UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Kristjan Jansons

Leader-follower System for Unmanned Ground Vehicle

Master's Thesis (30 ECTS)

Supervisor: Tambet Matiisen, MSc

Tartu 2017

Acknowledgments

My sincerest gratitude goes towards Tambet Matiisen, who guided me in writing this thesis and how to proceed in life in general, which has been a significant boost in getting to where I am now.

I would also like to thank Kristjan-Julius Laak, who was the one who pointed me towards the world of machine learning.

Last but not least, I have to thank the people who have been there for me throughout the journey - Lauri Tammeveski, Robert Roosalu, Markus Lippus and Zurabi Isakadze.

Leader-follower System for Unmanned Ground Vehicle

Abstract:
Unmanned ground vehicles can be utilized effectively, if they have autonomous capabilities. One of those capabilities is leader-follower functionality, which means that the robot has to follow a predefined object. Usually the predefined object is a person, which is still more effective than controlling the robot with teleoperation using a remote control. It is worthwhile, because teleoperation requires the full attention of the operator while leader-follower system allows the person to keep their situational awareness, which is essential in a military environment. Leader-follower systems often exploit multiple technologies: LIDARs, GPS, infrared markers, radar transponder tags and cameras. The aim of this thesis is to develop and test the proof of concept leader-follower system, which relies on camera vision only and is able to follow any object which means that the class of the object does not have to be predefined. Specifically, the approach uses behavioral cloning to train deep siamese network with convolutional layers to determine the velocity and turning commands of the vehicle based on input from camera. The results show that the proof of concept system works, but requires further development in order to make it robust and safe.

Keywords:
Neural networks, machine learning, object tracking, vision-based systems

CERCS: P170

Juht-järgijasüsteem mehitamata maismaasõidukile

Lühikokkuvõte:
Mehitamata sõidukeid saab kasutada efektiivselt, kui neil on autonoomse sõitmise võimekus. Autonoomse sõitmise üks funktsionaalsustest on juht-järgijasüsteem, mis tähendab, et robot peab järgi minema etteantud objektile. Tavaliselt on defineeritud objektiks inimene, mis võimaldab hoida inimeste käed vabana, võrreldes situatsiooniga, kus masina liigutamiseks tuleb kasutada pulti, mis muudab sõdurid militaarolukorras haavatavaks sihtmärgiks. Olemasolevad juht-järgija süsteemid kasutavad funktsioneerimiseks kombinatsiooni erinevatest sensoritest, kasutades nii LIDAReid, GPS-i, infrapunamarkereid kui ka kaameraid. Selle töö eesmärgiks on arendada ja testida juht-järgijasüsteem, mis kasutab ainult kaamerasisendit ning mis oma kontseptsioonilt peaks olema võimeline järgnema igale objektile, st objekti tüüp ja välimus ei pea olema eeldefineeritud. Täpsemalt kasutab süsteem käitumuslikku kloonimist sügavate konvolutsiooniliste tehisnärvivõrkude treenimiseks, et ennustada kaamera sisendi põhjal roboti liigutamiseks vajalikke kiiruse ja pööramise käske. Tulemused näitavad, et pakutud süsteem töötab, kuid vajab edasiarendust, et teha see reaalse olukorra jaoks piisavalt robustseks ning turvaliseks.

Võtmesõnad:
Tehisnärvivõrgud, masinõpe, objektide jälgimine, pilditöötlusel põhinevad süsteemid

CERCS: P170

Contents

1 Introduction

Unmanned ground vehicles can be utilized effectively, if they have autonomous capabilities. One of those functionalities is leader-follower functionality, which means that the vehicle, being the follower, has to follow a moving object, being the leader. Leader-follower systems are in active use in the logistics, military and agriculture. This thesis focuses on the point of view of the military, as it has been conducted in collaboration with a military company named Milrem.

Leader-follower systems are used in the military because it enables the soldiers to keep their situational awareness which is vital in dangerous environments. When vehicles are controlled via teleoperation with remote controls, the full attention of the operator is needed, meaning that the operator loses situational awareness and becomes an easier target for enemy troops.

The object being followed is usually a person, but it may also be another manned or unmanned vehicle to easily enable the formation of convoys. This makes it a complicated task for the military because the number of object types and the appearance of the objects may change, meaning that it is not reliable to use predefined classes of objects for the leader-follower systems.

Leader-follower systems often exploit multiple technologies: LIDARs, GPS, infrared markers, radar transponder tags and cameras. Military prefers to use non-emitting sensors in order to limit the detectability of troops, which means that LIDARs, infrared markers and radar transponder tags have their limitations. The GPS is often not accurate enough and military also has to function in GPS denied areas. The camera, being a passive sensor, is a preferred solution for the military as it does not need to emit signals for it to work.

The aim of this thesis is to develop and test the proof of concept leader-follower system based on machine learning, which relies on camera vision and is able to follow any object, meaning that the class of the object does not have to be predefined.

1.1 Contributions of the author

The author has devised a full approach from data collection, algorithm implementation, neural network architecture exploration to analysis of the results in order to build a proof of concept system for the leader-follower functionality.

The data collection process was conducted relying on the concept of behavioral cloning. This means that the aim was to train the neural network to mimic the behavior of a human driver. The data collection process was conducted using Milrem's THeMIS Type 3, which is a tracked unmanned ground vehicle. The vehicle was controlled by an operator and it followed a person walking in front of it, trying to keep a certain distance from the person and keep the angle so, that the person would be in the middle of the

image. For this the author implemented the code for communicating with the vehicle and for collecting training data.

The author experimented with multiple neural network architectures for solving the task. The concept of neural network architectures relies on siamese networks [CHL05]. Neural network architectures and training of neural networks was implemented using Python library Keras [C+15].

1.2 Outline

Section 2 describes the data which was collected for this thesis. It describes the concepts for collecting data for behavioral cloning, the protocol for collecting the data and the way it was implemented technically. In addition, data exploration is conducted in order to give an understanding of the distribution and nature of the data.

Section 3 describes the methods for implementing the leader-follower system. It describes the concepts of neural networks, convolutional neural networks and siamese networks which are the components of the system. Additionally, it covers the different types of network architectures and the techniques that were used to optimize the training process.

Section 4 describes different neural network architectures for solving different tasks and also gives an overview of their performance.

Section 5 discusses the difficulties encountered during training and compares the results of training different network architectures. Analysis is performed in order to understand what affected the performance. Finally, potential future directions are discussed, which could improve the performance of the system.

Lastly, section 6 draws the conclusions from the work conducted. It brings out the most important lessons and results.

2 Data

In this section the data collection principles and the process itself are described. In addition, data exploration is performed in order to give an understanding about the data that was collected.

2.1 Data collection

The leader-follower system was built with behavioral cloning approach. The purpose of behavioral cloning is to learn to mimic the behavior of human from the collected data.

As the system being developed is a leader-follower system, it was necessary to collect data that contains in itself the information about how the leader-follower system should work. The purpose of this thesis is to make a system, which could follow any object, but in order to simplify the data collection process, a human target was chosen. The simplification was necessary due to the fact that the leader needs to be on the move and it is much easier to coordinate the movement of a human leader compared to the movement of another vehicle in front of the follower-vehicle.

The appearance of the human leader was altered throughout the data collection process with clothing of eight different colors: yellow, purple, golden, red, orange, pink, white and multicolor (red and yellow). This means that the system that was trained is definitely biased towards following only humans, but the alteration in appearance still enables to demonstrate the proof of concept system as the target is only defined by the target image, meaning it does not get any additional input in terms of a type of object, features to extract, etc.

The physical setup consisted of the following:

- Milrem THeMIS Type 3 UGV as the vehicle platform for the follower.
- Logitech C930e webcam for taking pictures of the leader. Good resolution (Full HD 1080p) was necessary to get images with good quality and a wide field of view (90 degrees) was necessary not to lose sight of the leader in case of direction changes which might sometimes be unexpected and swift in case of a human leader.
- Microsoft Xbox 360 wired controller for the operator to give commands to the vehicle. Left stick button was configured to give velocity commands. Right stick button was configured to give turning commands. Left and right bumper buttons were configured as dead man's switches. "Start", "back" and diagonal pad buttons were configured for initializing and terminating the data collection sessions.
- Laptop for running the data collection algorithms and forwarding commands to the vehicle.

The collected data had the following fields:

- Initial target images about the leader taken at eight angles, starting from the backside of the leader and turning 45 degrees clockwise after every picture. The pictures are taken at a distance and angle that are the distance and angle the leader-follower system wishes to achieve.
- Live camera images that were taken after every 200 ms when the system was activated.
- Turning command values from -1.0 to 1.0, where -1.0 is full left and 1.0 is full right. Turning commands were averaged over every 200 ms, after which they were recorded and forwarded to the vehicle if the system was activated. Turning commands have a deadzone from -0.25 to 0.25, which means that those commands do not make the vehicle turn. The deadzone is necessary for the operator to be able to give stop commands to the vehicle with shaky hands.
- Velocity command values given to the vehicle from 0.0 to 1.0, where 0.0 corresponds to standstill and 1.0 to speed forward. Velocity commands were averaged over every 200 ms, after which they were recorded and forwarded to the vehicle if the system was activated. Velocity commands have a deadzone from 0 to 0.25, which means that within that range the commands have no effect and are equal to standstill.
- Timestamp with millisecond precision.
- Leader height in pixels and angle approximation from the image calculated using a person tracking system based on SqueezeDet [WIJK16].

Safety protocols that were necessary during data collection:

- Milrem THeMIS Type 3 is capable of moving at a speed of up to 24 km/h and a collision with a human might end in severe injuries or even death. For this reason the speed of the vehicle was limited to 5 km/h, which is the average walking speed of humans, so that the leader could easily move away from the vehicle.
- The area for collecting data was restricted, so that people unaware of the potential dangers would not be harmed. Only the leader and the operator were allowed in the area.
- Velocity and turning commands were effective only if the operator held down two dead man's switches under both index fingers.

- The initial target images set the distance and angle that the operator tried to hold or restore. For safety, the distance of the leader was roughly 4 m while taking initial target images, so that the operator would not try to get too close to the leader. If the follower got closer to the leader, the operator made the vehicle stop with the controller sticks or let go of the dead man's switches, which also stops the vehicle.

The protocol for data collection was the following:

1. 8 initial images about the leader at different angles at a distance and angle that the system would like to achieve. Leader was kept at the center of the image in order to minimize the incidents when the leader is lost out of sight. This means that the operator tried to hold or restore the situation where the vehicle angle was so that the leader was at the center of the image. If the leader would not have been at the center of the image, there would have been more cases when the leader would have gone out of the field of view of the webcam.

2. The leader started moving within the boundaries of the restricted area and the operator regulated the velocity and direction of the vehicle with a controller for it to start following the leader. As mentioned before, the operator tried to hold or restore the initial leader angle and distance.

3. The distribution of commands given to the vehicle was monitored by the operator. If the commands given became too tilted towards some pace or direction, the operator gave instructions to the leader about necessary modifications. For example, if the turning command distribution became too tilted towards right, the operator signaled that the leader should move more to the left in order to even out the distribution of data. The same logic was applied to velocity commands as well. For example, when too many commands were given at full throttle, the operator asked ythe leader to slow down to also collect data at slower velocity.

4. At the end of the session "stop" command data was gathered for cases when the leader was too close to the vehicle. This means that data was gathered for cases when leader was too close the vehicle and velocity and turning commands with value "0" were given.

5. With every color of clothing about 40 minutes of driving was conducted, which resulted on average about 12000 images per clothing.

2.2 Data exploration and analysis

The data was gathered in eight different colors: pink, golden, purple, yellow, multicolor, orange, white and red. Examples of them can be seen on Figure 1.

Figure 1. Initial target images from the backside of the leader with different type of clothing represented.

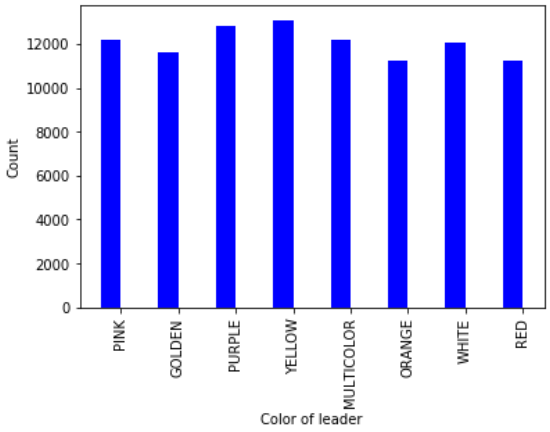

Figure 2. The amount of samples collected for every color of clothing.

The average amount of samples per color was 12055, yellow had the most at 13094 and orange had the least at 11254. The distribution of samples can be seen on Figure 2.

Next, unusable commands were thrown out. The movements of the follower were limited only to forward movement, which means that those commands, which commanded the velocity to go backwards, were thrown out. Another type of command that was disabled, was turning during standstill, because the vehicle acted very sharply and the leader easily got out of the field of view of the camera. On average, this only affected 57.25 commands per color, while the most was discarded for pink color with 133 samples.

Analysis was required to understand whether learning the velocity and turning commands from the images would even be possible it could even be possible to learn the

velocity and turning commands from the images. First, it was analyzed whether it is possible to learn the turning commands from the images. This means that there should be a correlation between the location of the person on the images and the turning commands. The operator tried to keep the direction of the vehicle so, that the person being followed would be in the middle of the image. This meant that when the person was on the left side of the image, the vehicle should turn left, vice versa for the right side and should keep straight while the person is roughly in the middle. For this task the approximate angle of the leader on the images was calculated with software based on SqueezeDet's pedestrian tracking [WIJK16]. An example output of the software based on Squeezedet's pedestrian tracking can be seen in Figure 3. The approximate angle α was calculated on the principle shown on Figure 4.

Figure 3. Output of the software based on SqueezeDet's pedestrian tracking. A bounding box is drawn around the person. The approximate angle of the leader on this image is 0.34 radians and the height of the bounding box is 176 pixels.

The distribution of the turning commands and approximate leader angles is shown on Figure 5. One can see that the approximate angles follow Gaussian distribution while the turning commands distribution is more uniform. For the turning commands we can see

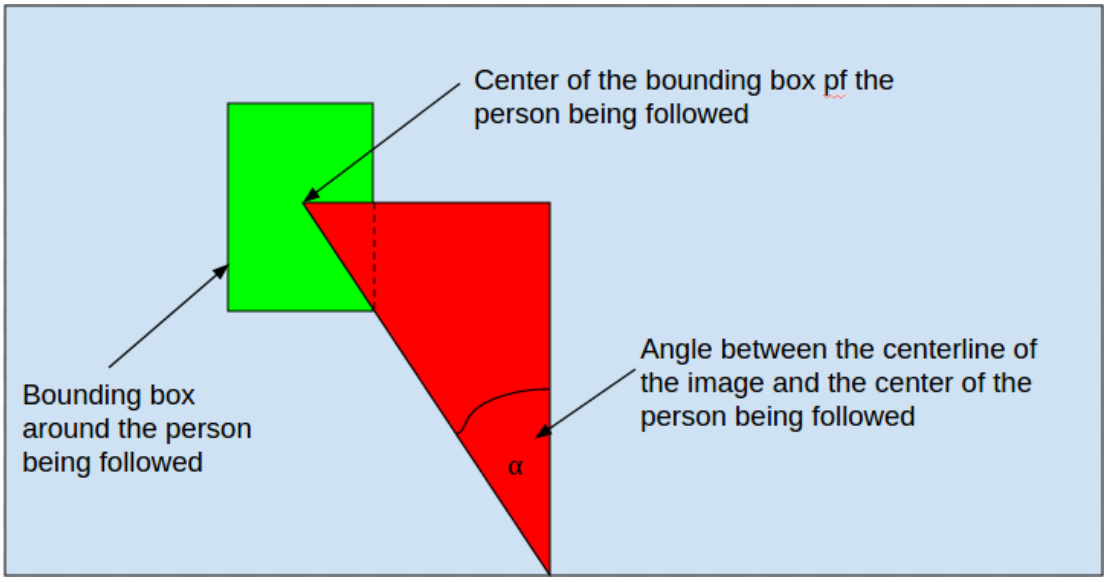

Figure 4. The approximate angle calculation of the leader in the images. The angle calculation uses the amount of pixel for catheti values in order to calculate the value of α

peaks near the center and the extremum values which indicate that the operator has often used maximum value in order to achieve the required angle, which might indicate poor handling of the vehicle or sudden changes in the moving direction of the leader. We can also see that the left and right turning command distributions are different which is due to the handling peculiarities of the vehicle, as one side weighs more than the other. It has to be noted that the turning command range from -0.25 to 0.25 is the range where the turning command has no effect and is equal to 0.0.

The correlation between the approximate angle and the turning commands can be seen on the histograms in Figure 6. The turning commands were separated into bins based on approximate leader angle and the contents of the bins were plotted. It can be seen as the target moves more to the left, the turning command average moves more to the left as well. Similar behaviour can be observed for the right direction. It has to be noted that the variance of the turning commands is quite high, which means that the operator has not given optimal commands, but has done a lot of under- and overcompensation to keep the vehicle directed towards the leader. It can be concluded that the correlation between the location of the leader on the images and turning command values given should be strong enough for the machine learning system to be able to learn from it.

Second, it was analyzed whether it is possible to learn the velocity commands from the images. This means that there should be a correlation between the person's height on the images and the velocity commands. For this, the same software based on SqueezeDet's pedestrian tracking [WIJK16] was utilized as was with approximate angle calculation. As the person moves farther, the bounding box height for the person becomes smaller

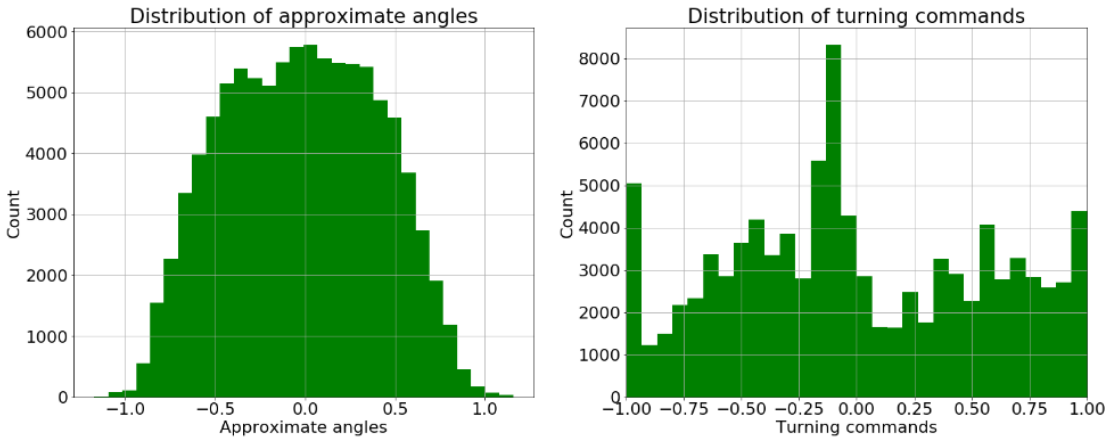

Figure 5. On the left the distribution of leader approximate angles. On the right the distribution of turning commands given to the vehicle by the operator.

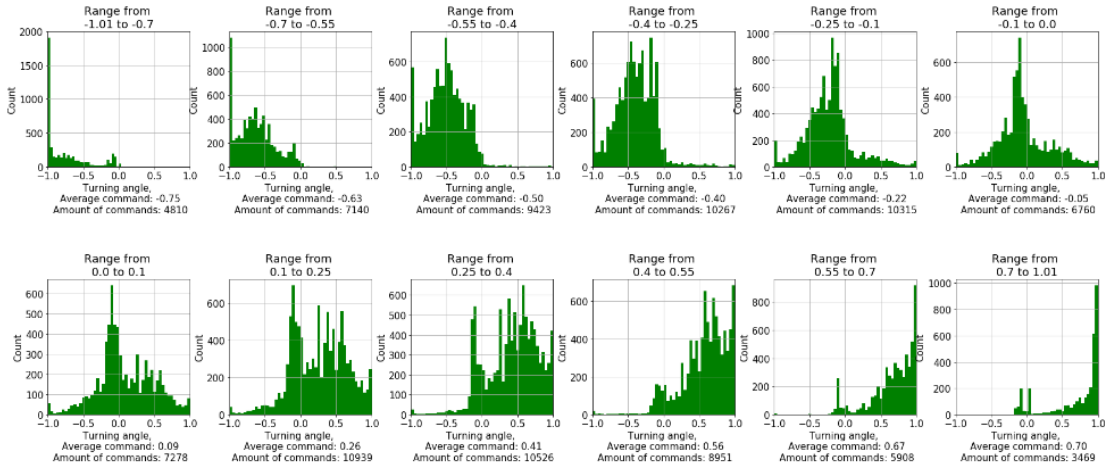

Figure 6. The distribution of turning commands dependent on the approximate angle. The range value on top of the plots represents the approximate angle range and the plots show how the turning commands are distributed for different ranges of the approximate angle values. It can be seen that there is a clear correlation between the approximate angle and the turning commands given by the operator.

and the velocity command should increase in order to restore the initial distance and vice versa for being closer. When the leader is too close to the follower, the vehicle should stop.

The distribution of bounding box heights and velocity commands can be seen on Figure 7. The average bounding box's height for the initial target images was 243 pixels, which means that most of the images have been taken while the leader is further than

that. This indicates that most of the velocity commands should be between 0.25 and 1.0 to recover the initial distance and the corresponding behavioral pattern can indeed be seen in the distribution of velocity commands. .

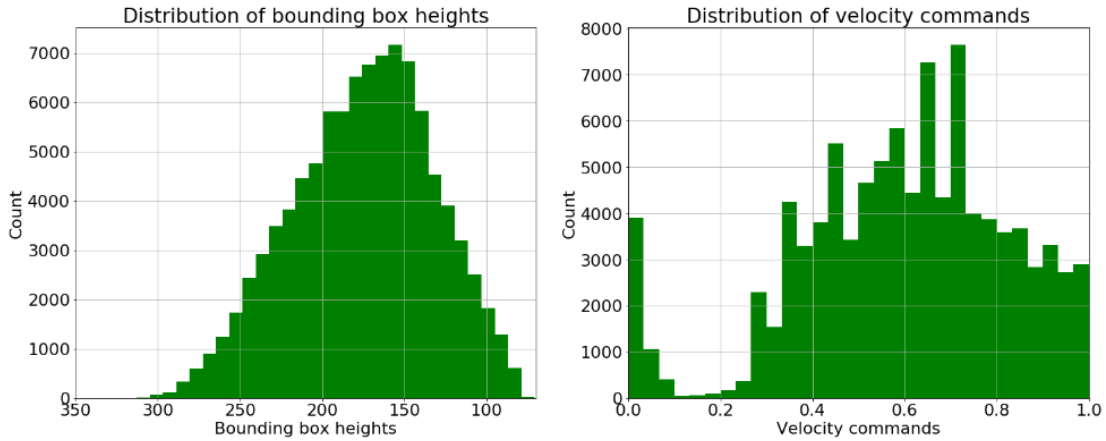

Figure 7. On the left the distribution of the bounding box heights which were created around the leader by software based on SqueezeDet's [WIJK16] pedestrian tracking. On the right the distribution of velocity commands given to the vehicle by the operator.

Figure 8 displays how the velocity commands are distributed in terms of the bounding box heights. It can be seen that as the leader moves further, meaning that the bounding box height gets smaller, the velocity command increases and vice versa. When the leader is too close to the vehicle, the stop command is given, meaning that the velocity command is between 0.0 and 0.25. From this it can be concluded that there should be information in the images to learn the velocity commands.

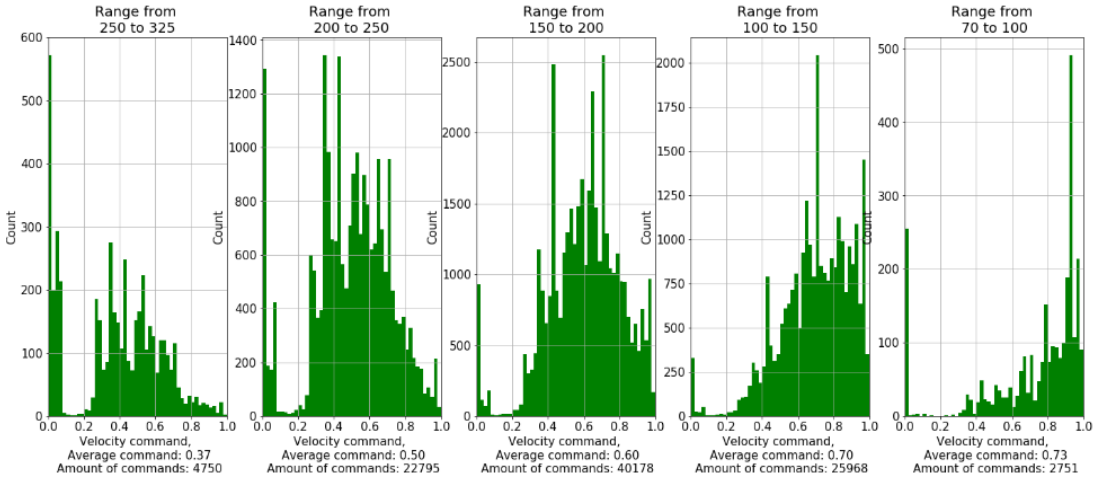

Figure 8. The distribution of velocity commands dependent on the bounding box heights which were created around the leader by software based on SqueezeDet's [WIJK16] pedestrian tracking. The range value on top of the plots represents the bounding box height range (bigger means the leader is closer) and the plots show how the velocity commands are distributed for different ranges of the bounding box height values. It can be seen that there is a clear correlation between the bounding box heights and the velocity commands given by the operator.

2.3 Data preparation

First, velocity commands were binned by values of [0.0, 0.25, 0.4, 0.55, 0.7, 0.85, 1.01] and turning commands were binned by values of [-1.01, -0.85, -0.70, -0.55, -0.4, -0.25, 0.25, 0.4, 0.55, 0.7, 0.85, 1.01]. The distribution of data in those bins can be seen on Figure 9. It can be seen that the distribution of commands is skewed towards those commands which command the vehicle to go straight.

Undersampling by capping the amount of commands in both command bins was one way of getting a more balanced dataset, especially in regards to output. Balancing the dataset in regards to movement commands improved the performance of the neural network, but it could also be done by input. The neural networks have to predict the movement commands and also whether the image is about the same leader or not. Of course, there is a correlation between the commands and the position of the leader on the images as was described in 2.2, but the approximate angle and bounding box's height calculation has a much better quality than the movement commands which are very noisy. Also, to make the ability of deciding whether the images are about the same leader or not more robust, it made sense to make the dataset more balanced in terms of where the leader is on the images, because otherwise the leader will be too often in the middle.

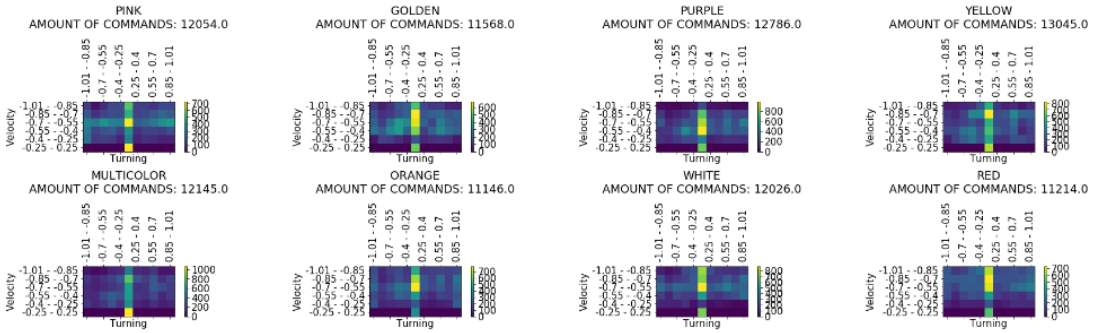

Figure 9. The distribution of data for every color of clothing taking into account both the velocity and turning commands. It can be seen that the data is skewed towards driving straight.

Therefore, bounding box's heights were binned by values of [50, 125, 150, 175, 200, 225, 300] and approximate angles were binned by values of [-1.01, -0.75, -0.60, -0.45, -0.30, -0.15, 0.0, 0.15, 0.30, 0.45, 0.60, 0.75, 1.01]. The amount of commands were placed in those bins accordingly and the distribution of data can be seen on Figure 10. It can be seen that the distribution of leader position is skewed towards leader being in the middle.

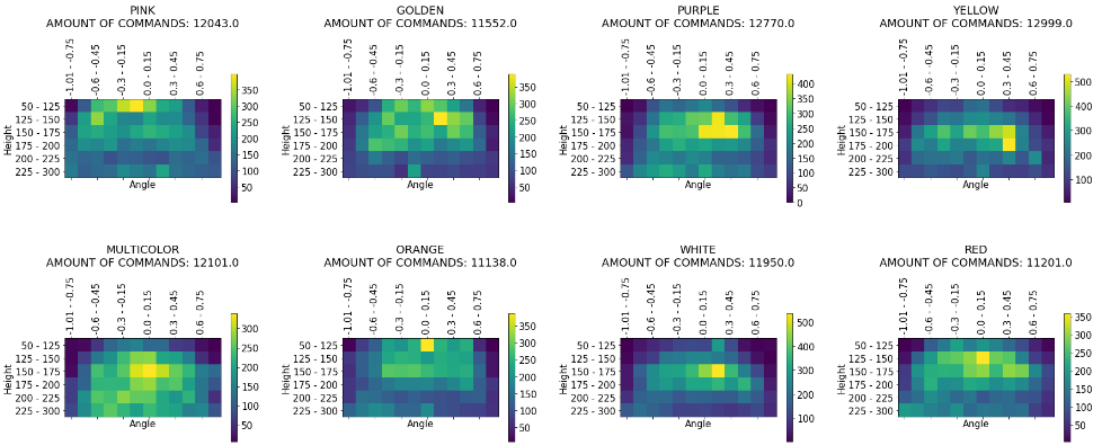

Figure 10. The distribution of data for every color of clothing taking into account both the velocity and turning commands. It can be seen that the data is skewed towards leader being in the middle.

Undersampling by capping the amount of commands in leader position bins was performed in order to get a more balanced dataset in regard to input. The cap was set at 100 images per position bin, which resulted in a distribution that can be seen in Figure 11. On average 52% of the data was kept. Higher data cap was tested, but it did not

improve the results. Due to the fact that there is a correlation between leader position and movement commands, the distribution of movement commands became more evenly distributed, which can be seen in Figure 12.

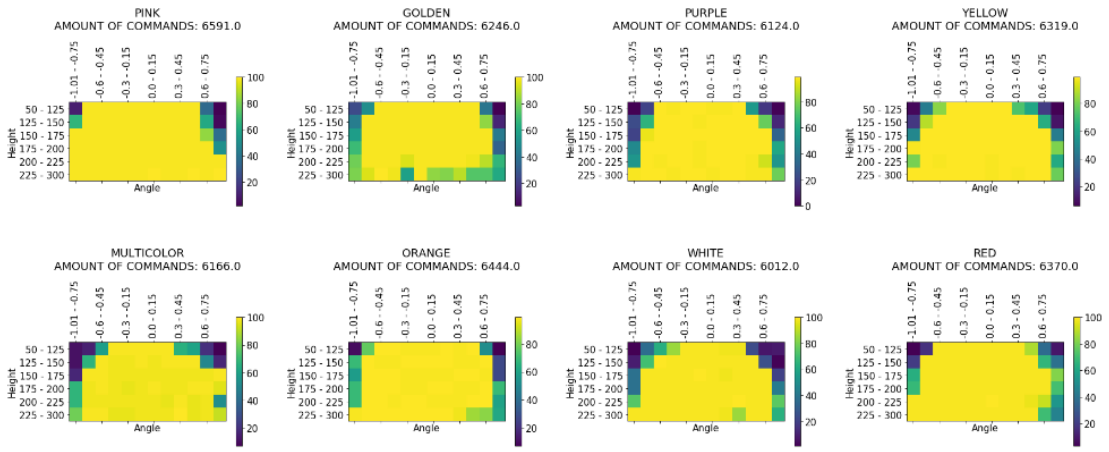

Figure 11. Undersampled data distribution by setting a cap of 100 images on every leader position bins. The distribution of data is a lot more balanced compared to Figure 10.

When predicting whether the image was about the same leader or not, the dataset was quite well balanced, but when predicting the commands, the effects of imbalance needed to be further adjusted. For this sample weights were created based on the frequency in command bins described above. The frequency calculation is based on the amount of commands in every turning and velocity bin after the capping process. The sample weights were in inverse correlation with the frequency, meaning that the more frequent the samples were, the lower was their weight and vice versa. Sample weights act as multipliers of sample loss (and therefore gradients) in the backpropagation algorithm.

A pairing of correct target leader images and camera images was created. Target leader images give information about how the leader looks like and the camera images give information about the leader's current location so that necessary velocity and turning commands could be predicted. Another important property of the leader-follower system is that it should stay still if the correct leader is not in sight. For this, false pairings between camera images of correct color and target leader images of other colors were created. The total amount of false pairings created was the same as correct pairings.

During the data collection process 8 initial target leader images were taken in the beginning of the session so, that the leader would be at 8 different angles. At first the neural networks were trained so, that only the 8 initial target images were used as target images, but it was unsuccessful as there had to be more diversity in the target images for the neural networks to generalize. To overcome this problem extra target images were sampled based on leader angle and bounding box's height so that the leader would be

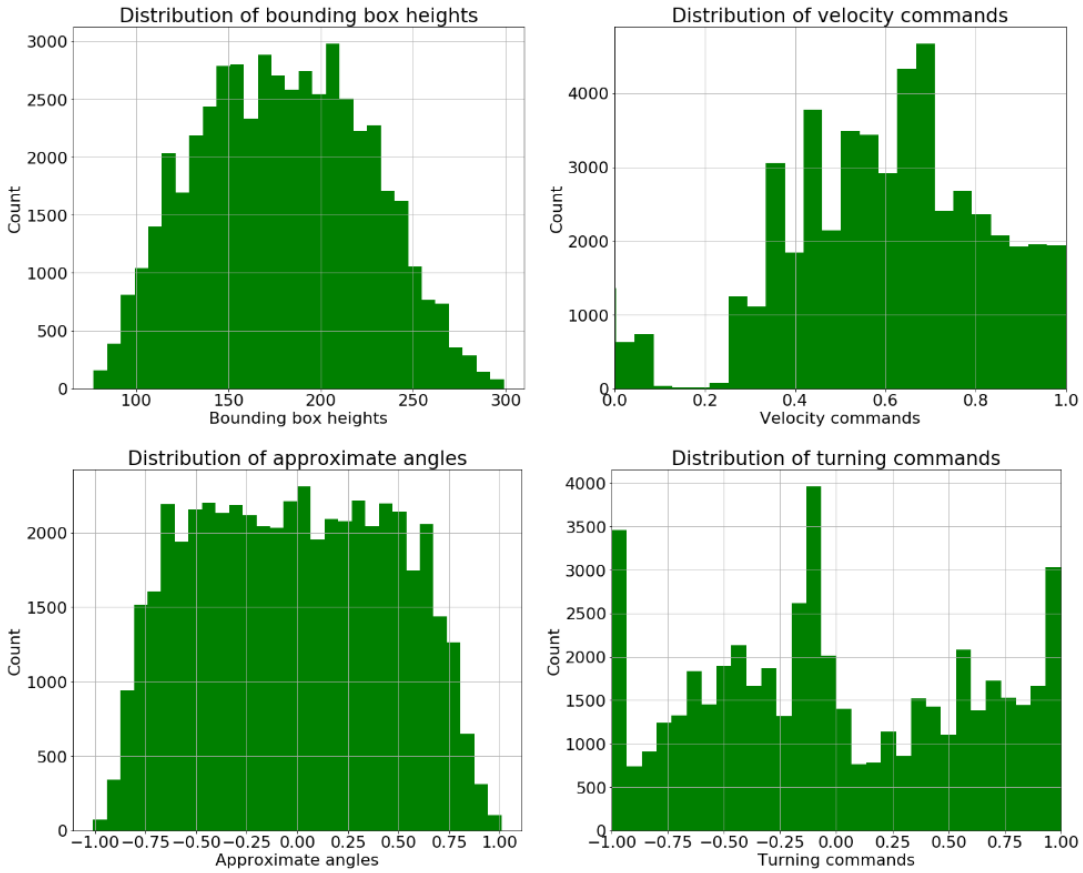

Figure 12. Distribution of movement commands after undersampling the dataset according to leader position on images. The distribution of commands is more balanced compared to Figures 5 and 7.

roughly in the middle and at the correct distance based on bounding box. This improved the generalization capabilities of the neural networks significantly.

In order to extract features from images the principle of transfer learning was used. Pretrained convolutional neural network ResNet50 [HZRS15] trained on ImageNet dataset were used. [DDS+09] Previous task-specific top layers were removed from pretrained model. The weights of the remaining layers were frozen, which means that they were not further trained when exposed to the new training dataset described above. Freezing the layers enabled to speed up the training process significantly since the images were ran through the convolutional neural networks once and the outputs of the convolutional neural networks were saved. The saved outputs of the convolutional neural networks were used as inputs to the leader-follower neural network.

3 Methods

In this section an overview of the algorithms used are given. In addition, different training configurations and techniques are described. The performance of the algorithms for leader-follower system is described in the next section.

3.1 Neural networks

Neural networks have been around for decades, but their popularity for image recognition has significantly risen after ImageNet competition was won in 2012 using convolutional neural networks [KSH12]. Three main reasons for this are increased computational power, especially the exploitation of GPUs, availability of large labelled datasets and algorithmic improvements like dropout [HSK+12] and rectified linear units [NH10].

Artificial neural networks are brain-inspired computational models used mainly for function approximation and classification tasks. Connection to the brain comes from the fact that artificial neurons behave similarly to real neurons as they are connected together in order to create networks, which is the same principle as in the brain. Artificial neurons are organized together into layers, in which they are not connected with each other, but connections are created between the layers. The general organization of layers can be divided in three: input layer, one or multiple hidden layers, which create non-linearities necessary to solve complex problems, and output layer for predictions. The general architecture of neural networks can be seen in Figure 13. [GBC16]

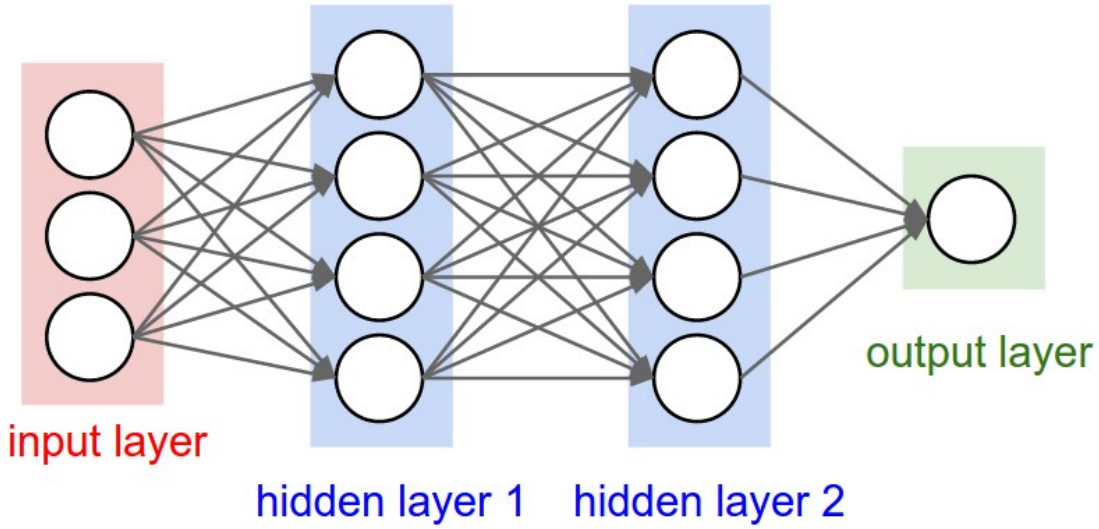

Figure 13. The general organization of layers in artificial neural networks. On the left the input layer, in the middle hidden layers for solving complex tasks enabled by non-linear activations, and the output layer for predictions. [Karb]

The computational process for every artificial neuron can be seen in Figure 14. Every incoming connection has its weight wi which is multiplied with the value of the input xi. The result of the multiplication is summed with bias b and the result of the summation is the input for the activation function. The activation functions are the source of non-linearity in artificial neural networks, which are necessary in order to solve complex problems. The result of the activation function is the output of the neuron, which is either the input for the next layer or the output of the whole network. The activation functions used in this thesis outside of pretrained ImageNet models are RelU [NH10], sigmoid [Kara], tanh [Kara] and ELU [CUH15] which can be seen in Figure 15.

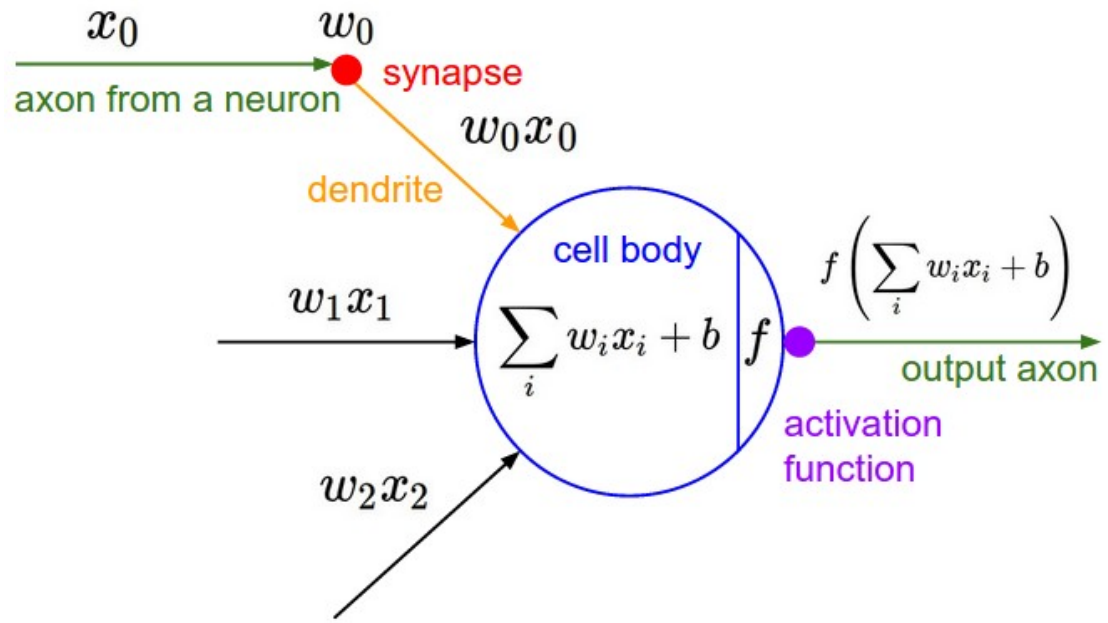

Figure 14. Computational process for artificial neurons and the relation to a real neuron. The artificial neuron uses weights, inputs values, bias and the activation function in order to calculate the output. The calculation done with the input is inspired by the way the neurons act in the brain as the input acts like a neuron, the weight of the connection can be thought as the synapse and weight-input calculation can be thought as the dendrite [Karb]

.

More neurons and layers in the neural networks increase the representational power of the network, which enables it to solve more complex problems, but can also easily lead to overfitting as they fit to noise. This means that the architecture of the neural network has to be optimal for the problem. The patterns in the data have to be complex enough in order to justify more neurons and layers. Overfitting can be a problem which is sometimes prevented with techniques like regularization or dropout [HSK+12]. Nevertheless, over

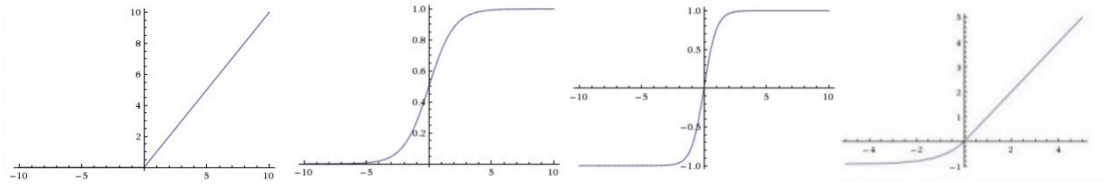

Figure 15. From left to right: RelU, sigmoid, tanh and ELU activation functions. RelU acts linearly if the input is greater than 0. Sigmoid squashes the output between [0; 1], tanh squashes the output between [-1; 1] and ELU which compared to RelU has mean zero leading to faster learning. [Karb]

.

the past years neural networks have become bigger and deeper, especially convolutional neural networks [HZRS15].

Neural networks are mostly trained with supervised learning using labelled data. Usage of labelled data is convenient as it enables to easily calculate the error in the output layer. A suitable loss function has to be defined for the training process to calculate the error in the output layer, for example mean squared error can be used for regression tasks and categorial cross-entropy can be used for classification tasks. The error is backpropagated through hidden layers towards input layer using the chain rule and the neural network parameters are updated using gradient descent. [GBC16]

In addition to the sheer amount of neurons and layers, an appropriate architecture has to be chosen as well for every problem. For example, it might be reasonable to solve regression tasks based on numerical features with simple feed-forward network which has only a few layers. But for images it is reasonable to use deep convolutional neural networks which enable to find hierarchical patterns and for a lot of natural language processing tasks recurrent neural networks are the reasonable choice.

3.1.1 Convolutional neural networks

First convolutional neural networks were assembled in the 1990's by Yann Lecun [LBBH98], but their triumph came in 2012 when AlexNet [HSK+12] won the ImageNet competition by significantly outperforming the runner-ups. In the coming years the performance of convolutional neural networks reached super-human levels for image recognition in the ImageNet competitions.[HZRS15] The strength of convolutional neural networks is the fact that convolutional layer enable to interpret hierarchical structures that exist within images and they do that with a lot less parameters compared to fully connected layers.

The layers of convolutional neural networks can be divided in three categories: convolutional layers, pooling layers and fully connected layers. Convolutional layers enable to interpret the hierarchical structure within the images using filters. Pooling

layers generally help with preventing overfitting and as they reduce the size of the output and therefore number of parameters in subsequent layers. Fully connected layers are used in the end of the convolutional neural networks for predicting required output by using the output from convolutional and pooling layers as features.

Neurons in convolutional layers are arranged similarly like pixels in images, meaning the configuration has width, height and depth. Connectivity of every neuron is defined by its receptive field which consists of kernel size defining the width and height for connections to be made. Depthwise the connections for neurons are usually done throughout the input depth, although more recent architectures like Xception [Cho16] also modifies the depthwise connectivity. Neuron connections to their receptive field can be seen on Figure 16. The neurons are organized in filters which process the image by sliding the filter by movements defined by strides. Sliding the receptive field over the input while maintaining the stride might require additional padding which means that zeros are added around the edges. The working principle of convolution operation can be seen in Figure 17.

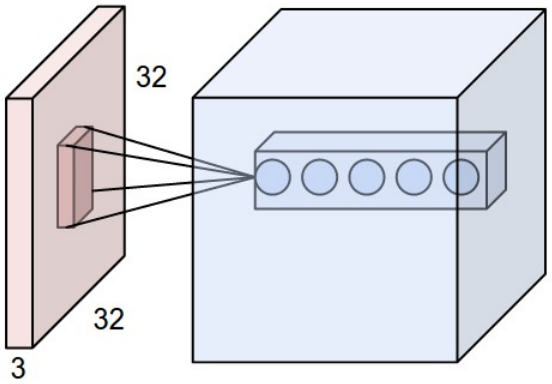

Figure 16. Artificial neuron connection to its receptive field in convolutional layers. The input is connected to a patch of the input which has its width, height and depth. [Karb]

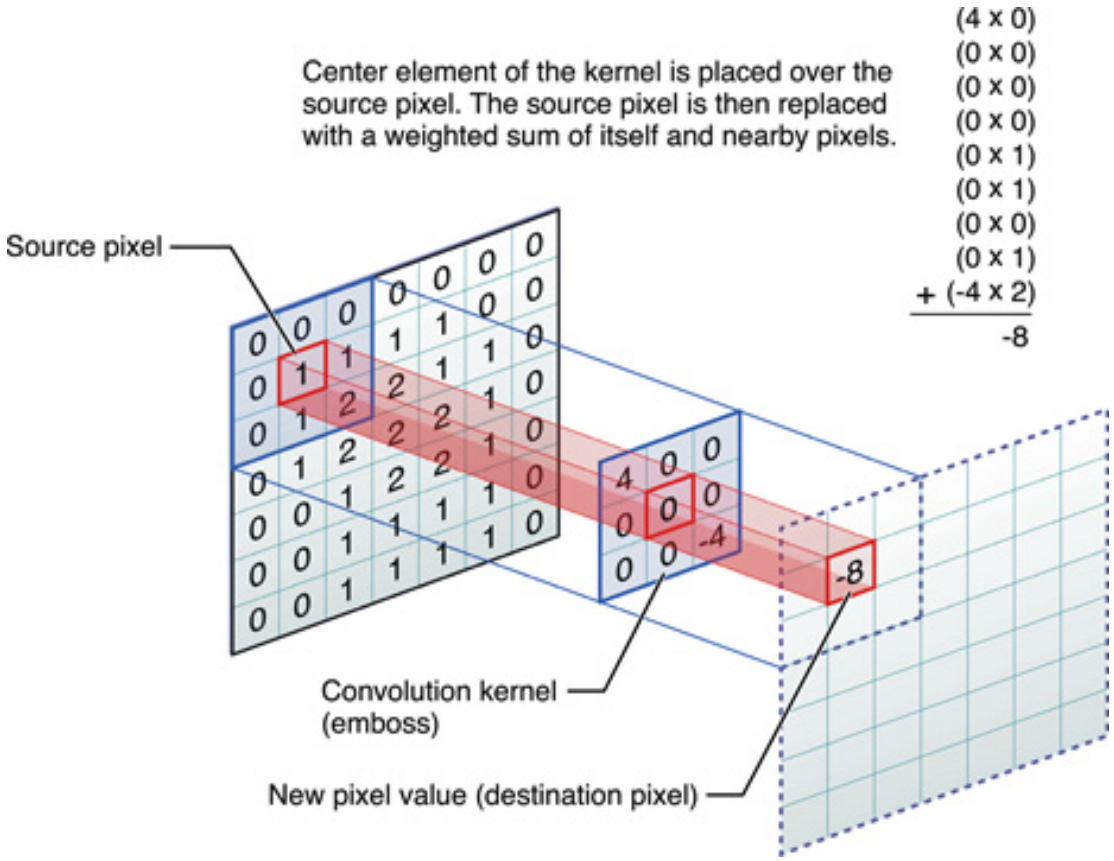

Figure 17. The working principle of convolution operation with a one channel image. The convolutional kernel center is aligned with the source pixel or pixels if it were a multichannel input. The input values are multiplied with the weights of the convolution kernel to calculate the output. In convolutional layers the output is summed with the bias and the output passes through the activation function as well which is not shown on this figure [App]

Image recognition tasks can be accomplished with fully connected neural networks as well, but it is feasible only with small images as the number of parameters with large images will increase to unreasonable levels. With convolutions it is possible to use parameter sharing by making an assumption that it is reasonable to use the same parameters to process input at different locations. In additions to a significant decrease in the number of parameters, it also makes the convolutional neural network location invariant as the same parameters are used to process input at different location.

Pooling layers are generally used to reduce the size of the input and reduce overfitting. Pooling layer parameters are the size of the window it affects, stride length and padding similar to convolutional layers. Pooling layers apply a function to the elements inside

the window it affect and produces an output. The most common functions to apply are average and max. An example of max pooling taking effect is shown in Figure 18.

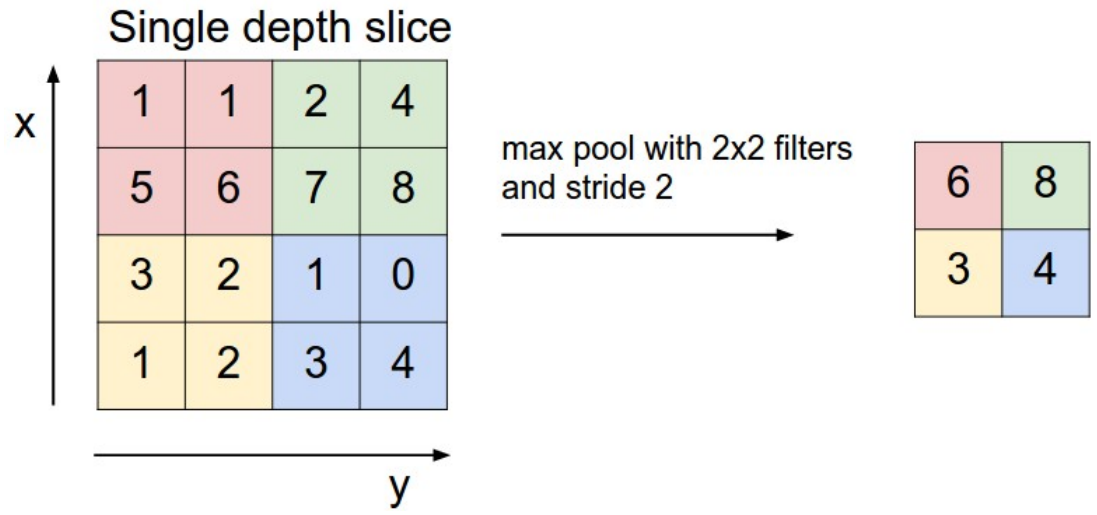

Figure 18. Max pooling layer with 2x2 window and stride of 2. [Kara]

A lot of other techniques have emerged which have shown to increase the accuracy of convolutional neural networks. Resnet's [SLJ+14] [HZRS15] residual modules which use shortcut connection to bring input closer to output lessens the problem of vanishing gradients which enables to build deeper networks and even helps to train the networks faster. Inception [SLJ+14] module stacks convolutions with different kernel sizes into one module which enables to extract features of different sizes from input. Xception [Cho16] module further develops the inception concept of separating convolutions also depthwise. These examples show that it is possible to add modules which are able to increase the performance of convolutional neural networks.

3.1.2 Siamese neural networks

Siamese neural networks [CHL05], [KZS15] are a type of neural network that contain subnetworks which have the same architecture and have identical weights. It is reasonable to use siamese neural networks if the subnetworks have to process comparable inputs in order to find similarities or relationships between the inputs. For this reason, it makes sense to use siamese neural networks in the current thesis as features extracted from initial images and camera images should have relationships between them. In addition, due to the fact that the weights are identical in two branches the number of parameters in the neural networks is decreased significantly.

3.2 Training configuration

Numerous architectures were tested for solving the leader-follower system task. The main overall principle of the architecture modifications is described in 3.2.1 while the results are discussed in section 4. Additionally, other methods which were exploited during the training process are described as well.

3.2.1 Network architecture

General overview of the architecture can be seen in Figure 19 inspired by a neural network architecture capable of visual navigation trained with reinforcement learning principles [ZMK+16]. Throughout the training process multiple configurations were tested, but the overall principle remained the same.

The inputs to the neural network are target image or images and camera image or images. The target image represents the object and gives information to the system about what to follow. The neural network trained in this thesis is biased towards following only a person since it has only seen that data, but in theory it could be any object, but then also the training data should be a lot more diverse. The camera image represents the current situation and gives information to the system about how to follow the object on the target image.

At the beginning of the data recording session eight different leader images were recorded at different angles of the leader. In case only one leader image was used in the neural network, the image with leader backside was used since it was the most common angle during the following procedure. If more than one image was used, those images where the leader had turned the least compared to the backside image were used. For example, if three initial images were used, the backside image and two images were the leader had turned 45 degrees clockwise and anticlockwise were used. As described in the data preparation section the approach of using randomly sampled images from the dataset where the leader is positioned similarly to initial images made the neural network generalize significantly better and the approach of using only initial target images for target image was dropped, but it could be returned to if there is a lot more diverse data.

Using multiple images as target images was tested on the idea of having a more diverse representation of the leader, but it did not improve the results and it was dropped. Additionally, multiple successive? camera images usage was tested on the idea of giving more information about how the target and the vehicle is moving, but it did not improve the results and it was dropped.

The ResNet50 model used in this architecture has its top layers removed, meaning that the convolutional layer output after flattening is used as features for the new layer on top of it.

For merging the features from ImageNet models two techniques were tested: multiplication and concatenation. Only with concatenation reasonable results were achieved.

In order to predict the velocity and turning commands from the image features fully connected layers were added to the network. The output of the fully connected layers was the input to velocity and turning command activations. For the velocity command sigmoid activation was used as it capped between 0 and 1 which is the same range where as it is for the velocity command, although later it was replaced with linear activation because of saturation. For the turning command tanh activation as it is capped between -1 and 1 which is the same range as it is for the turning commands. Both of these activations were concatenated together and an MSE loss function was applied to the network.

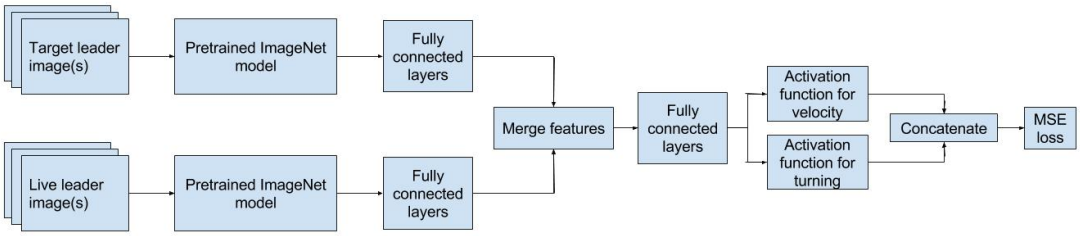

Figure 19. General architecture of the neural network which was implemented. Using one or multiple images for initial leader images and camera images was tested. Pretrained ResNet50 model was used [HZRS15]. The output of pretrained imagenet model went through fully connected layers which were merged and used as an input for fully connected layers. Different activation functions were tested for both velocity and turning which were concatenated together and MSE-loss was applied for both of them.

3.2.2 Other techniques implemented

Cross validation was applied in order to validate the results of the trained neural networks. The training process was performed on six colors and validation was performed on the seventh color which was not used during training. If the split would have been done on a randomly shuffled dataset where the both sets contain all colors the results would have been overly positive as training and validation sets would have contained a lot of images that were too similar to each other. Half of every training split contained correct pairings and the other half contained false pairings. The correct pairings contained initial leader images with 6 colors and the correct camera image with corresponding turning and velocity commands. The false pairings contained the images from 6 clothings combined so that the leader clothing on initial target images and the camera images did not match and the velocity and turning commands were set to 0. The eighth color red was used as testing and it was not included in the training or validation set.

When the neural network contained a lot of parameters it was relatively easy to achieve overfitting. In order to avoid overfitting L2 regularization was used.

Euclidean distance together with contrastive loss was used to verify whether it is possible to predict if the target image and camera image are about the same leader or not. Constrastive loss pulls the images with the same labels together and pushes others apart. [HCL06] Euclidean distance helps to determine whether the feature vectors representing the images contain information about them which helps to separate the images into matching and non-matching classes. [MKR16].

3.2.3 Training approach

In general the leader-follower system has to be able to perform two tasks. First, be able to distinguish if there is a correct leader on the images. This means that giving turning and velocity commands should be activated only if a matching leader is visible on the camera image. The other task is to predict appropriate turning and velocity commands if there is a matching leader on the camera image.

First, a regular convolutional neural network was configured which could be able to determine if it was possible to learn the turning and velocity commands from the camera images without the leader image. This sort of system could still be useful if the leader would always be a human.

Second, a setup was configured to determine whether it is even possible to distinguish which images are matching. A siamese neural network was constructed which could be able to determine whether the two images match using euclidean distance and contrastive loss.

Third, different siamese neural network configurations were tested in order to combine the capabilities of the two aforementioned neural networks. They had to be able to predict both the movement commands and determine whether the target image and camera image were of the same leader or not.

4 Results

This section describes what methods were used for evaluating the results and describes what sort of setups were implemented and what were the results achieved with different approaches.

4.1 Techniques for evaluation

For evaluating the performance of neural networks multiple approaches were needed. One of the subtasks was regression while the other was classification task and thus different approaches for evaluation were necessary.

As the initial data for velocity and turning commands is quite noisy, it is difficult to assess how much of the variance is captured by the neural network as there is also a lot of prediction error, noise and unaccounted input which are the cause behind the values of velocity and turning commands. For example, the inertia of the vehicle, charge of the vehicle battery and the surface on which the vehicle drove on affects how the operator had to give commands and can be thought of as unaccounted input while the shakiness of the operator hand increases noisiness of the commands. For assessing how much of the variance is captured by the neural network explained variance metric was used. Explained variance of 100% says that the network is able to explain 100% of the variance of the output with the input. The lower the value the worse it is.

Evaluating whether the target and camera image are about the same leader a classification task was performed. For the evaluation of this subtask accuracy metric was used.

Of course a lot of visualizations were necessary to understand the performance of the neural networks because metrics can not give all the necessary insights.

4.2 Results for learning commands without the siamese network

In the data exploration section it was determined that there is a correlation between the leader location on the images and turning command and between leader height on the image and velocity commands. For determining whether this correlation could be learned using neural networks a regular convolutional neural network was constructed which can be seen in Figure 20. It can be interpreted that the purpose of this network is to learn to track humans since only those objects are in the training dataset. If the dataset would have been more diverse, the network could have possibly learned to track salient objects.

The pretrained ResNet50 model was used as a feature extractor. It was followed by fully connected layers of size 32, 16 and 16 neurons. Those were fed into two separate activations. Sigmoid activation was used for velocity as the velocity command values are between 0 and 1. Tanh activation was used for turning as the turning command values

are between -1 and 1. This ensured that the command values would not go out of the allowable range. Activations were concatenated together and MSE loss was applied.

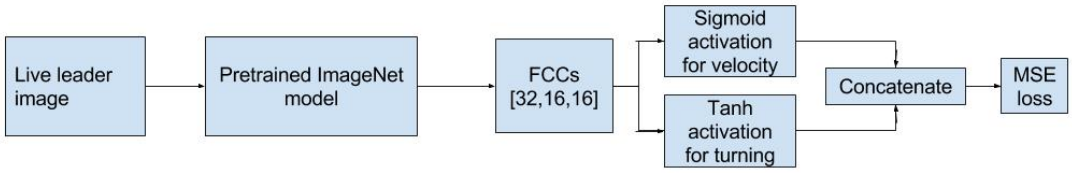

Figure 20. Neural network constructed for predicting velocity and turning commands.

The neural network was trained for 7 epochs and cross-validation showed that overfitting was very small while no regularization or dropout was used. The results were better for predicting turning commands were explained variance was over 0.65 and for velocity commands the explained variance was around 0.35. The results for explained variance can be seen in Figure 21 and 22 for turning and velocity respectively. It can be seen that test set results are quite close to validation set results which shows that validation set effect did not leak into the training set. The difference in explained variance between velocity and turning makes sense because of the peculiarities of the vehicle as the amount of velocity command needed depended a lot on the terrain, charge of the battery and inertia of vehicle which can not be captured at least fully on a single image. The turning command on the other hand acted on the vehicle quite expectedly. Velocity commands given by the operator have a lot more variance compared to turning commands.

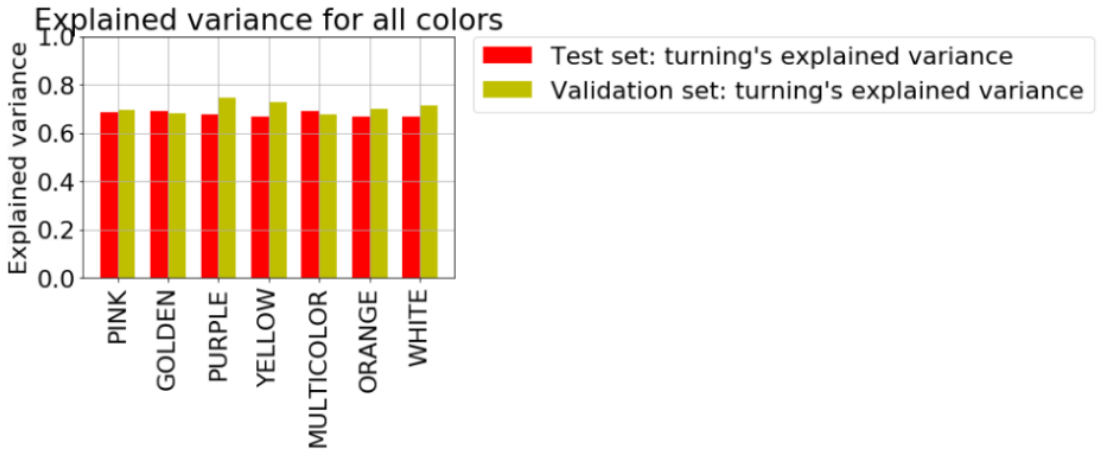

Figure 21. Turning command explained variance for test and validation sets. The colors on the bottom show the model which was used for making predictions. For example model "PINK" refers to the model which was trained with the data of other six colors and validated on "PINK" while "RED" remained the test set for all models.

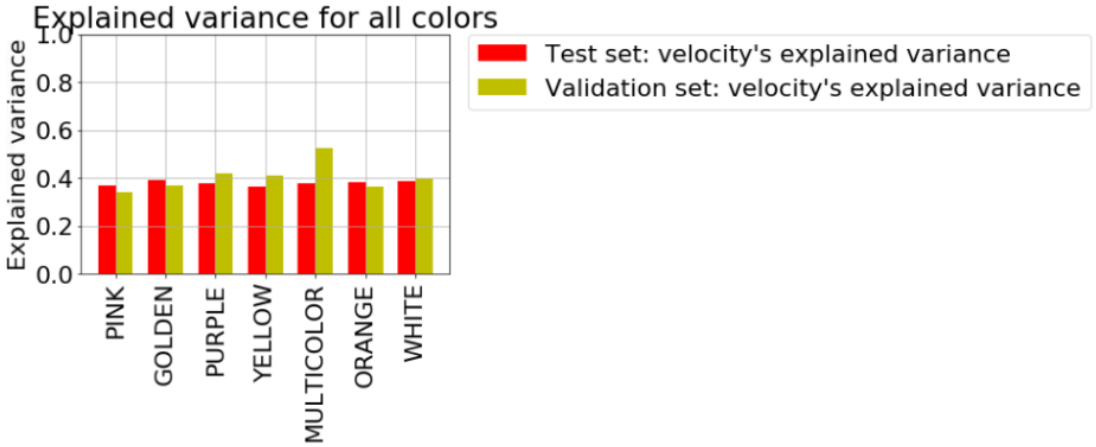

Figure 22. Velocity command explained variance for test and validation sets.

The distribution of original and predicted commands for test set can be seen on Figure 23 and the distribution for validation sets were very similar. Visually it seems that the system is able to predict the commands. The neural network has smoothed the velocity commands compared to original data. Turning commands roughly follow the original distribution.

The predictions were quite similar for all models and for presentation purposes "PURPLE" is randomly chosen. For test data the correlation between bounding box heights and predicted and original velocity commands have been brought out in Figure

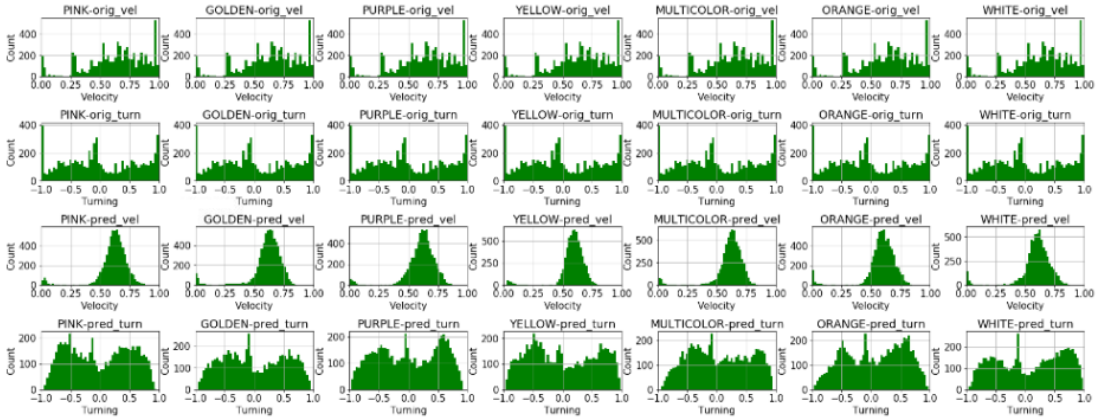

Figure 23. Distribution of original and predicted velocity and turning commands for test set. The first row represents the distribution of original velocity commands. The second row represents the distribution of original turning commands. The third row represents the distribution of predicted velocity commands. The fourth row represents the distribution of predicted turning commands.

24 and the correlation between approximate angle and predicted and original turning commands have been brought out in Figure 25. The predicted velocity are in correlation with the bounding box height and the predicted turning commands are in correlation with the approximate angle and we can conclude that it indeed is possible to predict the velocity and turning commands with convolutional neural networks based on the images in the dataset. Although it has to be noted that visually the correlation between turning and approximate angles looks better than the on between velocity and bounding box height.

Original velocity and turning commands distribution based on bounding box height and approximate angle can be seen on Figure 26 where one can see as was shown before that there is a correlation between velocity commands and bounding box heights as well as between turning command and approximate angle calculated from leader's bounding boxes. A similar correlation can also be seen on Figure 27 and Figure 28 where the predictions are represented, although the relationship is not so sharp, especially for velocity commands.

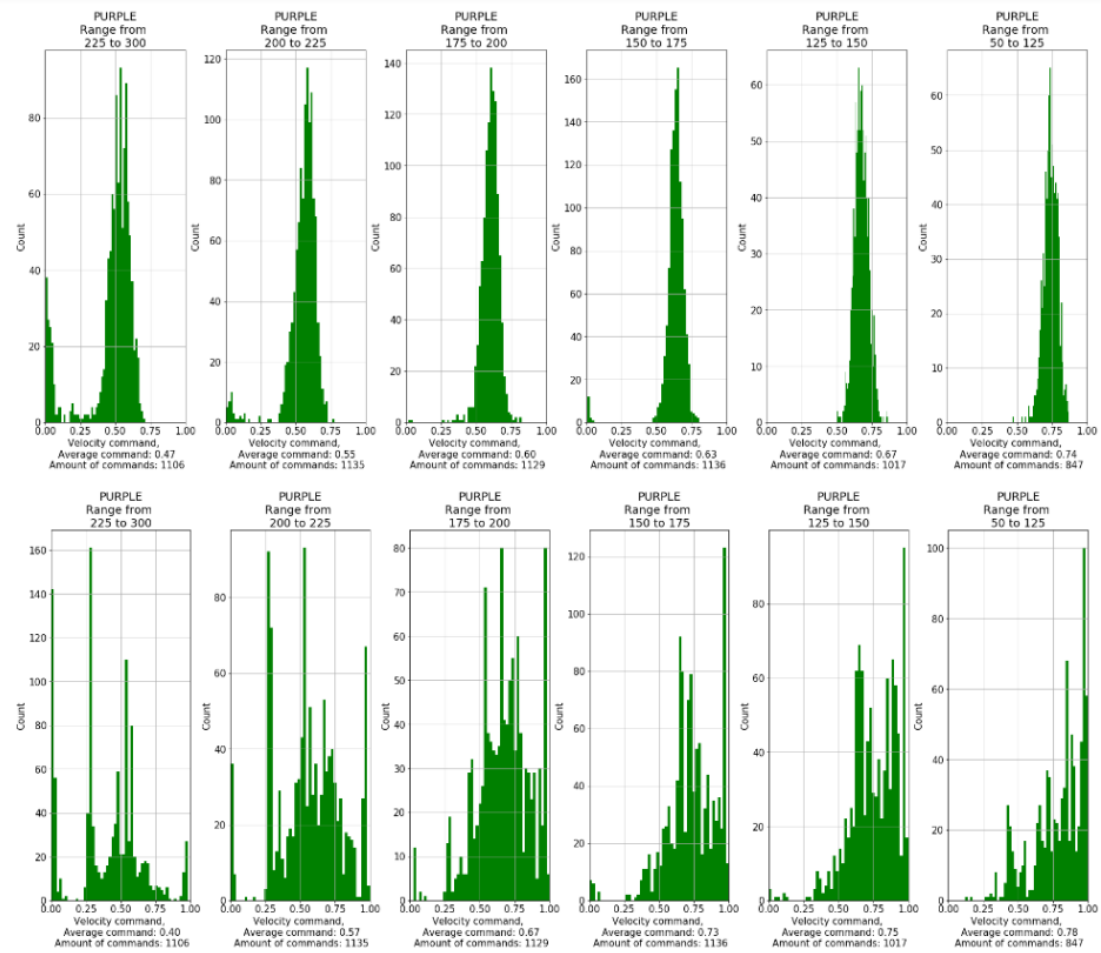

Figure 24. The distribution of predicted velocity commands dependent on the bounding box height for “RED” test set and model “PURPLE”. The top row represents the predicted velocity commands and the bottom row represents the velocity commands given by the operator.

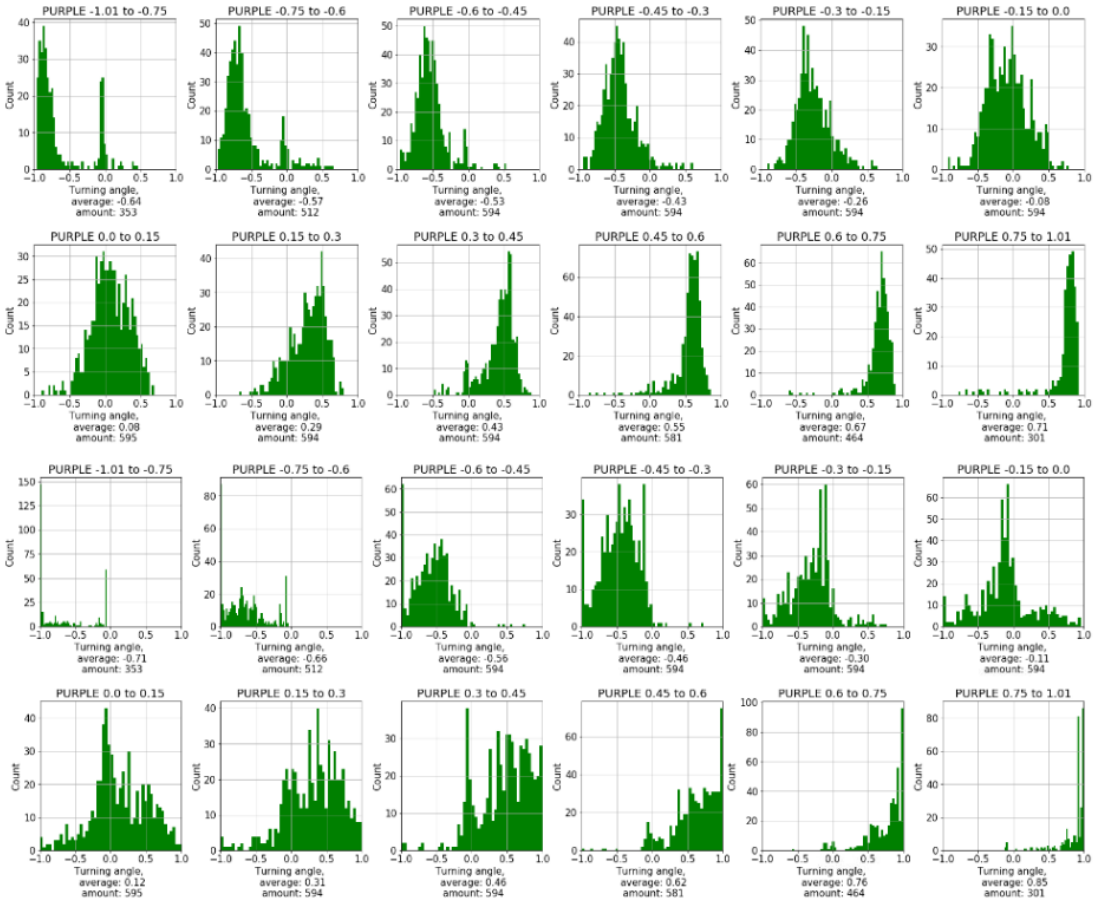

Figure 25. The distribution of predicted turning commands dependent on the approximate angle for “RED” test set and model “PURPLE”.

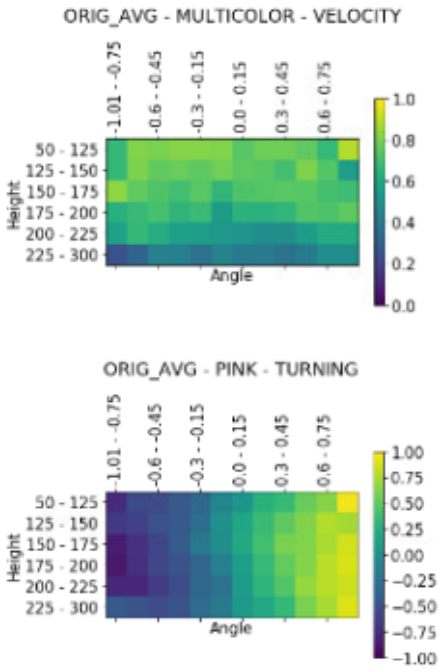

Figure 26. Colors represent original velocity and turning command distribution based on approximate angle and bounding box height.

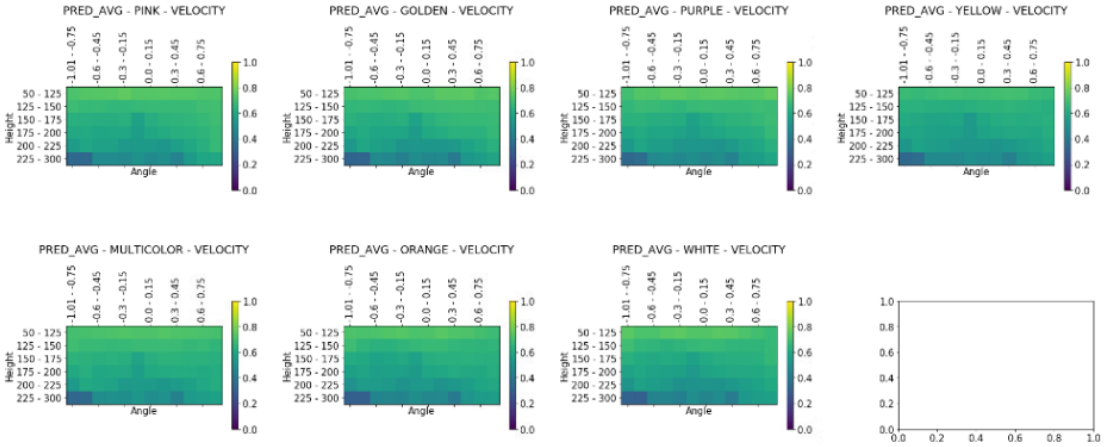

Figure 27. Colors represent predicted velocity based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.2, calculated on the predictions made on data points where the target and camera image had the same leader on it.

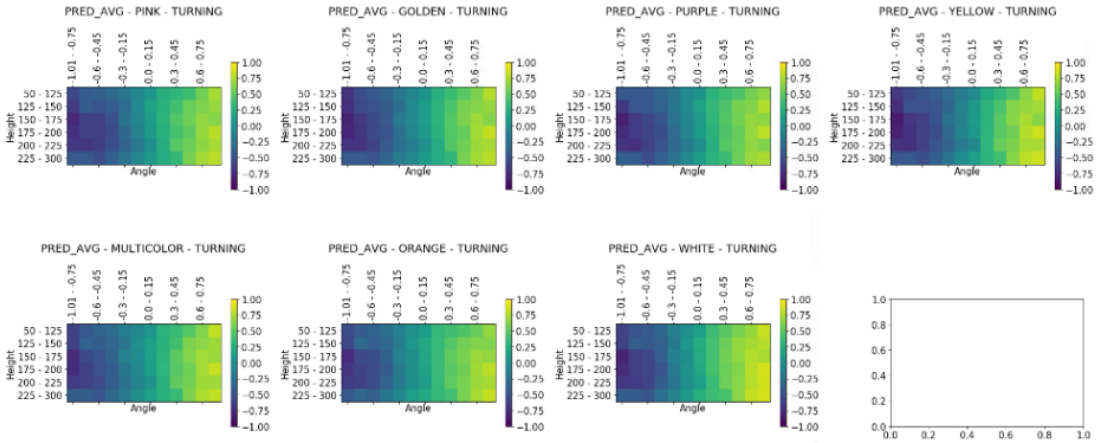

Figure 28. Colors represent predicted turning command based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.2, calculated on the predictions made on data points where the target and camera image had the same leader on it.

Potentially better results could be achieved with this or a similar architecture, but the goal of testing this architecture was to determine whether it is even possible to use convolutional neural networks to predict velocity and turning commands and it can be concluded that it indeed is possible.

4.3 Result for image matching classification

The leader-follower system has to be able to understand whether the target image and the camera image are about the same leader. If the images have a match, giving velocity and turning commands based on camera image is allowed. If there is no match the commands should be zero. This means that the neural network has to be able to predict whether the images are matching or not. For this, a neural network with architecture shown on Figure 29 was constructed. This architecture uses euclidean distance and constrastive loss to predict whether the images have the same leader or not [MKR16].

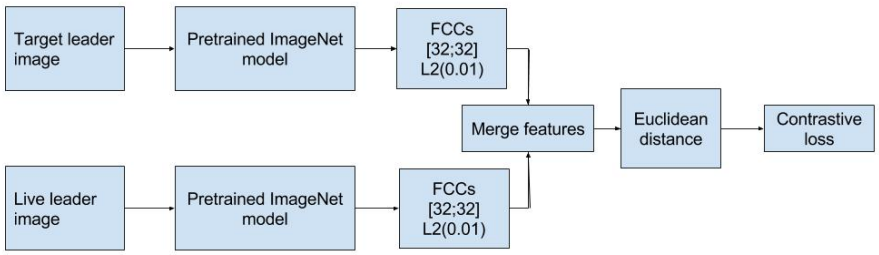

Figure 29. Neural network constructed for matching target and camera images.

In the beginning of every data collection session 8 images of the leader were taken which were planned to be used for the neural network to be able to understand whether the leader on the target image is the same as on the camera image. Using only these images to train the neural networked resulted in overfitting as the amount of images used in the target branch was very limited. It has to be noted that the error backpropagation from the camera image branch also affects the weights in the target image branch, but the same goes vice versa for the camera image branch. This is due to the fact that the branches share weights which is a characteristic of siamese neural networks. As simple regularization techniques were not able to solve this problem, more data was needed. Instead of using only the initial 8 target images from every recording session, all images where the same leader was roughly in the middle and at a similar distance as on the initial target images were taken into use for target branch data. The diversification of data enabled the neural network to learn to distinguish matching and non-matching images with an accuracy around 85% for both cross-validation and test data. The euclidean distance threshold was set at 0.5. The results of the accuracy for cross-validation and

test data can be seen in Figure 30. In order to understand where the mistakes come from precisions and recalls were plotted as well in which are seen on Figure 31, 32, 33 and 34. From those plots we can see that the classifier classifies the images as the same too often which is a logical mistake as most of the images are very similar while the only difference is the leader and the lighting conditions for the leader might not always be good. When looking at the accuracies based on bounding box locations on Figure 35 the results are all diverse and a conclusions whether the accuracy depends on leader location can not be drawn, although logically the leader has to be at least big enough that there would be a signal, but in this case it does not seem to be a problem.

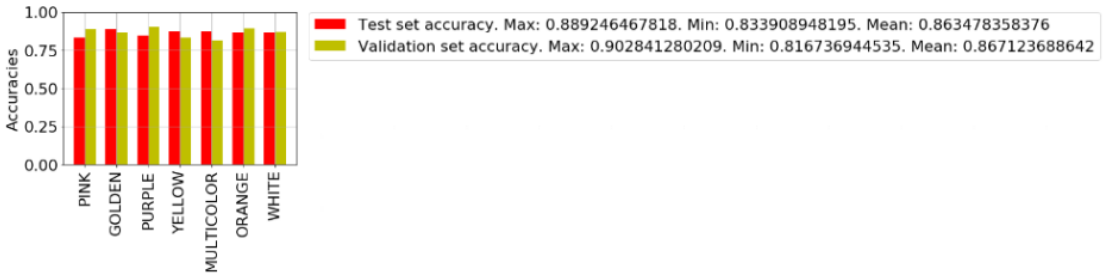

Figure 30. Classification accuracy for image matching.

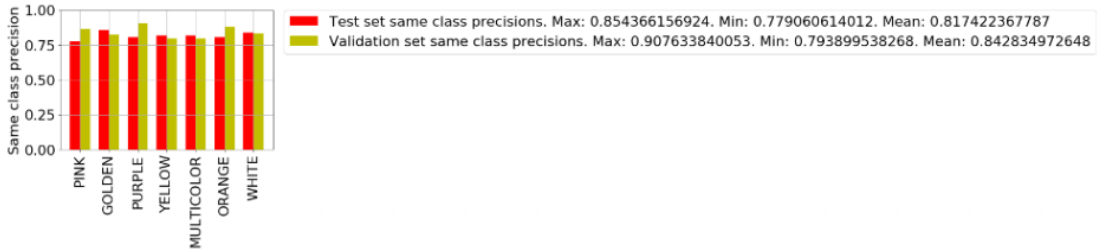

Figure 31. Precision for image classification when camera image and target image are about the same leader.

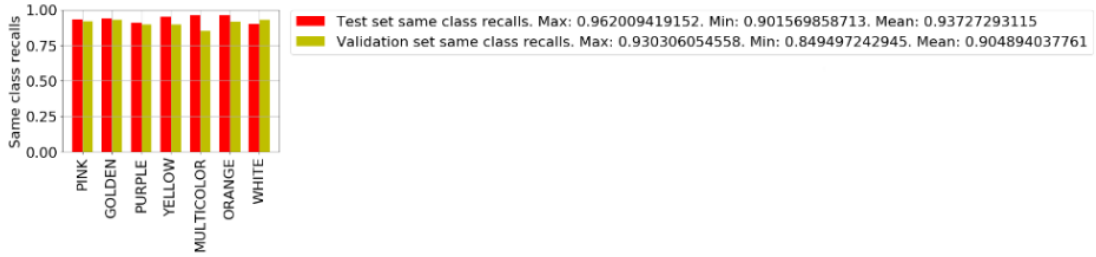

Figure 32. Recall for image classification when camera image and target image are about the same leader.

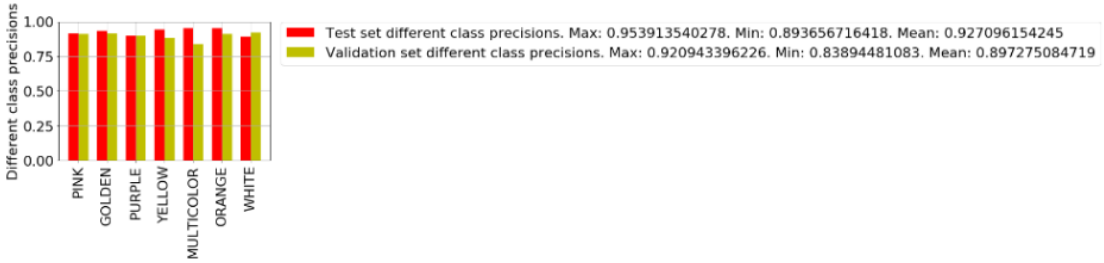

Figure 33. Precision for image classification when camera image and target image are about different leaders.

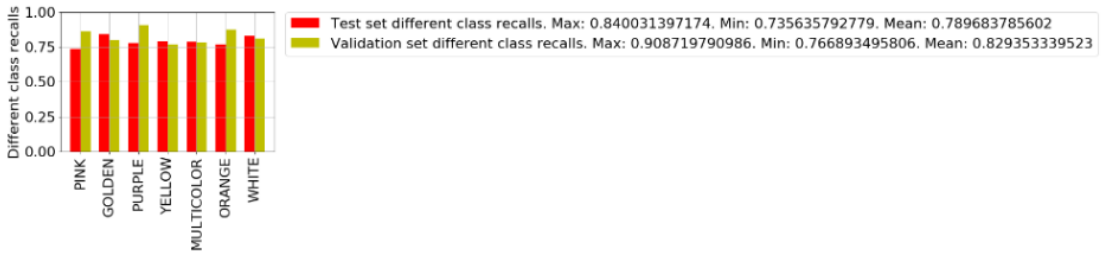

Figure 34. Recall for image classification when camera image and target image are about different leaders.

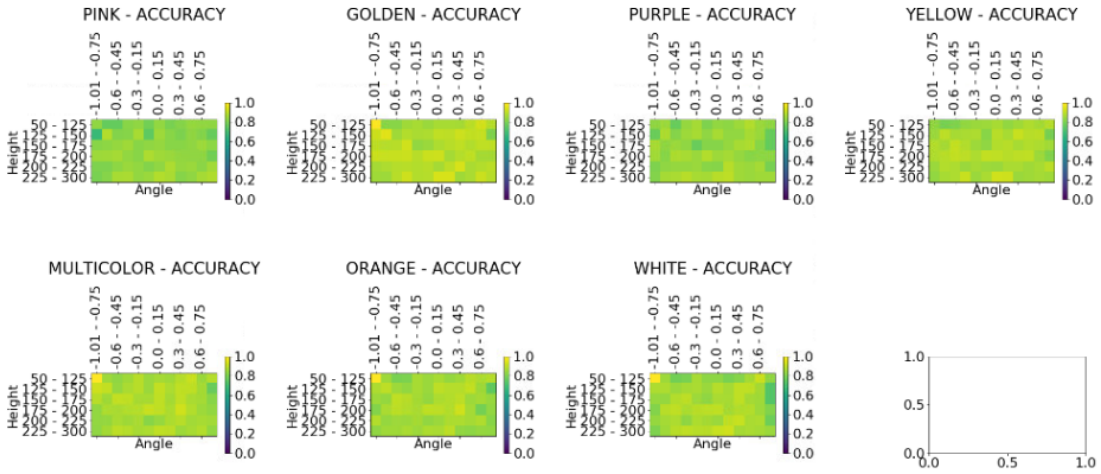

Figure 35. Test set accuracies predicted on different cross-validation models put into bins based on bounding box heights and approximate angles calculate from boundin box locations.

It can be concluded that neural networks are able to learn to match target and camera images with siamese neural networks, but close attention has to be put towards having a diverse dataset.

4.4 Results for predicting velocity and turning with siamese architecture

As it was determined that it is possible to learn velocity and turning commands from the caemra images and it is possible to classify images into matching and non-matching which are the subtasks of the leader-follower system it should be possible to make the system which gives movement commands if the the correct leader is in sight and gives standstill commands if the correct leader is not in sight. For this task a siamese neural network shown in Figure 36 was constructed. It has been constructed using insights from the two previous neural networks. The left part up to “concatenate features” acts as a feature extractor. The ResNet50 layers’ weights were frozen so it was necessary add at least one fully connected layer which would extract the necessary features. It was also necessary that the fully connected layer would be part of the siamese part of the setup because it is essential that the fully connected layer would extract features in the same manner in both the target and camera branches. Adding two fully convolutional layers in the siamese part did not seem to improve the results. The right part of the network acts as the command predictor and here the setup was kept similar as in the neural network which predicted the commands directly from images without the siamese network. For velocity the sigmoid activation was replaced with linear activation as sigmoid sometimes got saturated because there were a lot standstill-commands with value 0 and sigmoid had

trouble backpropagating the errors backwards. Also, separate network branches were created for velocity and turning, since the convergence of one command affected the other and vice versa, so it was more stable to separate the two. It has to be noted that the branches on the right have a similar architecture, but they do not share weights.

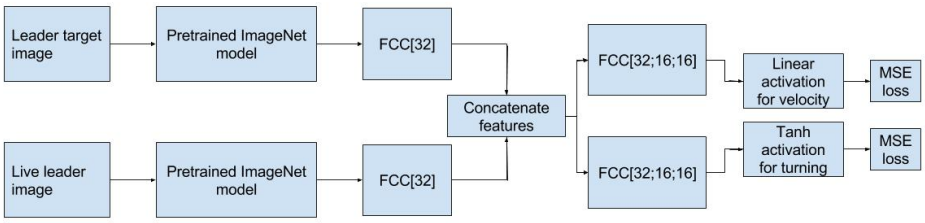

Figure 36. Neural network architecture for leader-follower task which predicts movement commands based on two input images.

It was definitely more complicated to get this neural network to converge, since the additional standstill commands from false image pairings made the command distribution heavily skewed towards 0 for both velocity and turning. Of course it was also more complicated for the neural network to learn to distinguish whether the images were of the same leader or not besides predicting only the movement commands. This resulted in a decreased performance for predicting the movement commands. The explained variance of both commands can be seen on Figure 37 and Figure 38. These are calculated only for the commands where the target and camera image had the same leader on it. Compared to neural network in 4.2 the explained variance decreased from about 0.65 to about 0.5 for turning and from 0.35 to below 0.2 for velocity respectively. Original velocity and turning command distribution based on approximate angle and bounding box height were depicted on Figure 25 and the predicted ones can be seen on Figure 39 and 40. From there we can see the same pattern as with explained variance, meaning that the velocity and turning command predictions follow a similar pattern like the original commands, but the quality of the predictions has slightly decreased compared to the performance in section 4.2.

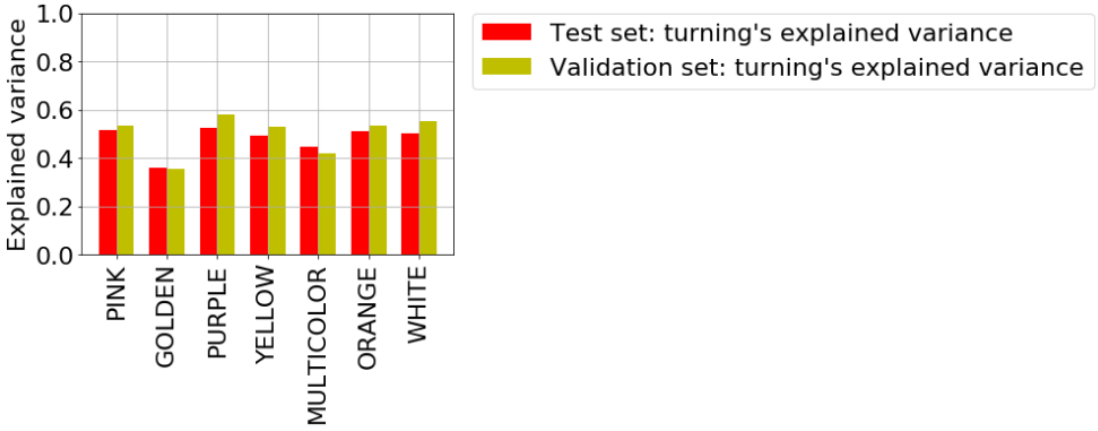

Figure 37. Turning command explained variance for the neural network described in section 4.4, calculated on the predictions made on datapoints where the target and camera image had the same leader on it.

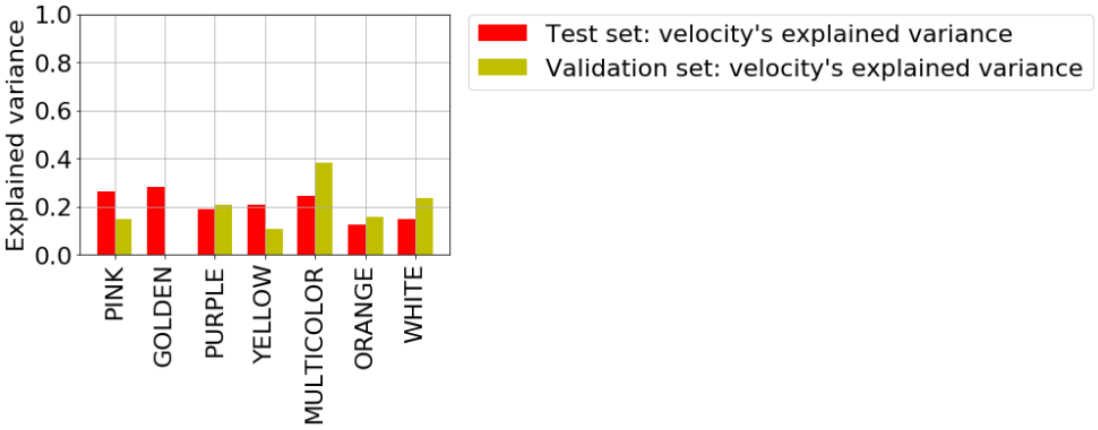

Figure 38. Velocity command explained variance for the neural network described in section 4.4, calculated on the predictions made on data points where the target and camera image had the same leader on it.

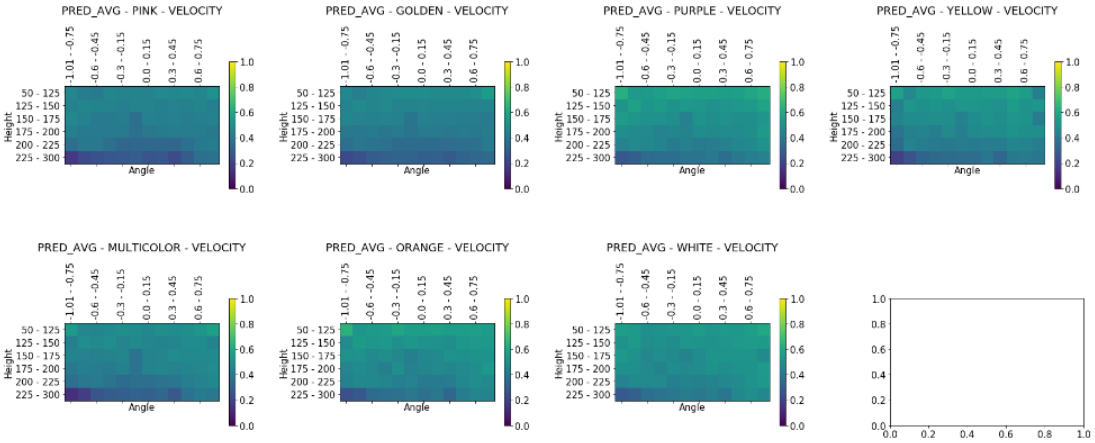

Figure 39. Colors represent predicted velocity based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.4, calculated on the predictions made on data points where the target and camera image had the same leader on it.

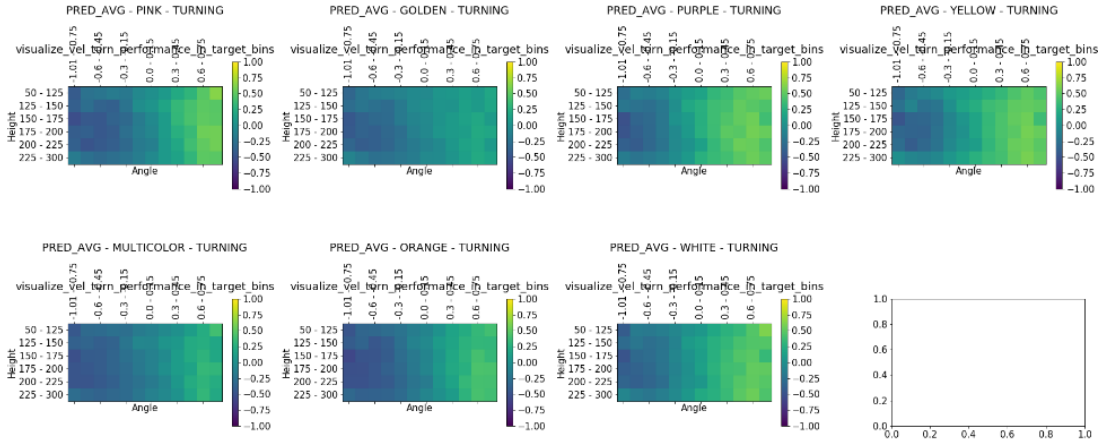

Figure 40. Colors represent predicted turning command based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.4, calculated on the predictions made on data points where the target and camera image had the same leader on it.

The original and predicted command distributions can be seen on Figure 41 and 42 respectively. On Figure 41 those predictions can be seen where the input target and camera image had the same leader on it while in the case of Figure 42 the leaders are different. It can be seen that the neural network has learned to distinguish between the two because the predictions where the leaders are different on the images are skewed towards 0 while the ones where the leaders are the same look like regular predictions similar to predictions by network defined in section 4.2. But the quality of the command predictions is somewhat poor, meaning that the actual system could not rely on it enough for it to be safe, meaning that the robot might be unsure whether the correct leader was in front or not and engage with the movement commands which could cause dangerous situations.

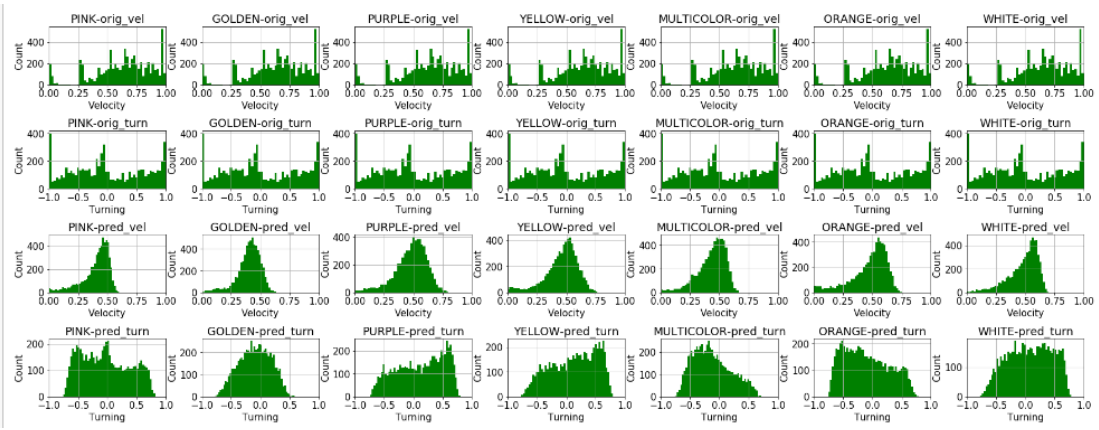

Figure 41. Original and predicted velocity and turning commands on test data for different cross-validation networks. From left to right are different cross-validation networks. The first and second row represents the distribution of original velocity and turning commands respectively if the target and camera image are about the same leader. The third and fourth row represents the distribution of predicted velocity and turning commands respectively if the target and camera image are about the same leader.

The last capability to evaluate is how well the neural network understands whether the target and camera image have different or same leaders on them. Those predictions where the velocity and turning commands are near 0, could mean that the neural network predicted that the images are about different leaders, although it could also mean that it thought that the images are about the same leader, but a standstill command was required since it thought it was too close to the leader. Those predictions where the velocity command value was below 0.25 and the absolute command value for the turning command was below 0.25 were classified as though the network had thought the images were about different leaders which should potentially mean that the recall of classifying them different should at least be high. Unfortunately, as can be seen from Figure 43, that

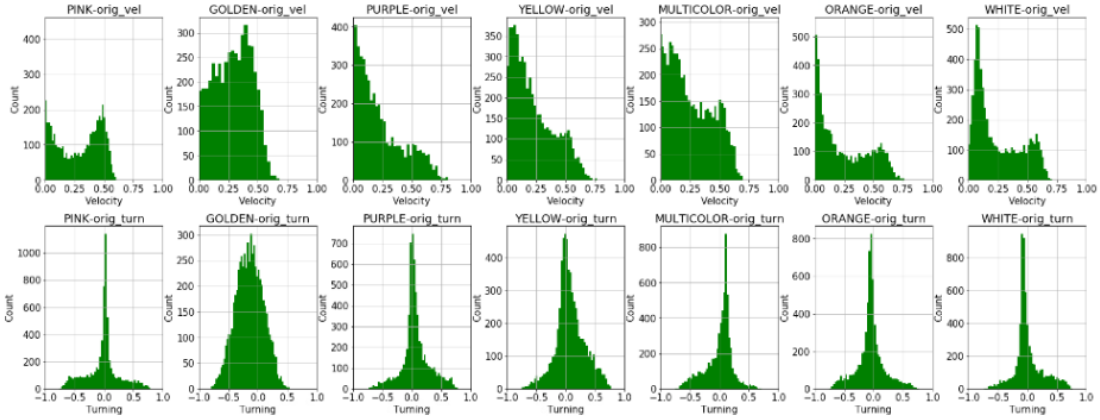

Figure 42. Predicted velocity and turning commands on test data for different cross-validation models if the target and camera image are about the different leaders. On the top row there are velocity predictions and on the bottom row turning predictions.

was not the case as the command predictions are too noisy and it is often the case that even when one command is in the allowable deadzone then the other command is out of it and vice versa.

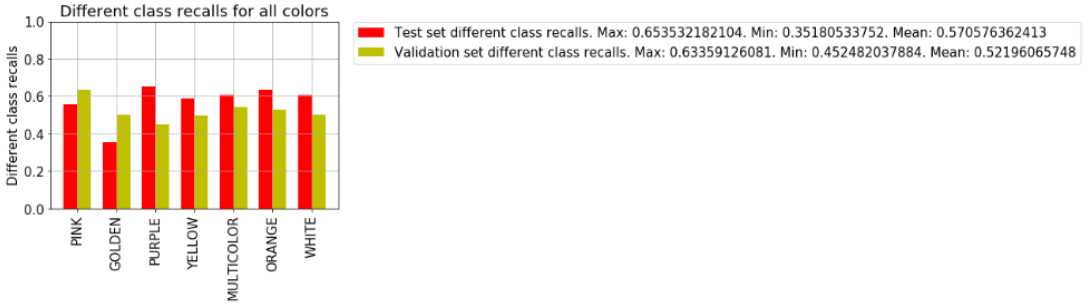

Figure 43. Recall for image classification when camera image and target image are about different leaders with neural network defined in section 4.4.

4.5 Results for predicting velocity, turning and image matching with siamese architecture

Due to the fact that it was difficult to evaluate with the architecture in 4.4 whether the neural network had predicted correctly whether the target and camera image had the same leader on it or not, a new neural network approach was tested which would combine the neural network from 4.3 and 4.4. The architecture can be seen on Figure 44. The

reasoning behind this was twofold. First, to improve the quality of the neural network's feature extractor layers on the left before the concatenation, so that the layers would extract information necessary for distinguishing whether the target and camera images have the same leader on them or not. Second, help to evaluate whether the neural network understood that the target and camera image contained the same leader or not.

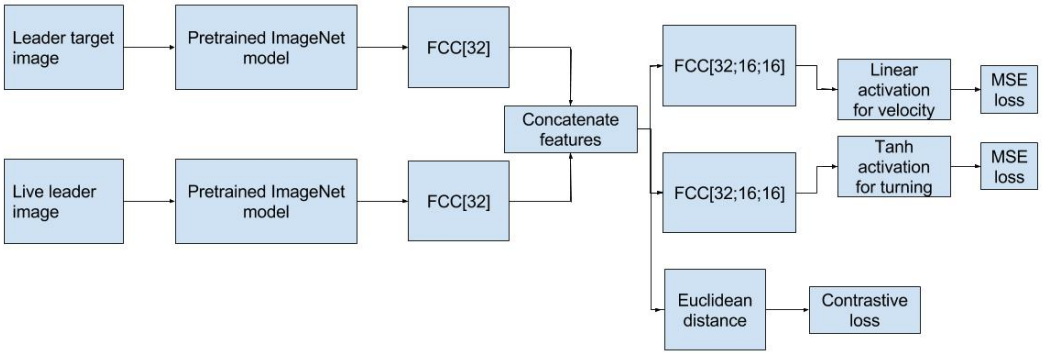

Figure 44. Neural network architecture for leader-follower task which predicts movement commands and whether the images contain the same leader or not based on two input images

This network indeed enabled to predict whether the images contained the same leader or not fairly well, although with decreased performance compared to network defined in 4.3. The accuracy is around 75% which can be seen on Figure 48 while it was around 86% for the architecture described in 4.3.

On the other hand, it decreased the performance of predicting movement commands. The velocity commands which were more complicated to predict before did not get an appropriate signal anymore and the prediction explained variance dropped below 0 which can also be seen from Figure 46 where the velocity predictions do not seem to be correlated with the input at least in a way they should be. The turning command predictions still had a decent explained variance as can be seen on Figure 45, although the performance decreased from 0.65 for the architecture from section 4.2 to around 0.5 for the current architecture. The turning command predictions are still nicely correlated with the location of the leader on the images as can be seen on Figure 47.

Although the velocity command predictions were poor in terms of explained variance, the signal still enabled to distinguish whether the leader was the same on both images or not. This can be seen on figures 49 and 50 where both the velocity and turning commands are skewed towards 0 when the leaders are different on images and have a similar shape to original commands when the leaders on the images are the same. The network could possibly be tuned in a way that all the predictions behave well by modifying the loss proportions.

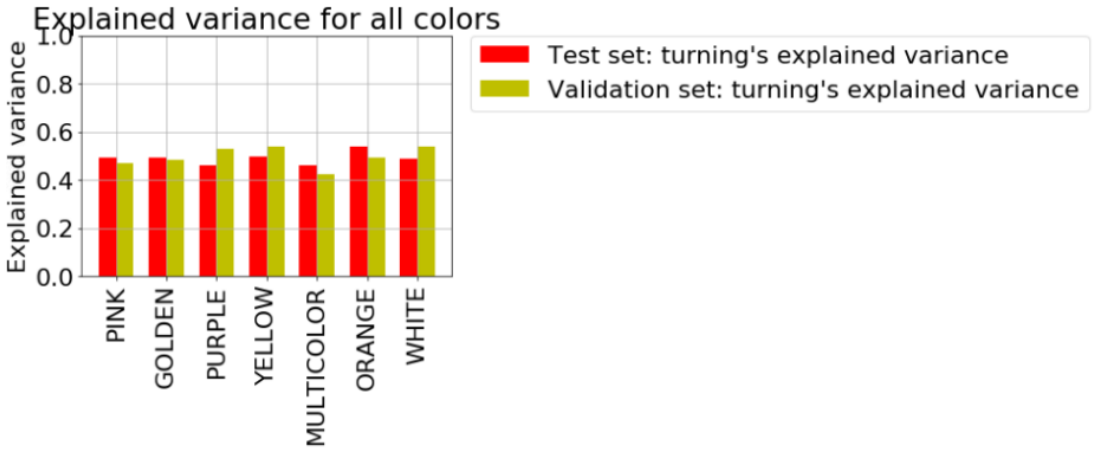

Figure 45. Turning command explained variance

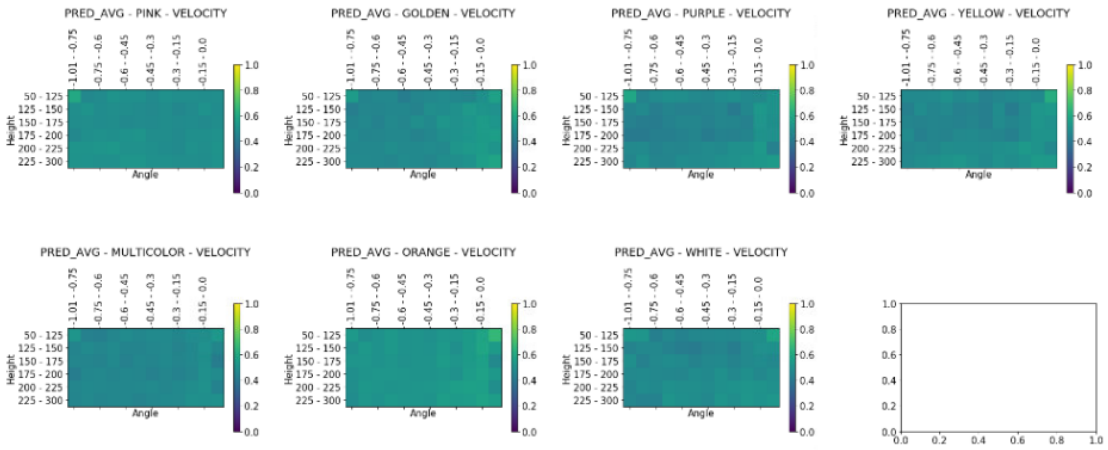

Figure 46. Colors represent predicted velocity based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.5, calculated on the predictions made on data points where the target and camera image had the same leader on it.

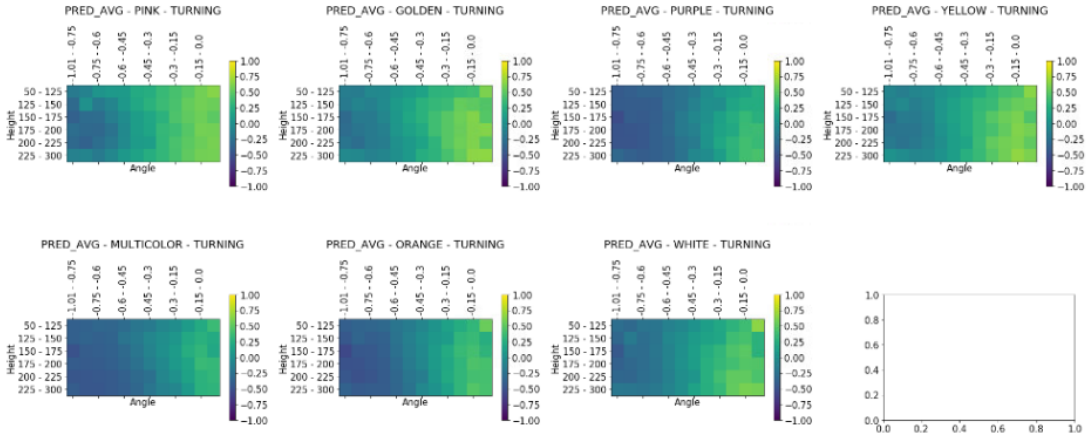

Figure 47. Colors represent predicted turning command based on angle and bounding box height. Predictions made on test data using different models from cross-validation training using architecture described in section 4.5, calculated on the predictions made on data points where the target and camera image had the same leader on it.

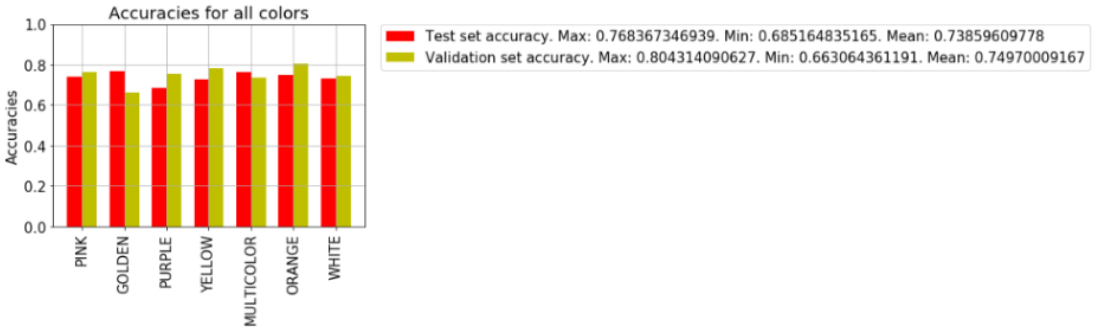

Figure 48. Classification accuracy

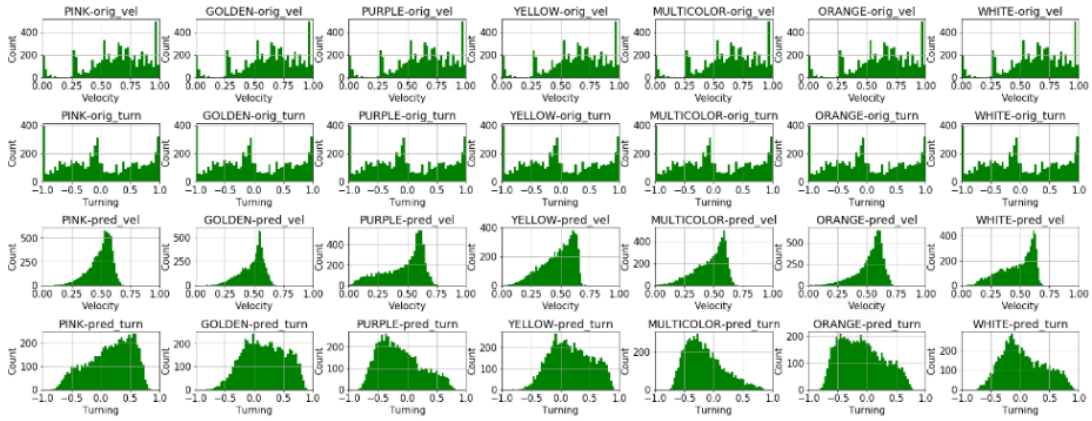

Figure 49. Original and predicted velocity and turning commands on test data for different cross-validation networks. From left to right are different cross-validation networks. The first and second row represents the distribution of original velocity and turning commands respectively if the target and camera image are about the same leader. The third and fourth row represents the distribution of predicted velocity and turning commands respectively if the target and camera image are about the same leader.

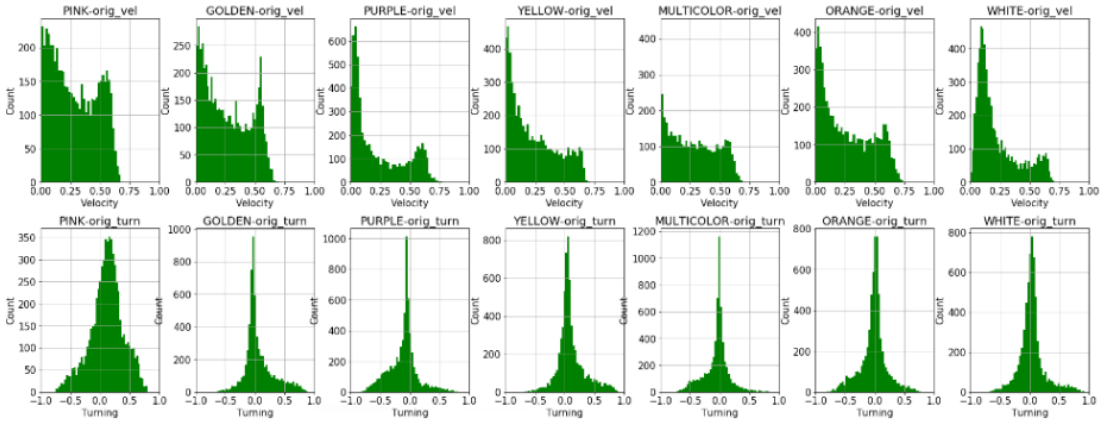

Figure 50. Predicted velocity and turning commands on test data for different cross-validation models if the target and camera image are about the different leaders. On the top row there are velocity predictions and on the bottom row turning predictions.

5 Discussions

This section describes what were the problems encountered during training and how those problems were overcome. An analysis is made on the results and ideas are given for potential future work as multiple approaches can be used to improve the current setup. It also discusses how to solve the leader-follower problem with other approaches like reinforcement learning.

5.1 Problems encountered during training

The initial dataset was quite unevenly distributed which made the velocity command predictions converge near the mean which affected the results significantly. Initially the distribution of data was evened by undersampling based on output movement commands, but better results for distinguishing whether the target and camera image were about the same leader were achieved when the distribution was evened by input image appearance. After the data was evened by input image appearance giving sample weights based on movement command frequencies improved the results for the command predictions.

When it came to image matching then overfitting was a huge problem. At first only initial target images were used in the target branch, but that approach overfitted heavily. When training a siamese neural network then the backpropagation affects both branches of the network, so if the variability is in one branch, it has an effect on that branch which has less variability in the input, but in this case it was more feasible to have the variability in both branches to converge at a reasonable result. Even with extra target images the feature extraction layers after ResNet50 output still required L2 regularization to avoid overfitting.

When predicting the movement commands using siamese neural network with both target and camera image the false pairings needed command values which would make the vehicle stop for which velocity 0.0 and turning 0.0 commands were appropriate. This made the command distributions heavily imbalanced. The velocity command's activation which was previously sigmoid often got saturated at 0 and was changed to linear. The quality of the movement command predictions fell significantly as the network had to do multiple tasks at once - decide whether the two images are about the same leader and what should be the appropriate commands to move the vehicle. The turning command predictions were a lot more robust to these changes, so a bigger weight was put on velocity command (by increasing the weight of velocity loss) to make the backpropagation stronger for velocity command predictions.

Understanding whether the neural network understood if the target and camera image were about the same leader or not was quite difficult with only velocity and turning commands. If the commands were in the deadzone limits which was from 0.0 to 0.25 for velocity and from -0.25 to 0.25 for turning commands then it was classified as if it

had thought that the target and camera image were about different leaders, but it simply could have been a stop command even if there was a match. For this reason an additional euclidean distance output was added which could at least indicate whether the neural network had an appropriate input signal to decide whether the images were about the same leader or not. In this way it could act at least as a failsafe system.

Initially the neural networks used ReLU as activations, but better results and faster convergence was achieved with ELU.

5.2 Analysis of results

Clearly, solving the regression and classification tasks separately had better performance. Turning command explained variance fell from about 0.65 to about 0.5 for those images where the target and camera image were about the same leader. Velocity command explained variance fell from about 0.35 to below 0.2 for those images where the target and camera image were about the same leader. The successfulness of classifying whether the target and camera image are about the same leader or not is harder to define as the command values might mean a mismatch between two images or a standstill in case there is a match. Nevertheless the siamese neural network picked up the signal, although far from perfect which is suitable for a proof of concept. In theory the siamese neural network could have the capability to follow any object, while the network described in 4.2 has the capability to follow only a specific type of object which in this case was a human or with a more diverse dataset a salient object which would make it useless if there was more than one potentially suitable target to follow. Of course the system would have to be tested a lot and additional data would have to be collected to make it both robust and safe, because if the system would make a mistake in a real situation, it could end up in casualties.

When the training process was tried with less colors, the performance of siamese networks decreased significantly, suggesting that a performance boost might be significant in case a more diverse dataset is collected. This was also seen by the need to use extra targets in the target branch of the siamese neural network. A more diverse dataset with a lot more different targets should be collected in order to understand the bottlenecks of the system better.

5.3 Future work

One approach of improving the performance of the movement commands would be to give additional input information to the system. In general the explained variance is rarely equal to 100% because the output signal is often influenced by other parameters as well, besides only the input signal. For example the velocity command which had a lower explained variance was definitely more influenced by other factors compared to turning command. For example knowing additional parameters like vehicle current

speed, battery voltage, IMU data, etc. would give additional information to the system which could improve the performance of the system notably as it is known that their signal has an effect on the commands that have to be forwarded to the robot.

As it was mentioned before, additional data should be collected. At the moment the data collection process was conducted on an existing real robot which made the process extremely time consuming. Using a virtual environment for data collection would speed up the data collection process and enable to easily and safely test out different scenarios. Of course, data collected in the real environment is more realistic and has more variability, but it has been shown it is possible to train a machine learning model in a virtual environment and then fine-tune the model to work in the real world as well [ZMK+16].

Another useful possibility arising from virtual environment besides data collection would also be the possibility of making the task into a reinforcement learning task. In the virtual environment the distance and angle of the leader to be followed would be known real time with great accuracy which would enable the reinforcement learning system to collect dense rewards and thus become smarter over time.

Additional approach would be to build the movement command prediction system on top of tracking neural networks [BVH+16] which generate a heatmap and the command prediction layers could be built on top of it.

6 Conclusion

The aim of this thesis was to test a proof of concept leader follower system. The process of testing it was very challenging and instructive. It started with setting initial targets to test, setting up and conducting the data collection process, followed by thorough analysis of the data, understanding whether the process could be solved with neural networks and implementing the neural networks with what to solve the problem at hand. The results showed that the system could potentially work, but it would take a significant amount of work to make it viably robust and safe in the current setup. A real leader-follower system would use a combination of approaches one of which could potentially be the development of the proof of concept leader follower system brought out in this thesis.

References

[App] Apple. Performing convolution operations. https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html. Accessed: 2017-08-01.

[BVH+16] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking, 2016.

[C+15] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[Cho16] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[HSK+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Kara] Karpathy, Andrej. Cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-1/. Accessed: 2017-08-01.

[Karb] Karpathy, Andrej. Organization of layers. http://cs231n.github.io/neural-networks-1/. Accessed: 2017-08-01.

[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.

[KZS15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[MKR16] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Image patch matching using convolutional descriptors with euclidean distance. In *Asian Conference on Computer Vision*, pages 638–653. Springer, 2016.

[NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[WIJK16] Bichen Wu, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, 2016.

[ZMK+16] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning, 2016.

Appendix

I. Source code

Code repository is available here: https://github.com/kjansons/BehavioralCloningForLeaderFollower

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kristjan Jansons**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Leader-follower System for Unmanned Ground Vehicle

   supervised by Tambet Matiisen

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 16.08.2017