Marius Landmann, Anna Dodik

Smart Systems

Convolutional Neuronal networks

Theory Script

Table of content

[List of Figures I](#_Toc45049565)

[List of Tables II](#_Toc45049566)

[List of abbreviations III](#_Toc45049567)

[1. Introduction 1](#_Toc45049568)

[2. CNN Applications 1](#_Toc45049569)

[2.1 Project “CNN truck” 2](#_Toc45049570)

[2.2 Project “DeepPiCar” 2](#_Toc45049571)

[3. Important software and hardware 3](#_Toc45049572)

[3.1 Raspberry Pi 3](#_Toc45049573)

[3.2 Arduino Nano 4](#_Toc45049574)

[3.3 Jupyter Notebook 4](#_Toc45049575)

[3.4 OpenCV 4](#_Toc45049576)

[3.5 TensorFlow 5](#_Toc45049577)

[3.6 Google Colab 5](#_Toc45049578)

[3.7 GPU 5](#_Toc45049579)

[3.8 TPU 6](#_Toc45049580)

[3.9 GitHub 7](#_Toc45049581)

[4. CNN Optimization approaches 8](#_Toc45049582)

[4.1 Faster R-CNN 9](#_Toc45049583)

[4.2 BLOB detection 9](#_Toc45049584)

[4.3 Image cropping 10](#_Toc45049585)

[4.4 Vehicle to infrastructure 10](#_Toc45049586)

[4.5 Colour spaces HSV vs. RGB 10](#_Toc45049587)

[5. Data set creation 12](#_Toc45049588)

[6. CNN Workflow 12](#_Toc45049589)

[6.1 Basic CNN workflow 13](#_Toc45049590)

[6.2 CNN workflow with transfer learning 14](#_Toc45049591)

[6.3 Basic CNN workflow adjusted for the TPU 15](#_Toc45049592)

[6.4 CNN workflow adjusted for TPU with transfer learning 16](#_Toc45049593)

[References 17](#_Toc45049594)

# List of Figures

[Figure 1.1 Overview about topics 1](#_Toc45049595)

[Figure 2.1 Some CNN applications 1](#_Toc45049596)

[Figure 2.2 MAN truck of the practice project 2](#_Toc45049597)

[Figure 2.3 The DeepPiCar project 3](#_Toc45049598)

[Figure 3.1 Overview of hard- and software 3](#_Toc45049599)

[Figure 3.2 Raspberry Pi for the project used 4](#_Toc45049600)

[Figure 3.3 Arduino Nano used for the project 4](#_Toc45049601)

[Figure 3.4 Basic GPU 5](#_Toc45049602)

[Figure 3.5 Matrix of a GPU 6](#_Toc45049603)

[Figure 3.6 TPU used for the truck project 6](#_Toc45049604)

[Figure 3.7 Matrix of the TPU 7](#_Toc45049605)

[Figure 3.8 Button to fork the repository 7](#_Toc45049606)

[Figure 4.1 Overview of the most important optimization approaches 9](#_Toc45049607)

[Figure 4.2 Faster R-CNN model 9](#_Toc45049608)

[Figure 4.3 Example for BLOB detection 10](#_Toc45049609)

[Figure 4.4 Example for image cropping 10](#_Toc45049610)

[Figure 4.5 Vehicle to infrastructure communication 10](#_Toc45049611)

[Figure 4.6 RGB colour cube 11](#_Toc45049612)

[Figure 4.7 HSV colour spectrum 11](#_Toc45049613)

[Figure 4.8 Colour scale 11](#_Toc45049614)

[Figure 5.1 Main process steps for data set creation 12](#_Toc45049615)

[Figure 6.1 Overview of different CNN workflow approaches 13](#_Toc45049616)

[Figure 6.2 Flowchart elements 13](#_Toc45049617)

[Figure 6.3 Basic CNN workflow 13](#_Toc45049618)

[Figure 6.4 CNN workflow with Transfer Learning 14](#_Toc45049619)

[Figure 6.5 Basic CNN workflow adjusted for the TPU 15](#_Toc45049620)

[Figure 6.6 CNN workflow adjusted for TPU with Transfer learning 16](#_Toc45049621)

# List of Tables

[Table 3.1 Important Git commands 9](#_Toc44747964)

[Table 5.1 Process steps data set creation 13](#_Toc44747965)

# List of abbreviations

Faster R-CNN Faster Region-based CNN

Google Colab Google Colaboratory

GPU Graphical Processing Unit

HSV Hue Saturation Value

RGB Red Green Blue

RPN Region Proposal Network

TLR Traffic Light Recognition

TPU Tensor Processing Unit

# Introduction

This course focuses on the creation of CNNs and the usage of them in traffic light recognition. The following chapters will give you an overview of CNN applications, mention hard- and software requirements, and show you how to start setting up your own CNN. Figure 1.1 shows all topics that will be discussed during the theory part of the lecture.



Figure 1.1 Overview about topics

# CNN Applications

**Learning objectives of this chapter:**

* You know which general applications exist for CNNs
* You can describe the rough project conditions of the truck project
* You know what the DeepPiCar project is

**Task:** Please name different CNN applications that are known to you.

Before we start to look more detailed into the projects, we want to clarify where CNNs get used in general. Some CNN applications are, for example, decoding facial recognition, analysing documents, or personalized advertising. Figure 2.1 illustrates an overview of some CNN applications. Further applications where CNNs are used are robot applications and especially applications in cars. Thereby, even partially automated driving with cars is possible today. Cars can drive partially automated with the supervision of the driver. The goal of the research and development in the automotive industry is to create cars that can drive autonomously. While driving autonomously, the car drives without human support. The car uses different parameters to calculate how to drive and behave in certain situations. The parameters are detected by sensors and the behaviour adapts by learning.

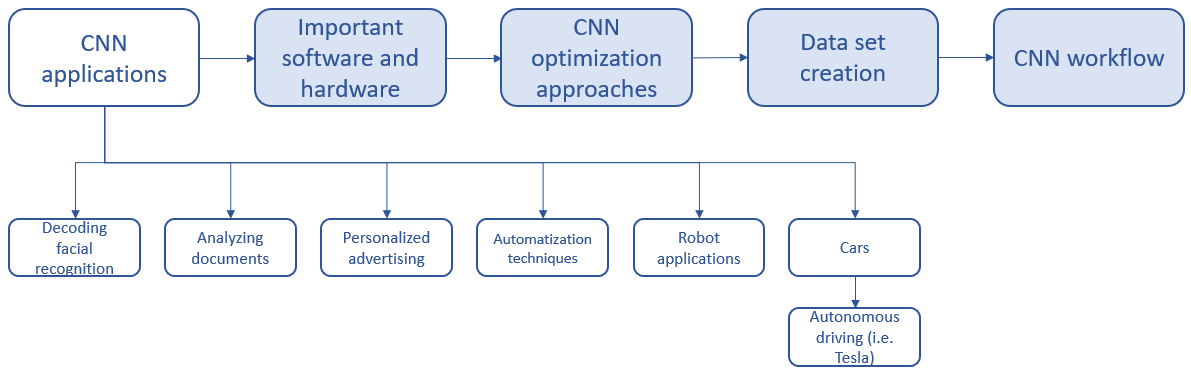


Figure 2.1 Some CNN applications

A big step in autonomous driving has recently been made. In April 2020, Tesla published a software update called “Traffic Light and Stop Sign Control” for their cars in the United States. By using this software, the car is now able to detect traffic lights and to slow down or to stop automatically. The software detects stop signs and all traffic light phases including switched off traffic lights. The driver can stop the slowdown process by briefly pressing the accelerator pedal. He still needs to spend attention to the road and needs to be prepared to act. The basis of the software is a convolutional neural network which allows the system to be trained and therefore to learn and improve its accuracy. But it still has some flaws and is not always able to detect the right traffic light phases. Due to certain regulations its implementation is not yet allowed in Europe.

## Project “CNN truck”

During the lab sessions, you will implement a CNN into a vehicle (Figure 2.2) that is supposed to drive autonomously. The truck should detect traffic light phases through the CNN and should react according to the status of the traffic light, for example stopping at a red traffic light and starting/continuing at a green one.



Figure 2.2 MAN truck of the practice project

*[1]*

The truck has a power unit which supplies the motor of the wheels with energy. A second power unit is connected to an Arduino Nano, a Raspberry Pi, and a webcam. The developed CNN will be implemented in the Raspberry Pi. Furthermore, the neuronal network must be able to detect images efficiently, must have a high classification accuracy rate, and a high detection speed.

## 2.2 Project “DeepPiCar”

The “DeepPiCar” is a project that aims at programming a self-driving robot car. The project set up is shown in figure 2.3. As hardware the PiCar kit from SunFounder is used. Furthermore, a TPU (Tensor Processing Unit) and a Raspberry Pi are used as processing units. Software tools include Python, OpenCV, and TensorFlow. The DeepPiCar performs lane detection and traffic sign recognition. For this purpose, CNNs, object Detection, and transfer learning are used. *[2]*

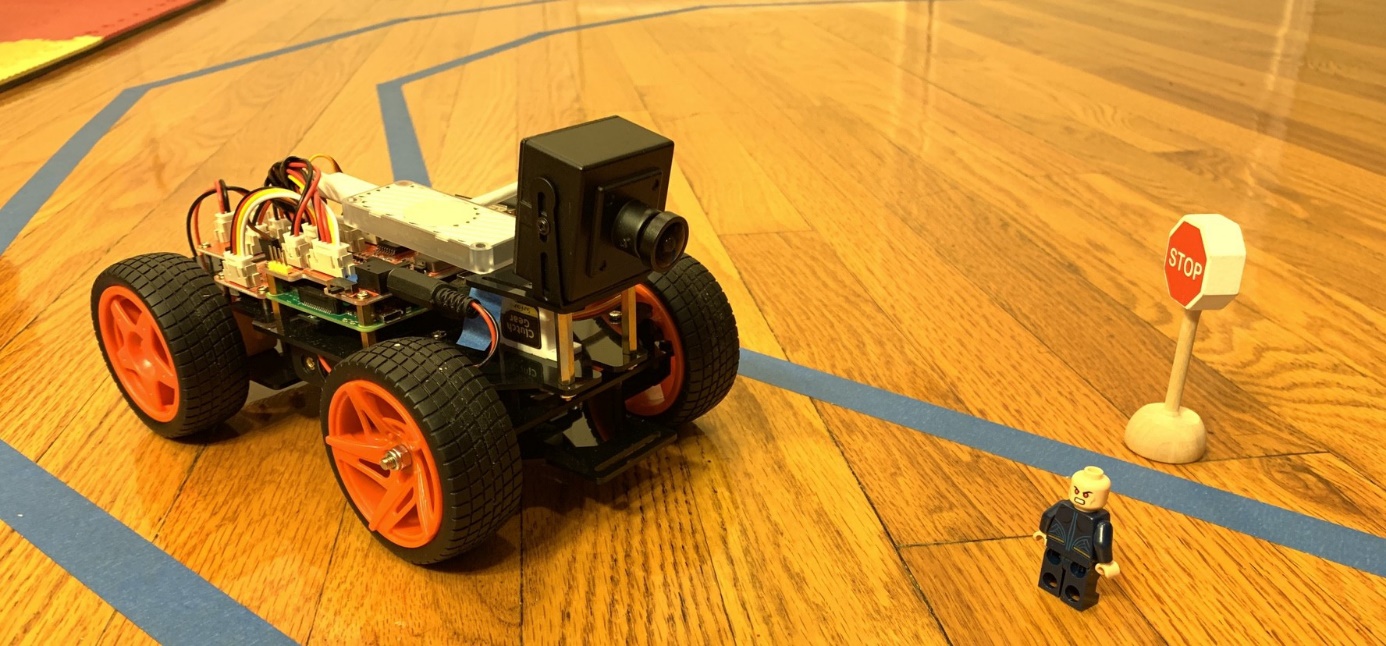


Figure 2.3 The DeepPiCar project

*[2]*

The truck project with traffic light recognition is similar to the DeepPiCar and CNNs will be used for recognition tasks.

# Important software and hardware

**Learning objectives of this chapter:**

* You know the most important soft- and hardware for CNN projects
* **Hardware:**
  + You know what a GPU is and can describe how it works
  + You can explain what a TPU is, and how it works
  + You know why in CNNs a TPU is more efficient than a GPU
* **Software:**
  + You can explain what GitHub is and where it is used
  + You know the meanings of important GitHub commands
  + You can explain what Google Colab is

**Task:** Please list different soft- and hardware that is important for the CNN truck project.

The following chapters provide an overview of the most important hard- and software (figure 3.1) and briefly explain how they work.

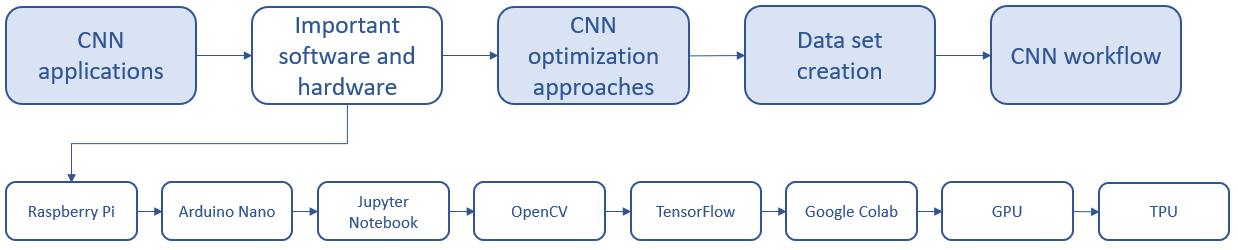


Figure 3.1 Overview of hard- and software

## Raspberry Pi

The Raspberry Pi, illustrated in figure 3.2, is an affordable computer as small as a credit card. It usually has two USB ports, an HDMI port, and can connect to a monitor, keyboard, and a mouse. For the project we added a third USB port that connects the Pi with the webcam. After the first setup, the Pi will always be connected via Wi-Fi. How to set up a Wi-Fi connection is explained in the truck manual documentation. *[3]*



Figure 3.2 Raspberry Pi for the project used

*[3]*

## 3.2 Arduino Nano

The Arduino Nano board, shown in figure 3.3, is a smaller version of the normal Arduino board. It consists of a microcontroller, different input and output pins, and some LEDs for testing. The program data is transferred to the Arduino as binary code. Therefore, the program written in Python will be converted into a binary code before the transfer. Just one program can be saved at the Arduino at any given point in time. If a new program is transferred to the Arduino, the old code will be overwritten. The program stored at the Arduino cannot be transferred back to the programming area because it would be too elaborate to decode everything back again to the Python (program) language.  *[4], [5]*

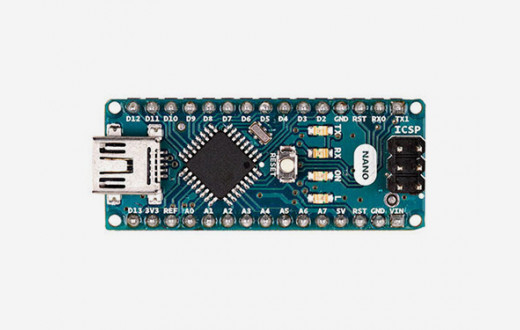


Figure 3.3 Arduino Nano used for the project

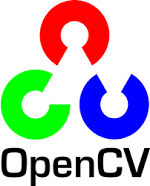
*[4]*

## 3.3 Jupyter Notebook

 *[6]*

Jupyter Notebook is an open source application that enables you to work online on your project. It is possible to code, create visualizations, do machine learning, and share created content. *[7]*

## 3.4 OpenCV

 *[8]*

OpenCV (Open Computer Vision) is an open source software library programmed for Python, C, C++, and Java. The software is often used for image and video recognition and even for deep learning.

## 3.5 TensorFlow

 *[9]*

TensorFlow is an open source machine learning library and is programmed for many different programming languages. The library is often used for image and video recognition as well as for deep learning.

## 3.6 Google Colab

 *[10]*

To use Google Colab (Google Collaboratory) it is necessary to sign up with a Google account. You immediately can start to work and have 2 GB free cloud space available. Colab is an online, cloud-based platform with which it is possible to write, run, and share codes. Furthermore, Colab is also able to execute machine learning, for example together with TensorFlow. In addition, you can comment, evaluate, and visualize the written code. The main advantage of Colab is that the GPU and TPU can be used for free. This significantly affects the runtime and allows faster results. *[11]*

## 3.7 GPU

A GPU (Graphical Processing Unit) is a processing unit that is specialized on processing graphics. It can handle tens of thousands of operations per cycle. The GPU is mainly used for graphics rendering, machine learning, model training, but also for general programming purposes. A basic GPU is shown in figure 3.4. *[12]*



Figure 3.4 Basic GPU

*[13]*

**How does a GPU work?**

A GPU has many tiny multipliers in a processor. This processor can do parallel tasks (also called matrix operations) very efficiently. The intermediate result is read after every single calculation. Therefore, GPU’s have a wide range of possible applications. Figure 3.5 illustrates a TPU matrix.

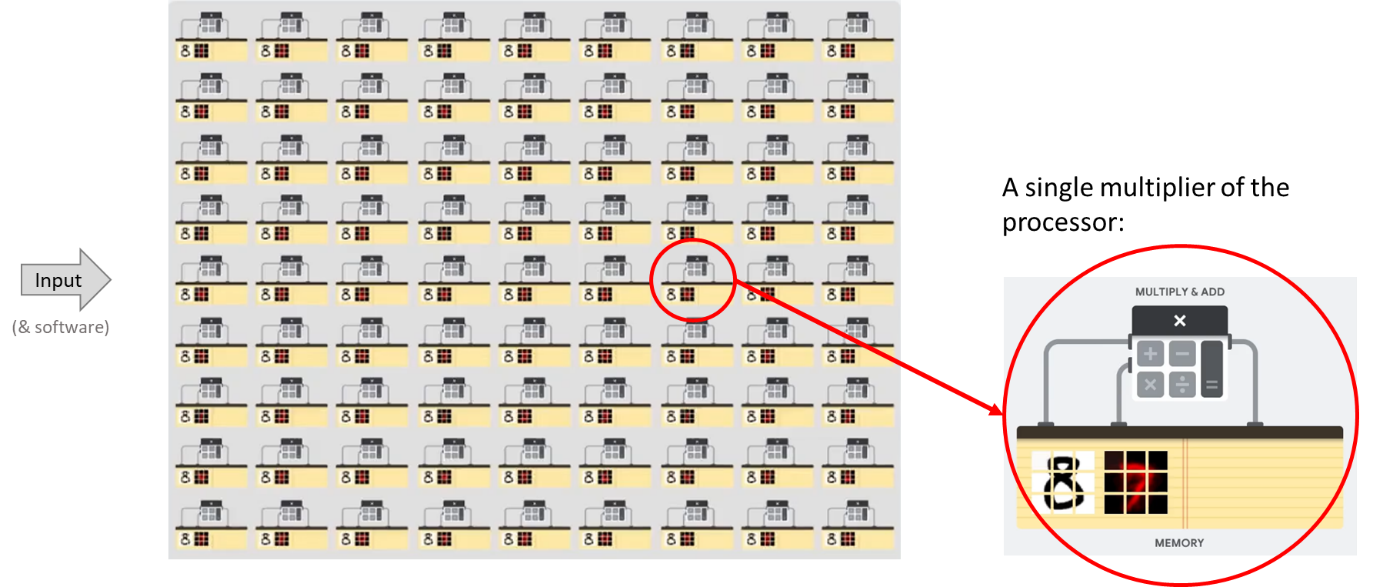


Figure 3.5 Matrix of a GPU

*[14]*

## 3.8 TPU

A TPU (Tensor Processing Unit) is a special processing unit developed by Google for optimized machine learning and fast processing. It can handle up to 256.000 operations per cycle and is mainly used for deep learning. The TPU is especially designed to use TensorFlow as a programming framework and can only be used with TensorFlow. Furthermore, it requires significant effort to do general programming on a TPU because it has a complex programming structure. But nevertheless, a TPU has many advantages. The TPU is around 30 times faster than a GPU and the newest generation of the TPU is 80 times more energy efficient as a GPU. The TPU used in the project is shown in figure 3.6.



Figure 3.6 TPU used for the truck project

*[15]*

But why is the TPU so much better for machine learning than the GPU? The reason is that the TPU is optimized for these tasks and the parameters are set to do them as efficient as possible. The TPU is especially designed to process matrix operations. Therefore, all operators are connected in a large matrix. Exercises of this type can be executed very fast. Figure 3.7 shows how a TPU matrix looks in theory.

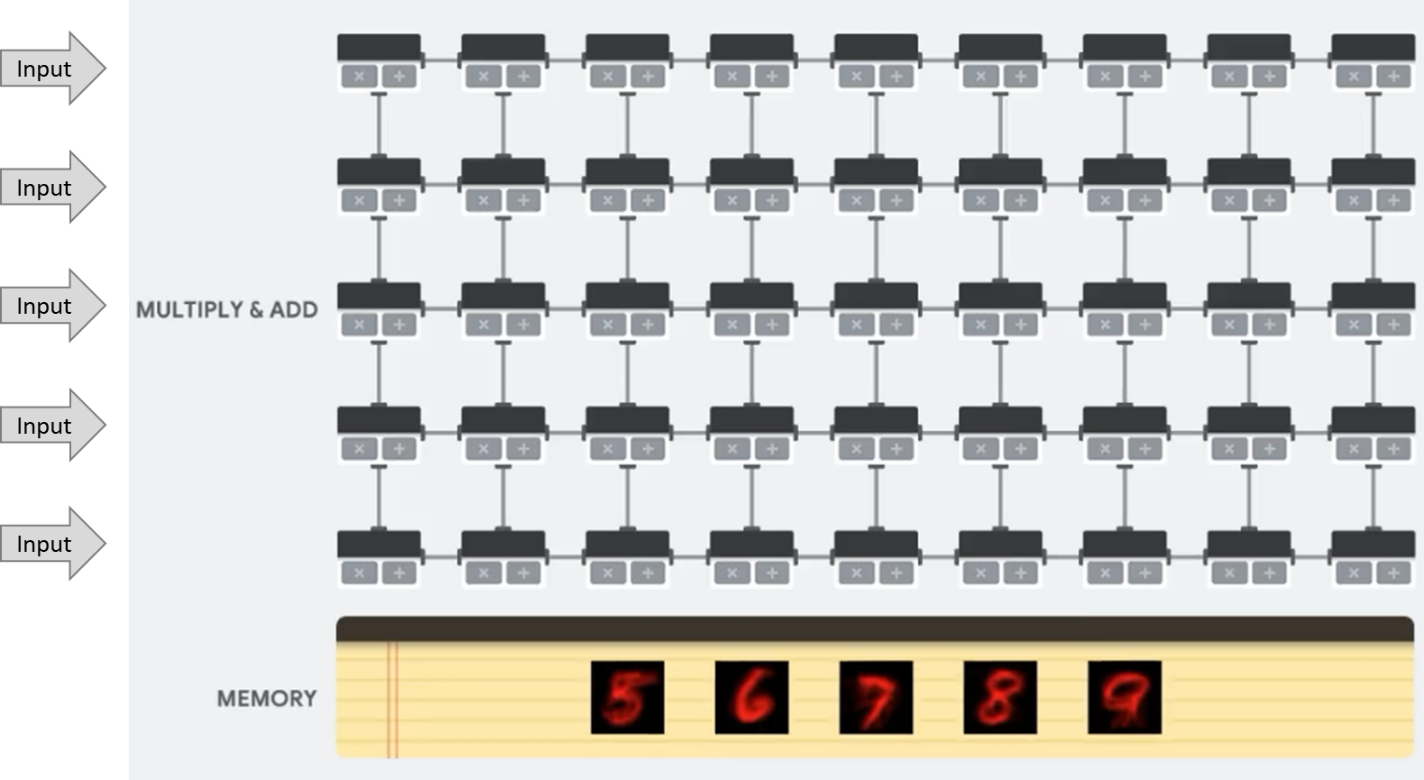


Figure 3.7 Matrix of the TPU

*[14]*

The CNN truck project needs a processor that supports the Raspberry Pi because its processing power is way too small and it would not be possible to run a CNN with it. Because of the reasons mentioned above, a TPU will support the Raspberry Pi in the truck project. *[14], [16]*

## 3.9 GitHub

 *[17]*

GitHub is a platform where people can work together on the same project. Each project is saved in a public dedicated or a private repository. The repository can also be connected with a local file on your computer. In order to use GitHub, you need to create a free account. Furthermore, Git shall be installed on your device to enable working locally. *[18], [19]*

**Git fork order:**

The **git fork** order is no command. The order duplicates the original repository, while the repository remains on your GitHub account. In this way you can experiment with changes without affecting the original project. To fork your repository, you just push the fork-button, shown in figure 3.8.



Figure 3.8 Button to fork the repository

Changes of the forked repository can be merged with the original repository by a pull request. The pull request also asks the owner for permission before the changes are saved. Table 3.1 shows important Git commands that will be used during the CNN truck project.

|  |  |  |
| --- | --- | --- |
| Command | Description | Command Example |
| git clone | Copies and downloads an existing Git repository to your local computer. Changes made can locally be pushed directly to the repository. | git clone url |
| git config | Sets the authors name and email address. | git config –-global user.name "name" git config –-global user.email "email address" |
| git init | Used to start a new repository. | git init repository name |
| git pull | Pulls changes from the remote server to the working directory. | git pull repository link |
| git add | Adds files to the staging area.  git add: single file git add \*: one or more files | git add file git add \* |
| git commit | Records or snapshots the file permanently in the version history. The message informs about the purpose of the file. | git commit -m "message" |
| git status | Lists all files that have been committed. It shows if changes were successful. | git status |
| git log | Lists the version history of the current branch. It is possible to see if the changes were successful. | git log |
| git push | Sends the committed changes of the master branch to the remote repository.  Attention: only push files with max. 100MB to GitHub. | git push variable name master |

Table 3.1 Important Git commands

# CNN Optimization approaches

**Learning objectives of this chapter:**

* You know general approaches for improving CNNs
* You know and can explain optimization methods
* You can explain the difference between HSV and RGB colour space and why the HSV colour space is preferred in the truck project

**Task:** Name possible solutions to optimize a CNN (besides transfer learning, deep learning).

There is no solution for the traffic light detection in autonomously driving cars yet. Many different approaches for detecting traffic lights are more reliable. Figure 4.1 gives an overview of the most important ones that will also be discussed in the next sections. The CNN of the truck project uses the HSV colour space for better detection reliability. What the HSV colour space exactly is and why it is more efficient to use this colour space, will be described in chapter 4.5.

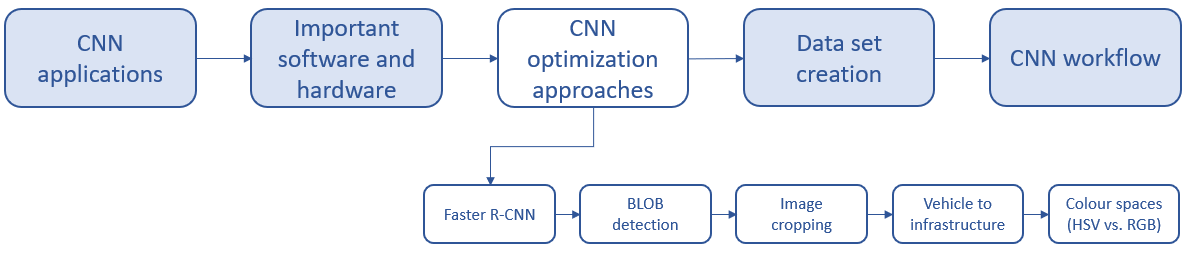


Figure 4.1 Overview of the most important optimization approaches

## 4.1 Faster R-CNN

To increase the detection speed of a CNN sometimes a Faster R-CNN (faster region-based CNN) is used. Faster R-CNNs use Region Proposal Networks (RPN), for example neuronal networks which split the input image into around 2000 different parts with selective search. How a faster R-CNN with RPN works is shown in figure 4.2. Selective search computes regions that have the highest probability of containing an object. *[20]*

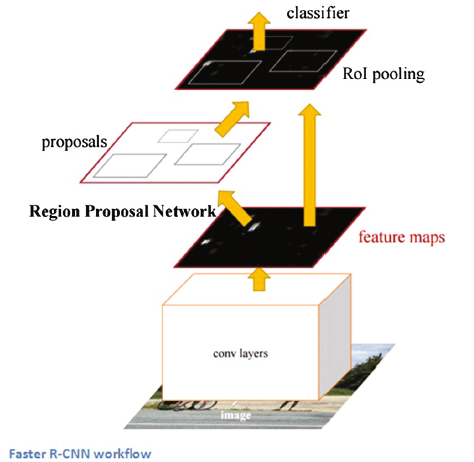


Figure 4.2 Faster R-CNN model

*[21]*

## 4.2 BLOB detection

BLOB detection uses multiple regions to check if there is an interesting circular shape (e.g. a traffic light in the truck project). If pixels with similar brightness are detected, a Blob is generated. Sometimes it also verifies a special colour for TLR (Traffic Light Recognition). Figure 4.3 shows the BLOB detection for a field of flowers as an example. *[22]*

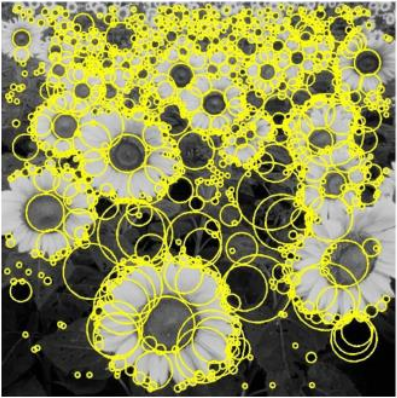


Figure 4.3 Example for BLOB detection

*[23]*

## 4.3 Image cropping

Image cropping means to remove a part of the picture that is not needed. For example, the lower part of an image is cut because the lights appear only in the upper part of the image. Figure 4.4 shows the application to a traffic light situation. The target is to shrink the picture in order to enable faster processing. *[22]*

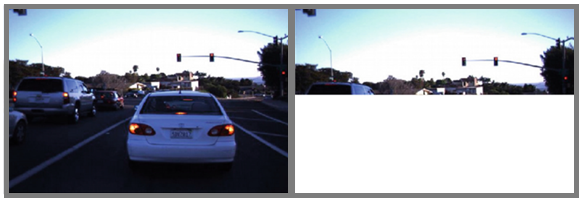


Figure 4.4 Example for image cropping

*[22]*

## 4.4 Vehicle to infrastructure

Vehicle to infrastructure means that the car is able to communicate with the traffic light and also with other vehicles to ask for their status. Figure 4.5 illustrates this idea. The traffic light communicates its state to the car and the car reacts according to the transmitted information.

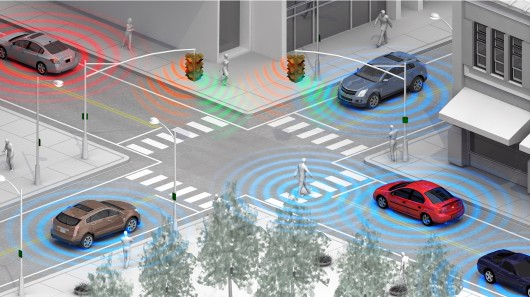


Figure 4.5 Vehicle to infrastructure communication

*[24]*

## 4.5 Colour spaces HSV vs. RGB

A colour space refers to a mathematical model that describes the range of colours with numbers. Usually, each colour has three dimensions and three values. The dimensions represent the three colour channels red (R), green (G), and blue (B). The RGB colour spectrum is shown as a colour cube in figure 4.6. The RGB channels approximately follow the colour receptors of the human eye. Each of the colours can take a value between 0 and 255. If all three colours have a maximum value of 255 and are added, the result will be white colour. Therefore, the RGB colour space is an additive colour space.

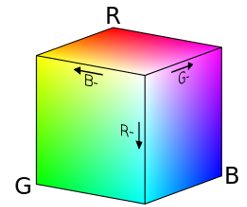


Figure 4.6 RGB colour cube

*[25]*

Other colour spaces do exist, but here we will mention the HSV colour space only. HSV is often used for traffic light detection and stands for hue (H), saturation (S), and value (V). These three variables form the HSV colour spectrum are as shown in figure 4.7. The value of hue ranges from 0° to 360°, see figure 4.8. The values for saturation and value are percentages. For example, if the value is 0% the colour is just black and if the value would be 100% the colour is very intense. The HSV colour space makes our traffic light system more invariant to illumination. This is especially helpful if the sun shines from behind the vehicle, for example in the evening. Furthermore, it reduces the number of feature spaces into two dimensions. *[26]*

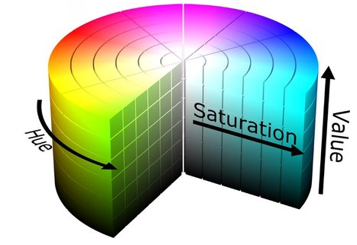


Figure 4.7 HSV colour spectrum

*[27]*

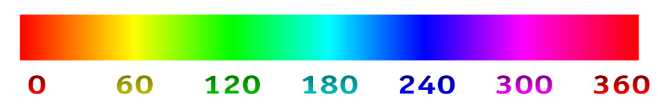


Figure 4.8 Colour scale

*[28]*

# Data set creation

**Learning objectives of this chapter:**

* You know the main steps to create a data set

**Task:** What steps are necessary for data set creation?

Data set creation includes different steps. They are shown in figure 5.1.

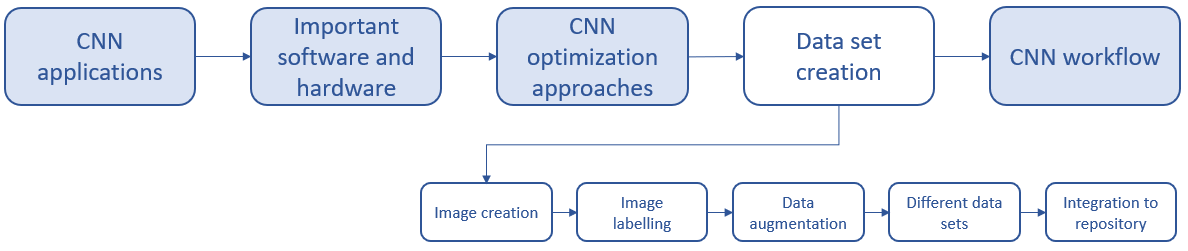


Figure 5.1 Main process steps for data set creation

**Image creation** means to take images of the things that should be identified by the CNN. If possible, the ones from the project should be used. During the **labelling process** all things that should be identified will be marked and lettered. **Data augmentation** means that existing images get changed just a little bit. For example, the picture section is changed, or the picture is mirrored. The goal of the data augmentation is to enlarge the dataset. The **creation of training and test data set** means to split all data into two groups. One group will be used to train the CNN and the other group will be used to validate the accuracy of the CNN. **The integration of the dataset into a repository** means to save the datasets in one central location. Later, the CNN can access this location. How these steps are carried out will be explained in detail in the practical part of the lecture.

# CNN Workflow

**Learning objectives of this chapter:**

* You know a basic CNN workflow
* You know how Transfer learning influences the workflow
* You know how a TPU influences the workflow

Before you start to program your own CNN, it is necessary to know the steps needed to work with a CNN. It helps to create a flowchart that contains all steps. Have it graphically in front of you to help yourself understand the process. The flowchart should mention all process steps.

**Task:** Please set up your own flowchart for a simple CNN. The flowchart should include the input and process steps which are necessary for creating a functional CNN. Please also consider the process of data set creation.

**Possible CNN workflow approaches:**

There are different approaches for the CNN workflow of the truck project. The CNN should correctly detect the colour of a traffic light and send this information to the system. Four different approaches were designed and figure 6.1 summarizes them.

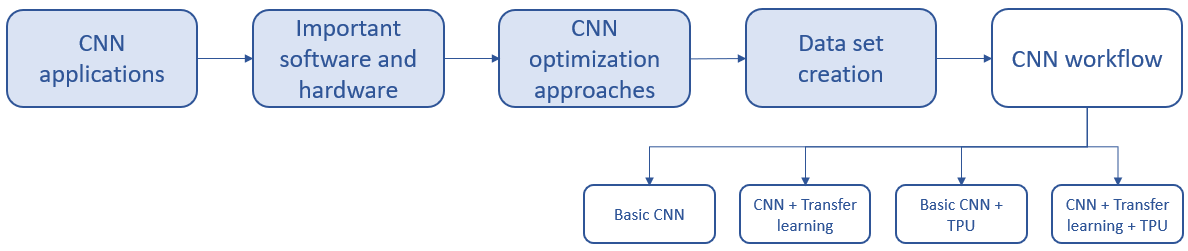


Figure 6.1 Overview of different CNN workflow approaches

All four approaches will be explained in the next sections and their workflows are presented in flow charts. Figure 6.2 explains the elements used in the flow chart.

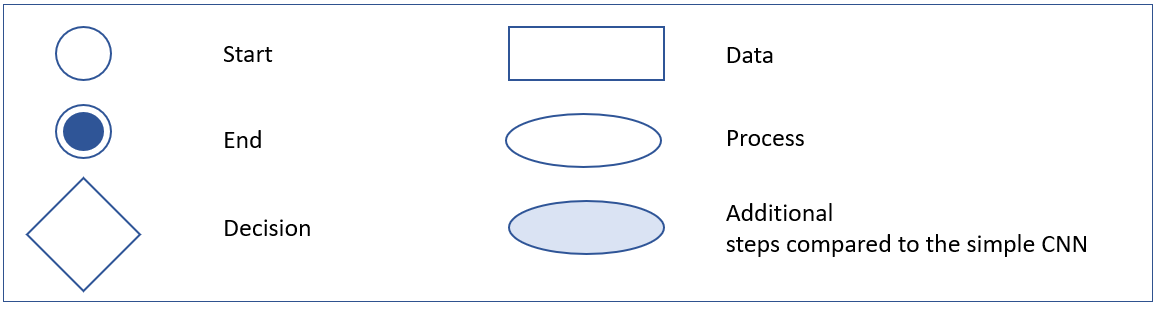


Figure 6.2 Flowchart elements

## 6.1 Basic CNN workflow

The first option is to create a CNN from scratch. This is the easiest variant and the existing knowledge can be easily adapted. The only disadvantage of setting up a CNN from scratch is that the accuracy of the network will be very low, especially in the beginning. The basic CNN workflow is shown in figure 6.3.

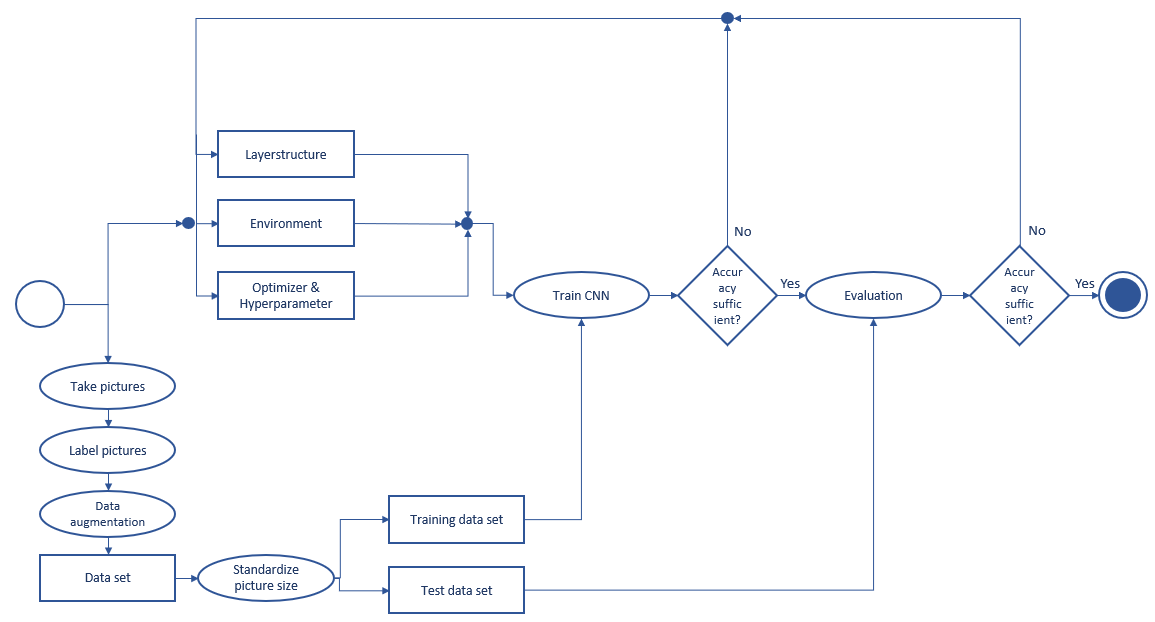


Figure 6.3 Basic CNN workflow

## 6.2 CNN workflow with transfer learning

The second option is to create a new CNN but to combine it with Transfer Learning. The CNN increases its accuracy by transfer learning, and it would probably be faster. The disadvantage of this method is that an already trained system which fits well to the conditions of the project has to be found. Since no pretrained network can be used this is not an option for you. Maybe this method can be used later to improve an already existing network. The general CNN layout with transfer learning is shown in figure 6.4. Note the boxes with a blue background that are new compared to the basic CNN workflow.

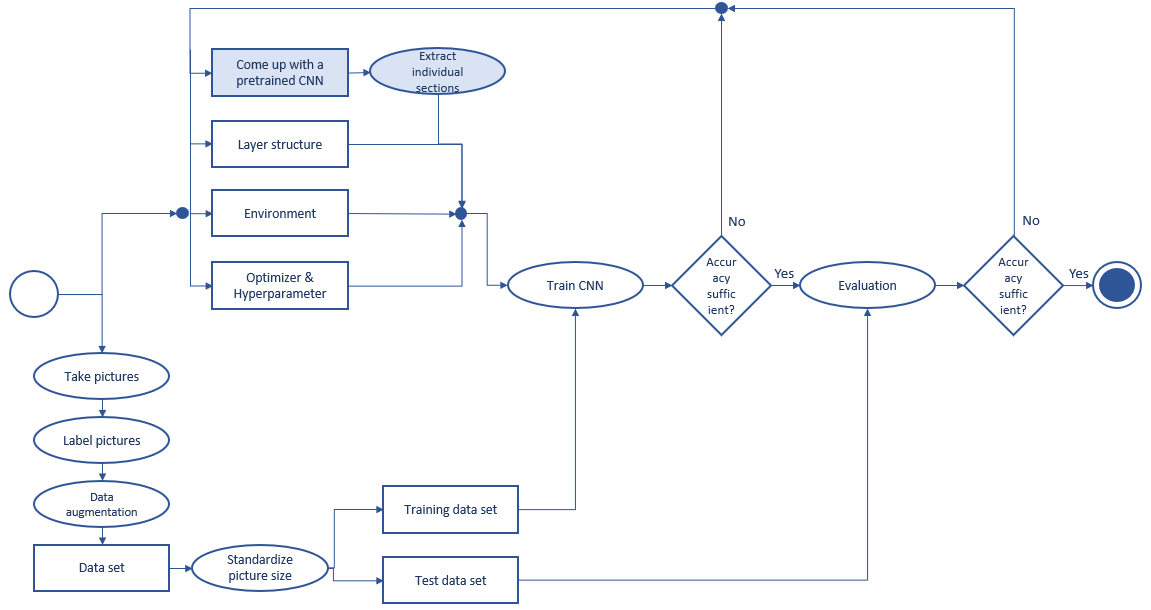
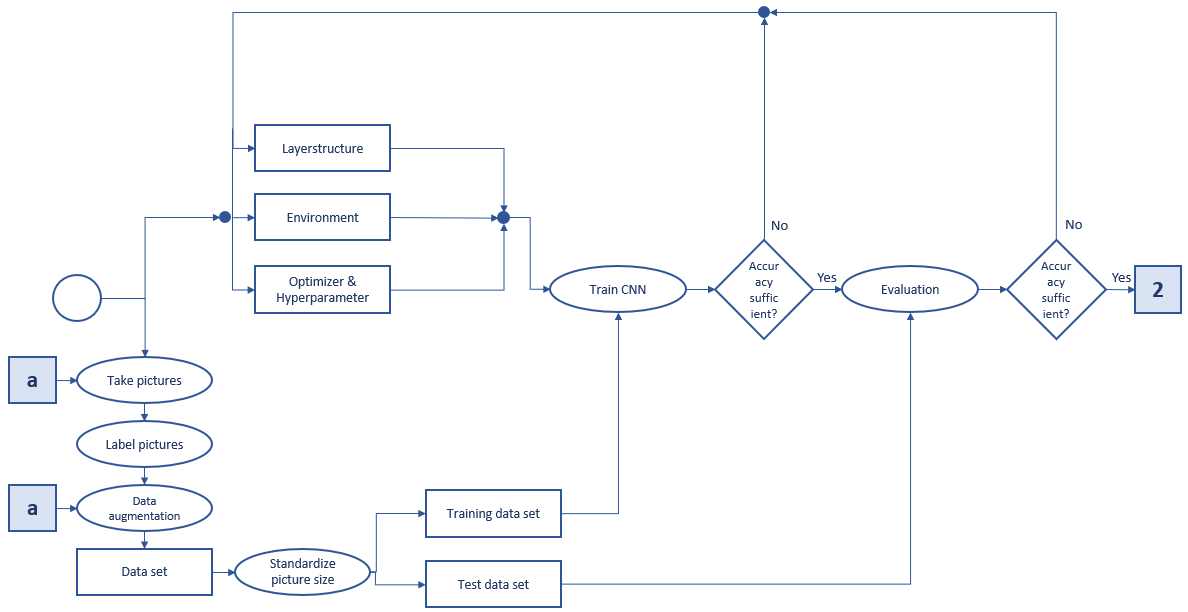


Figure 6.4 CNN workflow with Transfer Learning

## 6.3 Basic CNN workflow adjusted for the TPU

The two methods mentioned above do have a significant disadvantage. It is not possible to use the basic CNN in the truck project because the processing power of the TPU is needed to run the system. Therefore, the third option is to set up a CNN that is compatible with the TPU. To set up such a system is challenging because the TPU is relatively new on the market. Therefore, there is a lack of documentation and information in the web and only few projects relay on the TPU. Furthermore, the CNN with the TPU has a different network structure. The accuracy of the CNN will be very low even if the TPU is used. Figure 6.5 shows a basic CNN workflow with a TPU. The boxes with a blue background are new again compared to the basic CNN workflow.



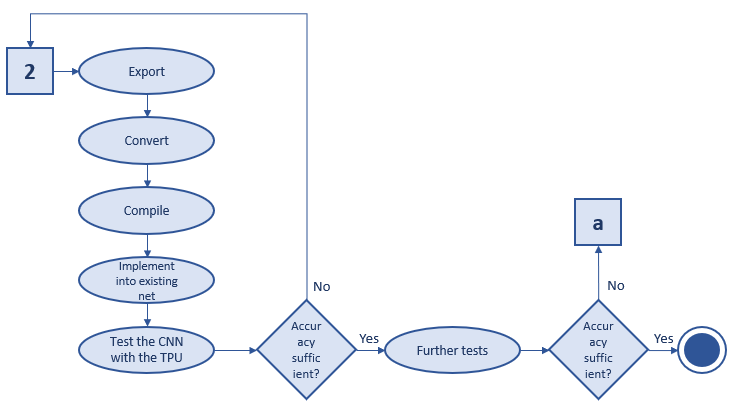
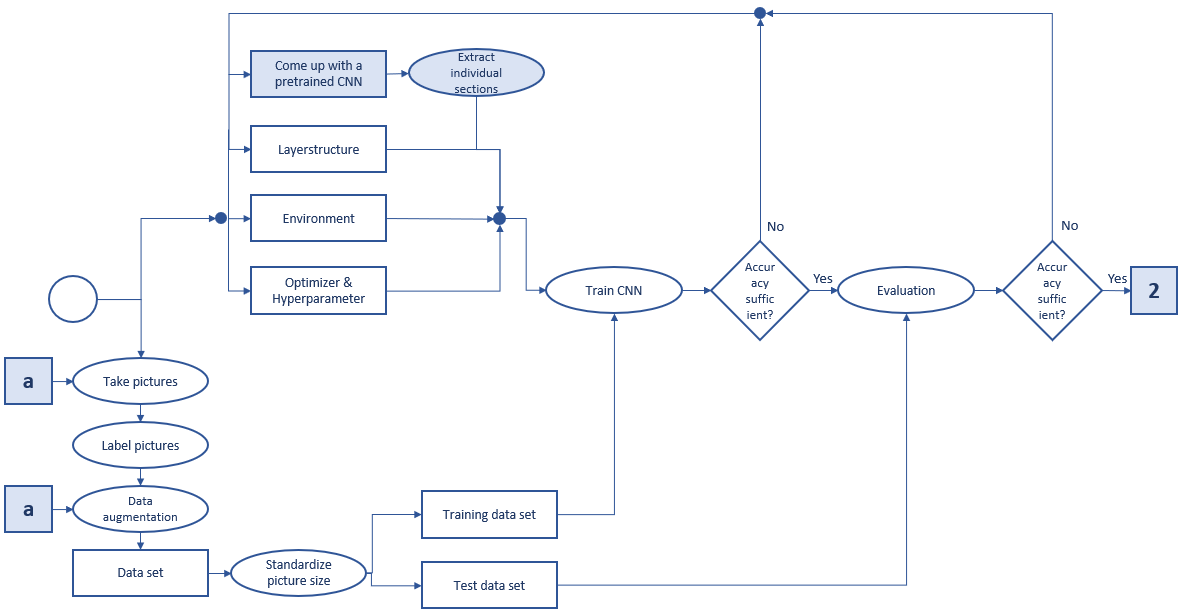


Figure 6.5 Basic CNN workflow adjusted for the TPU

## 6.4 CNN workflow adjusted for TPU with transfer learning

The fourth option is to combine the CNN with both the TPU and transfer learning. Since we do not have a pretrained network, this is currently not an option. The CNN workflow with a TPU and transfer learning is shown in figure 6.6. The boxes with a blue background are new again.



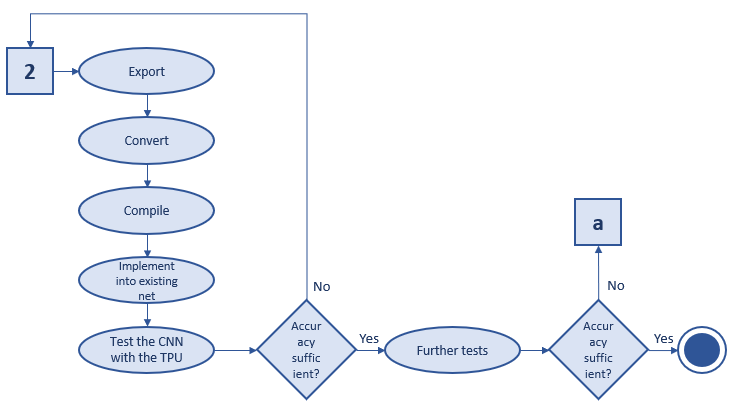


Figure 6.6 CNN workflow adjusted for TPU with Transfer learning

References

[1] MAN Truck, *Picture of MAN Truck.* [Online]. Available: https://​www.lindinger.at​/​at/​fahrzeuge-​und-​boote/​modelle/​lkw-​und-​baufahrzeuge/​tamiya-​man-​tgx-​18.540-​blau-​lackiert-​1-​14

[2] D. Tian, *DeepPiCar Series.* [Online]. Available: https://​towardsdatascience.com​/​tagged/​deep-​pi-​car

[3] Raspberry Pi, *What is a Raspberry Pi?* [Online]. Available: https://​www.raspberrypi.org​/​help/​what-​%20is-​a-​raspberry-​pi/​

[4] A. Aqeel, *Introduction to Arduino Nano.* [Online]. Available: https://​www.theengineeringprojects.com​/​2018/​06/​introduction-​to-​arduino-​nano.html

[5] P. Arduino, *What is Arduino.* [Online]. Available: https://​www.arduino.cc​/​en/​Guide/​Introduction

[6] Jupyter, *Jupyter Notebook (logo).* [Online]. Available: https://​jupyter.org​/​

[7] o. Jupyter, *Jupyter Notebook.* [Online]. Available: https://​jupyter.org​/​

[8] OpenCV, *Open CV (logo).* [Online]. Available: https://​opencv.org​/​

[9] TensorFlow, *TensorFlow (logo).* [Online]. Available: https://​www.tensorflow.org​/​

[10] Google Colaboratory, *Google Colaboratoy (logo).* [Online]. Available: https://​colab.research.google.com​/​notebooks/​intro.ipynb

[11] R. 2. Colab, *Google Colaboratory.* [Online]. Available: https://​colab.research.google.com​/​notebooks/​intro.ipynb

[12] o. Opengenus, *Central Processing Unit (CPU) vs Graphics Processing Unit (GPU) vs Tensor Processing Unit (TPU).* [Online]. Available: https://​iq.opengenus.org​/​cpu-​vs-​gpu-​vs-​tpu/​

[13] H. Computer, *Picture GPU.* [Online]. Available: https://​www.computerhope.com​/​jargon/​g/​gpu.htm

[14] C. Google, *Tensor Processing Unit.* [Online]. Available: https://​storage.googleapis.com​/​nexttpu/​index.html

[15] R. Colab, *Google Colaboratory.* [Online]. Available: https://​colab.research.google.com​/​notebooks/​intro.ipynb

[16] C. Google, *What makes TPUs fine-tuned for deep learning?* [Online]. Available: https://​storage.googleapis.com​/​nexttpu/​index.html

[17] GitHub, *GitHub (logo).* [Online]. Available: https://​github.com​/​

[18] S. Kappagantula, *Top 20 Git Commands With Examples.* [Online]. Available: https://​dzone.com​/​articles/​top-​20-​git-​commands-​with-​examples

[19] B. Kaniobi, *Overview.* [Online]. Available: https://​gist.github.com​/​Ben-​Kaniobi/​c528376a76765e99058c​#n%C3%BCtzliche-git-befehle

[20] E. Peng, F. Chen, and X. Song, “Traffic Sign Detection with Convolutional Neural Networks,” in vol. 710, *Cognitive Systems and Signal Processing*, F. Sun, H. Liu, and D. Hu, Eds., Singapore: Springer Singapore, 2017, pp. 214–224.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Jun. 2015. [Online]. Available: http://​arxiv.org​/​pdf/​1506.01497v3

[22] G. Symeonidis, P. P. Groumpos, and E. Dermatas, “Traffic Light Detection and Recognition Using Image Processing and Convolution Neural Networks,” in vol. 1084, *Creativity in Intelligent Technologies and Data Science*, A. G. Kravets, P. P. Groumpos, M. Shcherbakov, and M. Kultsova, Eds., Cham: Springer International Publishing, 2019, pp. 181–190.

[23] M. Kekoa, *Blob Detection.* [Online]. Available: https://​www.slideserve.com​/​maik/​blob-​detection-​powerpoint-​ppt-​presentation

[24] Transportationops, *Vehicle to infrastructure.* [Online]. Available: https://​www.transportationops.org​/​V2I/​V2I-​overview

[25] Wikipedia, *RGB colour cube.* [Online]. Available: https://​de.m.wikipedia.org​/​wiki/​Datei:​RGB\_​color\_​cube.svg

[26] V. Bulova, *Farbmodell.* [Online]. Available: https://​homepages.thm.de​/~​hg10013/​Lehre/​MMS/​SS01\_​WS0102/​Farbmodelle/​Kapitel/​Kapitel4.html

[27] V. Popov, M. Ostarek, and C. Tenison, “Practices and pitfalls in inferring neural representations,” *NeuroImage*, vol. 174, pp. 340–351, 2018, doi: 10.1016/j.neuroimage.2018.03.041.

[28] Wikipedia, *HSV Farbraum.* [Online]. Available: https://​de.wikipedia.org​/​wiki/​HSV-​Farbraum​#/media/Datei:HueScale.svg