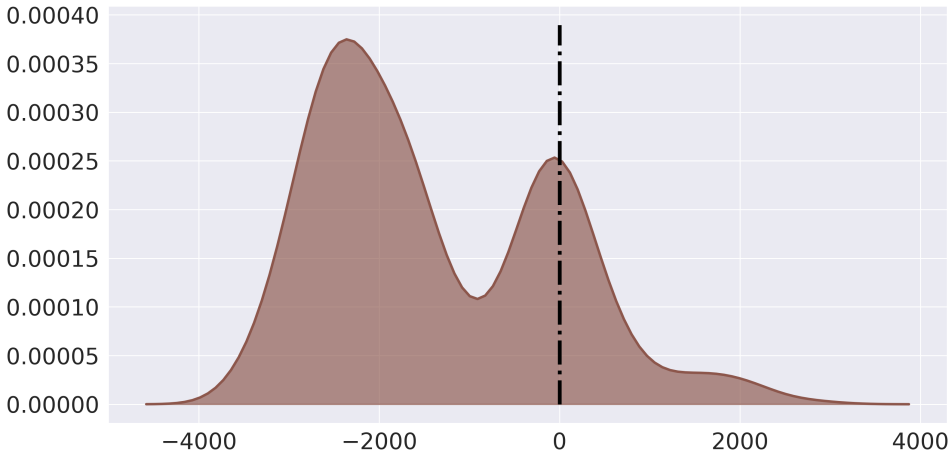


Figure 6.1: Distribution of the differences $S_{CS} - S_S$, i.e. the safety scores of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus lower is better.

6.3 Performance

As mentioned in Section 5.3, performance is represented by the distance that the agent is from the target at the end of an evaluation. We observe in Figure 6.3 that the Context+Skill network clearly outperforms the Skill-only network in this

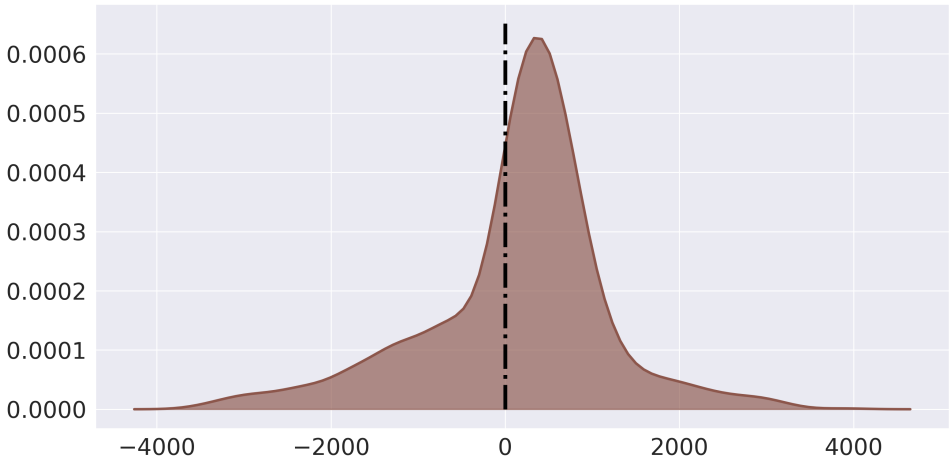


Figure 6.2: Distribution of the differences $S_{CS} - S_C$, i.e. the safety scores of the Context+Skill and Context-only networks. The problem is a minimization problem, thus lower is better.

metric. This confirms the hypothesis of this work: the Context+Skill network is able to extrapolate learned behavior in a way that a network that lacks context awareness is not. The extreme negative values indicate positions on the experiment grid where the Skill-only network likely crashed the car and therefore the agent was not able to find the target by the simulation timeout. For the same set of conditions, the Context+Skill network was able to adapt and complete the track. These findings are confirmed in Figure 6.5.

The Context-only aware network roughly achieves similar (if not slightly worse) generalization performance as compared to the Context+Skill network.

6.4 Lane Distance Penalty

As with the performance metric, the Context+Skill network vastly outperforms the Skill-only network as shown in Figure 6.7. This is in agreement with prior conclusion that the cars controlled by the Skill-only network crashed during the generalization simulations. If the car crashes and gets stuck off of the track, the lane distance penalty quickly adds up. The Context+Skill network and Context-only networks both adapted well to the unseen conditions.

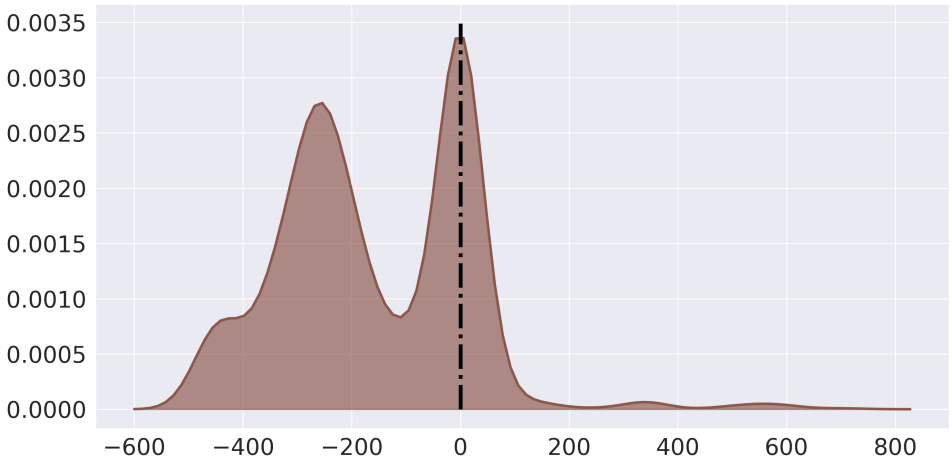


Figure 6.3: Distribution of the differences $P_{CS} - P_S$, i.e. the performance scores of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus lower is better.

6.5 Wobble Penalty

The wobble penalty metric measures smoothness of driving behavior. Jerky steering is penalized with the addition of the wobble penalty in the safety objective function. For this metric, it is observed that the Context+Skill network outperforms the Context-only network as illustrated in Figure 6.12. This difference implies that the skill module has a stabilizing effect on the overall neural network. The Context+Skill network is thus able to provide a smoother (and thereby safer) drive than the Context-only network.

The Skill-only network achieves roughly equal wobbliness as compared to the Context+Skill network, however this is expected given that the cars in these simulations crashed before reaching the target destination; if the car is crashed and idle it has no reason to wobble the car.

6.6 Training Time

During training it is unsurprisingly found that the Skill-only network required the least amount of training time in order to reach the training termination condition,

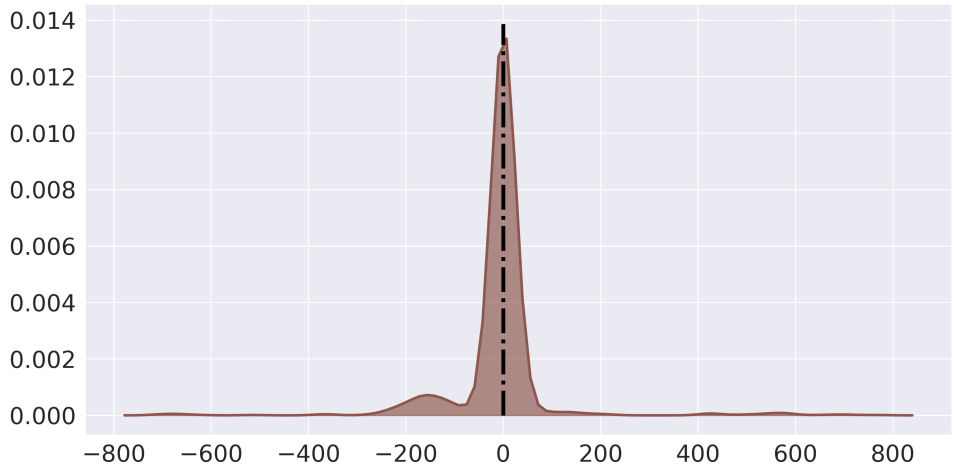


Figure 6.4: Distribution of the differences $P_{CS} - P_C$, i.e. the performance scores of the Context+Skill and Context-only networks. The problem is a minimization problem, thus lower is better.

due to the simplicity of the Skill-only network architecture. The Context+Skill network requires the greatest number of generations to reach the required conditions for training termination, due to the complexity of its architecture and the number of free parameters as compared to the other two networks. The training curves of all three networks are shown in Figure 6.15.

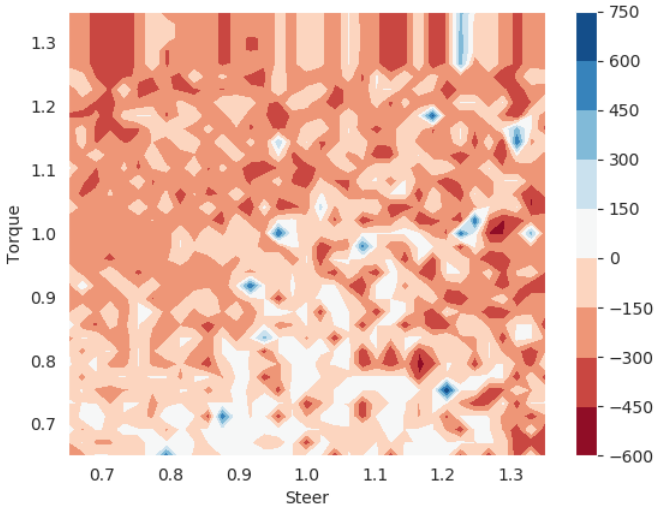


Figure 6.5: Heat map of the differences $P_{CS} - P_S$, i.e. the performances of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

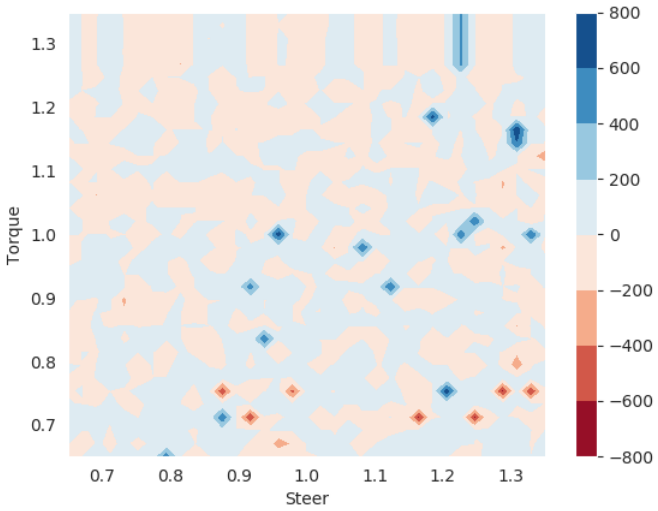


Figure 6.6: Heat map of the differences $P_{CS} - P_S$, i.e. the performance of the Context+Skill and Context-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

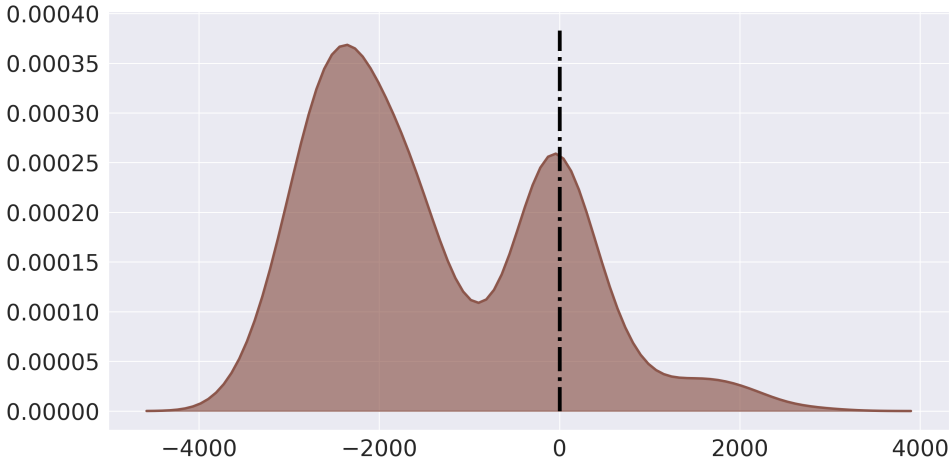


Figure 6.7: Distribution of the differences $LD_{CS} - LD_S$, i.e. the lane distance penalties of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus lower is better.

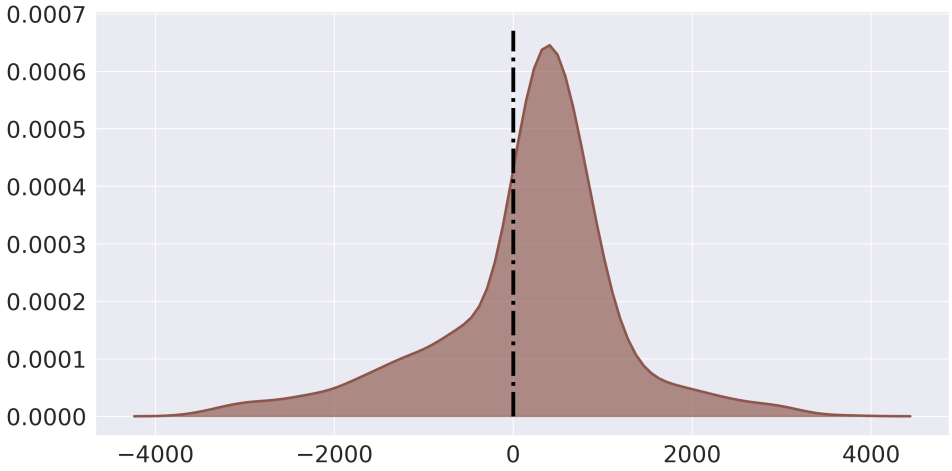


Figure 6.8: Distribution of the differences $LD_{CS} - LD_C$, i.e. the lane distance penalties of the Context+Skill and Context-only networks. The problem is a minimization problem, thus lower is better.

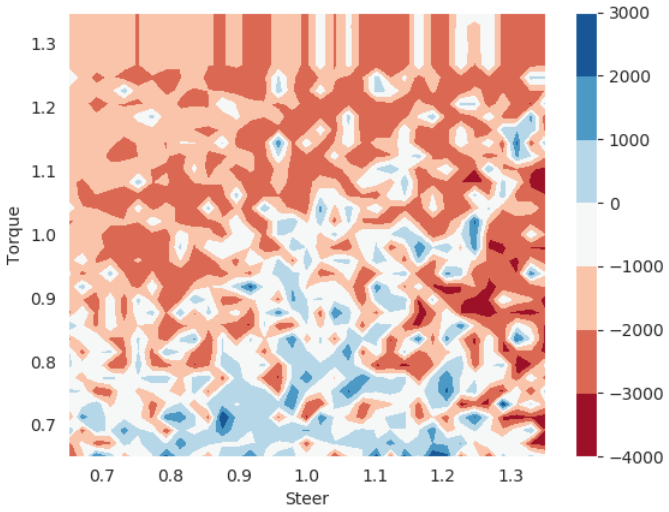


Figure 6.9: Heat map of the differences $LD_{CS} - LD_S$, i.e. the lane distance penalties of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

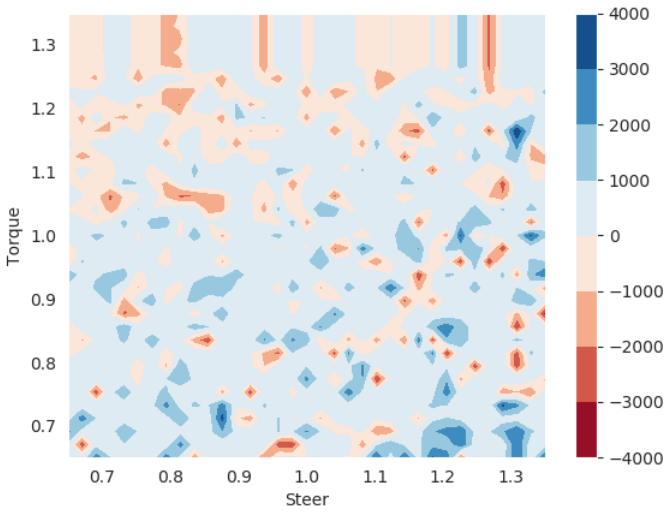


Figure 6.10: Heat map of the differences $LD_{CS} - LD_S$, i.e. the lane distance penalties of the Context+Skill and Context-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

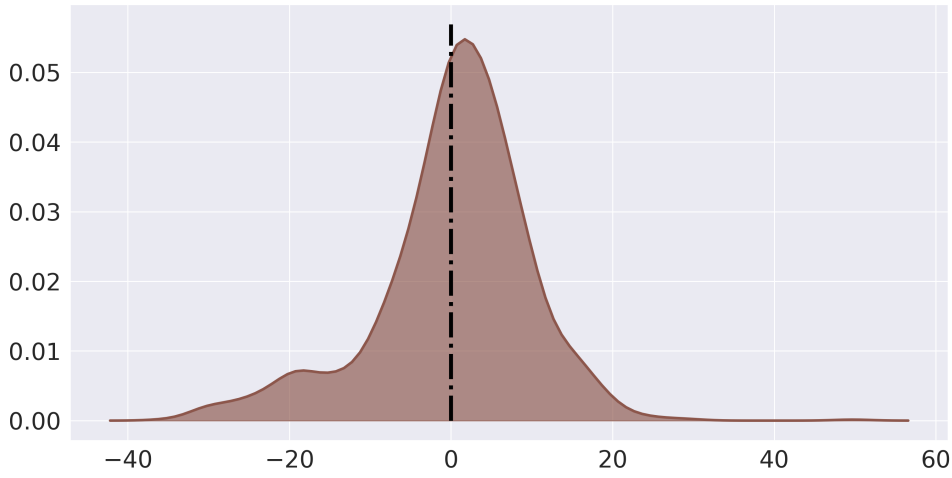


Figure 6.11: Distribution of the differences $W_{CS} - W_S$, i.e. the wobble penalties of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus lower is better.

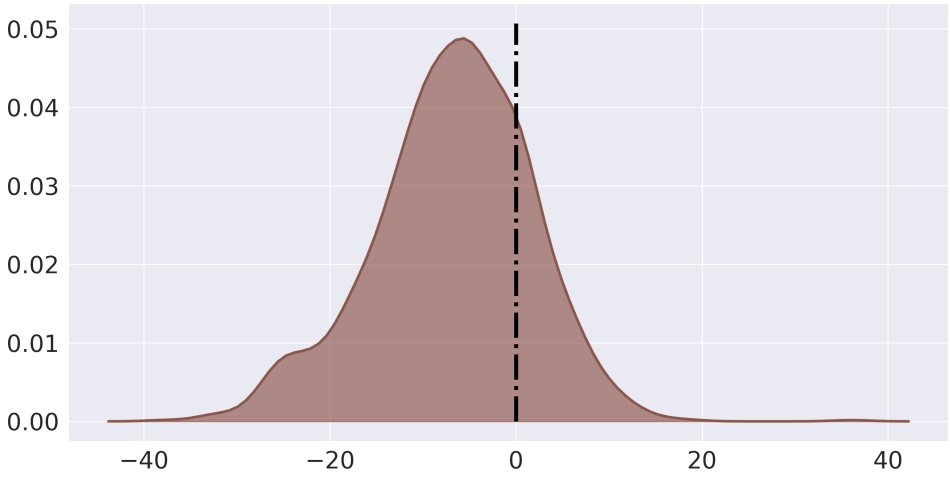


Figure 6.12: Distribution of the differences $W_{CS} - W_C$, i.e. the wobble penalties of the Context+Skill and Context-only networks. The problem is a minimization problem, thus lower is better.

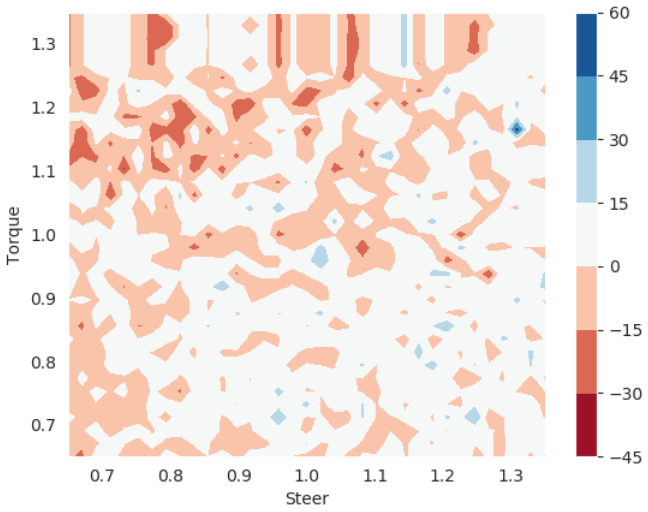


Figure 6.13: Heat map of the differences $W_{CS} - W_S$, i.e. the wobble penalties of the Context+Skill and Skill-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

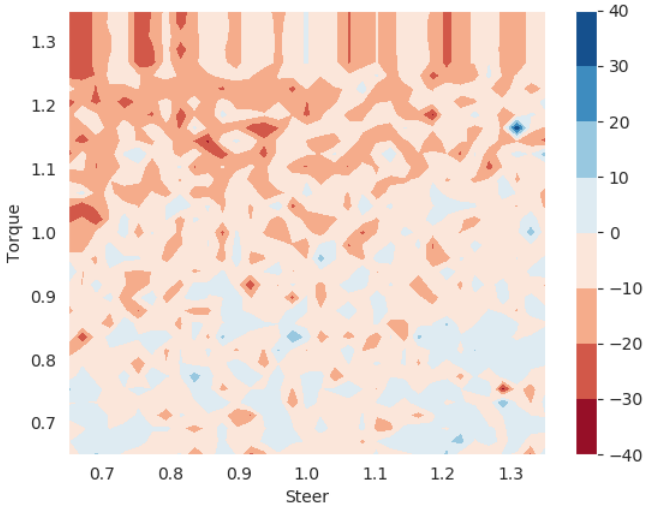


Figure 6.14: Heat map of the differences $W_{CS} - W_S$, i.e. the wobble penalties of the Context+Skill and Context-only networks. The problem is a minimization problem, thus negative values, represented by red, indicates superior Context+Skill generalization.

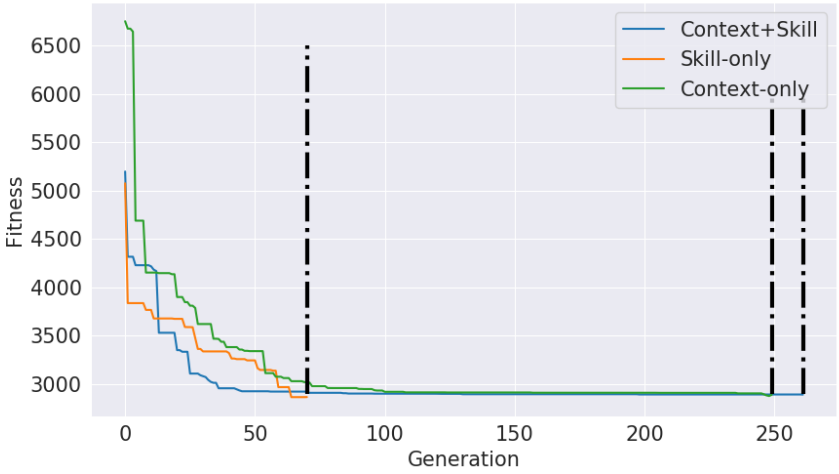


Figure 6.15: Training curves of the three networks tested. The Y-values correspond to the smallest euclidean distance from the origin of any individual in the pareto curve for the current generation of the given population. The X-values correspond to generation number.

Chapter 7

Discussion and Future Work

The Context+Skill neural network was proposed in this thesis as a solution to the problem of finite training data and subpar generalization abilities in state-of-the-art neural networks in the domain of autonomous vehicles. The Context+Skill network offers a hitherto unexplored architecture for adaptation to unseen driving conditions. The thesis builds on previous work in the autonomous vehicle domain as well as other domains in which the Context+Skill network has been successful in adapting to unseen conditions. The full range of generalizability of the Context+Skill network remains to be explored in future work. Given additional funding and time, a number of avenues for future research become available; this chapter explores some of those avenues.

7.1 Perception Modules

The experiments in this thesis relied on custom-designed rangefinders for determining vehicle position with respect to the track (See Section 4.2.3). Most contemporary experiments in the domain of autonomous vehicles used more advanced perception modules (such as Convolutional Neural Networks 9) for encoding sensory input into an internal representation of the world. The choice of rangefinders over other such perception modules was motivated by the intent to limit the confounding variables to only those directly related to the generalization abilities of the networks being compared.

A perception module comprised of a convolutional neural network may be

able to encode the agent state more efficiently than the rangefinders were. With the the manually designed rangefinders the state of the agent was compressed into five values that to a human seem intuitive. It is possible that a trained neural network might be able to come up with a more prudent encoding of state. It remains to be seen if superior performance for any of the three networks tested can be achieved by the addition of a more traditional perception module.

7.2 Weight Decay and Dropout

As discussed in section 2.4, the literature has ample potential solutions to the generalization problem. The goal of the present study was to present a novel neural network architecture with superior generalizability and evaluate its effectiveness in the CARLA domain. Further experimentation may show some of the less novel approaches discussed in section 2.4 can be combined to yield even superior generalization performance as compared to the results obtained and presented in this thesis.

Augmentations of the Context+Skill neural network inspired by techniques such as dropout and weight decay are likely to have the most benefit if applied to the controller module (Chapter 3). The purpose of the controller module is to enhance the decision of the skill network using the encoding of context obtained from the context module. Overfitting in the controller module is still possible however, and as mentioned in Section 2.4.1, smaller values for neural network parameters seem to alleviate this problem to an extent. Thus, performance of the Context+Skill network may benefit from implementing weight decay. Similarly, the resiliency of the controller module to the varied conditions seen during training may be improved by simulating the loss of certain neurons as done in dropout. Further research is needed to determine the effectiveness of these strategies for the task at hand.

7.3 Task Complexity

The tracks and evaluation parameters were chosen to represent major classes of scenarios that an agent might encounter while driving. The two tracks chosen during training incorporated both left and right turns, and inclines and declines. Additionally during testing a new track was evaluated that contained turns with radii

of curvatures significantly smaller as well as inclines much steeper than anything encountered during training. The Context+Skill network adjusted well to these changes. However it remains to be seen how well the agent might adapt to other differences such as terrain differences or full U-turns.

The evaluation parameters represented dampeners and amplifiers to the two major controls used when driving, steering and acceleration. However other vehicle parameters exist and could potentially have significant impact on driving behavior. Given additional time and compute resources other axes for extrapolation such as tire pressure, vehicle weight, and vehicle center of gravity could be investigated. It may be the case that context awareness does not benefit generalization along all dimensions as much as it did for the torque and steering multipliers. Insights about the effectiveness of context awareness for generalization will contribute to the understanding of the mechanics of neural networks and aid progress in the field of machine learning research at large.

7.4 Metrics

The metrics described in 4.3.2 were designed to optimize for specific driving behaviors deemed desirable for the specific experiments described in this thesis. High speed was rewarded implicitly in the ending distance metric (an agent that drives too slow will be further away from the target destination at the time of evaluation timeout). Short lane distance was rewarded to solve the problem of deviation from a specified lane.

In the real world, actual driving performance and safety might be represented by other objective metrics. Instead of a predefined lane center to which the agent must try learn to stay close to, the safety of the agent might be determined by how much distance it maintains from other vehicles on the road (roads were void of any other vehicles during the simulations run for these experiments). The safe response of an agent to erratic behavior of other vehicles on the road is arguably an even more coveted feature of self driving agents. Additional research in this particular type of generalization would pave the way for meaningful progress in the autonomous vehicle domain.

7.5 Training Termination Criteria

The Termination Criteria 5.4 chosen for the number of generations to train each model was determined somewhat subjectively. Videos of characteristic individuals were generated and observed; driving behavior beyond a certain performance and safety threshold were determined to be sufficient for the purpose of the experiments in this study; that is, the agents were able to traverse the course within acceptable time, jerkiness, and average distance from the center of the target lane. Of course it is entirely feasible that alternative thresholds for performance and safety might yield different results. For example, decreasing the performance and safety thresholds (the optimization is a minimization problem, therefore lower values for the metrics yield more desirable behavior) for training termination will ensure a greater level of maturity of the agents with respect to the given task. Greater maturity may benefit the networks unevenly, thereby widening the gap between generalization performances of the networks during testing.

It is also possible that some of the network architectures did not reach their true potential because of the chosen termination criteria. These criteria were necessary because of computation resource limitations, however given more resources a more appropriate termination criteria might be used. For instance, the change in population performance across generations might work well. That is, training stops only when the average performance of the last x generations is less than the a fixed multiple λ of the average performance of x generations immediately preceding the most recent x generations. In other words, training should only terminate when performance has plateaued. Such a stopping criteria would require significant compute resources. Improved familiarity with the task may not translate to evenly distributed improved generalization performance for all three networks. The hypothesis here is that upon maturation, the Context+Skill network achieves superior generalization performance as compared to the Skill-only and Context-only networks. Thus, it is possible that a stopping criteria that more appropriately reflects neural network maturation may yield results that more accurately reflect the potential of the Context+Skill neural network with respect to generalization ability.

These proposals for future work are offered in order to show that the results presented in this thesis constitute a starting point for the use of context awareness in a wider range of applications. Further experimentation could expand the applica-

bility of this approach within the autonomous vehicle domain and beyond. If further research yields that context awareness can contribute to the current state-of-the-art in the autonomous vehicle domain, the work in this thesis would be a stepping stone towards more robust Artificial Intelligence.

Chapter 8

Conclusion

Generalization remains an open problem in machine learning. Especially in the autonomous vehicle domain, the inability of agents to generalize learned behavior well is a barrier to the proliferation of autonomous vehicle technology. This thesis proposes an approach to generalization in the autonomous vehicle domain and juxtaposes it with others such as increased diversity during training, deep imitation learning, drop out, and weight decay.

The methodology for testing generalization performance provided in this thesis can serve as a framework for future endeavors in this area. Generalization ability is a less tangible attribute to optimize than e.g. sheer driving performance. Optimal driving behavior is comprised of many components, such as safety and performance (Section 4.3.2). The research and experiments presented in this thesis suggest a method for measuring generalization along multiple such dimensions.

The context+skill neural network is a novel architecture with potential to advance the field of autonomous vehicles. The need for context awareness in self-driving agents is a hitherto neglected insight, as demonstrated in Chapter 6. The ability of an agent to recognize that the environment parameters can change and that the current parameters should be taken into account when choosing actions is helpful for generalization, and the Context+Skill neural network offers a mechanism for achieving this awareness.

The applications of the Context+Skill approach are not be limited to the autonomous vehicle domain. Although the chosen domain for this thesis was autonomous vehicles, unlocking the ability of a neural network to extrapolate learned

behavior to conditions unseen during training will undoubtedly be a watershed moment in the broader field of machine learning research. Positive results in this domain should pave the way for additional applications of this novel approach into other domains such as recommendation systems, robotics, and healthcare. In each of these domains awareness of context, combined with state information will allow agents to make more informed decisions even when confronted with conditions different from those encountered during training.

The computational cost of running a simulation in CARLA limited the number of experiments that could be run with the resources available. Google Cloud Platform was used for the majority of the experiments, and costs associated with running high-fidelity CARLA simulations quickly became prohibitively expensive. As more compute becomes available, further research using the context+skill architecture along the avenues proposed in Chapter 7 should become possible.

In summary, the results presented in this thesis indicate that the context+skill neural network architecture is a promising approach to solving the generalization problem, at least in the autonomous vehicle domain. Of the neural networks tested, Context+Skill offers superior performance with equal or superior safety and comfort compared to its constituent components alone. The work in this thesis can be extended further by testing other dimensions of generalization (in addition to the steering and torque curve multipliers), extensions to the architecture, and augmenting the approach with conventional techniques including dropout and weight decay.

Bibliography

[car] *CARLA Python API reference.* https://carla.readthedocs.io/en/latest/python_api

[1] Ahmedov, Hasan B. ; Yi, Dewei ; Sui, Jie: Application of a brain-inspired deep imitation learning algorithm in autonomous driving. In: *Software Impacts* 10 (2021), 100165. http://dx.doi.org/https://doi.org/10.1016/j.simpa.2021.100165. – DOI https://doi.org/10.1016/j.simpa.2021.100165. – ISSN 2665–9638

[2] Alonso Raposo, María ; Grosso, Monica ; Mourtzouchou, Andromachi ; Krause, Jette ; Duboz, Amandine ; Ciuffo, Biagio: Economic implications of a connected and automated mobility in Europe. In: *Research in Transportation Economics* 92 (2022), 101072. http://dx.doi.org/https://doi.org/10.1016/j.retrec.2021.101072. – DOI https://doi.org/10.1016/j.retrec.2021.101072. – ISSN 0739–8859. – Innovations of modern technology and advanced modelling: new trends in transportation policy and economics

[3] Bojarski, Mariusz ; Testa, Davide D. ; Dworakowski, Daniel ; Firner, Bernhard ; Flepp, Beat ; Goyal, Prasoon ; Jackel, Lawrence D. ; Monfort, Mathew ; Muller, Urs ; Zhang, Jiakai ; Zhang, Xin ; Zhao, Jake ; Zieba, Karol: End to End Learning for Self-Driving Cars. In: *CoRR* abs/1604.07316 (2016). http://arxiv.org/abs/1604.07316

[4] Chang, Ming-Wei ; Ratinov, Lev ; Roth, Dan ; Srikumar, Vivek: Importance of Semantic Representation: Dataless Classification. In: *Proceedings*

of the 23rd National Conference on Artificial Intelligence - Volume 2, AAAI Press, 2008 (AAAI'08). – ISBN 9781577353683, S. 830–835

[5] Deb, K. ; Pratap, A. ; Agarwal, S. ; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. In: *IEEE Transactions on Evolutionary Computation* 6 (2002), Nr. 2, S. 182–197. http://dx.doi.org/10.1109/4235.996017. – DOI 10.1109/4235.996017

[6] Dingyi, You ; Haiyan, Wang ; Kaiming, Yang: State-of-the-art and trends of autonomous driving technology. In: *2018 IEEE International Symposium on Innovation and Entrepreneurship (TEMS-ISIE)*, 2018, S. 1–8

[7] Dosovitskiy, Alexey ; Ros, German ; Codevilla, Felipe ; Lopez, Antonio ; Koltun, Vladlen: CARLA: An Open Urban Driving Simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, S. 1–16

[8] Geng, Yuxia ; Chen, Jiaoyan ; Zhuang, Xiang ; Chen, Zhuo ; Pan, Jeff Z. ; Li, Juan ; Yuan, Zonggang ; Chen, Huajun: Benchmarking knowledge-driven zero-shot learning. In: *Journal of Web Semantics* 75 (2023), 100757. http://dx.doi.org/https://doi.org/10.1016/j.websem.2022.100757. – DOI https://doi.org/10.1016/j.websem.2022.100757. – ISSN 1570–8268

[9] Jain, Aditya K.: Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, S. 1630–1635

[10] Krogh, Anders ; Hertz, John: A Simple Weight Decay Can Improve Generalization. In: Moody, J. (Hrsg.) ; Hanson, S. (Hrsg.) ; Lippmann, R.P. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 4, Morgan-Kaufmann, 1991

[11] Li, Haidong ; Li, Jiongcheng ; Guan, Xiaoming ; Liang, Binghao ; Lai, Yuting ; Luo, Xinglong: Research on Overfitting of Deep Learning. In: *2019 15th International Conference on Computational Intelligence and Security (CIS)*, 2019, S. 78–81

[12] Li, Xun ; Miikkulainen, Risto: Dynamic adaptation and opponent exploitation in computer poker. In: *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018

[13] Loshchilov, Ilya ; Hutter, Frank: Decoupled Weight Decay Regularization. In: *International Conference on Learning Representations*, 2019

[14] Miikkulainen, Risto: *Neuroevolution*. New York : Springer, 2010 `http://nn.cs.utexas.edu/?miikkulainen:encyclopedia10-ne`

[15] Mishra, Yash ; Kumawat, Vijay ; Selvakumar, K.: Performance Analysis of Flappy Bird Playing Agent Using Neural Network and Genetic Algorithm. In: Gani, Abdullah B. (Hrsg.) ; Das, Pradip K. (Hrsg.) ; Kharb, Latika (Hrsg.) ; Chahal, Deepak (Hrsg.): *Information, Communication and Computing Technology*. Singapore : Springer Singapore, 2019. – ISBN 978–981–15–1384–8, S. 253–265

[16] Parekh, Vidhi ; Shah, Darshan ; Shah, Manan: Fatigue detection using artificial intelligence framework. In: *Augmented Human Research* 5 (2019), Nr. 1. `http://dx.doi.org/10.1007/s41133-019-0023-4`. – DOI 10.1007/s41133–019–0023–4

[17] Pérez-Gil, Óscar ; Barea, Rafael ; López-Guillén, Elena ; Bergasa, Luis M. ; Gómez-Huélamo, Carlos ; Gutiérrez, Rodrigo ; Díaz-Díaz, Alejandro: Deep reinforcement learning based control for autonomous vehicles in Carla. In: *Multimedia Tools and Applications* 81 (2022), Nr. 3, S. 3553–3576. `http://dx.doi.org/10.1007/s11042-021-11437-3`. – DOI 10.1007/s11042–021–11437–3

[18] Rojas, Raúl: *The Backpropagation Algorithm*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1996. `http://dx.doi.org/10.1007/978-3-642-61068-4{\_}7`. `http://dx.doi.org/10.1007/978-3-642-61068-4{\_}7`. – ISBN 978–3–642–61068–4

[19] Shir, Ofer M. ; Chen, Shahar ; Amid, David ; Boaz, David ; Anaby-Tavor, Ateret ; Moor, Dmitry: Pareto optimization and tradeoff analysis applied to meta-learning of multiple simulation criteria. In: *2013 Winter Simulations Conference (WSC)*, 2013, S. 89–100

[20] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014), Nr. 56, 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[21] Stanley, Kenneth O. ; Clune, Jeff ; Lehman, Joel ; Miikkulainen, Risto: Designing Neural Networks through Evolutionary Algorithms. In: *Nature Machine Intelligence* 1 (2019), 24–35. http://nn.cs.utexas.edu/?stanley:naturemi19

[22] Stanley, Kenneth O. ; Miikkulainen, Risto: Evolving Neural Networks through Augmenting Topologies. In: *Evolutionary Computation* 10 (2002), Nr. 2, S. 99–127. http://dx.doi.org/10.1162/106365602320169811. – DOI 10.1162/106365602320169811

[23] Staudemeyer, Ralf C. ; Morris, Eric R.: Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks. In: *CoRR* abs/1909.09586 (2019). http://arxiv.org/abs/1909.09586

[24] Tutum, Cem ; AbdulQuddos, Suhaib ; Miikkulainen, Risto: Generalization of Agent Behavior through Explicit Representation of Context. In: *2021 IEEE Conference on Games (CoG)*, 2021, S. 1–7

[25] Warner, Jamieson ; Devaraj, Ashwin ; Miikkulainen, Risto: Using context to make gas classifiers robust to sensor drift. In: *arXiv:2003.07292* (2020). http://nn.cs.utexas.edu/?warner:arxiv20

[26] Wen, Mingyun ; Park, Jisun ; Sung, Yunsick ; Park, Yong W. ; Cho, Kyungeun: Virtual Scenario Simulation and Modeling Framework in Autonomous Driving Simulators. In: *Electronics* 10 (2021), Nr. 6. http://dx.doi.org/10.3390/electronics10060694. – DOI 10.3390/electronics10060694. – ISSN 2079–9292

[27] Xian, Yongqin ; Lorenz, Tobias ; Schiele, Bernt ; Akata, Zeynep: Feature Generating Networks for Zero-Shot Learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018

[28] Yeh, Thomas Y. ; Reinman, Glenn ; Patel, Sanjay J. ; Faloutsos, Petros: Fool Me Twice: Exploring and Exploiting Error Tolerance in Physics-Based Animation. In: *ACM Trans. Graph.* 29 (2009), dec, Nr. 1. http://dx.doi.org/10.1145/1640443.1640448. – DOI 10.1145/1640443.1640448. – ISSN 0730–0301