RWTH Aachen University
Software Engineering Group

Comparison of Deep Learning Architectures on Simulated Environments

Seminar Paper

presented by

Bergerbusch, Timo

1st Examiner: Prof. Dr. B. Rumpe

2nd Examiner:

Advisor: Evgeny Kusmenko

The present work was submitted to the Chair of Software Engineering

Aachen, May 25, 2018

The present translation is for your convenience only.
Only the German version is legally binding.

Statutory Declaration in Lieu of an Oath

______________________________	______________________________
Last Name, First Name	Matriculation No. (optional)

I hereby declare in lieu of an oath that I have completed the present Bachelor's thesis/Master's thesis* entitled

____

____

____

independently and without illegitimate assistance from third parties. I have use no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

	The German version has to be signed.
______________________________	______________________________
Location/City, Date	Signature
	*Please delete as appropriate

Official Notification:

Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whosoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 to 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.

(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

I have read and understood the above official notification:

	The German version has to be signed.
______________________________	______________________________
City, Date	Signature

Abstract

The topic of autonomous driving using artificial intelligence increases in importance with the overwhelming amount of software usage within vehicles. For that *Convolutional Neural Networks* (CNNs), which try to figure out the importance of special areas of a single picture, have been shown to be promising.
In this paper we will give a general introduction to the topic of CNNs. We distinguish between the three main *deep learning languages* (DLLs) currently used and researched for autonomous driving agents: mediated perception, behaviour reflex and direct perception. Further we will compare different languages, which can be used to implement the different DLLs, based on the factors of usability, scope of functionality and the integration on a subject.
As a proof of concept we will train a CNN using the language *CNNArch* on the famous KITTI dataset in order to create a trained model. This model will then be tested on a test set created using either the simulation tool `MontiSim` or the open source racing game TORCS, containing multiple different challenging scenarios the agent has to manage.
Finally we evaluate the trained model on it's performance and try to reason, why it performed particularly good/bad, and give an overview based on the implemented test in order to state the similarities and differences of the languages.

Contents

Chapter 1

Introduction

The field of autonomous driving agents has rapidly increased in modern car manufacturing. Current research topic rises the agents from parking or lane keeping assistant fully autonomous driving agents obeying the traffic rules and having the ability to react to the volatile environment in a reasonable way.

For that machine learning techniques have proven themselves as an essential part. But in order to fulfil the security standards and create a sophisticated agent it has to be trained on hundred-thousands of scenarios each having a large set of data attached, for example sensor and camera data. The approach of *Convolutional Neural Networks* (CNNs) have been proven to be powerful enough to handle such many training iterations with a huge number of input variables, while maintaining the large learning capacity. [KSH12]

A CNN, as explained in Section 1.2, has a general structure, but can be altered to fit into the approach in various ways influencing the result. Therefore we introduce in Section 1.3 the three main approaches of using a CNN.

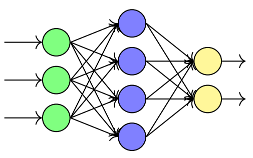
Further in Section 1.4 we see that there is the need of specific languages for the design and implementation of such agents. Two languages will be discussed and compared based on their suitability regarding the AlexNet, stated in Section 1.2.1.

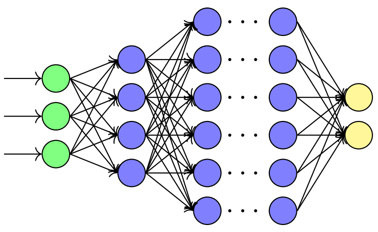
1.1 Neural Networks

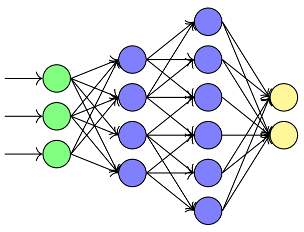
The neural networks are a construct adapted from biological processes. The general construct is very simple, but the expressiveness is very high but still not fully researched. A neural net is made out of neurons and has one very basic function: It takes a fixed number $n \in \mathbb{N}$ of the incoming values x_i and multiplies them each with a specific weight w_i, where $0 \leq i \leq n$. Also every neuron contains a bias b, which is a general value subtracted from the sum, so $\sum_{i=0}^{n}(x_i \cdot w_i) - b$. Often one applies an activation function to fix the value between 0 and 1. Such a function would be for example the sigmoid function or the ReLu. Without such a normalizing function the change in one neuron, which is very important could be very small compared to other values, which are not so important.

TODO: better example

This new value gets forwarded to the neurons of the next layer.


(a)


(b)


(c)

Those neurons are ordered in different groups often called layers, as seen in Figure 1.1a.

1. input layer (green):
   This layer gets fed with the input values of the problem, which can be for example sensor data or pixel color values.

2. hidden layer (blue):
   The hidden layer consists of neurons receiving the values from the previous layer, while not being obliged to have the same number of neurons (c.f. Figure 1.1a). Different hidden layer architectures can be distinguished to be deep (c.f. Figure 1.1b). This means, that there are multiple layers of neurons within the hidden layer itself. Also a variation within the hidden layer is the possibility of fully connectivity (c.f. Figure 1.1c). Thus some neurons don't forward their value to every neuron of the next layer.
   There is no rule of how to construct the best hidden layer, considering number of sub-layers, neurons per layer or the connectivity.

3. output layer (yellow): The output neurons contain the value the neural network produces. Depending on the neural networks purpose it can be for example a confidence value of a classification, like recognizing a stop sign, or the value of changing the steering wheel angle.

In order to train a neural network one has to define the behaviour the neural network should have. In an image classification example one should know what the correct class of a given image of a sign is, i.e. a speed limit sign.
A neural network can then be trained by giving it values for the input layer and comparing the values of the output layer with the solutions it should have resulted in. The difference can then be checked. Such a difference can be simply `true/false` or a value indicating how big the difference is. In the example of signs a classification of a "speed limit 70"-sign as "speed limit 50"-sign is still wrong, but as bad as a classification as a "stop"-sign.
Using this difference value the neural network can use linear algebra algorithms to adjust the weights w_i and biases b to improve the output iteratively.

For further information about the underlying training algorithms please see

---

TODO: training ref

1.2 Convolutional Neural Network (CNN)

A CNN is a special class of deep feed-forward neural networks. On of the main design goals of a CNN is that they require a minimal amount of preprocessing. This is an important aspect, because they are often fed with images. Preprocessing high resolution images is very costly in terms of computational time. In the context of autonomous driving the time is even more crucial, since the driving agent needs to be able to react to spontaneous events.

Like most parts of neural networks, also the CNNs are inspired by biological processes. It is mainly based on the connectivity pattern of an animals visual cortex, where special neurons respond only to stimuli of their receptive field, represented as rectangles lying in the image. Partially overlapping guarantees a complete coverage of the field of view. [Wik18]

The partitioning into those rectangles also has the advantage, that the size of the input image is not relevant. If there would be a direct correspondence of a pixel to one input value then a change of the size would infer null values or additional input values, where the weights are not directly well suited. But with partitioning it into sub-rectangles of the image the values can be unified by selecting these rectangles relative to the size.

These separation into those receptive fields has also the advantage that is reduces the effort to train a CNN. The weights and biases of neurons of each receptive field are equal. This is reasonable since for example a speed limit road sign should be identified independent whether it is located next to the road, like on a normal road, or above the road, like on an highway.

Further CNNs make strong and mostly correct assumptions about the nature of images, like stationary of statistics and locality of pixel dependencies. This leads to fewer connections and parameters, compared to a normal feed-forward neural net with similar sized layers, and therefore reduces the time it takes to be trained, while being only slightly worse in their best-performance. [KSH12]

TODO: erklären von kernel/filter

1.2.1 AlexNet

The *AlexNet* is on of the best performing CNN architectures currently known. It is trained on the ImageNet subsets of `ILSVRC-2010` and `ILSVRC-2012`[1] and became famous because of its result being way ahead of all other competitors.

A highly optimized GPU implementation of this architecture combined with innovative features is publicly available. Those features lead to improve performance and reduce training time. An important note is that the original test is dated back to 2012 and therefore was used with an overall GPU memory of 6GB, with which training took abound six days. With modern hardware the training can be done faster, or the model can be trained much more. [KSH12]

TODO: maybe calc the possible speed up based on “Hardware for Machine Learning”

[1]Further information: http://www.image-net.org/challenges/LSVRC/

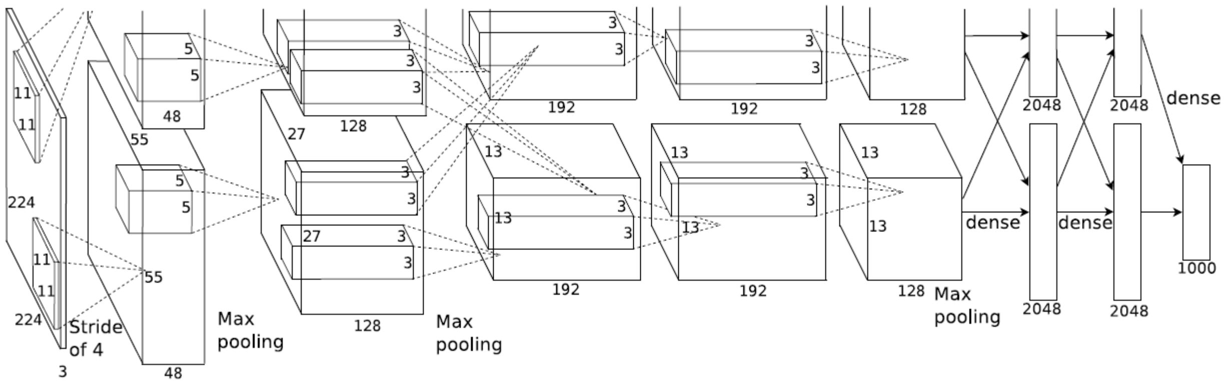


Figure 1.2: The AlexNet-architecture for two GPUs. It consists of 5 convolutional layers (at the beginning) and three fully-connected layers (at the end). The GPUs only communicate between two layers, but never within a layer.[KSH12]

TODO: Vereinfachen der Zeichnung

1.3 Available Deep Learning Approaches

There are various approaches of autonomous driving agents making a variety of assumptions and differ in numerous options. But they can be mostly categorized into two major groups of approaches: mediated perception approaches and behavior reflex approaches. [CSKX15]

In this paper we further analyze a suggested third group, called direct perception, which can traced to [Gib79] in the mid 50's, but was sharply criticized by researchers of the other two groups, i.e. in [Ull80].

All these three groups differ in the way of interpreting the given sensor data and whether or not to create a some what bigger picture based on consequent data.

1.3.1 Mediated Perception

The mediated perception approach is a multi-component continuous process. Every component recognizes specific aspects for driving. For example traffic signs, lanes, other cars. Those components are then combined into one single world state representing the cars surrounding based on the sensor data. [GLSU13]
These world states are 3D models of the current world. Cars are identified using a classifier and then often surrounded by a 3D bounding box. An example can be seen in Figure 1.3. By comparing different frames generated one can estimate the speed and distance to those objects and derive an A.I. based precedence behavior. [GLSU13][CSKX15]

The often stated problems with such approaches are, that computing such a scene is costly in terms of computation time. Some information is irrelevant, redundant or even misleading due to inaccuracy of sensors. To perform a right turn the sensor information of the distance to a car left behind me is irrelevant, but becomes very important when taking a left turn.
Additionally many of the subtasks are still open research topics themselves. For example

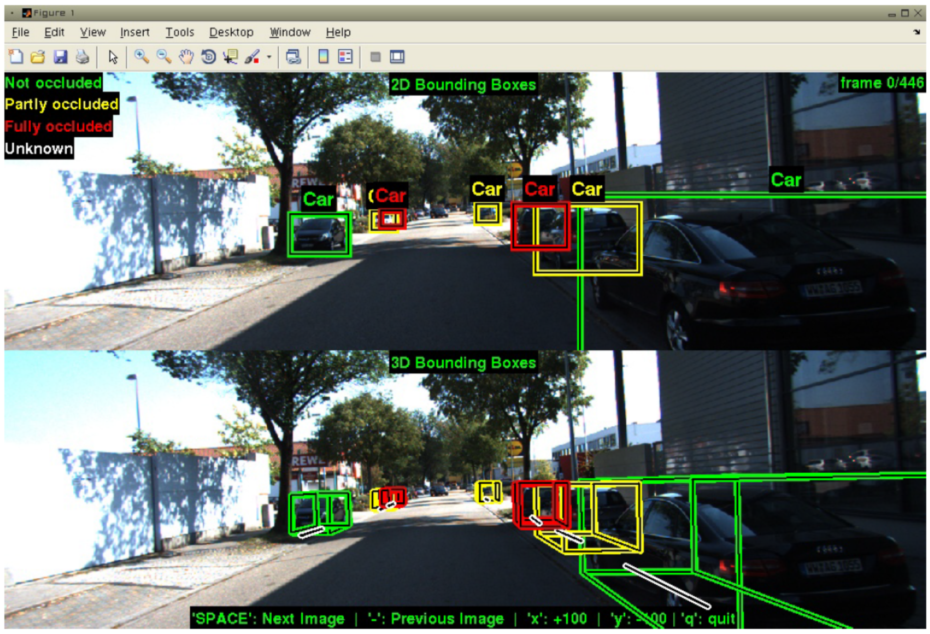


Figure 1
File Edit View Insert Tools Desktop Window Help
Not occluded
Partly occluded
Fully occluded
Unknown
2D Bounding Boxes
frame 0/446
Car
Car
Car
Car
Car
Car
3D Bounding Boxes
'SPACE': Next Image | '-': Previous Image | 'x': +100 | 'y': -100 | 'q': quit

Figure 1.3: An example of a scene using 3D bounding boxes. This image is taken from the MATLAB delvopment kit of [GLSU13]

a reliable lane detection throughout various weather conditions or even a new road not having any drawn lines yet. [Aly08]
Also mediated perception approaches require very detailed information up front, like up-to-date maps.

The approach of mediated perception is a reasonable and very sturdy way of handling such a complex task, but has its drawbacks regarding computational time and additional knowledge.

1.3.2 Behavior Reflex

The behavior reflex approach of constructing a reliable autonomous driving agent can be dated back to 1989 , where researchers tried to directly map a single frame to a decision of a steering angle. For such approaches a quite simple neural network were created.
The network ALVINN, shown in Figure 1.5, consisted of a single hidden layer, used back-propagation and is fed by two cameras: a 30×32 pixel video and a 8×32 pixel range finder retina. The input neurons fired depending on the blue color band of its pixel, because it is believed to infer the highest contrast between road and non-road. The difference in color of road and non-road was fed back to the network. The bias (activation level) is proportional to the proximity of the corresponding area, based on the likelihood and importance of having road in particular fields of the image.[Pom89]
For example having recognized that the road abruptly ends right in front of the car is more important than recognizing that there is a road in the top left corner.

Such systems, even though they are simple compared to the in Section 1.3.1 mentioned mediated perception approaches, have been proven to have the capability to perform simple tasks. It can elegantly be trained by having a human drive a car with the cameras

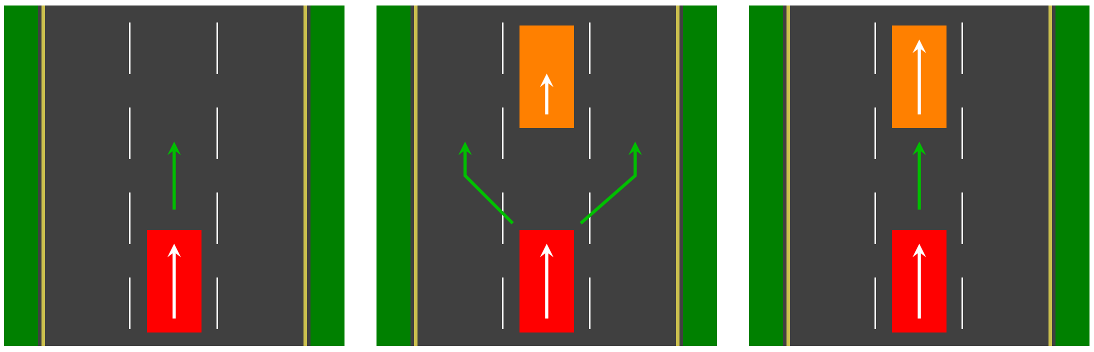

(a) free lane (b) slower car in the same lane (c) equal fast car in same lane

Figure 1.4: The 3 scenarios causing problems with behavior reflex approaches. The red block is the agent, the orange block the other car, the white arrows indicate the velocity and the green arrows the logically deduced behaviors.

equipped and forward the images to the neural network and adding the current steering angles as a label.[CSKX15]

The problem with behavior reflex approaches is, that they reach their limits very early when adding more complex scenarios. Having simple alternations to the trained scenarios, which enforce a different behavior, is very hard to train to such a neural network.
For example comparing a simple straight 3 lane road with the car in the middle, as sketched in Figure 1.7. The system is confidently able to hold the angle and make small adjustments to stay in the lane (Figure 1.4a). But what if on the same road there is an other car in the middle lane in front of the agent, which is slower? Having quite the same input the system would have to overtake the car left or right (considering an american highway) (Figure 1.4b). Now also considering a car in front, which has the same speed. One can simple stay in the lane (Figure 1.4c). This maneuver is very hard to train to a simple neural network like ALVINN.

TODO: noch mehr?

1.3.3 Direct Perception

The direct perception is the third group of approaches, which can be dated back to the 1954 and was initially mainly researched by James J. Gibson. [Gib54] The approach is based on analyzing a picture not simply deducing a steering angle, or velocity change, like the behavior reflex approaches (cf. Section 1.3.2), but also performing further computation without parsing it into a 3D world state model like the mediated perception approaches (cf. Section 1.3.1). [CSKX15]
So it is a third paradigm, which can be interpreted as a hybrid of the two other paradigms. The approach tries to identify only the meaningful affordance indicators and make a decision based on those parameters.

We further consider a design based on [CSKX15] and their way of training.

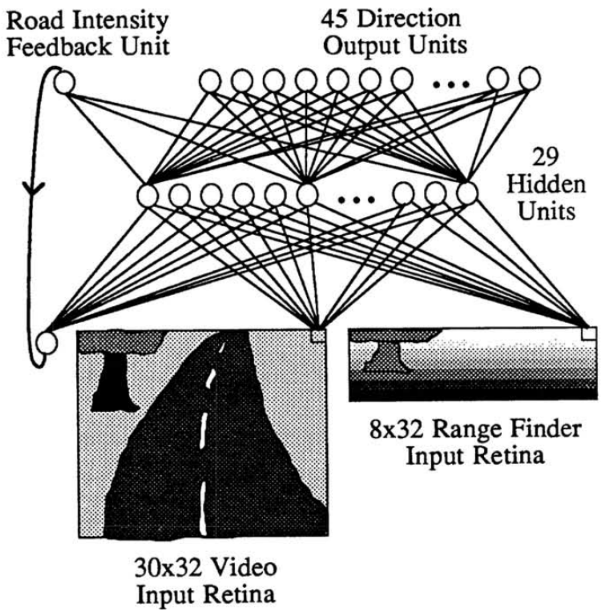


Figure 1.5: The neural network called ALVINN and used as a behavior reflex based autonomous agent. [Pom89]

The original paper [CSKX15] stated the a total number of 13 indicators separated in two states to be sufficient. The states are: in line driving (following the lane) and on line driving (changing lanes). The values themselves can be categorized as: preceding car distances, distances to the lane markers and the steering angle. The indicators and their affiliation to the states can be seen in Figure 1.6.

Based on the current state, all affordance indicators of the other state are not used, since the other state is defined to be inactive.
In order to identify the current state the host car is in, every state has their respective region, where they are active with an overlapping region for smooth transitioning.

TODO:

1. how to learn the affordance
2. how to conclude affordance to action

1.4 Deep Learning Languages

TODO:

1. why need a language
2. what should it in general do?

```
always:
  1) angle: angle between the car's heading and the tangent of the road
"in lane system", when driving in the lane:
  2) toMarking LL: distance to the left lane marking of the left lane
  3) toMarking ML: distance to the left lane marking of the current lane
  4) toMarking MR: distance to the right lane marking of the current lane
  5) toMarking RR: distance to the right lane marking of the right lane
  6) dist LL: distance to the preceding car in the left lane
  7) dist MM: distance to the preceding car in the current lane
  8) dist RR: distance to the preceding car in the right lane
"on marking system", when driving on the lane marking:
  9) toMarking L: distance to the left lane marking
  10) toMarking M: distance to the central lane marking
  11) toMarking R: distance to the right lane marking
  12) dist L: distance to the preceding car in the left lane
  13) dist R: distance to the preceding car in the right lane
```

Figure 1.6: The affordance indicators and their affiliation states

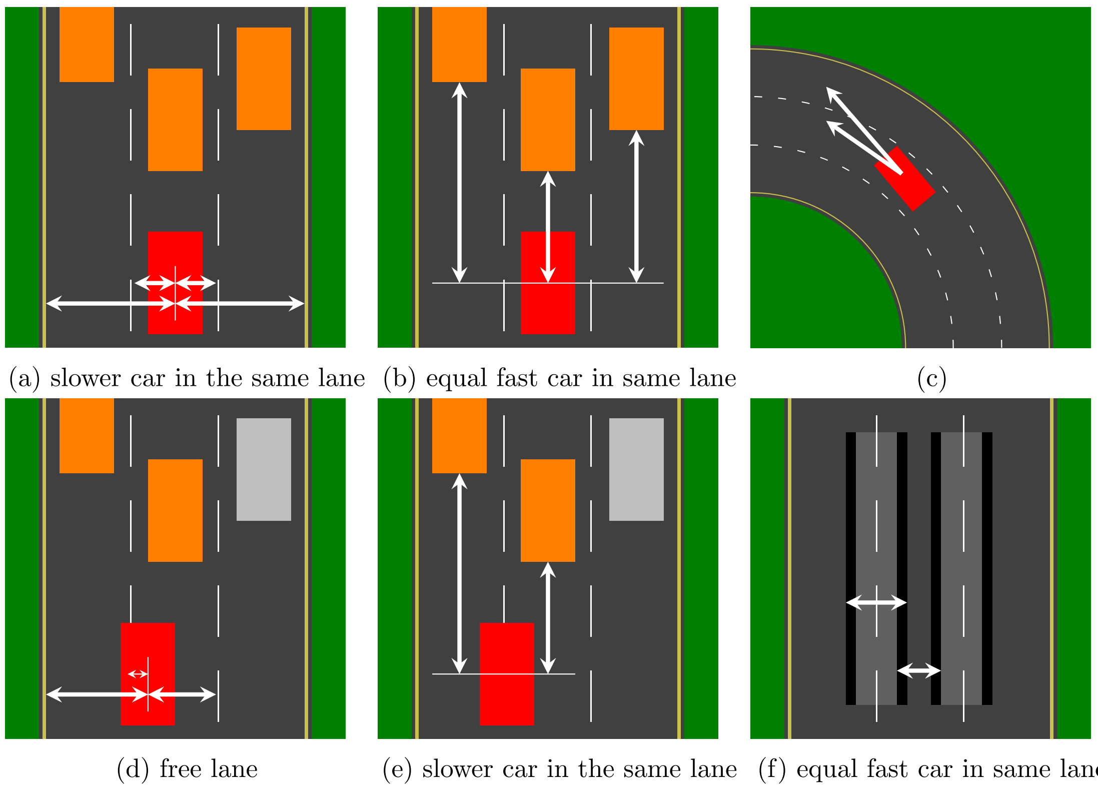

(a) slower car in the same lane (b) equal fast car in same lane (c)
(d) free lane (e) slower car in the same lane (f) equal fast car in same lane

Figure 1.7: The 3 scenarios causing problems with behavior reflex approaches. The red block is the agent, the orange block the other car, the white arrows indicate the velocity and the green arrows the logically deduced behaviors. The names of the indicators is given in Figure 1.6

TODO: name the arrows

3. state that different languages ex.

1.4.1 CNNArchLang

(The whole description is based on[TvWH17])

One language for modeling CNNs is CNNArchLang. This language is developed at the Chair of Software Engineering at the RWTH Aachen University and part of the MontiCar language family. Its basic structure is very similar to python.

In contrast to other languages for deep learning CNNArchLang does not denote the connections of layers directly, but tries to model the data flow through the network. For that specific task it contains two main operators:

->: Serial Connection:
This orders two elements sequentially. This means it denotes the first elements output as the second elements input.

|: Parallelization:
This allows the split of the network into separate data streams, which can be computed in parallel.

Since serial connection has the higher precedence one has to use brackets. Also to merge the splitted streams, created by |, one can use the operators: `Concatenate()`, `Add()` and `Get(index)`.

General Definitions

The general definitions of a CNN, which are the input, i.e. an image, maybe additional data in a specific file type, i.e. sensor data, and the output dimension, denoting the predictions or in our example the actions the car should perform.

Such definitions can be modeled in CNNArchLang as presented in Figure 1.8.

```
def input Z(0:255)^{3, h, w} image[2]
def input Q(-oo:+oo)^{10} additionalData
def output Q(0:1)^{3} predictions
```

Figure 1.8: A general definition of a CNN using CNNArchLang

Further analyzed the definition can be broken down to the following components:

- Keyword: `def`
  Every input and output can be introduced using the keyword `def`
- Direction: `input/output`
  Every definition, being a part of the 1.4.1, has to be defined to be either an `input` or an `output`

- Range of numbers:
  One can define the input to have special constraints. For example only integer values are denoted by a `Z` representing $\mathbb{Z}$. The same for `Q` and $\mathbb{Q}$.
  Also the range has to be given via `(x:y)`, where `x` and `y` either are numbers or `-oo(` or `oo)` to denote ∞.

- Size:
  The size of for example the input or the number of classes is denoted by a matrix like notation using `^{size}`. For the input image (line 1) the size `^{3,h,w}` determines the input image to have 3 channels with an image width of `w` and image height of `h`. The others are just defined as 1×10 or 1×3 vectors/tensors.

- Naming:
  At the end of the line there has to be a name to identify the corresponding input/output.
  Also through the `[2]` behind the name `image` one can define it to be a fixed length array of images.

1.4.2 Array handling

An array, like in the example shown in Figure 1.8, can be used either element wise:

```
image[0]->
```

which would access the first image and create a sequential connection (c.f. Section 1.4.1), whereas

```
image-> ≡ (image[0] | image[1]) ->
```

would result in two separate streams. If the receiving end of the sequential connection, for sake of example let it be called `out`, is also an array of size two one can also use

```
image -> out ≡ ([0]-> out[0] | [1] -> out[1])
```

1.4.3 Predefined Layers and Functions

Creating different CNN often use a similar set of layers. For that purpose there are some layers already defined by CNNArchLang to simplify the usage. There is for example the `FullyConnected`-layer with parameters for the number of units within and whether they should use a bias value (c.f. Section 1.1), the `Convolutional`-2D-layer with parameters for the kernel (rectangle) size, number of filters, the stride defining the distance of two rectangles, padding and the usage of biases. Further information on any of these parameters or other predefined layers can be found in [TvWH17].

Also there are already defined functions like the `Sigmoid`, `Softmax`, `Tanh` or `ReLu`.

TODO:

- structural specialties

1.4.4 Caffe

Caffe is a full deep learning framework, created by Yangqing Jia during hos PhD at UC Berkeley. It is a framework specially build to deal with multimedia input formats using state-of-the-art deep learning techniques. It comes as a BSD-license C++ library offering python and MATLAB bindings. One of its reasons why it's so well known and frequently used is because of its design based on expressiveness, speed and modularity.

1.4.5 MxNet

general information from [CLL+15]

Chapter 2

Running Example

The example net, AlexNet [KSH12] implemented as a Direct Perception approach

2.1 Implementation

Implementation of the net using CNNArch or MxNet (maybe discuss already implemented approaches to cover more details)

2.2 Training

The training of the implemented net based on the KITTI dataset [?]

Chapter 3

Evaluation

Test the trained set in a simulation environment

3.1 MontiSim

Short introduction of the tool used to evaluate the net in [Rea17]

3.2 Results

Evaluating the results of the test of the net in MontiSim

Chapter 4

Conclusion

Conclusion of differences and similarities between the frameworks

Also a general conclusion based on results and [Grz17]

Bibliography

[Aly08] Mohamed Aly. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 7–12. IEEE, 2008.

[CLL$^+$15] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[CSKX15] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2722–2730. IEEE, 2015.

[Gib54] James J Gibson. A theory of pictorial perception. *Audiovisual communication review*, 2(1):3–23, 1954.

[Gib79] James J Gibson. *The ecological approach to visual perception.* Psychology Press, 1979.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[Grz17] Adam Grzywaczewski. Training ai for self-driving vehicles: the challenge of scale. Technical report, Tech. rep., NVIDIA Corporation. URL https://devblogs. nvidia. com/parallelforall/training-self-driving-vehicles-challenge-scale, 2017.

[JSD$^+$14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Pom89] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[Rea17] Bernhard Rumpe and et. al. Montisim. https://github.com/MontiSim, 2017.

[TvWH17] Thomas Timmermann, Michael von Wenckstern, and Malte Heithoff. Cnnarchlang. https://github.com/EmbeddedMontiArc/CNNArchLang, 2017.

[Ull80] Shimon Ullman. Against direct perception. *Behavioral and Brain Sciences*, 3(3):373–381, 1980.

[Wik18] Wikipedia contributors. Convolutional neural networks — Wikipedia, the free encyclopedia, 2018. [Online; accessed 11-May-2018].