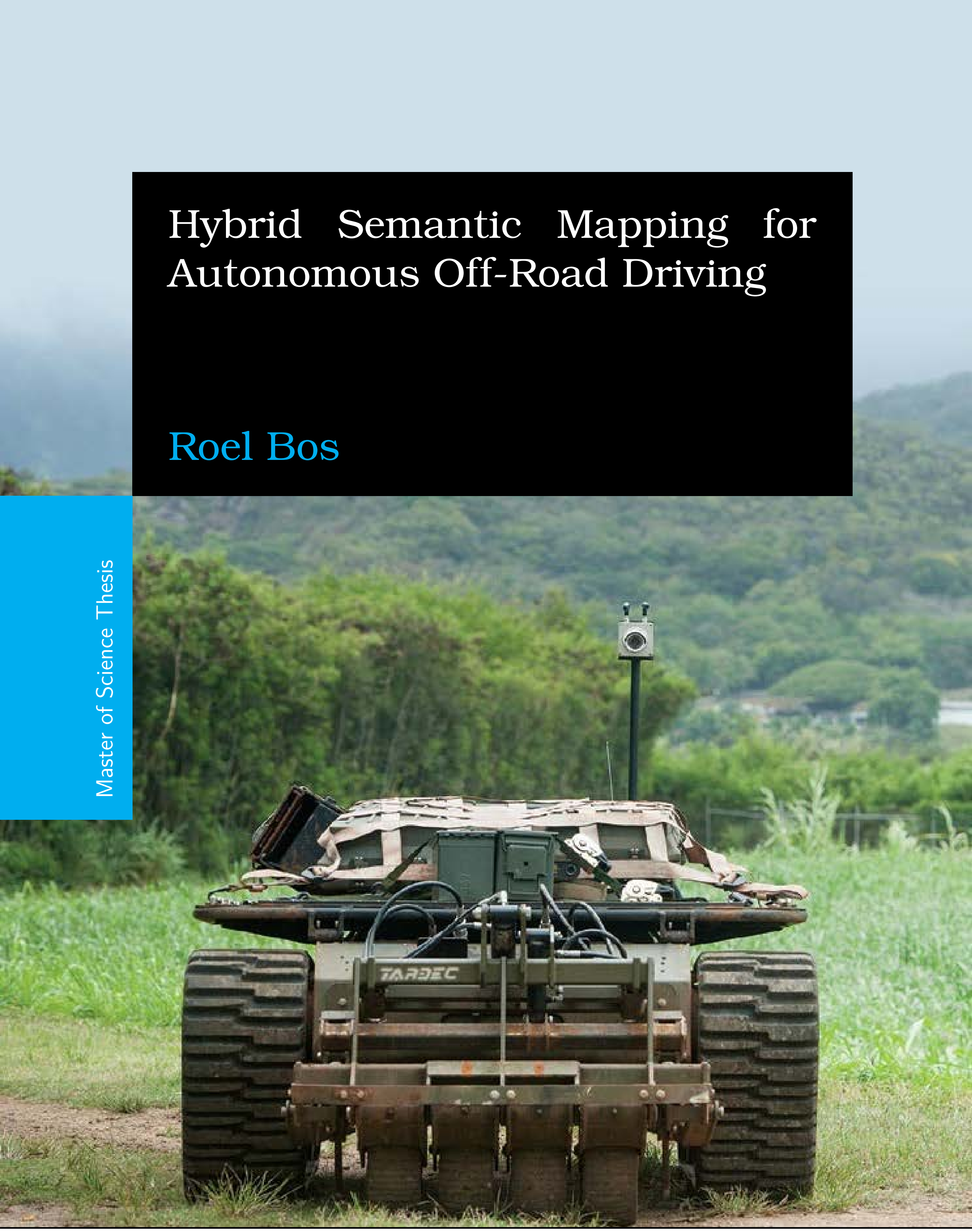
Hybrid Semantic Mapping for Autonomous Off-Road Driving

Roel Bos

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Hybrid Semantic Mapping for Autonomous Off-Road Driving

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Roel Bos

March 2, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by the Netherlands Organisation for Applied Scientific Research (TNO). Their cooperation is hereby gratefully acknowledged.

Front cover images was adapted from: A.C. Turner [55].

Abstract

Unmanned Ground Vehicle (UGV) navigation in unstructured off-road environments can benefit from accurate traversability estimation. Often, experiments with UGVs use semantic segmentation networks for visual scene understanding. Based on the pixel-wise classification of a semantic segmentation network, the UGV can distinguish traversable from non-traversable terrain. However, it is still an open challenge to design a model which is able to accurately estimate traversability in a variety of environments. Variation in terrain characteristics and different levels of structuredness requires a model with a high level of generalisability. Limited generalisability will result in inaccurate traversability estimation, which in the worst-case scenario can cause the UGV to crash.

In order to overcome limited generalisability, a hybrid semantic segmentation framework is presented that can switch between different operation modes. The hybrid framework contains multiple environment-specific segmenters. For each input frame, the hybrid framework selects an environment-specific segmenter, based on a decision parameter. In this work, two hybrid frameworks containing different decision parameters are designed. The first hybrid framework contains multiple Bayesian segmenters, which quantifies prediction uncertainty in addition to the pixel-wise classification. This uncertainty quantification is obtained by Monte Carlo sampling to generate a posterior distribution of pixel class labels. The second hybrid framework consists of multiple environment-specific segmenters and autoencoders. Every segmenter has a corresponding autoencoder trained on the same environmental dataset. The output of the environment-specific autoencoder is a reconstructed image of the input image. The error between the original input image and the reconstructed image is used as a decision parameter for selecting the best performing segmenter.

We experimented with a hybrid segmentation framework and observed that it could outperform a single semantic segmentation network with a 2.6% Intersection over Union (IoU) increase. The hybrid framework with the autoencoder approach resulted in a model selection precision of 99.3% on all the test images. Therefore, we can conclude that UGV navigation can benefit from a hybrid semantic segmentation framework.

Table of Contents

List of Figures

List of Tables

Acknowledgements

The work that lies in front of you marks the end of my time as a student at the TU Delft. I would not have been able to write this thesis without the support and help of many people who I would like to thank in particular.

First of all, I would like to express my gratitude to my supervisors Dylan Kalisvaart, Carlas Smith, Frank ter Haar and Pieter Piscaer, for their time and effort, and for all the insights they provided me with during my thesis. In particular, I would like to thank Dylan for his enthusiastic guidance and discussions during the weekly digital meetings. Secondly, I want to thank Carlas Smith for his guidance of my thesis in general and his feedback. Furthermore, I would like to thank Frank ter Haar and Pieter Piscaer from the Intelligent Imaging department of TNO who gave me the opportunity to expand my knowledge on neural networks and images processing.

Last, but certain not least, I would like to thank my mom, dad, brothers, girlfriend and friends who supported me during my entire time in Delft. Your support has made the whole process fun and exciting. Thank you all!

Delft, University of Technology Roel Bos
March 2, 2022

Chapter 1

Introduction

Autonomous vehicles and systems can significantly impact society. Said autonomous vehicles and systems are being developed with the promise of preventing accidents, reducing energy consumption, transporting the mobility-impaired, and reducing driving-related stress [12]. In addition, autonomous vehicles and systems have the potential to play a crucial role in military operations [59]. For example, an Unmanned Ground Vehicle (UGV) can be designed to aid and complement soldiers [11]. UGVs can assist in tasks that exceed the limits of human endurance [49].

Most importantly, UGVs can reduce casualties by increasing the combat effectiveness of soldiers on the battlefield. Said UGVs can assist in repetitive tasks such as surveillance. Another crucial benefit is that UGVs can replace soldiers in hazardous situations, i.e. target acquisition, mine clearing, and disposal of unexploded ordnance [11].

A UGV can only assist in these military operations when it is able to safely navigate through its environments. In contrast to commercial vehicles, UGV operation is not limited to structured environments. There already exist models that can accurately estimate traversability in these structured urban environments [2, 44, 64, 65]. However, our navigation task exceeds the boundaries of these structured urban environments. The UGV also has to safely navigate through off-road and unstructured environments, which is known to be very complex [31, 56], especially when operating at high speeds [37]. Off-road environments differ from urban environments. They have more diverse obstacle classes, three-dimensional surfaces and no defined road networks or driving rules. In particular, there are obstacle classes and surfaces where the UGV can drive over, such as tall grass or bushes, but these must be distinguished from obstacles and surfaces that the UGV must avoid, such as large boulders or water puddles [56].

Often, safe navigation of autonomous vehicles is obtained by self-contained but inter-connected modules such as perception, localization, planning and control [45]. A modular approach offers the advantage of well-defined sub-tasks and enables engineers to independently make improvements across the whole stack [53]. In this research, we focus on the perception task of the UGV, in which we try to estimate the traversability of the terrain. This task requires cameras that obtain visual information of the UGV's surroundings. The images captured by the camera can be processed by so-called deep semantic segmentation networks, which generates a semantic segmentation mapping.

Semantic segmentation is a visual scene understanding problem formulated as a K-class classification task [63]. This deep learning approach relies on Convolutional Neural Network (CNN) techniques that can extract information out of high dimensional data that contain spatial information [34]. A semantic segmentation network aims to pixel-wise classify an input image into a predefined number of classes. The main benefit of semantic segmentation is situation understanding. Therefore, it is used in a wide variety of applications such as autonomous driving [4, 50], satellite images [35], medical imaging [20], and precision agriculture [30] as a first step to achieve visual perception. The generalization of semantic segmentation networks relies on the size, annotation richness, scene complexity and variability captured in the pixel-wise labelled training datasets.

Recent efforts in annotating data for autonomous driving have resulted in a variety of datasets captured in urban, forest and off-road environments [9, 10, 39, 56]. These datasets differ in class distribution, class characteristics and level of structuredness. Our semantic segmentation model should thus be able to deal with heterogeneous classes in order to approach real-world complexity. Therefore, the focus in this thesis is on a semantic segmentation model that can deal with the different levels of structuredness and characteristics captured in these diverse environment-specific segmentation datasets.

1-1 Research objective

In the presented context, the problem is formulated that the current semantic segmentation networks for autonomous driving vehicles focus on traversability estimation for a specific environment. However, during military operations, a UGV can cross varying environments from structured urban environments to more complex unstructured off-road environments. Therefore, the research question of this thesis is formulated as

How can a hybrid framework improve traversability estimation in unseen unstructured off-road environments for unmanned ground vehicles?

To answer this question, we first need to define and assess methods for traversability estimation. Hence, the first sub-question to the research question is formulated as

1. *In what ways can unseen unstructured off-road environments be classified?*

The hybrid semantic segmentation approach requires a decision parameter. For both the hybrid frameworks, we quantify the uncertainty of a segmenter its prediction. During operation. we compare the uncertainty of the segmenters for the current input frame. The segmenter with the lowest corresponding uncertainty value is selected for traversability estimation. These decision parameters are evaluated on a pixel-level and image level to see what these values represent. Hence, the second sub-question to answer the research question is formulated as

2. *How can uncertainty of a semantic segmentation prediction be quantified?*

1-2 Thesis contribution

This research aims to compare the semantic segmentation performance of a hybrid semantic segmentation framework with a single semantic segmentation network. We design two different hybrid frameworks. Hybrid framework 1 contains multiple environment-specific semantic segmentation network, which outputs in addition to the pixel-wise classification a measure of model uncertainty. This model uncertainty is obtained by Monte Carlo sampling with dropout at test time to generate a posterior distribution of pixel class labels. This uncertainty measure will perform as the decision parameter in hybrid framework 1, which selects the best performing segmenter. Hybrid framework 2 uses the reconstruction error of an autoencoder instead of the Bayesian uncertainty measure to select the best performing segmenter. Ideally, we capture the environment-specific features of the training datasets in the autoencoder its weights. This will result in environment-related differences in the reconstructed output images of the environment-specific autoencoders. Moreover, the datasets used in this research are captured in an urban, a forest and an off-road environment. Hence, this study focuses on the semantic segmentation performance of the different environmental segmenters in these diverse environments. We offer insights into the best performing hybrid framework based on semantic segmentation performance, model selection performance and uncertainty estimation.

1-3 Thesis outline

This thesis comprises six more chapters, which are structured as follows. Chapter 2 will describe the theory behind Artificial Neural Network (ANN) and CNN, such that the fundamentals of semantic segmentation networks are understood. Furthermore, we describe Bayesian Neural Network (BNN) and how to train these different neural networks. In Chapter 3, we present Bayesian SegNet and describe autoencoder networks. Additionally, we introduce the semantic segmentation datasets used in the experiments performed in this thesis. In Chapter 4, the designed hybrid frameworks 1 and 2 are explained together with the preprocessing step of the images and evaluation metrics used. Answers to the research questions are obtained by performing different experiments in Chapter 5. The results of these experiments are shown to the reader. In Chapter 6, the thesis is concluded with a summary of the main finding. Furthermore, recommendations for improving this research and future work are presented. Lastly, additional results, neural network architectures and underlying data can be found in the Appendices A, B and C.

Chapter 2

Theory

In this chapter, the fundamentals of deep learning are explained. Some common architectures as Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Bayesian Neural Network (BNN) are discussed. We show the reader the different layers and activation functions for neural networks. Neural networks are trained using the backpropagation algorithm which is explained in the last section of this chapter.

2-1 Artificial Neural Networks

An ANN (*or a neural network*) is inspired by the biological neural system. A neural network consists of an input, a hidden, and an output layer. Every layer consists of multiple perceptrons. A neural network is also known as a multilayer perceptron.

The neural network architecture determines the approximation properties of the function. In a 1989 paper [13], mathematician George Cybenko proved the universal approximation theorem: an ANN with one hidden layer can accurately approximate any continuous function f as long as the number of perceptrons per layer is large enough.

Cybenkos Theorem Let σ be any continuous discriminatory function. Then finite sums of the form:

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{2-1}$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum $G(x)$, of the above form, for which:

$$|G(x) - f(x)| < \epsilon \qquad \forall x \in I_n \tag{2-2}$$

Therefore given a desired function $f(x)$, which is to be computed with certain accuracy $\epsilon > 0$, Cybenkos Theorem states than when using enough hidden neurons, there is always a network with output $G(x)$ which satisfies $|G(x) - f(x)| < \epsilon$ [13].

2-1-1 Single perceptron

As described above, a neural network consists of multiple interconnected perceptrons. A single perceptron is inspired by how a single neuron functions in our biological neural system. The schematic overview of a single perceptron is shown in Figure 2-1.

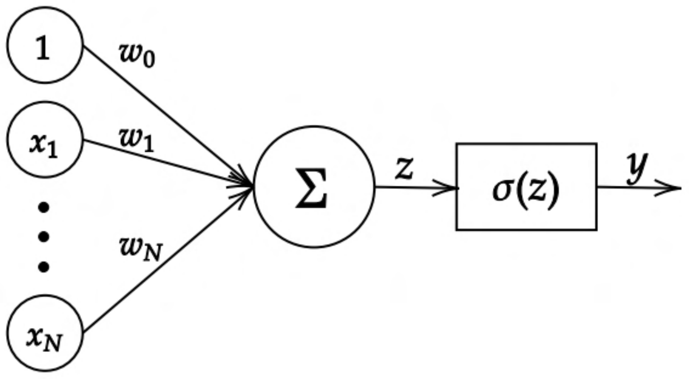


Figure 2-1: Representation of a single perceptron with input x_i, weights w_i and activation function σ and output y.

A neuron receives multiple inputs x_i, which are transported over the model weights w_i and are eventually summed up (Σ). When this sum z reaches a certain threshold, the neuron is able to produce an output. The output production rate of a neuron is modeled by an non-linear activation function σ. The activation of a neuron is modeled as shown in Equation 2-3. This equation models decision-making, by varying the weights and activation function the output is affected.

$$y_i = \sigma(\Sigma_i(w_i x_i + w_0)) \tag{2-3}$$

y_i represents the activation of a neuron, $\sigma(z)$ is the non-linear activation function, w_i is the network weights, x_i is the neuron input and w_0 is the neuron bias. Popular activation function in neural networks are modeled the sigmoid, hyperbolic tangent or Rectified Linear Unit (ReLU) function, which are further described in Section 2-2-3.

2-1-2 Multilayer perceptron

A multilayer perceptron (*neural network*) is created by interconnecting multiple single perceptrons in layers. The output of a single perceptron of a particular layer is the input of the consecutive layer. These networks are called feedforward neural networks. A neural network consists of an input layer, hidden layers, and an output layer as shown in Figure 2-2. The amount of hidden layers scale with the depth and complexity of a neural network. In the first layers of a neural network, relatively simple decisions are made. Layers placed deeper in the network make more complex decisions based on the previous layers' input. The term deep in deep learning refers to a neural network containing multiple hidden layers, a deep neural network.

These neural networks are designed to map input data x to output data y. This mapping is learned from training data. The learning process consists of updating the weights w_i from Equation 2-3. The process of updating and adjusting the weights is called backpropagation, which is described in Section 2-4.

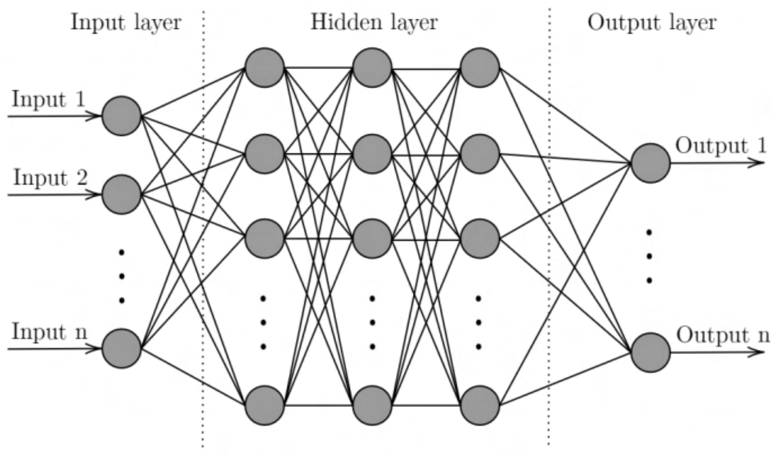


Figure 2-2: A neural network with an input layer, three hidden layers and an output layer. Image adapted from: [8].

2-2 Convolutional Neural Networks

A CNN is a special type of neural network that is designed to extract information out of high dimensional data that contains spatial information such as images. An advantage of a CNN is that the number of parameters needed is invariant to the size of the input image [33]. The spatial information of neighbouring pixels in the image is captured by so-called kernels in a convolutional layer. The neurons in convolutional layer are arranged in three dimensions, width w, height h and depth d. Depth refers to the depth of the input image, which is three for Red-Green-Blue (RGB) images. Deeper in the network, the depth refers to the number of feature maps of the convolutional layer. The neurons in a CNN are only connected to a small region of neurons in the layer before. This strongly decreases the number of weights compared to a regular ANN.

A CNN consist of convolutional layers, activation functions and pooling layers. A CNN architecture that is designed for image classification is shown in Figure 2-3. This image classification networks consist of a feature learning part and a classification part. Different computer vision tasks can be solved by replacing the classification part with another end network. Other applications that also contain a feature learning part are, for example, segmentation and object detection networks. The different layers in the feature learning part are further elaborated on in this section.

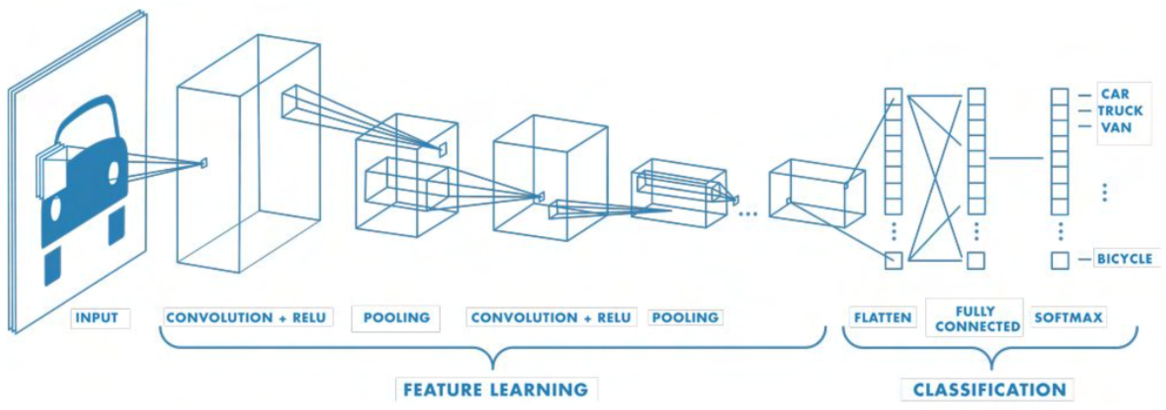


Figure 2-3: CNN architecture for image classification. The input image is processed through the convolutional neural network containing convolutional layers, non-linear activation functions and pooling layers. After the feature learning part, the classification of the input image is performed. This is obtained by a fully connected layer and the softmax function. The softmax function converts the information of the feature learning part into a probability distribution over the predicted output classes. Image adapted from: [38]

2-2-1 Convolutional layer

In a CNN the convolutional layer is the first and most fundamental layer. The parameters of the convolutional layers are stored in kernels (or learnable filters), which have a small receptive field and apply a convolution operation to the input [34]. The first convolutional layer extracts simple features from the input images, such as edges and lines. The result of this convolution is then passed on to the next layer.

Images exist out of pixels that contain a pixel value. RGB images can thus be represented by matrices with size $h \times w \times d$, where h is the image height, w the width and d the colour channels. Let K be a kernel with x rows, y columns and depth d. Then a kernel with size $(K_x \times K_y \times d)$ works on a receptive field $(K_x \times K_y)$ of the input image, where $K_x < h$ and $K_y < w$. The kernel slides over (convolves) the image, producing a feature map, which is shown in Figure 2-4. Convolution is the sum of the element-wise multiplication of the kernel and the image.

The size of the kernel is a parameter that can be set in the design phase of the CNN. Besides this parameter, one also has to define the stride s of the convolution operation. The stride represents the number of pixels the kernels shift over the image. The stride influences the output size of the convolution operation. Larger strides number results in smaller output size. The relation between the input image I and output size O after convolution with kernel K and stride s is given in Equation 2-4.

$$\begin{aligned} O_x &= \frac{I_x - K_x}{s} + 1 \\ O_y &= \frac{I_y - K_y}{s} + 1 \end{aligned} \tag{2-4}$$

Each convolutional layer contains a non-linear activation function, which in this case is the ReLU non-linearity. This activation function performs an element-wise operation to every value in the activation map. ReLU and other non-linear activation functions are described in Section 2-2-3. After the convolutional layer with the ReLU activation function, we observe a pooling layer in Figure 2-3.

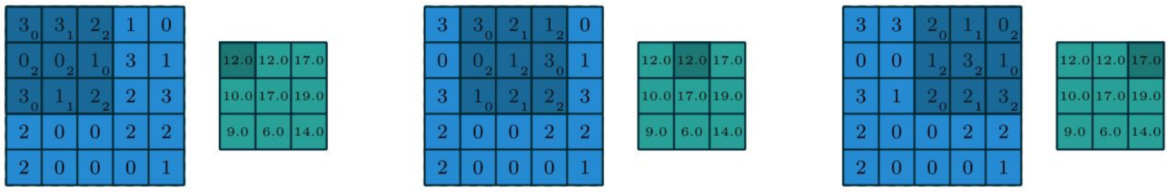


Figure 2-4: The convolutional layer operation. The input image matrix shown in light blue with size $h \times w$. The kernel K in dark blue with size $K_x \times K_y$. The kernel slides over the input image with stride $s = 1$. The values in the green matrix represent the activation map. Image adapted from [17].

2-2-2 Pooling layer

The pooling layers in CNN architectures reduce the spatial size of the feature map [23] and are also known as downsampling layers. Decreasing spatial size is crucial in achieving a more efficient training process. Commonly used CNN pooling techniques are AveragePooling and MaxPooling. AveragePooling takes the average score in the pooling window. MaxPooling selects the maximum score of the pooling window, as can be seen in Figure 2-5.

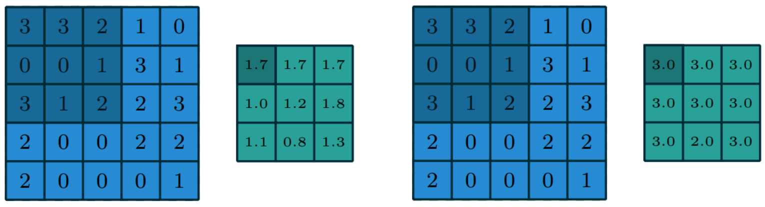


Figure 2-5: The pooling layer operation. The values in light blue is the input image matrix. The dark bleu matrix represents the receptive field. The output values are shown in green. On the left side we show an example of AveragePooling and on the right side we show a MaxPooling operation. Image adapted from [17].

2-2-3 Non-linear activation functions

As already mentioned in the previous sections, neural networks have non-linear activation functions. Popular activation functions are the hyperbolic tangent and the sigmoid function, given in Equation 2-5 and Equation 2-6 respectively.

$$\sigma(z) = \tanh(z) \qquad \in (-1, 1) \tag{2-5}$$

$$\sigma(z) = \frac{1}{1 + \exp^{-z}} \qquad \in (0, 1) \tag{2-6}$$

Another interesting activation function is the Rectified Linear Unit (ReLU). ReLU has as advantage that it speeds up the convergence of the training process, as it does not contain computational expensive exponentials. The ReLU activation function is given by:

$$\sigma(z) = \max(0, z) \qquad \in [0, \infty) \tag{2-7}$$

2-2-4 Dropout layer

Overfitting is a problem that arises in neural networks during the training phase. The model's fit on the training data determines the generalizability capacity of the network. Overfitting occurs when a model learns the training data including the noise to such a great extent that it has failed to capture the underlying general information. This will result in a network with bad generalizability performance on unseen data. CNNs are sensitive to overfitting since the networks often have a large number of network weights and small-sized training datasets.

In order to overcome the problem of overfitting, dropout layers can be implemented during the training phase. Dropout layers randomly set network weights to zero, which constrains the network adaptation to the training data. Consequently, it prevents the weights from being overfitted on the training data. A schematic overview of a neural network with and without dropout layers is shown in Figure 2-6. The implementation of dropout layers will eventually result in a decreased difference between training and validation data performance. Dropout layers are only implemented during the training phase to prevent overfitting. Gal and Ghahramani use dropout layers during inference to estimate the uncertainty of the model its prediction [19].

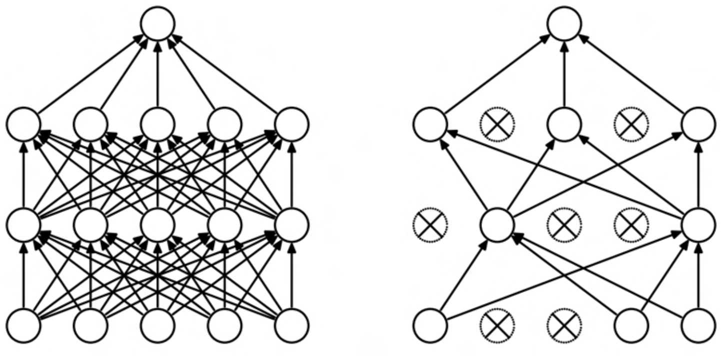


Figure 2-6: A schematic overview of a neural network before and after applying dropout. Image adapted from [52].

2-2-5 Deconvolutional layer

The image classification task shown in Figure 2-3 can be modified to an object detection or a semantic segmentation task. For both these tasks the network needs to output an image. In object detection, the aim is to locate a predefined object in the input image and encircle it. Semantic segmentation is the pixel-wise labelling of an image into a predefined number of classes.

This process is obtained by replacing the classification part with a deconvolution network. A deconvolution network consists of deconvolutional and unpooling layers. These layers perform the opposite operation of the pooling and convolutional layers, as shown in Figure 2-7. In the pooling layer, the location of the maximum activation value can be stored in switch variables. These values can be restored in the unpooling layer [42]. The output of an unpooling layer is sparse, as it is an enlarged version of the input map. The deconvolutional layer produces a dense output map from the unpooling layer output. In Chapter 3 we further elaborate on deconvolutional network in CNNs.

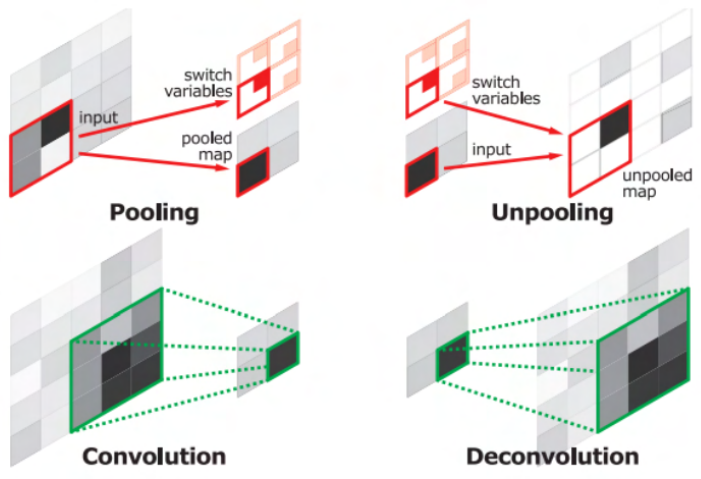


Figure 2-7: Schematic representation of pooling, unpooling, convolution and deconvolution operations. Image adapted from [42].

2-2-6 Fully connected layer

The CNN architecture, as shown in Figure 2-3, ends with a classification part. In this classification part, the features obtained in the feature learning part are processed through a flattening layer, fully connected layer and a softmax function. The network flattens the three-dimensional output feature map to a one-dimensional array by a vectorisation. The fully connected layer connects every input of the one-dimensional array to output values by the learnable weights. To summarise, the network extracts the features using the convolutional and pooling layers. Afterwards, the fully connected layer maps the extracted features to model the final estimate, the probability for each class. The softmax function is a commonly used output function for classification tasks, which predicts probabilities for each class. The sum of the probabilities over all classes adds up to one. The class with the highest probability is considered the neural network's output. The softmax function is given by:

$$\begin{aligned} \hat{\boldsymbol{y}}_{\boldsymbol{i}}(\boldsymbol{x}_i, \Theta) = \sigma(a_k) &= \frac{e^{a_k}}{\Sigma_{j=1}^{K} e^{a_j}} \\ p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \Theta\right) &= \sigma(a_k) \end{aligned} \tag{2-8}$$

here $k \in K$ represents the number of classes, a_k the input of the softmax function and Θ the network parameters.

2-3 Bayesian Neural Networks

The above described neural networks operate as black boxes, making it hard to understand the model's reasoning. It lacks explainability [60]. The model might also generalise overconfident in situations it has not seen before [26, 41]. This property, in addition to the inability of a neural network to output "I do not know", can cause dangerous situations, especially when autonomous vehicles decisions rely on these models. Incorporating Bayesian statistics offers a method to understand and quantify the uncertainty associated with the neural network's prediction. This section explains how uncertainty can be quantified in deep learning models.

Neural networks that model uncertainty are known as BNN. They offer a probabilistic interpretation of deep learning models by inferring distributions over the network parameters as shown in Figure 2-8. We define a BNN by placing a prior distribution over the network parameters. The prior information is incorporated in the posterior distribution by using Bayes rule. The goal is to obtain the predictive distribution, which contains the variance and uncovers the uncertainty associated with the prediction.

For simplicity, we assume Θ represent the network parameters, $\mathbf{X}$ the input data and $\mathbf{Y}$ the desired output which is captured in training dataset $\mathcal{D}$. A neural network can be written as a probabilistic model, $p(y_i|x_i, \Theta)$ for given input $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$ from the training data. The likelihood, $p(\mathbf{Y}|\mathbf{X}, \Theta)$, is the distribution for the target values given inputs $\mathbf{X}$ and Θ. Bayes rule is used to compute the posterior distribution $p(\Theta|\mathbf{X}, \mathbf{Y})$ with the likelihood $p(\mathbf{Y}|\mathbf{X}, \Theta)$ and prior for the network parameters $p(\Theta)$ by:

$$p(\Theta|\mathbf{X}, \mathbf{Y}) = \frac{p(\Theta)p(\mathbf{Y}|\mathbf{X}, \Theta)}{p(\mathbf{Y}|\mathbf{X})} \tag{2-9}$$

The objective of a BNN is to include parameter uncertainty in the predicted output y_i for the unseen data x_i:

$$p\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \mathcal{D}\right) = \int p\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \Theta\right) p\left(\Theta \mid \mathcal{D}\right) d\Theta \tag{2-10}$$

This calculates the predictive distribution by integrating over all possible model configurations for Θ. This is similar to using an ensemble of an infinite number of neural networks, which unfortunately is computationally impossible. To obtain the posterior distribution, we consider variational methods. Variational methods can approximate the intractable distribution using an approximating variational inferences technique to estimate the unknown distribution of the weights in Equation 2-10.

Gal and Ghahramani [19] introduced a framework that uses dropout as approximate Bayesian inference over the network weights. Their framework quantifies model uncertainty without requiring any additional model parameters. This is achieved by Monte Carlo sampling with dropout at test time.

2-3-1 Monte Carlo dropout

A popular way to estimate uncertainty is by inferring predictive distributions $p(\boldsymbol{y}_i|\boldsymbol{x}_i, \mathcal{D})$ with BNNs. We can obtain the variance from this predictive distribution and uncover the uncertainty. Gall and Ghahramani designed a Monte Carlo dropout technique which provides a scalable way to learn a predictive distribution. This is obtained by dropout layers, as described in Section 2-2-4, at test time.

First, we define the Bayesian classification neural network. We want to find the network parameters Θ of the function $y = f_\Theta(x)$. Using the Bayesian approach, we define a prior

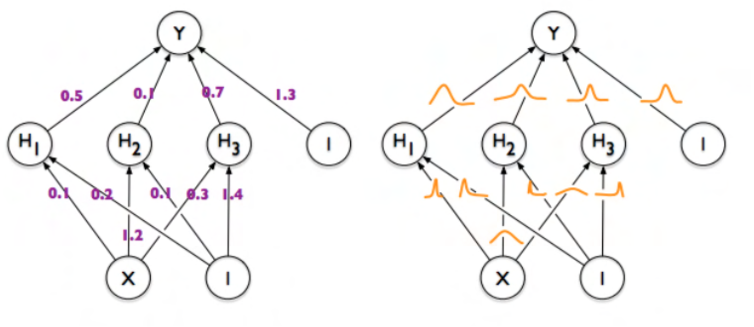


Figure 2-8: A standard neural network design where each weight is represented by a fixed value. And a BNN with a probability distribution over the network weights. Image adapted from: [5]

distribution over the network parameters before observing any data points. We place a standard matrix Gaussian prior distribution over the model parameters $\Theta = \{\Theta^{(l)}\}_{l=1}^{L}$:

$$\Theta^{(l)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{2-11}$$

The likelihood distribution for classification tasks is defined by the softmax likelihood as defined in Equation 2-8. We want to obtain the predictive distribution:

$$\begin{aligned} p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \mathcal{D}\right) &= \int p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \Theta\right) p\left(\Theta \mid \mathcal{D}\right) d\Theta, \\ &\approx \int p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \Theta\right) q_\theta^*(\Theta) d\Theta. \end{aligned} \tag{2-12}$$

The posterior distribution $p\left(\Theta \mid \mathcal{D}\right)$ is intractable, therefore, we need to approximate the distribution of these network parameters [16]. We can use variational inference to approximate it [24]. This technique allows us to learn the distribution over the network parameters, $q_\theta(\Theta)$, by minimising the Kullback-Leibner (KL) divergence between this approximating distribution and the full posterior:

$$\mathbf{KL}\left(q_\theta(\Theta) \mid\mid p(\Theta|\boldsymbol{X}, \boldsymbol{Y})\right). \tag{2-13}$$

We define the approximate variational distribution $q_\theta(\Theta^{(l)})$ for every layer l of the network as:

$$\begin{aligned} \Theta_i^{(l)} &= M_i^{(l)} \cdot \operatorname{diag}([z_j^{(l)}]_{j=1}^{K^{(l)}}), \\ z_j^{(l)} &\sim \operatorname{Bernoulli}(p^{(l)}) \end{aligned} \tag{2-14}$$

for the number of hidden layers, $l = 1, ..., L$ with hidden units, $j = 1, ..., K^{(l-1)}$. Where $z_j^{(l)}$ are random Bernoulli distributed variables with probability $p^{(l)}$ for each layer l and hidden unit j. The variational parameters vectors in $\theta = M^{(l)}, p^{(l)}$, could be optimised for, such that for θ^* the distribution $q_\theta^*(\Theta)$ is as close as possible to the true distribution.

Finally, we can estimate the predictive distribution in Equation 2-12 with a Monte Carlo integration to sample from the posterior,

$$\begin{aligned} p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \mathcal{D}\right) &\approx \int p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \Theta\right) q_\theta^*(\Theta) d\Theta, \\ &\approx \frac{1}{T} \sum_{t=1}^{T} p\left(\hat{\boldsymbol{y}}_i = k \mid \boldsymbol{x}_i, \hat{\Theta}_t\right), \quad \text{s.t.} \hat{\Theta}_t \sim q_\theta^*(\Theta) \end{aligned} \tag{2-15}$$

Here T represent the total number of dropout samples, T times a stochastic forward pass through the deep network. We illustrate the Monte Carlo dropout in Figure 2-9.

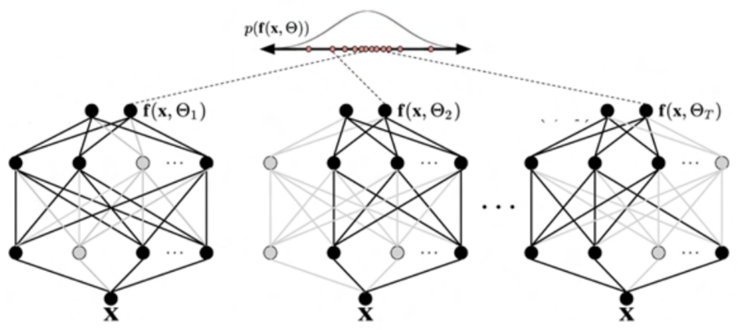

$p(\mathbf{f}(\mathbf{x},\Theta))$
$\mathbf{f}(\mathbf{x},\Theta_1)$
$\mathbf{f}(\mathbf{x},\Theta_2)$
$\mathbf{f}(\mathbf{x},\Theta_T)$
$\mathbf{x}$
$\mathbf{x}$
$\mathbf{x}$

Figure 2-9: Schematic overview of Monte Carlo dropout. In each forward pass neurons are switched off shown in grey, the prediction then relies on the neurons shown in black. The predictive distribution is sampled from T thinned networks. Image adapted from: [1]

2-4 Backpropagation algorithm

The weights or kernel parameters of a neural network are adjusted and updated by the backpropagation algorithm. This algorithm optimises the network weights for the given training data. Backpropagation is an efficient method for computing the gradients required to perform gradient-based optimisation of the weights in a neural network. The weight are optimized for a defined loss function L, different loss function are defined in Section 2-4-1. This optimisation method requires the computation of the gradient of the loss function at each iteration. Therefore, the loss function should be both continuous and differentiable.

A gradient descent algorithm updates the weights of a neural network. This gradient descent algorithm is used by the backpropagation algorithm, which consists of several steps:

1. *Initialisation:* the neural network weights are initialised at small random values.
2. *The forward pass:* a data sample is given to the neural network, an output $\hat{y}$ is returned, and the value of the loss function is calculated.
3. *The backward pass:* the gradient of the loss function to all weight is calculated. This is straightforward for the last layer, the hidden layer to the output layer. However, for deeper layers, the chain-rule is needed to calculate the gradient:
$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial v_j}\frac{\partial v_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} \tag{2-16}$$
4. *Update:* the weights are updated with a gradient descent step.

During training of neural networks, a mini-batch gradient descent algorithm is often proposed, since calculating the gradient for all data takes too long. In literature, it is often referred to as the stochastic gradient descent algorithm [23]. Stochastic gradient descent is a common way to optimize the weights of a neural network.

2-4-1 Loss function

The loss function is an essential aspect of training a neural network. This function measures the error between the output $\hat{y}$ of a neuron and the desired output value y. The two loss functions considered in this thesis are the Mean Squared Error (MSE) and the cross-entropy loss.

The MSE loss function is a simple quadratic loss function that calculates the average of the squared difference between the predicted value $\hat{y}$ and the ground truth value y. The function is given:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2-17}$$

The cross entropy cost function is commonly used in convolutional neural networks that are designed to do a classification task. The function is calculated as the average cross-entropy across all data samples and is defined by:

$$L = \frac{1}{N}\sum_{i=1}^{N}[\hat{y}_i \ln y_i + (1 - \hat{y}_i)\ln(1 - y_i)] \tag{2-18}$$

For N data points where y_i is the truth value in classification task either 0 or 1 and $\hat{y}_i$ is the probability of the i^{th} data point.

2-4-2 Stochastic gradient descent

The gradient descent method is a first-order iterative algorithm that uses the negative gradient of the loss function to update the network parameters. This update rule tries to minimise a loss function iteratively. The iteration process is built upon the following function:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla L\left(\mathbf{w}_k\right) \tag{2-19}$$

where $\mathbf{w}_k$ represents the iteration point at iteration step k, $\nabla L(\mathbf{w}_k)$ the gradient of the differentiable loss function L, and α_k the learning rate. The algorithms start from a predefined initial point $\mathbf{w}_0$.

In standard gradient descent methods, the loss function is the sum of all the N data points $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$ used for training the neural network:

$$\nabla L\left(\mathbf{w}_k\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L\left(\mathbf{w}_k, \mathbf{x}_i, \mathbf{y}_i\right)}{\partial \mathbf{w}_k} \tag{2-20}$$

The larger the dataset $\mathcal{D}$, the more time-consuming optimising the weights becomes. To improve computation time, methods that apply an approximation of the gradient are often used [7]. These so-called Stochastic gradient descent (SGD) methods use only one training sample at each iteration instead of all the datapoints N in the data set. The gradient of the data set is thus approximated by one single random selected sample. Instead of one single datapoint, one can also consider a set of data points for the update rule. This is called mini-batch stochastic gradient descent. This method has as advantage that it reduces the variance of parameter updates. Larger mini-batches reduce the variance of SGD updates by taking the average of the gradients in the mini-batch. To further improve training time one can apply learning rate scheduling with the Momentum [47], RMSprop [15] or Adam [32] method.

2-4-3 Learning rate scheduling in gradient descent optimisation

In deep learning approaches, several modifications to the update rule are available to improve the training process of the neural network weights. These approaches try to determine the best learning rate for optimisation. Below we describe three gradient descent optimisation algorithms.

Momentum Momentum can be used in stochastic gradient descent. The momentum approach increases the step size when the algorithm goes in the same direction. This prevents the algorithm from getting stuck at a non-optimal solution, such as a local minimum. Momentum accumulates a fraction $\beta \in [0, 1)$ of the gradient in $\mathbf{m}_k$ (an exponentially decaying average of the gradients):

$$\begin{aligned} \mathbf{m}_k &= \beta \mathbf{m}_{k-1} - \alpha_k \nabla L(\mathbf{w}_k) \\ \mathbf{w}_{k+1} &= \mathbf{w}_k + \mathbf{m}_k \end{aligned} \tag{2-21}$$

where $\mathbf{m}_k$ is the sequence that keeps track of previous gradients and can accelerate the algorithm when going in the same direction and reduce oscillations when the parameters are changing. The sequence $\mathbf{m}_k$ is usually initialized at zero.

RMSprop RMSprop (stands for Root Mean Square Propagation) stores an exponentially decaying average of past squared gradients in $\mathbf{v}_k$ and uses this value to calculate a different learning rate for all the weights [48]. This results in a decrease of the step-size over time and is larger when the gradients are large:

$$\begin{aligned} \mathbf{v}_k &= \gamma \mathbf{v}_{k-1} + (1-\gamma) \nabla L(\mathbf{w}_k) \odot \nabla L(\mathbf{w}_k) \\ \mathbf{w}_{k+1} &= \mathbf{w}_k - \frac{\alpha_k}{\sqrt{\mathbf{v}_k} + \epsilon} \odot \nabla L(\mathbf{w}_k) \end{aligned} \tag{2-22}$$

where $\odot$ is the element-wise multiplication operator and ϵ a small number in the order of 10^{-6} that avoids division by zero. The fraction of the squared gradient that is added to $\mathbf{v_k}$ is denoted by $\gamma \in [0, 1)$. A commonly used value for γ is 0.9, and $\mathbf{v}_k$ is initialised at zero.

Adam Adam algorithms [32] combines the Momentum and RMSprop algorithms and is nowadays the standard for training neural networks. Since the first moment $\mathbf{m}_k$ (the mean) in Equation 2-21 and the second moment $\mathbf{v}_k$ (the variance) in Equation 2-22 are initialized at zero, they seem to be biased towards zero [32]. The Adam algorithm counteracts by calculating a bias-corrected first and second-order moment and using these to update the weights:

$$\begin{aligned} \mathbf{m}_k &= \beta \mathbf{m}_{k-1} - (1-\beta) \nabla L(\mathbf{w}_k) \\ \mathbf{v}_k &= \gamma \mathbf{v}_{k-1} + (1-\gamma) \nabla L(\mathbf{w}_k) \odot \nabla L(\mathbf{w}_k) \\ \hat{\mathbf{m}}_k &= \frac{\mathbf{m}_k}{1-\beta^k} \\ \hat{\mathbf{v}}_k &= \frac{\mathbf{v}_k}{1-\gamma^k} \\ \mathbf{w}_{k+1} &= \mathbf{w}_k - \alpha_k \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon} \end{aligned} \tag{2-23}$$

where $\hat{\mathbf{m}}_k$ and $\hat{\mathbf{v}}_k$ are the bias corrected first and second-order moment, β^k and γ^k are β and γ to the power k. Values for all the parameters are proposed in [32]: $\beta = 0.9, \gamma = 0.999, \epsilon = 10^{-8}, \alpha_k = 0.001, \forall k$.

Chapter 3

Related work

In this thesis, several projects have been used to create new insights. Some of the concepts are reused and combined to develop and evaluate a novel method for semantic segmentation in unstructured environments. This chapter briefly describes the most relevant articles associated with this thesis and describes the work of the articles from which most of the inspiration has been acquired. We also introduce the semantic segmentation datasets that are used in the experiments.

3-1 Encoder-decoder networks

Encoder-decoder networks are a type of Artificial Neural Network (ANN)s that can be employed in both supervised and unsupervised learning problems. Encoder-decoder networks have been used in many fields, examples are natural language processing [25], image processing [14], object detection [43], biometric recognition [62], anomaly detection [54] and data analysis [36].

An encoder-decoder model learns to map data points from an input domain to an output domain via a two-stage network. In the first stage, the encoder, with its encoding function $z = g_{\varphi}(x)$, compresses the input data into a lower dimension called latent-space representation. In this latent space representation the most important features of the input data are captured. The type and number of layers determine the latent space size and thus the number of features. The consecutive decoder network, with function $y = f_{\vartheta}(z)$ aims to reconstruct the output from the latent space representation [2].

Examples of unsupervised tasks in encoder-decoder networks are file compression and image reconstruction. Supervised encoder-decoder networks are able to classify the input data. An example of a supervised encoder-decoder network is a semantic segmentation network. This thesis focuses on both supervised and unsupervised encoder-decoder networks applied to image-related tasks.

3-1-1 Convolutional autoencoders

Convolutional autoencoders are a variant of Convolutional Neural Network (CNN)s that are used in dimensionality reduction, image compression or image denoising tasks. The convolutional autoencoder aims to generate a new set of images similar to the original input images. The auto-encoder network does not need a labelled dataset since the input image must be reconstructed. Auto-encoder models are, therefore, examples of unsupervised learning.

The convolutional filters capture the features of the training datasets in their network weights. These features are then reconstructed in the output image of the network. During the autoencoder training phase, we try to minimize a loss function. For autoencoders, the loss function can be the Mean Squared Error (MSE) or the binary cross-entropy loss. The loss function depends on the type of input and output data the autoencoder has. For image-related tasks, the most popular loss function for reconstruction is the MSE. The binary cross-entropy loss is preferred as reconstruction loss when input and output values are either 0 or 1. These values occur when performing classification tasks.

The reconstruction performance of an autoencoder is highly influenced by the latent space size of the autoencoder. The latent space size is determined by the amount, type and size of the layers in the autoencoder. The ratio between the input shape and the latent space size determines the compression rate. An input shape of 360x480x3 is used for all our RGB experiments, corresponding to 518 400 data points. If we divide this number by the amount of data points in the latent space size, we obtain the compression rate of an autoencoder.

3-1-2 Semantic segmentation networks

Image segmentation is the process of dividing an image into multiple segments. Semantic segmentation is the process of pixel-wise classifying each pixel of these segments to a particular class. This task is solved by semantic segmentation networks, which is used in, for example, road scene understanding applications [2]. Semantic segmentation networks use a similar encoder-decoder network as a convolutional autoencoder. The output of both these networks differ. The convolutional encoder reconstructs the original input image, while a semantic segmentation network generates a pixel-wise annotated image. Most of the state of the art semantic segmentation networks rely on an encoder-decoder architecture to create semantic segmentation mappings [65, 64, 2, 44, 42].

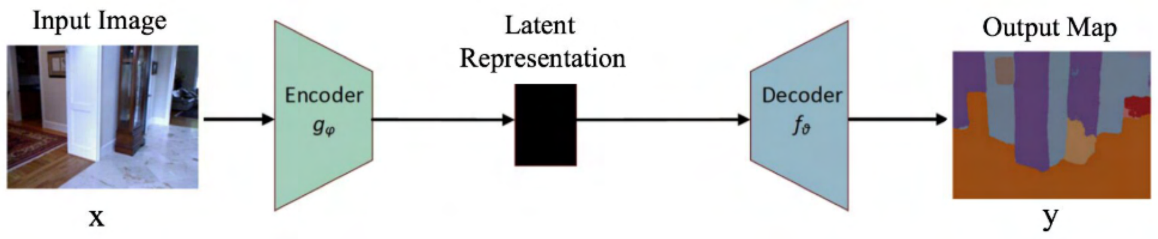


Figure 3-1: The encoder-decoder architecture. The input image is compressed by the encoder ($z = g_\varphi(x)$) into a latent space representation. The decoder ($y = f_\vartheta(z)$) then decodes it back to the original input image size, with the pixels annotated. Image adapted from: [40].

In Figure 3-1 an overview of an encoder-decoder model is illustrated. These supervised learning models are trained by minimizing the loss function $L(y, \hat{y})$, which measures the difference between the ground-truth output y and the semantic segmentation prediction $\hat{y}$. A semantic segmentation network is thus a form of supervised learning, where for example, the network is trained to segment images into different classes, as can be seen in Figure 3-1.

In autonomous driving systems, segmentation models are required to run at real-time computation. Real-time is considered to be around 25 frames per second. Figure 3-2, although it is not up to date, shows the accuracy of the different models with its inference speed. The accuracy and inference speed is calculated on the Cityscapes test dataset [10].

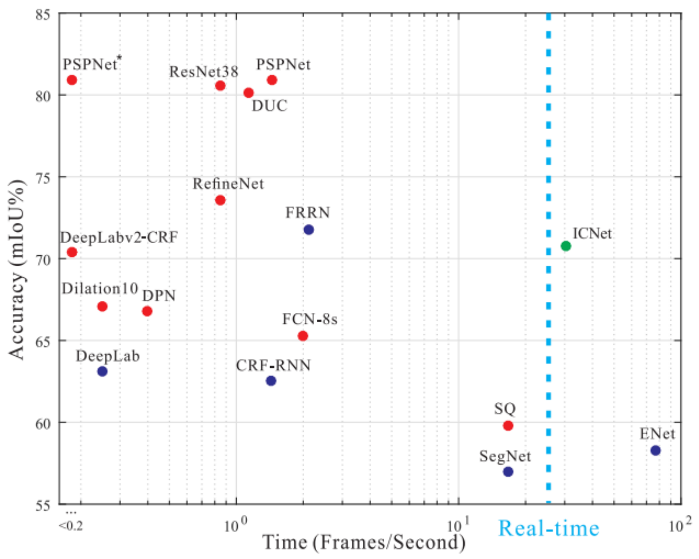


Figure 3-2: Inference speed and mean Intersection over Union (mIoU) performance of different semantic segmentation networks on Cityscapes test dataset. The bleu marked networks are tested with downsampled images. Image adapted from [64].

3-2 SegNet

This thesis uses the SegNet architecture as a semantic segmentation network. This network consists of an encoder network, a corresponding decoder network, followed by a pixel-wise classification layer [2]. The encoder network is identical to the 13 convolutional layers in the VGG16 network [51]. Each encoder layer has a corresponding decoder layer, and hence the decoder network has 13 layers. The decoder network maps the low-resolution feature maps to full input resolution feature maps. The decoder output is fed to a multi-class softmax classifier to independently produce class probabilities for each pixel. In Figure 3-3 the architecture of SegNet is shown, where the input image is transformed by convolutional, pooling, upsampling and softmax layers into a pixel-wise classified image.

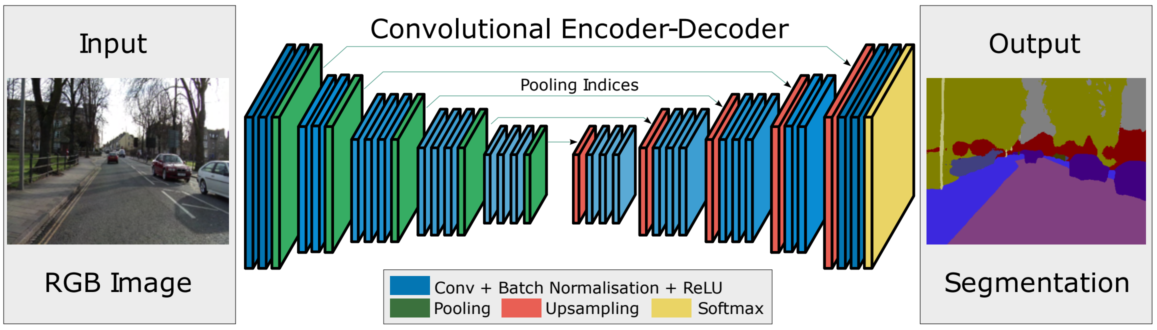


Figure 3-3: An illustration of the SegNet architecture [2]. An RGB input image is fed to the encoder, containing convolutional and pooling layers. A decoder upsamples its input using the pooling indices from the corresponding encoder. It then performs convolution with a trainable filter bank to densify the feature map. The soft-max classifier layer generates a per class probability. The feature maps are transformed into a semantic segmentation mapping.

3-2-1 Architecture

Each encoder in the network performs convolution with a filter bank to produce a set of feature maps. These are then batch normalized [27, 3]. Then an element-wise rectified-linear non-linearity (ReLU) is applied. Following that, max-pooling with a 2×2 window and stride 2 (non-overlapping window) is performed, and the resulting output is sub-sampled by a factor of 2. We store the max-pooling indices, i.e. the location of the maximum feature value in each pooling window is memorized for each encoder feature map. In principle, this can be done using 2 bits for each 2×2 pooling window and is thus much more efficient to store than memorizing feature map(s) in float precision.

The appropriate decoder in the decoder network upsamples its input feature map using the max-pooling indices from the corresponding encoder feature map. This step produces sparse feature map(s). These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A batch normalization step is then applied to each of these maps. Note that the decoder corresponding to the first encoder (closest to the input image) produces a multi-channel feature map, although its encoder inputs have three channels (RGB). This is unlike the other decoders in the network, which produce feature maps with the same size and channels as their encoder inputs. The high dimensional feature representation at the output of the final decoder is fed to a trainable softmax classifier. This softmax function classifies each pixel independently. The output of the softmax classifier is a K channel image of probabilities where K is the number of classes. The softmax function is defined in Equation 2-8

3-2-2 Bayesian model uncertainty

A probabilistic variant of SegNet, which outputs in addition to the semantic segmentation map a measure of model uncertainty, is proposed in [29]. At test-time dropout is used to obtain this model uncertainty as described in Section 2-3-1. This is achieved with no additional parameterisation.

The Bayesian SegNet architecture with dropout layers is shown in Figure 3-4. We obtain the probabilistic output from Monte Carlo samples of the model by these dropout layers during test time. The number of Monte Carlo samples can be set to any value. The authors state that the Monte Carlo sampling converges after 30 samples, no further significant improvement of semantic segmentation performance is obtained beyond this point. We, therefore, fix the number of Monte Carlo samples to 30.

These different Monte Carlo samples are used to obtain the final semantic segmentation and model uncertainty. The mean softmax value of these samples is set as the final softmax output.

$$\bar{s}(z)_i = \frac{1}{n}\sum_{i=1}^{n} s(z)_i \tag{3-1}$$

where n represents the number of Monte Carlo samples. The class with the highest mean probability at a pixel is set as final semantic segmentation prediction. The variance per class v_i between the softmax samples for a pixel is used as the uncertainty.

$$v_i = \frac{1}{n}\sum_{i=1}^{n}(s_i - \bar{s}_i)^2 \tag{3-2}$$

The pixel uncertainty measure above represents the uncertainty per class per pixel. We calculate the overall pixel model uncertainty $\bar{v}$ by taking the mean of the per-class pixel variance.

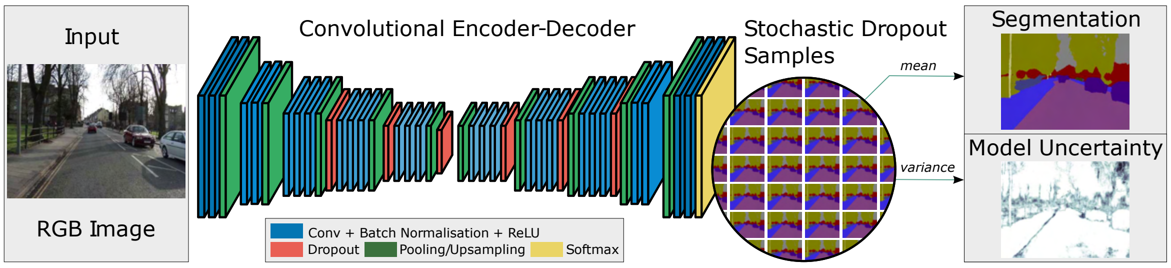

Input
RGB Image
Convolutional Encoder-Decoder
Stochastic Dropout Samples
mean
variance
Segmentation
Model Uncertainty
Conv + Batch Normalisation + ReLU
Dropout
Pooling/Upsampling
Softmax

Figure 3-4: An illustration of the Bayesian SegNet architecture [29]. An RGB input image is fed to the encoder, containing convolutional, pooling and dropout layers. A decoder upsamples its input using the pooling indices from the corresponding encoder. It then performs convolution with a trainable filter bank to densify the feature map. The probabilistic output is obtained from Monte Carlo samples of the model with dropout at test time. The mean of these softmax samples represents the final semantic segmentation, and the variance represents the model uncertainty for each class.

Bayesian SegNet improves semantic segmentation performance by 2-3% compared to the original SegNet. Improvements are especially observed in smaller datasets where modelling uncertainty is more effective. A disadvantage of the Bayesian variant is the additional inference time needed for measuring the model uncertainty. SegNet and Bayesian SegNet both use the cross-entropy loss as the objective function for training the network. The networks can be trained end-to-end using stochastic gradient descent.

3-3 Datasets

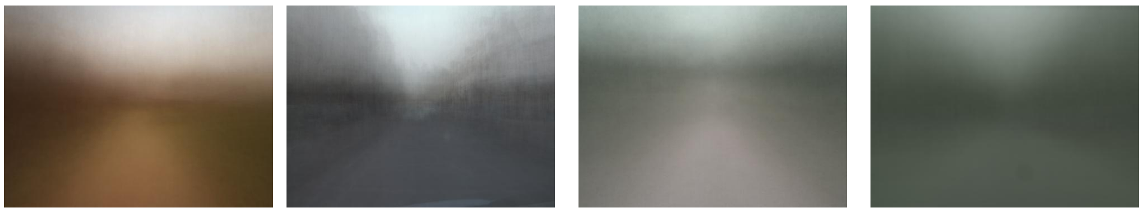
In order to define traversable areas in the RGB images one needs to train a semantic segmentation network. Training these CNN requires labelled datasets. An overview of different semantic segmentation autonomous driving datasets is given in Table 3-1.

Table 3-1: Overview of autonomous driving datasets. For every dataset the environment type, number of annotated images, number of classes and image dimensions is given.

Name	**Environment**	**#Annotations**[1]	**#Classes**[2]	**Dimension**
DeepScene [56]	Forest	366	6	880x480
CamVid [9]	Urban	701	11	480x360
Yamaha-CMU Off-Road [39]	Off-road	783	8	1024x544
Cityscapes [10]	Urban	3475	30	2048x1024

[1]Number of images annotated, [2]Number of classes annotated

The semantic segmentation datasets differ based on the environment in which they are captured, the number of images and classes and the image dimension. We show a comparison of dataset statistics in Figure 3-5, here we see the average RGB image of the different training datasets. In the DeepScene average image, we observe that the road defined in this dataset has a more yellow colour. In contrast, the urban datasets, CamVid and Cityscapes, have a grey colour for traversable areas. We observe a more light grey colour for the traversable pixels in the YCOR average training image. The datasets are described in more detail in the following sections.



(a) DeepScene **(b)** CamVid **(c)** YCOR **(d)** Cityscapes

Figure 3-5: The average training image of the CamVid, DeepScene, YCOR and Cityscapes training dataset.

3-3-1 DeepScene

DeepScene is a forest environment dataset containing 366 images with pixel-level ground truth annotations, which were manually annotated [56]. The data is collected on three days to obtain enough variability in lightning conditions. The original images are of the size 880x480 pixels. In this dataset, six classes are defined: sky, grass, trail, vegetation, tree and obstacle. Examples of the input image with its corresponding ground-truth image can be seen in Figure 3-6.

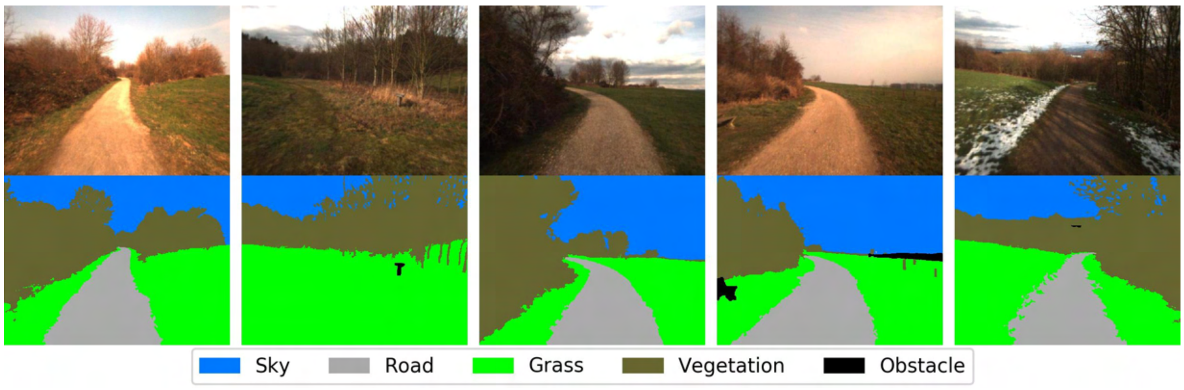


Figure 3-6: DeepScene forest dataset images [56]. The top row is the original input image, with the ground truth shown in the second row. The legend contains the 5 classes defined in the annotated images.

3-3-2 CamVid

CamVid is an urban road scene understanding dataset with 367 training images, 101 validation images and 233 testing images of the day and dusk scenes [9]. The original image size is 480x360; therefore, no resizing of the images is needed. In this dataset, 11 different classes are defined. Examples are roads, buildings, cars, pedestrians, signs, poles, sidewalks. Example images with their labelled ground truth can be seen in Figure 3-7.

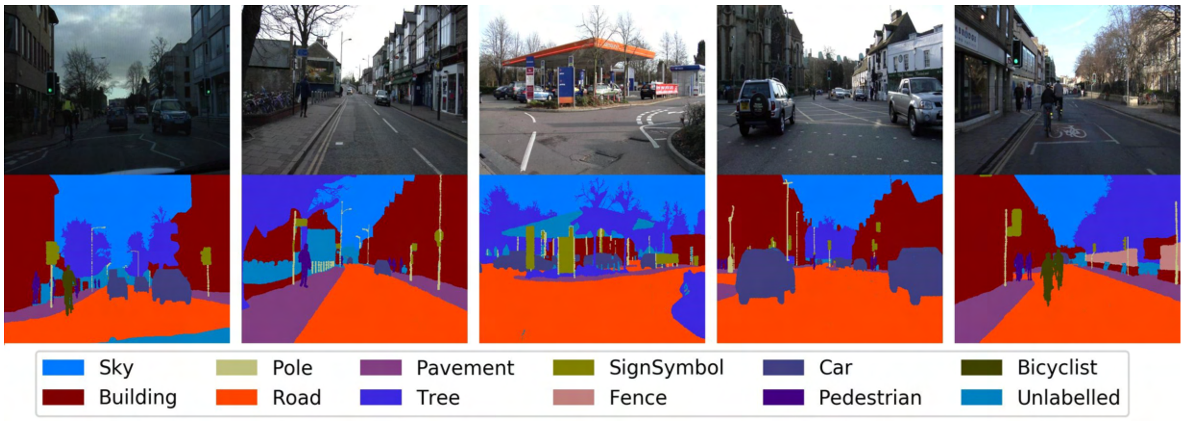


Figure 3-7: CamVid urban dataset images [9]. The top row is the original input image, with the ground truth annotation shown in the second row. The legend contains the 12 label classes with the corresponding colors.

3-3-3 YCOR

Yamaha-CMU Off-Road is an off-road dataset in which eight classes are defined. The dataset consists of 563 training, 134 validation and 86 test images. The original image size is 1024x544 pixels. The images are first cropped to 592x444 pixels and then resized to 480x360 pixels. Example images of this dataset can be found in Figure 3-8.

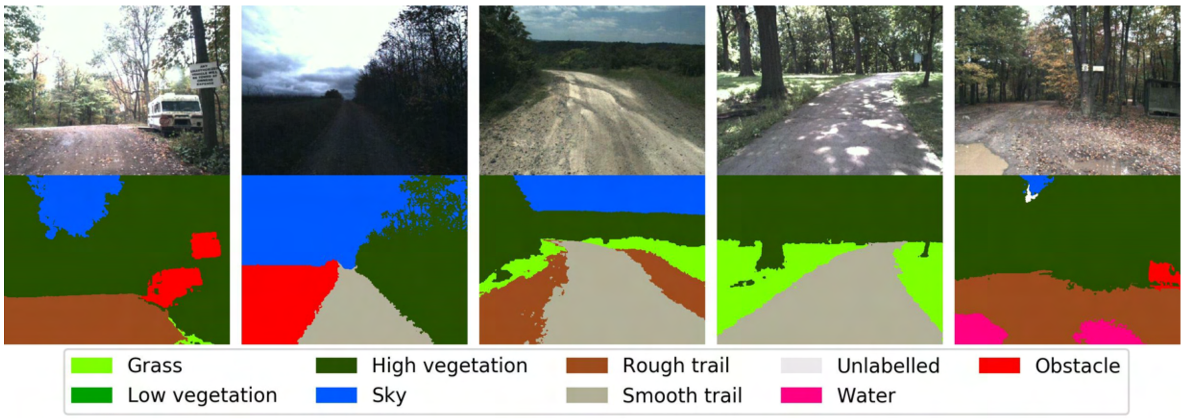


Figure 3-8: Yamaha-CMU Off-Road images [39]. The first row contains the RGB input image, with the ground truth shown in the second row. The legend shows the nine classes encountered in the YCOR dataset.

3-3-4 Cityscapes

The Cityscapes dataset is recorded in different German cities. A total of 5000 images with its ground truth segmentation is available for training, validating and testing the performance of a semantic segmentation network. The original image size was 2048x1024. The images are cropped to a dimension of 1000x750 pixels and then resized to 480x360 pixels. Samples from this dataset are shown in Figure 3-9.

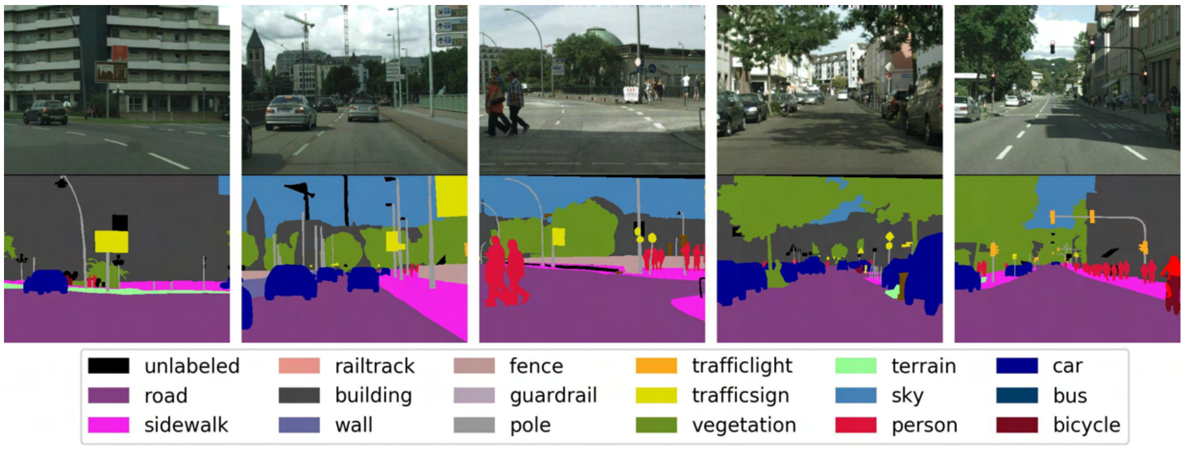


Figure 3-9: Examples images of the Cityscapes dataset [10]. The first row contains the RGB input image, with the ground truth shown in the second row. The legend does not contain all labels used in the dataset.

Chapter 4

Methodology

The previous chapters provided descriptions of Bayesian SegNet and autoencoder networks. This chapter describes how we apply these neural networks in a hybrid framework for traversability estimation in unseen unstructured environments. Hybrid in this context stands for a changing operation mode for changes in environment type. By designing environment-specific models, an improvement of the overall semantic segmentation performance is expected.

We design two different hybrid frameworks, hybrid framework 1 consists of Bayesian SegNet. Hybrid framework 2 contains SegNet and an additional autoencoder. The hybrid frameworks both require a decision parameter for selecting the best performing model for the current input image. In hybrid framework 1, the Bayesian uncertainty is used as a decision parameter. In hybrid framework 2, either the pixel reconstruction error (RMSE) or the Structural Similarity Index Measure (SSIM) will be implemented. Besides improving the model's overall accuracy, we try to measure uncertainty and detect situations it has not seen before. In the following sections, both approaches will be further described.

This chapter starts with the data preprocessing step of the images. After that, an overview of the hybrid frameworks with their different architectures are shown. The last section defines different metrics for evaluating and comparing both approaches.

4-1 Preprocessing of images

The datasets described in Chapter 3 will be used for training, validating and testing the semantic segmentation performance of the hybrid frameworks. The images of the datasets have different widths and heights, which causes difficulties for the input of the neural networks. The semantic segmentation networks are designed for an input image size of 480x360 (4:3 aspect ratio). Therefore, we crop and resize the images and labels of all datasets into this dimension.

We measure the semantic segmentation performance of the hybrid frameworks by the Intersection over Union (IoU) and the Pixel Accuracy (PA). These measures consider the number of classes defined in the datasets. The semantic segmentation performance of the segmenters can only be compared if we rewrite the labels into similar classes for every environmental model. We rewrite all the original label classes to traversable, non-traversable and sky. Example input images with the original labels and the rewritten labels are shown in Figure 4-1.

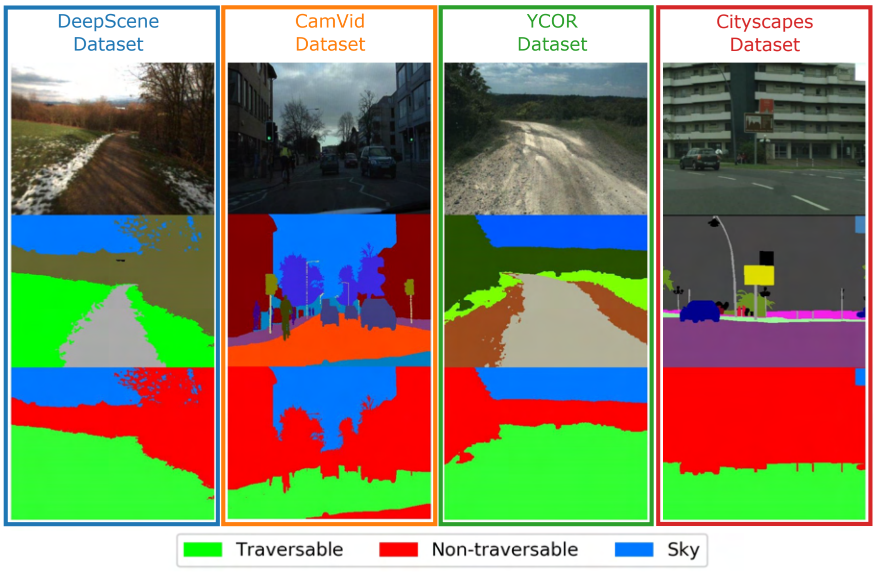


Figure 4-1: RGB images, original labels and rewritten labels examples of the DeepScene, CamVid, YCOR and Cityscapes dataset. The legend contains the classes of the rewritten labels.

4-1-1 Class weights

Before training SegNet or Bayesian SegNet, some variables related to the dataset need to be set. For semantic segmentation networks, the occurrence frequency and the pixel size of the classes influence the network's performance. During training, some classes occur more often than others and which cause the model to prefer some label over another. Roads, for example, tend to be both the biggest and most occurring object in the training images. On the other hand, pedestrians are relatively small objects and are not always present in an image. SegNet corrects this by introducing class weights in the cross-entropy loss function. These class weights are calculated using median frequency balancing [18], shown in Equations 4-1, 4-2 and 4-3. Larger classes in the training set have a weight smaller than one, and the weights of the smallest classes are the highest.

$$\text{weight}(c) = \frac{\text{medianFrequency}}{\text{frequency}(c)} \tag{4-1}$$

where:

$$\text{frequency}(c) = \frac{\sum_{i=1}^{I} \#\text{pixelsOfClass}(c)\text{InImage}(i)}{\#\text{imagesContainingClass}(c) \times \text{imageSize}} \tag{4-2}$$

and:

$$\text{medianFrequency} = \frac{\sum_{c=1}^{N} \text{frequency}(c)}{N} \tag{4-3}$$

The class weights of the different datasets are shown in Table 4-1. These class weights are then passed into the cross-entropy loss function. The loss is summed up over all the pixels in a mini-batch.

Table 4-1: Class weights for the classes traversable, non-traversable and sky in the different datasets. The class weights are calculated using the median frequency balancing method.

Dataset	DeepScene	CamVid	YCOR	Cityscapes
Traversable	0.8826	1.0	0.9664	1.0
Non-traversable	1.0	0.7689	1.0	0.6655
Sky	1.6615	2.1398	3.1564	3.1686

4-2 Hybrid framework 1: Bayesian Uncertainty

In hybrid framework 1, we use the Bayesian uncertainty as a decision parameter C_θ. An overview of the general hybrid framework is shown in Figure 4-2. This overview consists of N environmental models. Every environmental model consist of a environment-specific segmenter S_θ and a corresponding uncertainty measure C_θ. Hybrid framework 1 consists of multiple Bayesian SegNets S_θ with its Bayesian uncertainty measure C_θ. Every model in this hybrid framework will have the same RGB image as input from the camera installed on the Unmanned Ground Vehicle (UGV). The different models will output different semantic segmentation mappings. We hypothesise that when traversing, for example, a forest, a segmenter specifically trained on forest environmental images will outperform a segmenter trained on images from multiple environments.

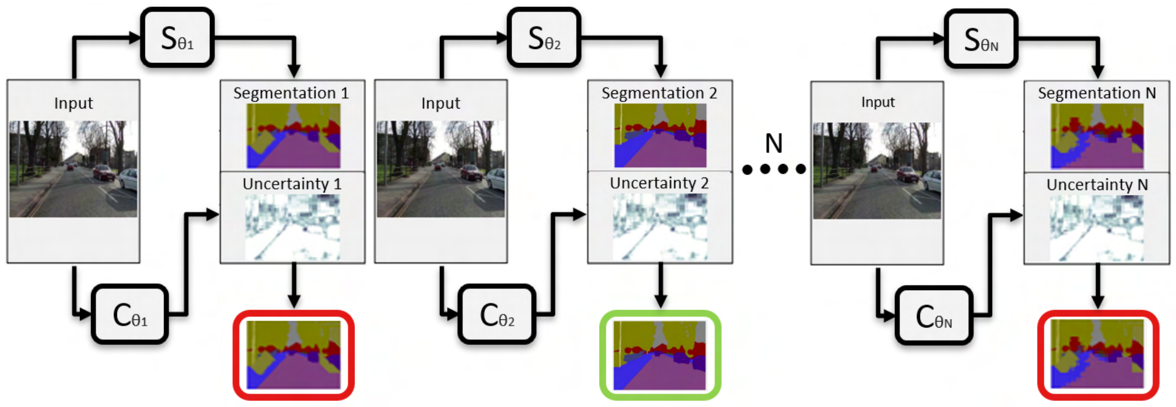


Figure 4-2: Overview of the designed hybrid semantic segmentation framework with N environmental models in parallel. Every environment-specific model consist of a segmenter $S_{\theta i}$ and uncertainty measure $C_{\theta i}$. We compare the uncertainty values of the different segmenters. The segmenter S with the lowest uncertainty value is selected for traversability estimation. In this example, the second environmental model has the lowest uncertainty value C_θ. Therefore, this environment-specific segmenter is selected for traversability estimation.

4-2-1 Bayesian pixel uncertainty (σ)

The N models will output different semantic segmentation mappings. One the environment-specific segmenters S_θ has to be selected for traversability estimation during operation. In hybrid framework 1, this will be based on the Bayesian uncertainty measure C_θ. Every environmental model will present its model uncertainty as a 360x480x1 pixel image. The pixels in this mapping contains a value $v_{h,w}$ between 0 and 0.5. The higher the value, the more uncertain the model's prediction and vice versa. For every N model, the Bayesian uncertainty (σ) is measured by:

$$\text{Bayesian pixel uncertainty } (\sigma) = \sum_{h=1}^{360} \sum_{w=1}^{480} \frac{\bar{v}_{h,w}}{360 \times 480} \tag{4-4}$$

where $\bar{v}$ represents the pixel model uncertainty as calculated in Equation 3-2, h the height of the uncertainty mapping and w the width. Eventually, the segmenter (S_θ) with the lowest uncertainty σ or C_θ will be used for estimating traversability:

$$S_\sigma = \arg\min\Big([\sigma_1, \sigma_2, ..., \sigma_{N-1}, \sigma_N]\Big) \tag{4-5}$$

here σ represent the uncertainty of the different environmental models for the current input image.

4-3 Hybrid framework 2: Reconstruction error

In hybrid framework 2, we introduce another method to quantify uncertainty. In this approach, every segmenter S_θ has an additional autoencoder C_θ, which has to perform as the decision parameter in this framework. The segmenter S_θ and autoencoder C_θ couple in an environmental model are trained on the same training dataset. These CNNs differ in architecture, which eventually result in different outputs. The semantic segmentation network is trained to pixel-wise classify the input image. In contrast, the autoencoder is trained to reconstruct the input image from a latent space representation.

The semantic segmentation network thus outputs a pixel-wise classification while the autoencoder produces a reconstructed RGB image. Hybrid framework 2 compares the reconstructed output images of the autoencoders C_θ to select a segmenter S_θ for the current input frame.

The autoencoders of the environmental models output different reconstruction images since they are trained on different environment-specific datasets. During the training stage, the autoencoder tries to capture the most relevant features of the training dataset in its network weights. For example, an autoencoder that is trained on forest related images will try to capture features such as trees, bushes and leaves in its network weights, while the an autoencoder trained on urban images tries to learn the features as cars, buildings and pedestrians. The training datasets thus determines the network weights, which will result in different reconstructed output images.

These differences in network weights become visible when the decoder reconstructs the latent space representation into a RGB image with same dimensions as our original input. During the decoding phase the autoencoders try to incorporate these environment related features of the training dataset in the reconstruction. This will therefore result in different outputs of the environment-specific autoencoders.

The differences in reconstructed images can be reflected in a reconstruction metric. This reconstruction metric has to measure the difference between the original input image y and the reconstructed image $\hat{y}$. Two commonly used options for measuring this difference are the pixel reconstruction error (RMSE) or the SSIM [22, 57, 58]. One of these measures will eventually be used for model selection in hybrid framework 2. A more detailed description of the RMSE and SSIM are given in the following sections. After that, we describe the considered autoencoder architectures used in this thesis.

4-3-1 Pixel reconstruction error (RMSE)

In the autoencoder approach, a reconstruction error is used to define the model uncertainty of an environmental model. The output of the autoencoder is a reconstructed image, defined by $\hat{y}$, of the size 360x480x3 pixel image. This image has the same dimension as the input image y. The reconstruction error can be calculated as the RMSE between y and $\hat{y}$:

$$\text{Pixel reconstruction error (RMSE)} = \sum_{c=1}^{3}\sum_{h=1}^{360}\sum_{w=1}^{480} \frac{\sqrt{(y_{h,w,c} - \hat{y}_{h,w,c})^2}}{360 \times 480} \tag{4-6}$$

where c represents the RGB color channels, h the height and w the width in pixels. This measure will eventually result in a single value that represents the pixel reconstruction error of the whole image. The pixel reconstruction error functions as the decision variable in hybrid framework 2.

4-3-2 SSIM

The other method for calculating similarities between images is the SSIM [57]. SSIM is a metric that measures the perceived change in structural information between two images windows. Instead of measuring the differences between images on pixel value level, SSIM tries to identify structural information in an image or a window of the image. This metric compares two images based on three key features: luminance, contrast and structure. The SSIM between the image windows of y and $\hat{y}$ is given by the equation:

$$\text{SSIM} = \frac{(2\mu_y\mu_{\hat{y}} + C_1)(2\sigma_{y\hat{y}} + C_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + C_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + C_2)} \tag{4-7}$$

The image luminance features μ_y and $\mu_{\hat{y}}$ are calculated by taking the average of all pixel values in the image window. One can compare the luminance of two image windows using the following formula $\frac{2\mu_y\mu_{\hat{y}}+C_1}{\mu_y^2+\mu_{\hat{y}}^2+C_1}$. C_1 is there to stabilize the division with weak denominators. The function $\frac{2\sigma_{y\hat{y}}+C_2}{\sigma_y^2+\sigma_{\hat{y}}^2+C_2}$ is used to compare the contrast of two images. The contrast features σ_y and $\sigma_{\hat{y}}$ are obtained by taking the standard deviation of all pixel values. Finally the structural similarity is given by $\frac{\sigma_{y\hat{y}}+C_3}{\sigma_y\sigma_{\hat{y}}+C_3}$, where $\sigma_{y\hat{y}}$ is the covariance of the pixels in both images. The mean SSIM will be calculated for the whole image if the SSIM is calculated on a window level. The outcome of the SSIM is a unitless number between -1 and 1, where 1 represents two identical images, and -1 no match at all.

4-3-3 Autoencoder architectures

In this section, we design five different autoencoder architectures for hybrid framework 2. The autoencoder architectures are presented in Table 4-2. They differ based on trainable parameters, latent space size, compression rate, and the number of layers in the encoder and decoder. The architecture names are defined by the number of data points captured in the latent space size. For hybrid framework 2, we aim for an autoencoder architecture that outputs a reconstructed image, which reflects on what the segmenter has seen before. This is, as described above, measured by either the pixel reconstruction error (RMSE) or the SSIM. The precision and recall metrics will determine which autoencoder architecture and decision parameter perform the best in hybrid framework 2.

All the autoencoder have the same input and output dimension. Every encoder consists of multiple convolutional layers followed by a non linear activation function and a MaxPooling layer. The consecutive decoders have instead of a MaxPooling layer an UpSampling layer. The non linear activation functions in the network are the ReLu function. The ReLU function is used since it is known to be computational not expensive. Only the last layer in the autoencoder contains a different activation function. We implement the sigmoid function since we want the output values of the autoencoder to have a lower bound and upper bound. This is for RGB images 0 and 255, we however have normalized these value such that the input and output values range between 0 and 1. Therefore, we use the sigmoid function since this non linear activation function maps its output values onto this range. The autoencoder architectures in Table 4-2 grow in depth as the latent space size decreases. This is obtained by implementing an extra stack of layers in the decoder and encoder. Every stack exists of a convolutional layer, non-linear activation function and a MaxPooling or Upsampling layer.

The first three architectures, 345600, 86400 and 21600, differ an order of four in compression rate. This difference is obtained by adding a (2,2) MaxPooling layer in the encoder and an (2,2) UpSampling layer in the decoder. The fourth 3600 autoencoder architecture has a latent space size six times smaller than the previous autoencoder. This is caused by a new kernel size design of the MaxPooling and UpSampling layer.

We had to consider a new kernel size since adding an extra MaxPooling and UpSampling layer with a (2,2) kernel size causes difficulties in calculating the pixel reconstruction error or the SSIM. Implementing the (2,2) kernel would eventually result in a latent space size of (22.5x60x8), which is rounded up to (23x60x8). The decoder network then passes this latent space representation through its layer which results an output shape of (368x480x3). This output image shape can not be pixel-wise compared with the original input image since the height of the output image differs from the original input image. Therefore, we designed a different kernel size for the MaxPooling and UpSampling. Autoencoder architecture 3600 and 600 therefore have an additional layer with kernel size (3x2) instead of (2x2).

4-4 Evaluation metrics

We describe two segmentation performance measures and two model selection metrics to evaluate the hybrid framework performance. Besides the semantic segmentation and model selection performance, we also consider the inference speed of the models.

Table 4-2: Overview of the five autoencoder architectures designed for hybrid framework 2: reconstruction error. The architectures differ in number of trainable parameters, latent space sizing, compression rate and the amount of layers. The code for the different autoencoder architectures is displayed in Appendix B.

Architecture	Trainable parameters	Latent space size	Compression	Convolutional layers	Maxpooling/ UpSampling layers
345600	1419	180 x 240 x 8	1.5	3 x (8,3,3) 1 x (3,1,1)	2 x (2,2)
86400	2467	90 x 120 x 8	6	1 x (8,2,2) 4 x (8,3,3) 1 x (3,1,1)	4 x (2,2)
21600	3635	45 x 60 x 8	24	1 x (8,2,2) 6 x (8,3,3) 1 x (3,1,1)	6 x (2,2)
3600	4803	15 x 30 x 8	144	1 x (8,2,2) 8 x (8,3,3) 1 x (3,1,1)	6 x (2,2) 2 x (3,2)
600	5971	5 x 15 x 8	864	1 x (8,2,2) 10 x (8,3,3) 1 x (3,1,1)	6 x (2,2) 4 x (3,2)

4-4-1 Pixel Accuracy (PA)

Pixel accuracy measures the ratio between the properly classified pixels, divided by the total number of pixels in the image.

$$\mathrm{PA} = \frac{\sum_{i=0}^{K} p_{ii}}{\sum_{i=0}^{K}\sum_{j=0}^{K} p_{ij}} \tag{4-8}$$

Where K represents the number of classes, p_{ij} the number of pixels of class i predicted as belonging to class j. The above-described function is also known as the global accuracy of a prediction. This pixel accuracy measure can also be transformed into a class average accuracy. Here the ratio is computed per class and then averaged over the total number of classes.

4-4-2 Intersection over Union (IoU)

The Jaccard index or the IoU is another metric to assess the performance of a semantic segmentation network [28]. IoU measures similarity between finite sample sets, and is defined as the size of intersection divided by the size of union of the sample sets. For semantic segmentation prediction this is defined as the area of intersection between the predicted segmentation map and the ground truth, divided by the area of union between the predicted segmentation map and the ground truth.

$$\mathrm{IoU} = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4-9}$$

Where A denotes the ground truth, and B represents the predicted segmentation map. The mean Intersection over Union (mIoU) is the average IoU over all classes. The mIoU is the standard used in research to measure and compare the performance of segmentation models.

4-4-3 Model classification

The hybrid approach requires a classification in which the correct semantic segmentation has to be selected for the current input frame. This selection will be based on the Bayesian uncertainty measure or the reconstruction error measures described above. To compare the model selection performance of the hybrid frameworks, we select two measures, precision and recall. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved based on relevance.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4-10}$$

and

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4-11}$$

We explain the recall and precision based on an experiment with images from the CamVid and Deepscene test datasets and their corresponding models, CamVid and DeepScene. For the CamVid precision and recall measures, True Positive (TP) represents the number of CamVid test images where the CamVid model is selected. False Positive (FP) is the number of DeepScene images where the CamVid model is selected, FP is also known as the Type I error. False Negative (FN) is the number of CamVid images where the DeepScene model is selected, FN is also known as the Type II error. True Negative (TN) is the number of DeepScene images where the DeepScene model is selected.

4-4-4 Inference time

The UGV will drive at high speeds through the different environments. In Figure 3-2 different semantic segmentation networks with their inference speed are shown. In this figure Frames per second (FPS) for real-time is around 25 frames. However, for these obtained results, other hardware is used. Eventually, the inference time of both the hybrid frameworks can be compared to see which one is faster. The specifications of the hardware used in this thesis are:

Hardware specifications:
CPU: Intel Core i7-6800K, 6 cores @ 3.40GHz
GPU: GeForce GTX 1080, 8 GB

Chapter 5

Experiments and Results

In this research, we conducted a set of experiments using two different hybrid semantic segmentation frameworks. We described hybrid framework 1: Bayesian uncertainty (σ) and hybrid framework 2: Reconstruction error (RMSE) in detail in Chapter 4.

In Section 5-1, we define the potential semantic segmentation performance of a hybrid framework compared to a single deep semantic segmentation network. Then we assess the performance of hybrid framework 1 based on the Bayesian uncertainty measure as a decision parameter. The performance of the different approaches is measured by the Intersection over Union (IoU) and Pixel Accuracy (PA) on the semantic segmentation test datasets.

In Section 5-2, we experiment with the different autoencoder architectures for hybrid framework 2. We assess the influence of colour on the autoencoder reconstruction performance. For hybrid framework 2, we have two candidate decision parameters, the pixel reconstruction error (RMSE) and the Structural Similarity Index Measure (SSIM). We investigate which decision parameter is better for this application. We measure the model selection performance of the different decision parameters by the precision and recall evaluation metrics.

In the third experiment, both the hybrid frameworks process a sequence of images. The sequential dataset enables us to measure the Bayesian uncertainty and reconstruction error over time. By tracking these values over time, we can define the number of model switches within this sequence of images for both the hybrid frameworks. Another important metric for traversability estimation is the inference time. The inference time is a factor that limits the maximum speed of the Unmanned Ground Vehicle (UGV) during operation. Therefore, we also perform a comparison on inference time.

In the last section, we compare the semantic segmentation performance of the two hybrid frameworks based on their decision parameters. First, we compare the model selection performance and then we compare the performance on a pixel-wise uncertainty level. The pixel-wise uncertainty comparison differs from the earlier experiments since we compare every pixel individually instead of all the pixels of image.

5-1 Hybrid semantic segmentation experiments

In this experiment, the potential of a hybrid framework containing N=4 Bayesian SegNets in parallel is calculated. These semantic segmentation networks are trained on the DeepScene, CamVid, YCOR and Cityscapes datasets. In addition to the hybrid framework, we train one single deep semantic segmentation network on a combination of all these training datasets. An overview of the different environmental segmenters and their datasets is given in Table 5-1.

The dataset split size is determined by the authors of the different datasets. Therefore, the train, validation and test dataset sizes vary between the segmenters. Changing the split would result in similar-looking frames ending up in both the train, validation and/or test dataset, which eventually influences the semantic segmentation performance. We train the semantic segmentation networks to classify pixels of the input image as traversable, non-traversable or sky. In addition to the pixel-wise prediction, Bayesian SegNet outputs an uncertainty mapping of its prediction.

Table 5-1: Overview of semantic segmenters with their corresponding dataset split and environment type. The dataset of the combined segmenter is a combination of all the datasets.

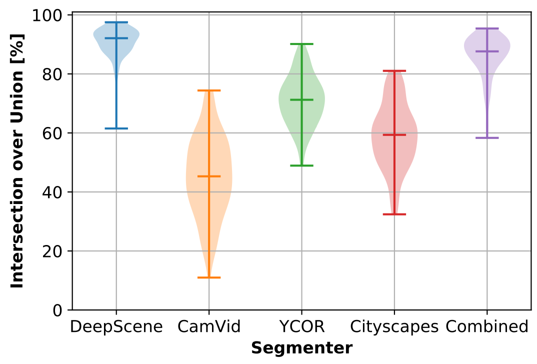
Segmenter	**Environment**	**Train size**	**Validation size**	**Test size**
DeepScene	Forest	197	52	117
CamVid	Urban	367	101	233
YCOR	Off-road	563	134	86
Cityscapes	Urban	2510	500	174
Combined	All	3637	787	610

The five segmenters defined in Table 5-1 use the cross-entropy loss as the objective function. The loss is summed up over all the pixels in a mini-batch. The variation in the number of pixels in each class in the training set is compensated for by using median frequency balancing [18]. The weights of these different classes are given in Table 4-1. Apart from the class weights and datasets, segmenters are trained using identical hyperparameters. The segmenters make use of the Adam optimizer with learning rate $\alpha = 0.001$, exponential decay rate first moment estimates $\beta = 0.9$, exponential decay rate for the second-moment estimates $\gamma = 0.999$, $\epsilon = 0.0001$ and batch size = 3. We train the segmenters until convergence, when we observe no further reduction in training loss.

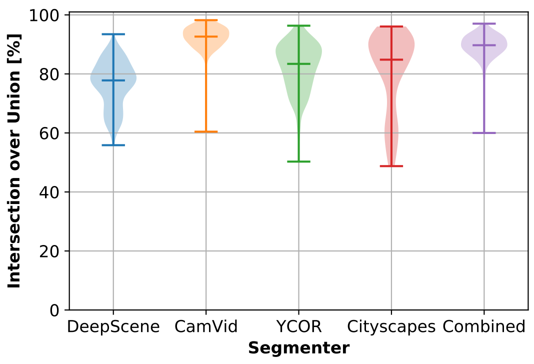
5-1-1 Semantic segmentation

In this section, we compare the semantic segmentation performance of the five segmenters defined in Table 5-1. The predictions differ based on what the segmenter has seen before, which relates to the training dataset. We define the segmenters' quantitative performance by comparing the different accuracies of the N-predictions on the test datasets. The semantic segmentation performance is measured by the PA and IoU as described in Section 4-4.

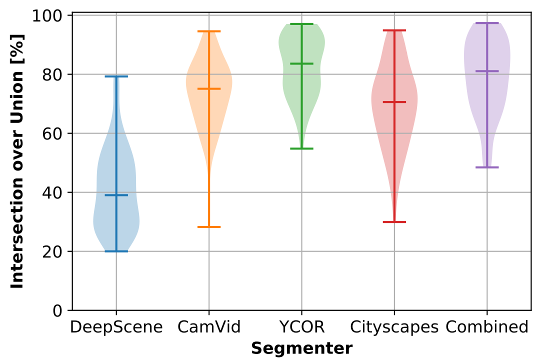
The semantic segmentation performance of the individual segmenters on the different test datasets is shown in Figure 5-1. The DeepScene segmenter outperforms all the other segmenters on the DeepScene test dataset as shown in Figure 5-1a. It obtains the highest minimum 61.5%, median 92.1% and maximum 97.5% IoU scores. In Figure 5-1b, the CamVid test dataset is fed to the five segmenters. Here we see similar results. The CamVid segmenter obtains the highest minimum 60.4%, median 92.6% and maximum 98.2% IoU-scores. The YCOR segmenter outperforms the other segmenter on the YCOR test dataset, with a minimum 54.8%, median 83.6% and maximum 97.1% IoU-score as shown in Figure 5-1c. In Figure 5-1d, we observe that the Cityscapes segmenter obtains a median IoU of 94.6% while the combined segmenter achieves a median IoU score of 95.0%. The combined segmenter thus performs better than the Cityscapes segmenter. However, in this comparison the IoU scores are close, making it hard to state that one is better than the other.



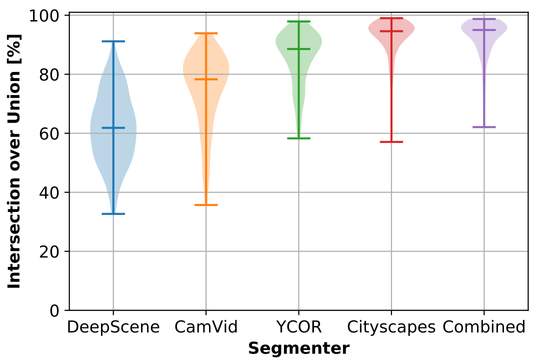
(a) DeepScene test dataset



(b) CamVid test dataset



(c) YCOR test dataset



(d) Cityscapes test dataset

Figure 5-1: IoU performance of the segmenters on (a) the DeepScene test dataset, (b) the CamVid test dataset, (c) YCOR test dataset and (d) the Cityscapes test dataset. The segmenters differ based on their environmental training dataset. The combined segmenter is trained on combination of all training datasets.

Based on the above-performed experiment, we conclude that a semantic segmentation network trained for a specific environment achieves higher IoU and PA scores on the corresponding test dataset than one that is trained for another environment. The environmental specific segmenters thus better capture the features of the training dataset than the combined segmenter.

These results of this experiment also enable us to calculate the potential of a hybrid semantic segmentation framework. The potential performance of a hybrid framework consists of the performance of an environmental segmenter corresponding to a specific dataset. The potential performance thus represents a merged performance of the DeepScene, CamVid, YCOR and Cityscape segmenter on their corresponding test datasets. In practice, this means that the

UGV selects the DeepScene segmenter when it is traversing a DeepScene related environment and the YCOR segmenter when it travels a YCOR environment. In the following section we compare the semantic segmentation performance of the combined segmenter, the potential of hybrid framework and hybrid framework 1: Bayesian uncertainty on the 610 test images.

5-1-2 Bayesian uncertainty

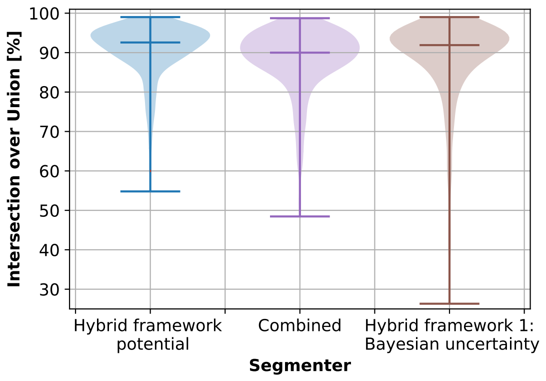
In this experiment we use the uncertainty mappings generated by Bayesian SegNet as a decision parameter for hybrid framework 1. The IoU and PA of the hybrid framework potential, the combined segmenter and the hybrid framework with Bayesian uncertainty is shown in Figure 5-2.

The hybrid potential framework has a higher minimum, median and maximum IoU than the combined segmenter. The median IoU and PA scores of the hybrid potential framework are 92.6% and 97.0%, respectively. In comparison, the combined segmenter has a median IoU and PA score of 90.0% and 95.7%. An increase of 2.6% in IoU and 1.3% in PA can be obtained by designing a hybrid framework. The data points in the hybrid potential framework are thus denser represented at higher IoU and PA scores than the data points of the combined segmenter. The minimum IoU scores also increase from 48.5% in the combined segmenter to 54.8% in the hybrid framework potential. The minimum PA scores increases from 62.0% to 63.8%.

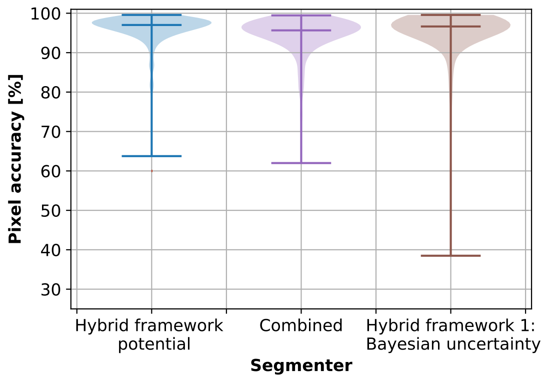
We also observe that hybrid framework 1: Bayesian uncertainty (σ) does not achieve the full potential of the hybrid framework. This is caused by false model selections, where the wrong segmenter is selected for the current input frame based on the Bayesian uncertainty measure. The median IoU score of the Bayesian uncertainty approach is 91.9%, which is 0.7% lower than the potential hybrid framework performance. The Bayesian uncertainty approach, however, has a higher median IoU score than the combined segmenter, 90.0%. The PA also increases in the Bayesian approach. A median of 96.6% is obtained, while the combined segmenter gains a median PA score of 95.7%. The minimum values of hybrid framework 1 are for both the IoU and the PA lower than that of the combined segmenter, 26.3% and 38.5%, respectively. The frames with the lowest semantic segmentation scores are the most critical since for these frames, the UGV is more likely to crash than in frames with high semantic segmentation performance.

Example input images of the DeepScene test datasets and output images of hybrid framework 1 are shown in Figure 5-3. Therefore, we expect the highest semantic segmentation performance of the DeepScene segmenter. For the first input image, both the DeepScene segmenter and the combined segmenter obtain high IoU-scores with 87.5% and 84.6%, respectively. The other three segmenters, CamVid, YCOR and Cityscapes, have lower IoU scores ranging from 48.2% to 64.9%.

We observe this performance difference also in the predictions of the different segmenters. In the first prediction and uncertainty mapping of the DeepScene segmenter, the segmenter is most uncertain on the left edge between the traversable and non-traversable area. The prediction differs from the ground truth annotation in this area, the pixels are falsely classified. For the CamVid, YCOR and Cityscapes segmenter, we obtain higher uncertainty scores which correspond to less confidence about their prediction.



(a) Intersection over Union (IoU)



(b) Pixel Accuracy (PA)

Figure 5-2: Semantic segmentation performance of different segmenters, measured by (a) IoU and (b) PA. The hybrid framework potential represents the optimal performance, for every image in the test dataset the correct model is selected. Hybrid framework 1: Bayesian uncertainty represents the performance when the Bayesian uncertainty measure is implemented as decision parameter.

In the second example, the DeepScene and combined segmenter again produce the best segmentation predictions with respectively an IoU score of 81.5% and 81.6%. For this input image, the CamVid segmenter only classifies a small number of pixels as traversable, which results in an IoU score of 46.3%. The prediction of the CamVid segmenter produces the lowest uncertainty value $\sigma_{\text{camvid}} = 0.003$, this will cause a false model selection. In the next section, we will further elaborate on model selection performance.

5-1-3 Model selection

In the previous subsection, the potential of a hybrid semantic segmentation framework is shown. A decision parameter is designed to select the best semantic segmentation prediction for the current input image for this hybrid framework. In hybrid framework 1, Bayesian uncertainty mapping is used to select the corresponding segmentation. If we choose the prediction with the lowest Bayesian uncertainty value, a median IoU score of 91.9% is obtained. In contrast, the potential median IoU score is 92.6% as shown in Figure 5-2.

The performance decrease occurs due to false model selection based on the Bayesian uncertainty value (σ). The segmenter (S_σ) corresponding to the lowest Bayesian uncertainty (σ) value is selected. In Figure 5-3, we show an example of a correct and false model selection. The first row contains an input image of the DeepScene test dataset. For this input image the DeepScene segmenter has the lowest Bayesian uncertainty pixel value $\sigma_{\text{deepscene}} = 0.006$. While the other environmental specific segmenters have $\sigma_{\text{camvid}} = 0.012$, $\sigma_{\text{ycor}} = 0.013$ and $\sigma_{\text{cityscapes}} = 0.011$. For this input frame the prediction of the DeepScene segmenter is selected for traversability estimation resulting in an IoU of 87.5%.

A false model selection is shown in the second example of Figure 5-3. In this example the CamVid segmenter has the lowest Bayesian uncertainty pixel value $\sigma_{\text{camvid}} = 0.003$. While the other environmental segmenters have an uncertainty value of $\sigma_{\text{deepscene}} = 0.01$, $\sigma_{\text{ycor}} = 0.011$ and $\sigma_{\text{cityscapes}} = 0.04$, which are all higher. This results in a false model selection, the CamVid

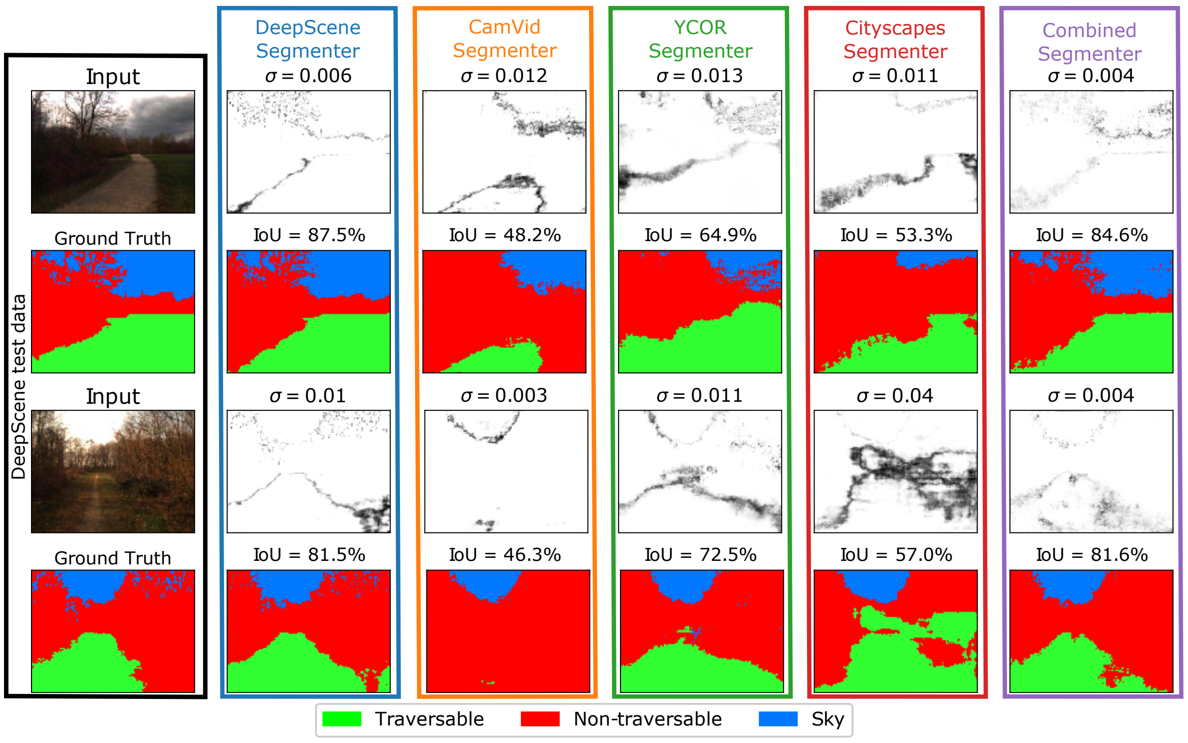


Figure 5-3: Hybrid framework 1: Bayesian uncertainty (σ) semantic segmentation input and outputs. The first column contains input images with ground truth annotations from the DeepScene test dataset. The other columns contain the segmentation and uncertainty mapping of the DeepScene, CamVid, YCOR, Cityscapes and combined segmenter. The combined segmenter is not part of the hybrid framework but is visualised for comparison purposes only. The Bayesian uncertainty pixel values (σ) and IoU scores are printed above the uncertainty mapping and prediction, respectively.

segmenter is selected while the DeepScene segmenter is the preferred one. This false model selection eventually results in a decrease to an IoU score of 46.3%.

The total number of false model selections of hybrid framework 1: Bayesian uncertainty on all the 610 test images is shown in Table 5-2. We measure the recall and precision, resulting in an average of 76.4% and 81.9%, respectively. On the Cityscapes dataset, the highest recall score of 98.9% is obtained. From all the Cityscapes test images, only two instances were wrongly classified. At those two images, the YCOR segmenter had a lower Bayesian uncertainty value (σ) than the Cityscapes segmenter. We observe from the precision scores that images where the DeepScene segmenter had the lowest Bayesian uncertainty value (σ) did not result in false model selections.

5-1-4 Conclusion

From the results shown in Figure 5-2 we conclude that a hybrid semantic segmentation framework can potentially outperform a single segmenter that is trained on a combination of multiple environment-specific datasets. Both the minimum and median IoU and PA scores increase with the design of a hybrid semantic segmentation framework. The hybrid framework po-

Table 5-2: Model selection performance of Hybrid framework 1: Bayesian uncertainty on 610 images of the test datasets. Perfect model selection would be if all the values are on the diagonal. Off-diagonal values represent the number of false model selections. A precision of 100% is obtained by the DeepScene model, however the other models are not able to achieve this performance.

		Test dataset				
		DeepScene	**CamVid**	**YCOR**	**Cityscapes**	**Precision**
Segmenter	**DeepScene**	91	0	0	0	100%
	CamVid	23	114	4	0	80.9%
	YCOR	0	6	69	2	89.6%
	Cityscapes	3	113	13	172	57.1%
	Recall	77.8%	48.9%	80.2%	98.9%	

tential has a more dense distribution of data points at higher performance scores. However, using the Bayesian uncertainty measure as a decision parameter will not fulfil the potential of the hybrid approach. This is caused by false model selection for some input images of the test datasets. These false classifications causes an IoU drop to scores lower than the combined segmenter.

5-2 Autoencoder experiments

In this section, we experiment with hybrid framework 2: Reconstruction error (RMSE). This approach uses SegNet for semantic segmentation predictions and an additional autoencoder to select a segmenter. An overview of hybrid framework 2 is shown in Figure 4-2. For this hybrid framework we need to implement one of the designed different autoencoder architectures as defined in Table 4-2. Besides the different autoencoder architectures designs we also have to decide which decision parameter to use in this framework. The two candidate decision parameters are the pixel reconstruction error (RMSE) and the SSIM. We select the best performing segmenter for a specific input image based on this decision parameter. For hybrid framework 2, we thus have to select an autoencoder architecture and choose one of the decision parameters.

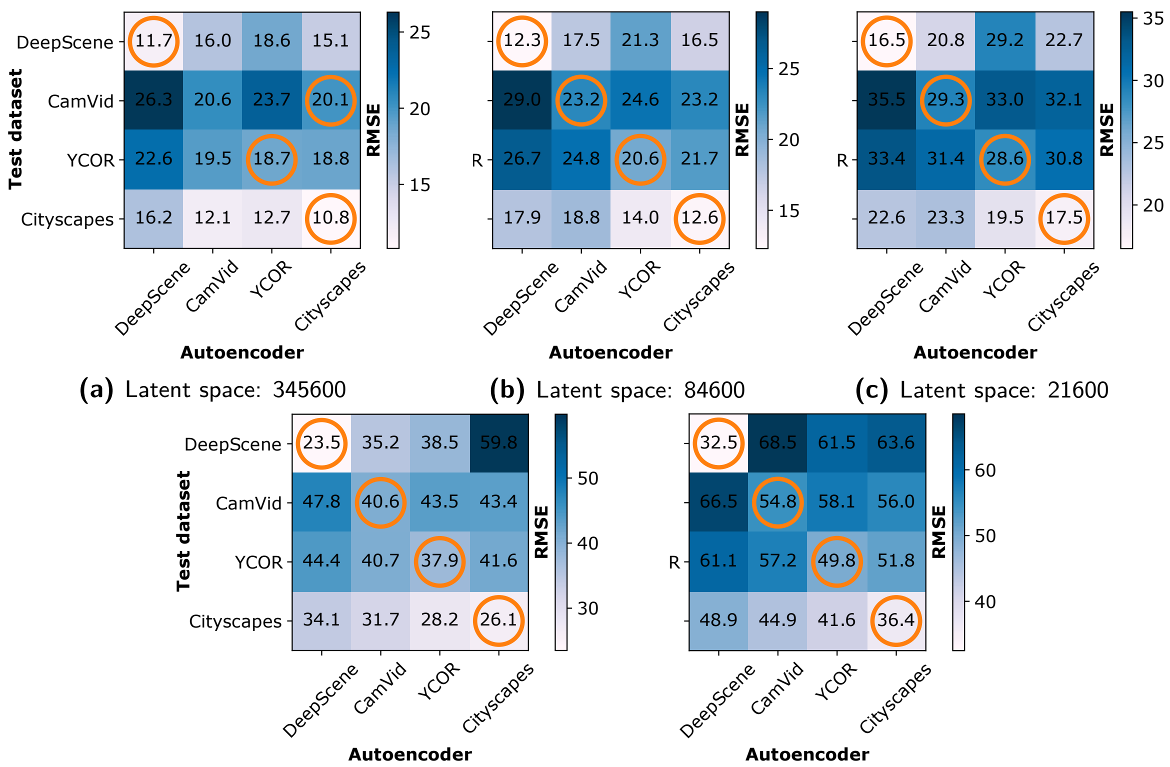
Our autoencoder approach aims to capture relevant features from the training dataset into the autoencoder its weights. The information captured in the weights represents what the environmental segmenter has seen before. Suppose an image from the DeepScene test dataset is fed to the autoencoders. In that case, we expect the best reconstruction of the DeepScene autoencoder and less precise reconstructions of the other environmental autoencoders. We assess the difference between the original input image and reconstructed image by the reconstruction error (RMSE) or the SSIM. We aim for improving the model selection performance obtained by hybrid framework 1.

For every architecture described in Table 4-2 we train four different environmental autoencoders on the DeepScene, CamVid, YCOR and Cityscapes training datasets. This thus results in a total of 20 different autoencoders that differ based on their training dataset and autoencoder architecture. The autoencoders are trained using the Mean Squared Error (MSE) as loss function. The weights are optimized using Adam with learning rate $\alpha = 0.001$, exponential decay rate first moment estimates $\beta = 0.9$, exponential decay rate for the second-moment estimates $\gamma = 0.999$, $\epsilon = 1e-07$ and batch size $= 3$. We train the autoencoders until convergence, when we observe no further reduction in training loss.

5-2-1 Image reconstruction performance

In this experiment, we assess two different decision parameters, the pixel reconstruction error (RMSE) and the SSIM as described in Section 4-3. These decision parameters measure the output image reconstruction quality of an autoencoder. An overview of the reconstruction performance measured by the pixel reconstruction error (RMSE) is shown in Figure 5-4. The mean RMSE of an environmental autoencoder is calculated for the different test datasets.

We observe in most plots the lowest pixel reconstruction error (RMSE), circled in orange, on the diagonal. The environmental autoencoders with latent space size 345600 has one lowest RMSE value off-diagonal. The Cityscapes autoencoder has the lowest mean pixel reconstruction error for the CamVid test dataset. This, however, can be explained by the fact that both the CamVid and Cityscapes autoencoder are trained on urban images. We also observe that the RMSE values increase over the different latent space size plots. In Figure 5-4a with latent space size 345600 the mean RMSE range from 10.8 to 26.3. While in Figure 5-4e the mean RMSE ranges from 32.5 to 68.5. Autoencoders with a smaller latent space representation thus result in less precise reconstruction performance.

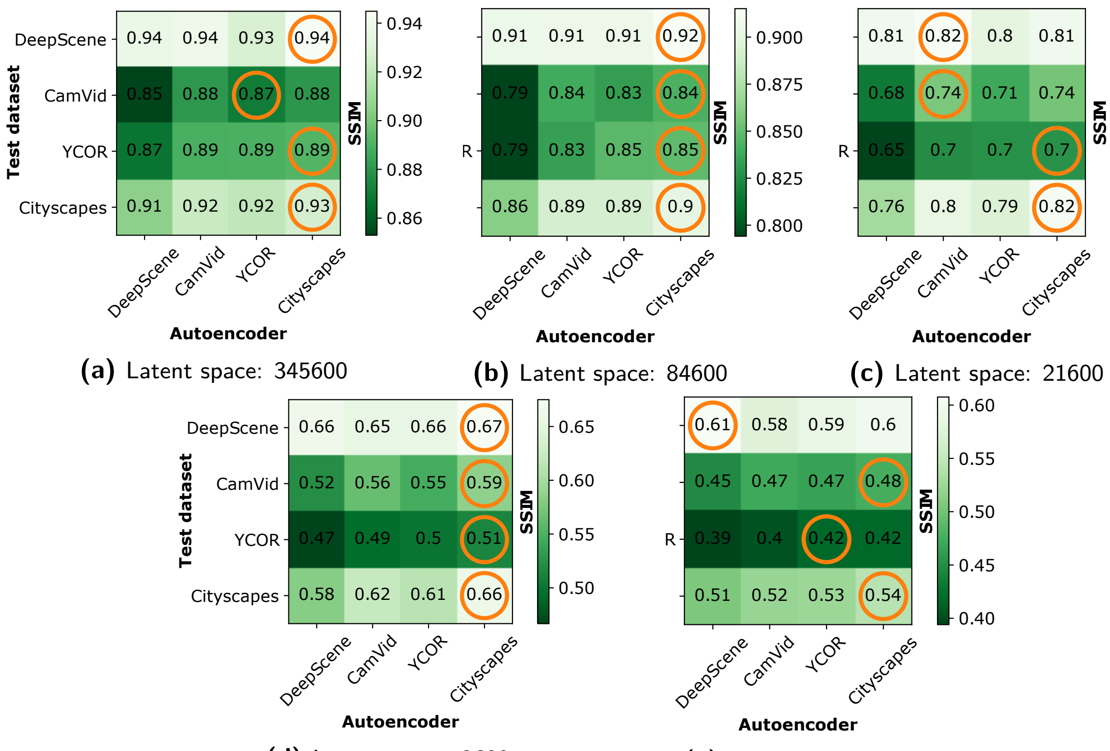


(a) Latent space: 345600 **(b)** Latent space: 84600 **(c)** Latent space: 21600

(d) Latent space: 3600 **(e)** Latent space: 600

Figure 5-4: Autoencoder reconstruction performance measured by RMSE for the different latent space size (a) 345600, (b) 86400, (c) 21600, (d) 3600 and (e) 600. The mean RMSE is calculated on the different test datasets. The orange circles indicate lowest RMSE on a row level.

In Figure 5-5 we replace the mean pixel reconstruction error (RMSE) by the mean SSIM. A value close to 1 represents a high structural similarity. In this matrix, values circled in orange represent the highest SSIM on a row level. We observe that not all the orange circles are on the diagonal. Only Figure 5-5e with an autoencoder latent space size of 600 has 3 out of 4 highest values on diagonal.



(a) Latent space: 345600 **(b)** Latent space: 84600 **(c)** Latent space: 21600

(d) Latent space: 3600 **(e)** Latent space: 600

Figure 5-5: Autoencoder reconstruction performance measured by SSIM for the different latent space size (a) 345600, (b) 86400, (c) 21600, (d) 3600 and (e) 600 on the different test datasets. The orange circles indicate lowest RMSE value on a row level.

We observe that the RMSE approach has more orange circled values on the diagonal than the SSIM approach. The relative differences in RMSE within a row are also more significant than the SSIM. Therefore, we conclude that the pixel reconstruction error (RMSE) is a better decision parameter for the hybrid semantic segmentation framework.

As an example, we show some reconstructed images of the different architectures and environmental models of hybrid framework 2 in Figure 5-6. Every autoencoder architecture is trained on four different training datasets. We observe that the reconstruction quality decreases by decreasing the latent space size. The environmental autoencoders with smaller latent space sizes have difficulties reconstructing the colour and structural details. This can, for example, be seen in the colour of the grass. Here the urban environmental autoencoders have difficulties reconstructing grass's colour in latent space sizes of 21600 and smaller. This effect can have two different causes. It can either occur because these autoencoders have not seen the colour of grass before or the autoncoders did not classify it as an essential feature.

These differences in structure and colour are also present in the output of the decision parameters. We observe a decrease in pixel reconstruction error (RMSE) as the latent space size of an autoencoder increases. This does not hold for the DeepScene autoencoder. The DeepScene autoencoder with latent space size 345600 has a RMSE value of 11.0 while the 86400 autoencoder has a RMSE of 10.8. This difference is, however, not observable in Figure 5-4. Here the average RMSE value of the DeepScene autoencoders are 11.7 and 12.3 for the 345600 and 84600 autoencoder respectively. This example in Figure 3-6 does thus not correctly represent the averages obtained on the DeepScene test dataset.

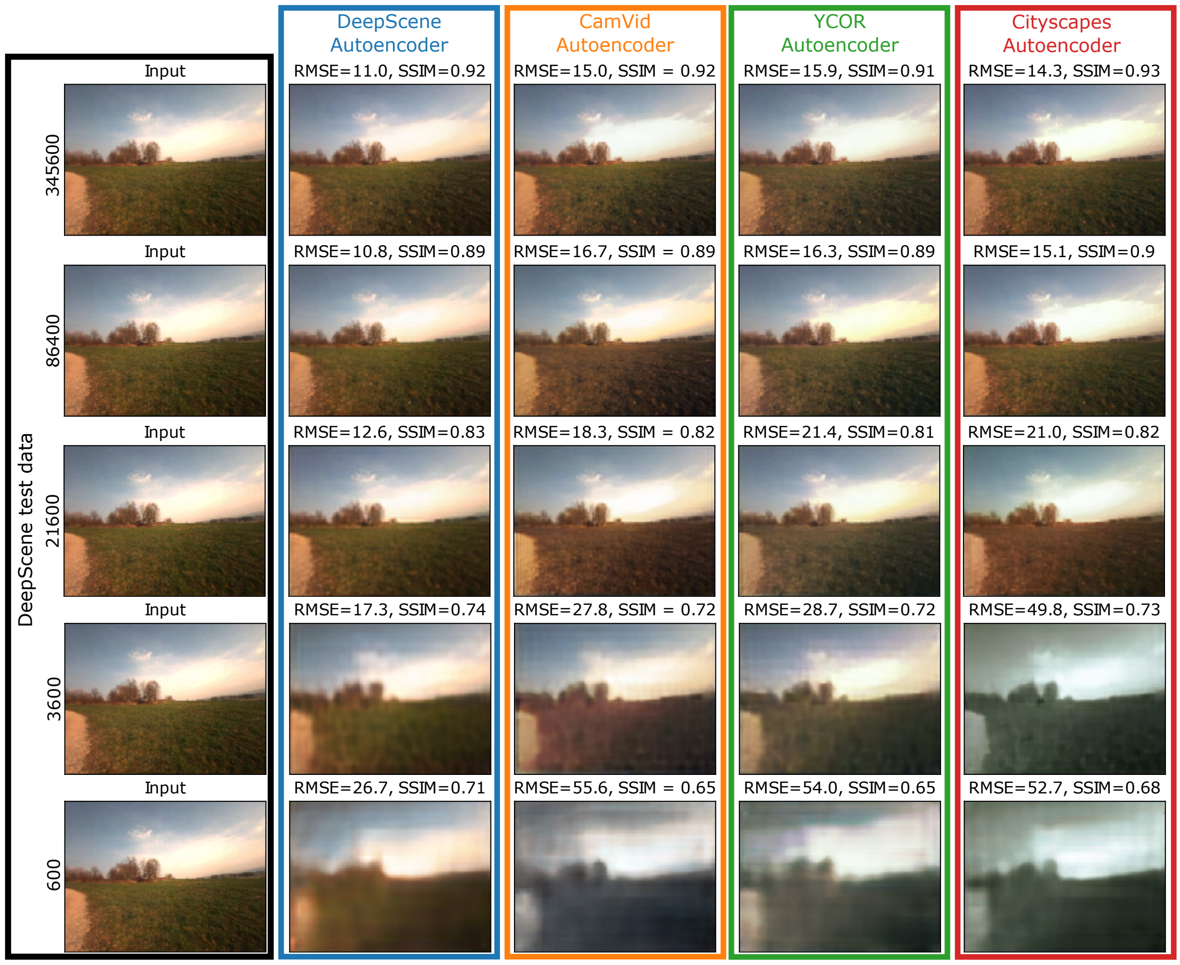


Figure 5-6: Reconstructed input images of different autoencoder architectures and models. Every row contains another autoencoder architecture and every column contains another environmental autoencoder. The pixel reconstruction error and SSIM are shown in the title of the plots.

5-2-2 Model selection

In this section, we test the model selection performance of the different autoencoder architectures. We consult the DeepScene, CamVid, YCOR and Cityscapes test datasets for this experiment. We measure the model selection performance by the precision and recall metrics as described in Section 4-4.

In the previous section is shown that for this application the pixel reconstruction error (RMSE) is a better measure for comparing images than the SSIM. Therefore, we select the RMSE as decision parameter in hybrid framework 2. Based on this decision parameter a segmenter will be selected for the semantic segmentation prediction. The correct environment-specific segmenter is selected if the RMSE of the corresponding environment-specific autoencoder is the lowest for an input image from a specific test dataset. If not, a false model selection is obtained.

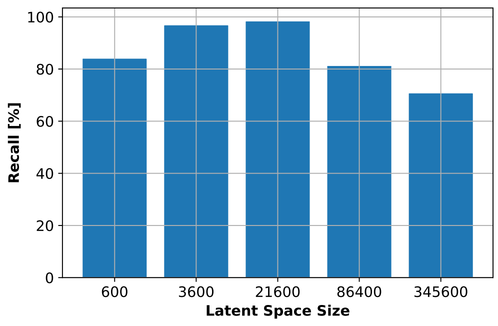
In the example image of the DeepScene test dataset can be seen how the RMSE value changes over the different environmental autoencoder architectures shown in Figure 5-6. For this input image, the DeepScene autoencoder in the second column always has the relative lowest pixel reconstruction error value compared to other environmental autoencoders. In the first row, with latent space size 345600, all the reconstructed images look similar to the input image, resulting in relatively small differences in pixel reconstruction error. This relatively small difference result in a relative higher change in false model selections. We need to determine which autoencoder architecture results in the best model selection performance for all test images to overcome this problem.

The model selection performance of the different autoencoder architectures on the test datasets is shown in Figure 5-7. The results presented in this plot can be consulted in Appendix A. We observe that both the precision and recall increase from architecture 600 to 3600 and from 3600 to 21600. The autoencoders architectures 600 obtain a recall of 84.2% and precision of 89.5%. The 3600 autoencoder architecture increases to 97.0% recall and precision score. The 21600 autoencoder architecture eventually result in the highest recall and precision scores with a recall of 98.5% and a precision of 99.3%. Both the 86400 and 345600 autoencoders result in lower recall 81.4% and 70.9% and a lower precision 87.8% and 85.9%, respectively. These architectures thus result in more false model selections on the test datasets. The 21600 autoencoder architecture thus outperforms all the other autoencoder architectures on the test datasets. It is better in reconstructing dataset related features in the output image. Therefore, we select autoencoder architecture with a latent space of 21600 for hybrid framework 2.

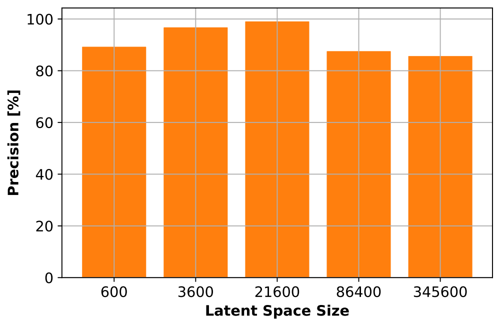
The recall on individual test datasets and the precision of the individual autoencoders is shown in Figure 5-7c and 5-7d, respectively. We observe that the 21600 architecture has a recall of 94.2%, which corresponds to a total of 5 false classifications on the YCOR dataset. These false classifications occur due to the Cityscapes autoencoder, the precision of the 21600 Cityscapes autoencoder is 97.2% as shown in 5-7d. The Cityscapes autoencoder causes most of the false model selections over all the autoencoder architectures.

Three examples of the 21600 autoencoder architecture, in which the wrong model is selected, are shown in Figure 5-8. These false classifications occur due to better reconstruction performance of the Cityscapes autoencoder. It can be seen that the RMSE between the different autoencoders are close. For the first image, the YCOR autoencoder has an RMSE of 22.3

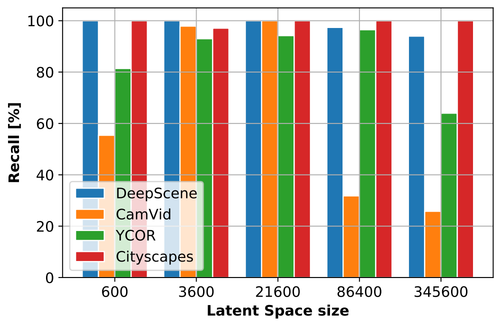
while the Cityscapes RMSE was 21.6. For the second and third examples, an even smaller difference of 0.1 in RMSE is obtained.



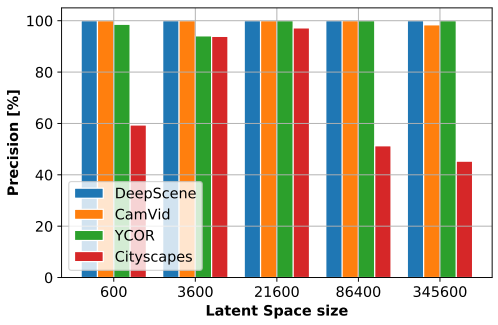
(a) Overall Recall



(b) Overall Precision



(c) Recall on individual test datasets



(d) Precision of individual autoencoders

Figure 5-7: Model selection performance of hybrid framework 2 Reconstruction error (RMSE), measured by (a) recall and (b) precision. The x-axes contain different autoencoder architectures. The architecture with latent space size 21600 obtains the highest recall and precision score, 98.5% and 99.3% respectively. The underlying recall and precision are shown in (c) and (d).

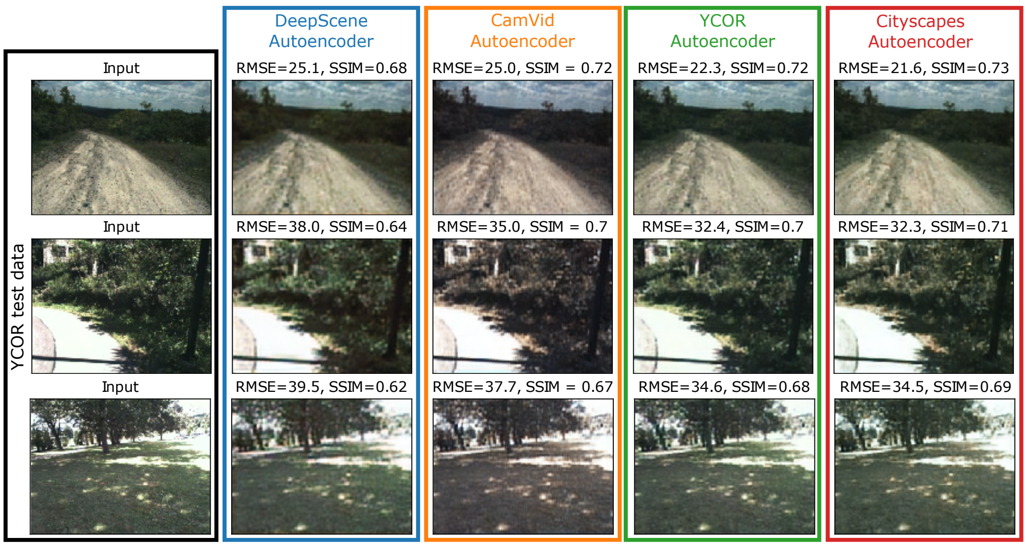


Figure 5-8: Three examples of false model selection based on pixel reconstruction error (RMSE) of the 21600 autoencoder architecture. The Cityscapes autoencoder produces the lowest pixel reconstruction error while YCOR test images are processed by hybrid framework 2.

5-2-3 Grayscale autoencoder

The reconstructed images in Figure 5-6 show that colour is an aspect that potentially influences the pixel reconstruction error. To assess the influence of color we perform an experiment with grayscale images and variations on the RGB input images. The previous experiments all use images with dimension $360 \times 480 \times 3$. We convert these images to grayscale images to obtain the following image dimensions $360 \times 480 \times 1$. The luminosity method [46] is designed to transform RGB images into grayscale images. This weighted equation is the standard in MATLAB's function "rgb2gray", and is frequently used in computer vision [6]. This method is designed to match the human brightness perception by using a weighted combination of the RGB channels. The weights correspond to color its wavelength. The lumosity method is as follows:

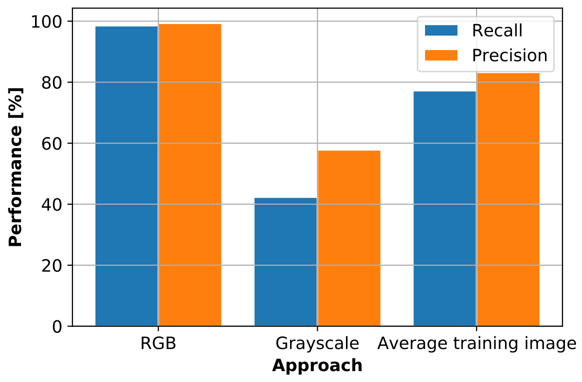
$$\text{Grayscale} = \frac{299}{1000} \cdot \text{R} + \frac{587}{1000} \cdot \text{G} + \frac{114}{1000} \cdot \text{B} \tag{5-1}$$

Here R, G and B represent the RGB values of the original images.

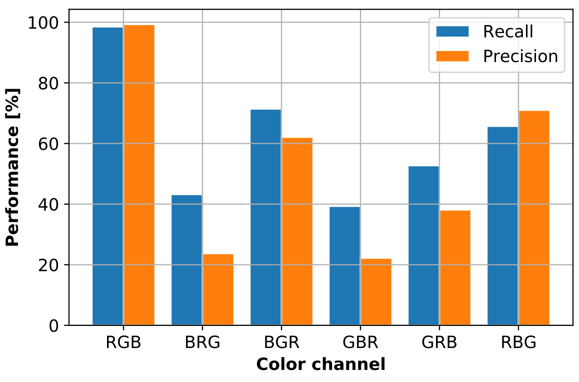
The model selection performance of the grayscale autoencoder and variations of the RGB input order channel is shown in Figure 5-9. The 21600 autoencoder architecture with pixel reconstruction error as a decision parameter is used in this experiment. First, we train the autoencoders on the converted grayscale images as described above. This results in a decrease of model selection performance to a recall of 42.3% and precision of 57.8% as shown in Figure 5-9a.

In addition to the grayscale experiment, we perform an average training image experiment. In this experiment, we measure the RMSE between the input image and the average training image as shown in Figure 3-5. The image of the test datasets are somehow related to the training dataset. They are captured in the same or a similar environment. In this experiment we try to find out if we can select the correct environment-specific segmenter with this simple approach. By measuring the RMSE between the average training image and the current input frame we obtain a decision variable for every environmental model. If we select the segmenter with the lowest RMSE we might can obtain correct model selection. This simple approach results in a recall of 77.2% and precision of 83.2% on the test datasets. The experiments with the RGB autoencoder obtained a recall and precision score of 98.5% and 99.3%, respectively. Based on the difference between those experiments, we conclude that the autoencoder can learn features from the training dataset.

We perform a third experiment where we assess the influence of the colour channels on the model selection performance. In this experiment, variations of the RGB test images are fed to hybrid framework 2. The RGB order of the images from the test datasets are transformed into BRG, BGR, GBR, GRB and RBG. The model selection performance, shown in Figure 5-9b, of all these variations, decreased with respect to the original RGB input frames. Interestingly, the difference between the input channels that are entirely shuffled, BRG and GBR, and the ones that still have one input channel in its original position, BGR, GRB and RBG. The model selection performance of the autoencoders on BRG and GBR images is the lowest. In these images, no original information of the RGB image is retained. While if one of the RGB channels stays in place, a higher model selection performance is obtained.



(a) RGB autoencoder, grayscale autoencoder and average training image approach



(b) RGB autoencoder with different input order channels

Figure 5-9: A comparison of different color input modalities is performed based on recall and precision. In (a), three different approaches are shown. The first two are autoencoders with latent space size 21600, one that is designed for RGB images and one that is designed for grayscale images. The third approach is the average training image approach. Here we select the model that has the lowest pixel reconstruction error (RMSE) between the input image and average training image. In (b) the RGB autoencoder is used. The order channels of the input frames are changed. By changing the order of channels the different precision and recall scores are obtained.

5-2-4 Conclusion

From the experiments with the different autoencoder architectures and decision parameters can be concluded that the autoencoder architecture with the pixel reconstruction error (RMSE) and a latent space size of 21600 performs the best for model selection. A recall and precision of respectively 98.5% and 99.3% are obtained with this autoencoder architecture.

The pixel reconstruction error (RMSE) as a decision parameter results in better model selection performance than the SSIM. We conclude that structural differences between the reconstructions of the different environmental autoencoders are less present than colour differences on a pixel level.

The autoencoder architectures with a latent space size smaller and equal to 21600 have difficulties reproducing specific colour compositions. Autoencoders with bigger latent space sizes do not have difficulties in the reconstructed images' colours. There is little difference between the environmental models. To assess the influence of colour on model selection performance, we performed an additional experiment with models trained on grayscale images using the 21600 autoencoder architecture. This resulted in a decrease of the recall and precision score to 42.3% and 57.8%, respectively. We thus conclude that colour significantly influences the model selection performance of the reconstruction error approach.

The average training image approach results in recall and precision scores of 77.2% and 83.2%, respectively. The 21600 colour autoencoder approach also outperforms this simple model selection approach. This substantiates that the autoencoder does indeed learn features from the training dataset and can reconstruct those in the decoding phase of the autoencoder.

5-3 Sequential data experiments

The images processed by our hybrid semantic segmentation framework will contain sequential information. Input frame i is likely to have similar features as input frame $i-1$. We, therefore, perform experiments with sequential data to compare the two decision measures, the Bayesian uncertainty (σ) and the reconstruction error (RMSE). In this experiment, we measure the model selection performance of the systems over time.

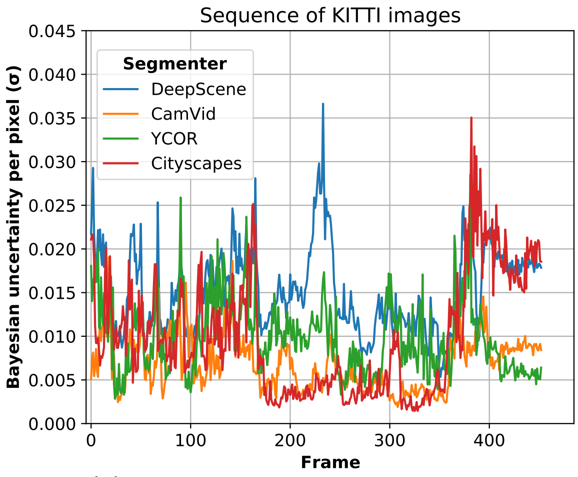
5-3-1 Sequence of KITTY City images

We visualise the decision parameter output values of the hybrid frameworks over a sequence of 432 images from the KITTI City dataset [21] in Figure 5-10. This sequence of images is captured in an urban environment, example images from this dataset are shown in Figure 5-5d. We, therefore, expect that either the CamVid or Cityscapes segmenters is preferred by the decision parameters during this experiment.

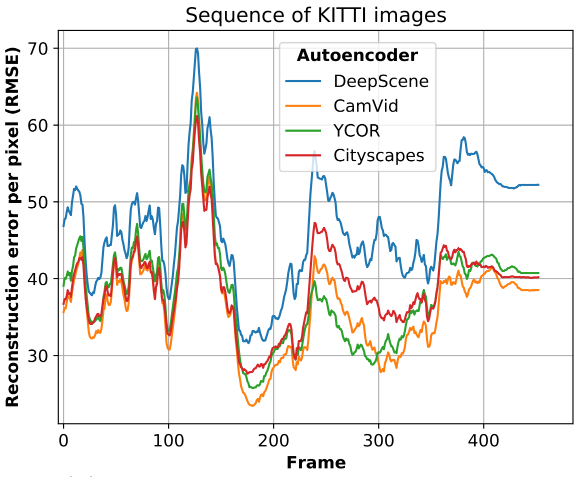
In Figure 5-10a, we observe a non-smooth trend in the Bayesian uncertainty plot. This occurs due to the prediction uncertainty involved in the Bayesian uncertainty approach. Bayesian SegNet quantifies uncertainty by approximating the predictive distribution, which is obtained by random dropout layers. The randomness involved in this process cause the prediction uncertainty, the non-smooth trend in this plot. The random parameter result in different outputs for the same input image. The above described predictive uncertainty eventually results in 110 model switches as shown in Figure 5-10c. In this plot we also tracks the number of model selections over time for the environmental segmenters. Here we see that the Camvid segmenter is consulted 183 times, the YCOR segmenter 101 times, the Cityscapes segmenter 167 times and the DeepScene segmenter 2 times. We aimed for zero occurrences of the YCOR segmenter and DeepScene since this data sequence is captured in urban environments. Hybrid framework 1: Bayesian uncertainty (σ) thus results in 103 times a false model selection.

In the output of hybrid framework 2: reconstruction error (RMSE) we observe a more smooth trend as can be seen in Figure 5-10b. The RMSE scales with changes in the input frame, this can be seen overtime but also by comparing the different autoencoders output values. The different autoencoders share a trend which occurs due to the underlying input frame. Input images with more edges and contrast result in relative higher RMSE. At 339 frames, the CamVid autoencoder has the lowest reconstruction error. The YCOR is 52 times consulted, and the Cityscapes segmenter 62 times. A total of 14 model switches occur, which is less than the number of switches in the Bayesian uncertainty approach. A total of 62 false model selections in hybrid framework 2 occurred due to the YCOR autoencoder.

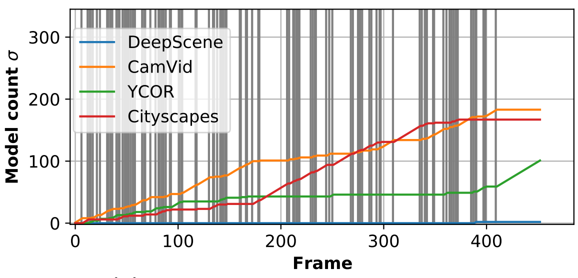
The inference time of both approaches is also measured during this experiment. Hybrid framework 1: Bayesian uncertainty (σ) can process 0.73 frames per second. While Hybrid framework 2: pixel reconstruction error (RMSE) obtains an FPS of 30.7. Hybrid framework 2 is thus 42 times faster than the Bayesian approach. This difference is caused by the number of dropout samples generated by Bayesian SegNet. In Hybrid framework 1, a total of 30 semantic segmentation predictions are generated to assess the model uncertainty, while for hybrid framework 2, this is only 1.



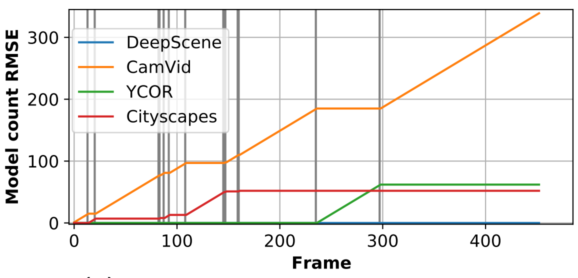
(a) Bayesian uncertainty (σ) over time



(b) Reconstruction error (RMSE) over time



(c) Model selection count lowest σ



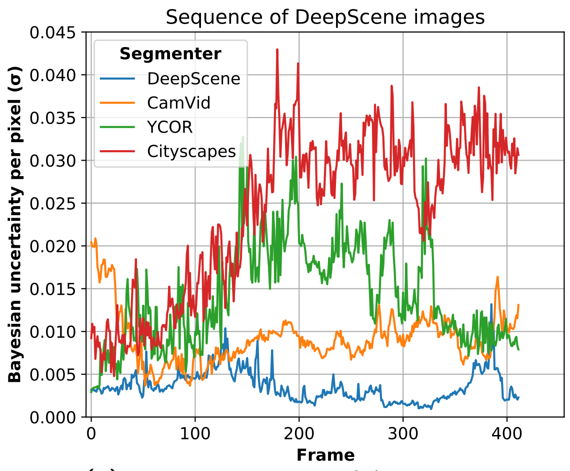
(d) Model selection count lowest RMSE

Figure 5-10: Sequential experiment based on (a) Bayesian uncertainty (σ) and (b) reconstruction error (RMSE) approach on sequence of 432 KITTI images. The KITTI images are captured in an urban environment. The vertical grey lines represent the frames at which the model switches to another environmental segmenter. In hybrid framework 1 the system switches 110 times between the different models (c). This is due to the inconsistent behaviour of the Bayesian uncertainty value. In hybrid framework 2 a more smooth output is obtained resulting in 14 model switches (d).

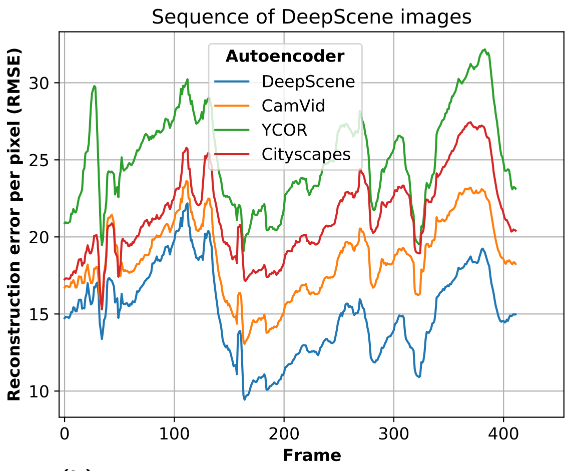
5-3-2 Sequence of DeepScene images

In this experiment, we process a sequence of 412 images from a sequential DeepScene dataset. Samples from this sequence are shown in Figure C-2 in the appendix. The decision parameter outputs of both the hybrid framework are shown in Figure 5-11. Hybrid framework 2 obtains a total of 27 false model selections. These false model selections occur at the beginning of the sequence, around frame 100 and at the end. Hybrid framework 1 switches a total of 23 times in this sequence. The DeepScene segmenter in the Bayesian uncertainty plot has a quite smooth trend. However, predictive uncertainty still occurs but is less than the other segmenters. Hybrid framework 2 in plot 5-11b has no false model selections. The DeepScene autoencoder had the lowest RMSE for all image frames in this sequence.

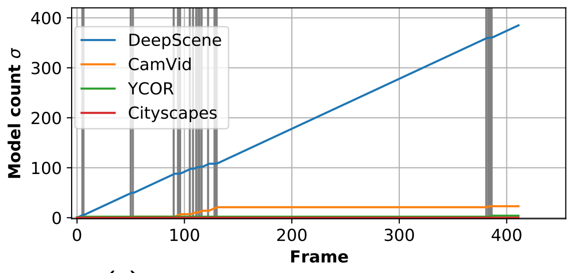
The difference in the number of false model selections can be due to the Bayesian SegNet's method to measure the uncertainty of its prediction. For obtaining the Bayesian uncertainty, dropout is used, this method uses a random parameter. The number of Monte Carlo samples is set to be 30. This results in different uncertainty values for the same image frame. The autoencoder approach does not incorporate any random variable. For that reason, it will



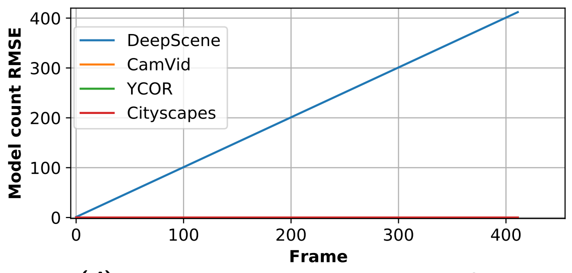
(a) Bayesian uncertainty (σ) over time



(b) Reconstruction error (RMSE) over time



(c) Model selection count lowest σ



(d) Model selection count lowest RMSE

Figure 5-11: Sequential experiment based on (a) Bayesian uncertainty σ and (b) reconstruction error (RMSE) approach on sequence of 412 DeepScene images. Lowest Bayesian uncertainty and pixel reconstruction error is expected for DeepScene the model. The vertical grey lines represent the frames at which the model switches to another environmental segmenter. However, in Bayesian uncertainty approach false model selection occur (c) while in the reconstruction error approach the correct model is selected for every input frame (d).

always result in the same reconstruction error for a particular frame, and a value that scales with changes in the input frame. Therefore, the reconstruction error of the current input frame i relates more to the previous frame $i-1$ than in the Bayesian case. In the DeepScene sequence experiment, we also measure the inference time. This results in a similar FPS rate as the KITTI sequential data experiment.

5-3-3 Conclusion

We conclude that using the autoencoder with its pixel reconstruction error as a decision parameter for the hybrid system is better than the Bayesian uncertainty measure. It results in 41 less false model selections and 96 less model switches in the KITTI sequence. In the DeepScene sequence experiment the Bayesian approach resulted in 27 times a false model selection while the RMSE approach did not result in any false model selection. In addition, the pixel reconstruction error measure is more reliable than the Bayesian uncertainty measure since no random parameter is involved. This approach results in more consistent behaviour and better model selection performance and recall. Furthermore, the autoencoder approach is 42 times computationally faster than the Bayesian approach.

Unfortunately, no dataset with a sequence of images and corresponding ground truth annotations is available. The training datasets in this thesis have annotated images but do not contain sequential data. Therefore, we cannot measure the semantic segmentation performance, IoU and PA, over time.

5-4 Bayesian uncertainty vs Reconstruction error

The previous experiments show how hybrid frameworks 1 and 2 perform on the test and sequential datasets. This section compares both the hybrid frameworks based on model selection performance and on a pixel-wise performance level.

5-4-1 Model selection decision parameter

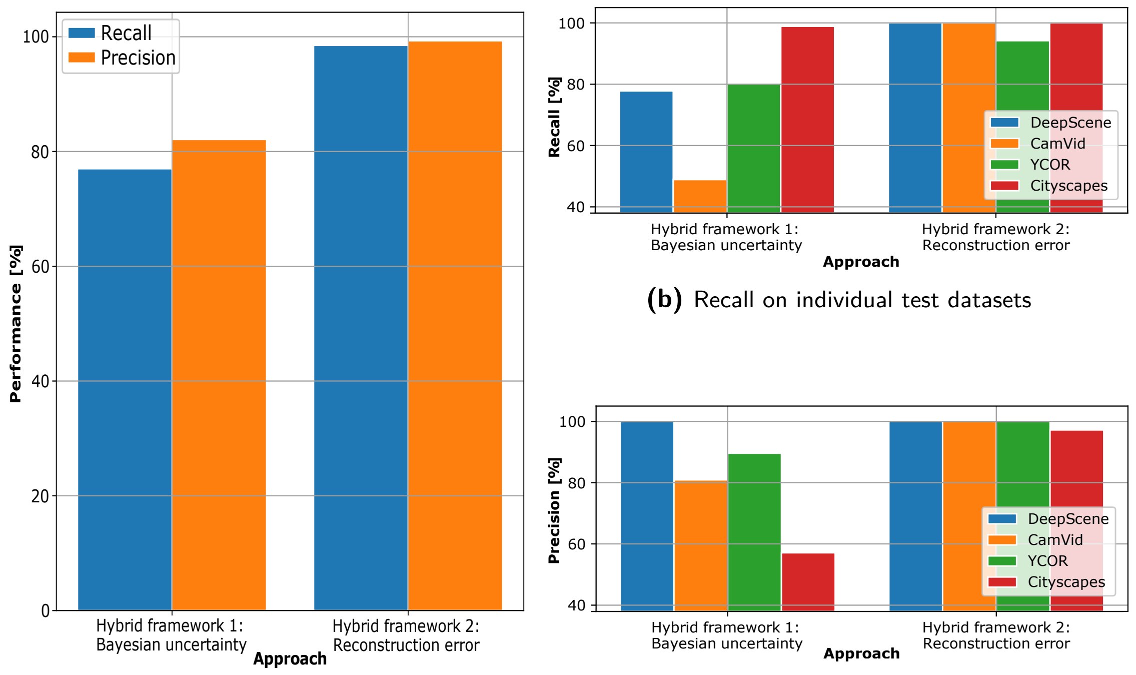
We compare the model selection performance of the two approaches discussed above in Figure 5-12. Hybrid framework 1: Bayesian uncertainty (σ) has a recall and precision of 77.0% and 82.1%, respectively. These values were calculated in Section **??** and given in Table 5-2. The other approach in this figure, hybrid framework 2: reconstruction error (RMSE), obtains a recall and precision score of 98.5% and 99.3%, respectively.

In Figure 5-12b and 5-12c, the recall on the individual test datasets and precision of the individual models is shown. In hybrid framework 1, the DeepScene segmenter has a model selection precision of 100%. The Cityscapes segmenter obtained the lowest model selection precision with 57.3%. We observe that in the underlying recall and precision plots of hybrid framework 2, only false model selections occur for the YCOR test dataset. The recall of YCOR is not 100% but 94.2%. These errors occur due to the Cityscapes autoencoder, which results in a precision of 97.2%.

The reconstruction error approach results in the highest recall and precision scores compared to the Bayesian approach as can be seen in the bar charts in 5-12. We thus conclude that Hybrid framework 2 with the pixel reconstruction error (RMSE) as the decision parameter outperforms the Bayesian uncertainty framework.

The correlation between the two designed decision parameters, mean Bayesian pixel uncertainty (σ) and mean pixel reconstruction error (RMSE), and the mean IoU is shown in Figure 5-13. For every environmental model, DeepScene (5-13a, 5-13b), CamVid (5-13c, 5-13d), YCOR (5-13e, 5-13f) and Cityscapes (5-13g, 5-13h), we plot the output value pairs of all the test images.

In the left column, the output data points of Hybrid framework 1: Bayesian uncertainty (σ) are plotted. We fit a line between the mean Bayesian pixel uncertainty and mean IoU data points and plot the correlation coefficient r in the figures for all the segmenters. The correlation coefficients of hybrid framework 1 are: $r_{\text{deepscene}} = -0.821, r_{\text{camvid}} = -0.41, r_{\text{ycor}} = -0.675, r_{\text{cityscapes}} = -0.74$, which indicates a negative correlation between the Bayesian uncertainty value (σ) and the mean IoU. We observe outliers at lower Bayesian uncertainty values in Figure 5-13c, which is reflected in the correlation coefficient of this segmenter. An example of such a data point was shown in Figure 5-3, where a relative low Bayesian uncertainty value of 0.003 is obtained with a corresponding IoU of 46.3%.



(a) Model selection performance **(b)** Recall on individual test datasets **(c)** Precision of individual models

Figure 5-12: Model selection performance comparison of Bayesian uncertainty and reconstruction error approach. The precision and recall are calculated over a total of 610 test images from the Camvid, DeepScene, YCOR and Cityscapes test datasets. The autoencoder approach with its reconstruction error as decision parameter outperforms the two other approaches. The underlying recall and precision of the hybrid frameworks are shown in (b) and (c) respectively.

The mean pixel reconstruction error (RMSE) and mean IoU pairs of hybrid framework 2 are shown in the right column of Figure 5-13. Here we already observe less regularity in the plots. The correlation coefficients of hybrid framework 2 are: $r_{\text{deepscene}} = -0.124, r_{\text{camvid}} = 0.282, r_{\text{ycor}} = -0.036, r_{\text{cityscapes}} = -0.009$. These correlation coefficients range from negative to positive. We, therefore, conclude that there is no clear correlation between the mean RMSE and mean IoU. The autoencoder does not assess the model uncertainty of the prediction. From the example shown in Figure 3-8 it can be seen that the RMSE values for the different input images differ a lot. By implementing the RMSE as a decision parameter for model selection, we can select the right model for the current input frame. However, the absolute mean RMSE of an image does not represent any form of uncertainty.

5-4-2 Pixelwise performance

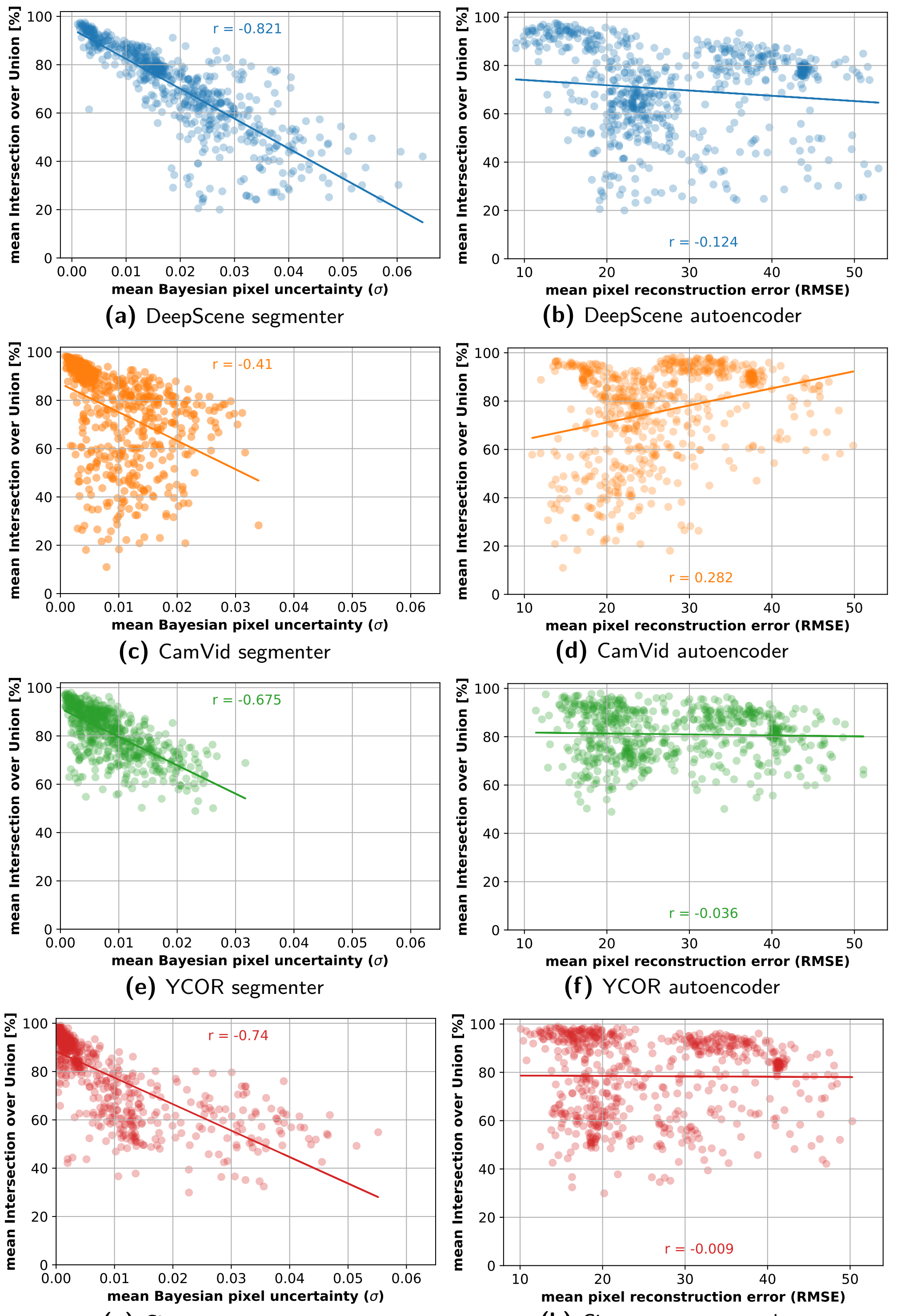
We investigate the relation between the two decision parameters, Bayesian uncertainty (σ) and the reconstruction error (RMSE), and semantic segmentation performance on a pixel level in Figure 5-14. We group all the predicted pixels of the 610 test images in percentile ranges sorted by either the Bayesian uncertainty pixel value (σ) or the Pixel reconstruction error (RMSE) shown in Figure 5-14a and 5-14b, respectively. Percentile range 90-100 thus contains 10% of the pixels with the highest Bayesian uncertainty value or pixel reconstruction error (RMSE). For every percentile range, we calculate the percentage of correctly classified pixels.

We observe accuracies higher than 95% in the lower Bayesian model uncertainty percentile ranges. The trend over the different percentile ranges demonstrates that Bayesian model uncertainty effectively measures pixel-wise confidence in prediction accuracy. We do not observe the same trend in the pixel reconstruction error plot as in the Bayesian uncertainty approach. We, however, observe a slight negative trend over the percentile ranges but cannot conclude that low classification accuracy occurs for high pixel reconstruction errors.

5-4-3 Conclusion

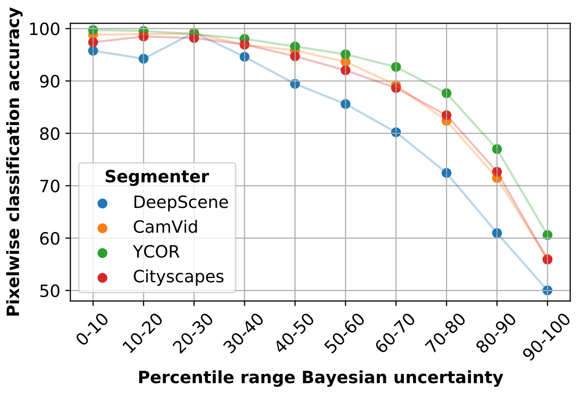
We conclude that hybrid framework 2 outperforms hybrid framework 1 on model selection performance. The pixel reconstruction error (RMSE) as a decision parameter with autoencoder architecture 21600 results in almost perfect model selection performance. A model selection precision of 99.3% is obtained on images of the test dataset and a recall of 98.5%. Hybrid framework 1 with Bayesian uncertainty as a decision parameter has for both the precision and recall lower scores on the test datasets.

Hybrid framework 2 is, however, not able to quantify uncertainty correlated to the models prediction. No clear correlation between the RMSE and segmentation performance is observed. Hybrid framework 1, on the other hand, shows a negative correlation between Bayesian uncertainty and IoU. This is assessed on both an image level as a pixel level. We thus conclude that the Bayesian uncertainty measure is an effective measure of semantic segmentation accuracy.

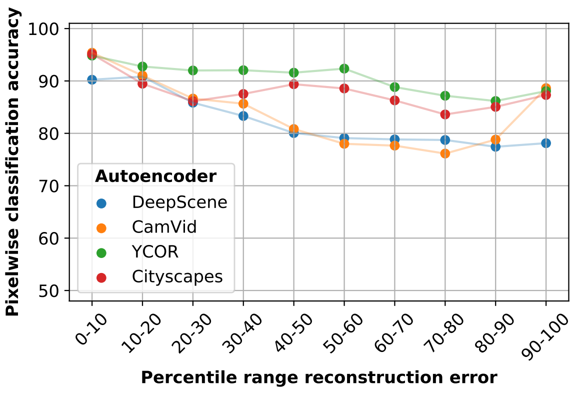


(a) DeepScene segmenter
(b) DeepScene autoencoder
(c) CamVid segmenter
(d) CamVid autoencoder
(e) YCOR segmenter
(f) YCOR autoencoder
(g) Cityscapes segmenter
(h) Cityscapes autoencoder

Figure 5-13: Performance of segmenters, measured by the IoU as a function of (a,c,e,g) mean Bayesian pixel uncertainty (σ) and (b,d,f,h) mean pixel reconstruction error (RMSE) for 610 test images. In the left column we observe that for increasing Bayesian pixel uncertainty less precise predictions are made. The Bayesian line fits and correlation coefficients show a negative correlation for every segmenter. In the right column there is no relation between the mean IoU and mean RMSE, the correlation coefficient r ranges from negative to positive values. Images with high mean RMSE do not correspond to bad semantic segmentation performance.



(a) Bayesian pixel uncertainty (σ)



(b) Pixel reconstruction error (RMSE)

Figure 5-14: Pixel-wise classification accuracy as a function of (a) Bayesian pixel uncertainty and (b) pixel reconstruction error. Every percentile range represents 10% of the pixels in the test dataset, sorted by either the pixel Bayesian uncertainty (a) or the pixel reconstruction error (b). Percentile range $90 - 100$ contains 10% off the pixels with the highest Bayesian uncertainty or pixel reconstruction error. The lines connecting the datapoints indicate the trend over the percentile ranges.

Chapter 6

Conclusion and Future work

This chapter discusses and concludes on the methodology and obtained results. First, in Section 6-1, we will summarize the main findings of this study. In Section 6-2 we answer the research questions and the two sub-questions. Lastly, we will propose future research possibilities in Section 6-3.

6-1 Summary of results

In this thesis, it is investigated how the introduction of a hybrid semantic segmentation framework can improve traversability estimation in unseen unstructured off-road environments for an Unmanned Ground Vehicle (UGV). The motivation behind this research is that a UGV has to be able to estimate traversability in a variety of environments including unstructured off-road environments. Challenges arise when the UGV enters these unstructured environments, where we encounter more diverse classes and less regularity than in urban environments. Roads in urban environments are designed to be primarily straight and regularized. Traffic signs and road surface markings indicate whether areas can be safely traversed, information which is unfortunately not available in unstructured off-road environments.

The diverse structured and unstructured environments require an approach that can estimate traversability for various situations. Therefore, we designed a hybrid semantic segmentation framework. In the hybrid framework, four environment-specific semantic segmentation networks are present that can accurately estimate traversability for a variety of environments. The semantic segmentation networks in the hybrid framework differ based on their training datasets. The datasets considered are the DeepScene, CamVid, YCOR and Cityscapes datasets, respectively, a forest, urban, off-road, and urban dataset. We evaluated the semantic segmentation performance of these segmenter by the Intersection over Union (IoU) and Pixel Accuracy (PA) measure. In the first experiment, we compared the potential performance of a hybrid framework with a single semantic segmentation network. The outcomes of these experiments showed that a hybrid framework outperforms a single deep semantic segmentation network with an increase in median IoU-score of 2.6% from 90.0% to 92.6%. The median

PA scores increased with 1.3% from 95.7% in the combined model to 97.0% in the hybrid potential framework. The full potential of a hybrid framework can only be reached if for every input image the best performing segmenter is selected. This selection is based on a decision parameter. In this thesis, we designed two hybrid frameworks with different decision parameters.

Hybrid framework 1 uses a Bayesian uncertainty measure as a decision parameter. The Bayesian semantic segmentation network approximates the predictive distribution by Monte Carlo dropout. The predictive distribution contains the variance from which model uncertainty can be estimated. The pixel-wise uncertainty is thus calculated by measuring the variance between the sampled predictions. For every input we have to select one of the four segmenters. We compare the Bayesian uncertainty mappings of the different segmenter. We eventually select the segmenter with the lowest mean Bayesian pixel uncertainty value for traversability estimation. This results in a median IoU score of 91.9%, which is 1.9% higher than the combined segmenter. However, it did not level the full potential of a hybrid semantic segmentation framework, which is 92.6%. The full potential of the hybrid framework can only be obtained if for every image of the test datasets the best performing segmenter is selected. In hybrid framework 1 this is not obtained due to false model selection based on the Bayesian uncertainty value. The model selection performance is evaluated by the precision and recall metrics. The Bayesian uncertainty measure as decision parameter results in a recall of 77.0% and a precision of 82.1%.

Hybrid framework 2 uses in addition to the environment-specific segmenter an autoencoder to obtain a reconstruction error. We measure the pixel reconstruction error between the input image and the reconstructed output image of the environment-specific autoencoder. This pixel reconstruction error (RMSE) functions as the decision parameter in this hybrid framework. The segmenter and the autoencoder are both trained on the same training dataset. The autoencoder encodes the image to a latent space representation. This latent space representation captures a limited amount of data from the input image. The original input image dimension is restored by a decoder from the latent space representation. This results in a reconstructed image of the input image. The reconstructed images of the four autoencoders differ based on what the autoencoder has seen before. The autoencoder weights are optimized for different features in the different environment-specific training datasets. We select the semantic segmentation prediction corresponding to the lowest pixel reconstruction error. We performed model selection experiments with different autoencoder architectures. We found that the autoencoder with a latent space size of 21600 datapoints results in the best model selection performance. A recall of 98.5% and precision of 99.3% is obtained.

The obtained results from the different experiments substantiate the answers to the research and sub-questions of this thesis. The two hybrid frameworks' performance is evaluated using the test datasets and an additional sequential dataset. The performance of the hybrid frameworks is quantified by evaluation metrics for semantic segmentation, IoU and PA, and model selection, precision and recall.

6-2 Conclusions

The implementation of a hybrid semantic segmentation framework for traversability estimation has been researched in this thesis. The research question of this thesis is:

How can a hybrid semantic segmentation framework improve traversability estimation in unseen unstructured off-road environments for unmanned ground vehicles?

To answer this research question, the formulated sub-questions are first answered:

1. *In what ways can unseen unstructured off-road environments be classified?*

   Literature has shown that deep semantic segmentation networks are a popular tool for traversability estimation. These deep learning models are trained to pixel-wise classify images. The obtained semantic segmentation mappings are used for traversability estimation through its environment.

   The performance of the different environmental models in our hybrid framework can only be compared if we have the same labels for all models. We, therefore, rewrite the labels of the DeepScene, CamVid, YCOR and Cityscapes datasets into the three labels: traversable and non-traversable and sky. Those three labels contain enough information to safely navigate the UGV through its environment.

   The experiments performed on the test datasets show an accurate performance of the semantic segmentation networks. The combined segmenter, which is trained on a combination of all training datasets, obtains a median IoU-score 90.0% and a PA of 95.7%.

2. *How can uncertainty of a semantic segmentation prediction be quantified?*

   Quantifying uncertainty in computer vision applications is of great importance in autonomous driving systems, such as a UGV. This is due to the fact that deep learning models are often overly confident even when they are operating in unseen or unstructured environments. In these situations, the UGV might be incompetent, while the underlying deep learning model itself may not be aware of this.

   We designed two different approaches for uncertainty quantification. Hybrid framework 1: Bayesian SegNet, uses dropout layers to obtain the model's prediction uncertainty and hybrid framework 2: Reconstruction error, which uses an autoencoder to estimate uncertainty by comparing the reconstructed image with the original input image.

   This subquestion can be answered by the obtained results of the test datasets experiments. We observed a negative correlation between the Bayesian pixel uncertainty and IoU-score in hybrid framework 1. We are thus able to correctly estimate model uncertainty by Monte Carlo sampling.

   However, the hybrid framework with the autoencoder approach, was not able to correctly quantify model uncertainty. No clear correlation between the pixel reconstruction error (RMSE) and IoU is observed. The absolute value of the pixel reconstruction error did not contain any information regarding uncertainty. This can be due to the fact that the pixel reconstruction error is not directly linked to the segmenter. It is trained on the same datasets but does not contain a direct link to the prediction.

With the two sub-questions answered, the answer to the research question is substantiated. There are multiple advantages of implementing a hybrid framework over a single deep semantic segmentation network. The primary advantage is the increase in semantic segmentation performance on IoU and PA. Both hybrid frameworks results in higher IoU scores compared to a single semantic segmentation network. Hybrid framework 2 with its pixel reconstruction error, outperforms hybrid framework 1 on model selection performance and therefore results in better semantic segmentation performance.

We proved that an autoencoder approach as decision parameter for the hybrid framework is able to select the segmenter corresponding to an environmental image. The autoencoder was able to learn features from the training dataset and capture these in the network its weights. This resulted in reconstructions that differ based on the environment it has been trained on. This difference was reflected in the pixel reconstruction error (RMSE). However, the autoencoders were not able to reconstruct the structural information of the training datasets in their reconstruction. The Structural Similarity Index Measure (SSIM) metric and grayscale autoencoder approach showed the autoencoders model selection performance mostly relied on the color compositions captured in the datasets.

Furthermore, we observed in the sequential data experiments that the Bayesian uncertainty approach results in a non-smooth uncertainty trend over-time. This trend occurs due to the method in which the uncertainty is estimated. The Monte Carlo sampling method uses dropout during approximation of the predictive distribution. In this method a random parameter is involved that affects the prediction and the uncertainty mapping. This randomness influenced the model selection performances in the sequential data and the test dataset experiments. This behaviour results in a lot of model switches in the sequential data experiments and bad model selection performance on the test dataset experiments.

Another practical advantage of a hybrid framework, is that additional environmental segmenters can be added after the system is deployed. This has as advantage that we do not need to retrain the whole model, which is very time-consuming.

To conclude, the designed hybrid semantic segmentation frameworks can outperform a single deep semantic segmentation network. Both the designed approaches, however, have their advantages and disadvantages. Hybrid framework 1 with the Bayesian pixel uncertainty has proven to be able to correctly quantify prediction uncertainty. A disadvantage of this framework is the bad model selection performance. Hybrid framework 2, on the other hand, has accurate model selection performance but is not able to quantify prediction uncertainty. Ideally, we want to have the best of both worlds. This, however, will result in an even more complex framework. The designed hybrid semantic segmentation framework thus should be further improved to resolve the disadvantages. As a start, we make some recommendations for future research in the following section.

6-3 Future research

This thesis shows how a hybrid semantic segmentation framework can be designed for traversability estimation in a variety of environments. It would have been interesting to investigate this field further. Suggestions for future work on the continuation of this thesis and different problems encountered in the construction of this thesis are:

1. **Virtual world data:** To overcome the problem of limited data, simulations in virtual environments can be performed. Using virtual environments for data collection has as advantage that ground truth annotations can be generated directly from the virtual

data. This will result in the ability to collect a huge amount of data within different environments using the same camera settings, such as camera type, height, resolution and lighting conditions. More experiments can be performed with this approach, and hopefully, structural information can be extracted from the different environmental datasets.

2. **Sequential data:** With the virtual form of data collection, sequential information can be incorporated into the model. This will enable us to combine the hybrid framework with a recurrent neural network since image i will likely have similar features as image $i-1$. Long Short-Term Memory (LSTM) are a type of recurrent neural networks which captures sequential information over time. The sequential annotated data will also enable comparison based on semantic segmentation performance since we then have the ground truth annotation of the sequence.

3. **Number of classes:** We rewrote all the labels of the original datasets into traversable, non-traversable and sky. This was for performance evaluation and comparison of the environment-specific segmentation networks. It would be interesting to define more classes in the segmentation networks in future research. The hybrid framework with the autoencoder can have different classes defined in the in parallel placed environmental segmenters since the decision parameter does not rely on semantic segmentation prediction.

4. **Autoencoder architectures:** In this research, we experimented with five different autoencoder architectures, consisting of convolutional, MaxPooling and UpSampling layers. In future research, we are interested in incorporating other neural network layers, such as average pooling layers and dilated convolutional layers [61]. Different autoencoder architectures are might be able to quantify uncertainty or detect situations it has not seen before.

5. **Depth information:** For the pixel-wise annotated predictions, we used RGB images as the input to the deep semantic segmentation networks. In addition to the RGB values, we can incorporate depth information. This requires some relatively simple modifications to the neural network design. Instead of the three channel RGB input image we have an additional channel containing the depth information. We thus have to adjust our network to an input shape with four channels, RGBD. Here the D represents the depth information, which can be captured by stereovision or LiDAR cameras. This can improve semantic segmentation performance as proved by [56].

6. **Human-machine interaction:** The implementation of a hybrid semantic segmentation framework enables human operators to interact with the machine. A human operator can assist the UGV in selecting the best predictions since there are multiple predictions to choose from. Interaction with a human operator is necessary if the UGV gets, for example, stuck, no traversable area is defined. The human operator can then visually inspect the previous predictions of the different environmental segmenters and select the best prediction.

Chapter 7

Glossary

List of Acronyms

ANN	Artificial Neural Network
BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
FPS	Frames per second
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mIoU	mean Intersection over Union
MSE	Mean Squared Error
PA	Pixel Accuracy
SGD	Stochastic gradient descent
SSIM	Structural Similarity Index Measure
TNO	Netherlands Organisation for Applied Scientific Research
TN	True Negative
TP	True Positive
UGV	Unmanned Ground Vehicle

Appendix A

Supportive Tables

A-1 Semantic Segmentation Performance

Table A-1: Intersection over Union (IoU) of different segmenters on test datasets. Last row contains the combined segmenter that is trained on a combination training images of the other segmenters.

	DeepScene test data			CamVid test data			YCOR test data			Cityscapes test data		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
DeepScene	61.50	90.81	97.50	55.84	76.59	93.45	20.00	41.17	79.27	32.69	61.81	91.17
CamVid	10.98	45.31	74.38	60.43	92.12	98.22	28.24	73.76	94.57	35.69	74.71	93.90
YCOR	48.92	70.96	90.13	50.28	82.18	96.34	54.80	82.51	97.05	58.28	85.54	97.90
Cityscapes	32.42	59.16	81.04	48.75	80.88	96.06	29.92	69.35	94.89	57.07	92.52	99.01
Combined	58.32	86.04	95.39	59.99	89.43	97.03	48.45	78.96	97.36	62.09	92.16	98.73

Table A-2: Semantic segmentation performance of the different approaches measured by the IoU and Pixel Accuracy (PA) on the test dataset.

	Intersection over union				Pixel accuracy			
	min	mean	median	max	min	mean	median	max
Hybrid framework potential	54.80%	90.63%	92.57%	99.01%	63.75%	95.88%	97.02%	99.57%
Combined	48.45%	88.08%	89.99%	98.73%	62.00%	94.28%	95.66%	99.45
Hybrid framework 1: Bayesian uncertainty	26.32%	88.30%	91.90%	99.01%	38.49%	93.81%	96.64%	99.57%

A-2 Model Selection Performance

Table A-3: Model selection based on Bayesian uncertainty measure. The numbers in bold represent correct model selections.

		Test Dataset				
		DeepScene	CamVid	YCOR	Cityscapes	Precision
Model	DeepScene	**91**	0	0	0	100%
	CamVid	23	**114**	4	0	80.9%
	YCOR	0	6	**69**	2	89.6%
	Cityscapes	3	113	13	**171**	57.1%
	Recall	77.8%	48.9%	80.2%	98.9%	

Table A-4: Model selection based on RMSE between average training image and input image. The numbers in bold represent correct model selections.

		Test Dataset				
		DeepScene	CamVid	YCOR	Cityscapes	Precision
Model	DeepScene	**95**	1	5	0	94.1%
	CamVid	5	**142**	16	9	82.6%
	YCOR	3	0	**62**	0	95.4%
	Cityscapes	14	90	3	**165**	60.7%
	Recall	81.2%	60.9%	72.1%	94.8%	

Table A-5: Model selection based reconstruction error of autoencoder architecture 21600. The numbers in bold represent correct model selections.

		Test Dataset				
		DeepScene	CamVid	YCOR	Cityscapes	Precision
Model	DeepScene	**117**	0	0	0	100%
	CamVid	0	**233**	0	0	100%
	YCOR	0	0	**81**	0	100%
	Cityscapes	0	0	5	**174**	97.2%
	Recall	100%	100%	94.2%	100%	

Appendix B

Autoencoder Architectures

Listing B.1: Autoencoder architecture with latent space size 600

```python
def autoencoder_600():
    input_img=Input(shape=(360,480,3))
    x=Conv2D(8,(2,2),activation='relu',padding='same')(input_img)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((3,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((3,2),padding='same')(x)
    encoded=Conv2D(8,(3,3),activation='relu',padding='same')(x)

    direct_input=Input(shape=(5,15,8))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(direct_input)
    x=UpSampling2D((3,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((3,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    decoded=Conv2D(3,(1,1),activation='sigmoid',padding='same')(x)

    encoder=Model(input_img,encoded)
    decoder=Model(direct_input,decoded)
    autoencoder=Model(input_img,decoder(encoded))
    autoencoder.compile(optimizer='ADAM',loss='mean_squared_error')
    return autoencoder,encoder,decoder
```

Listing B.2: Autoencoder architecture with latent space size 3600

```
def autoencoder_3600():
    input_img=Input(shape=(360,480,3))
    x=Conv2D(8,(2,2),activation='relu',padding='same')(input_img)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((3,2),padding='same')(x)
    encoded=Conv2D(8,(3,3),activation='relu',padding='same')(x)

    direct_input=Input(shape=(15,30,8))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(direct_input)
    x=UpSampling2D((3,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    decoded=Conv2D(3,(1,1),activation='sigmoid',padding='same')(x)

    encoder=Model(input_img,encoded)
    decoder=Model(direct_input,decoded)
    autoencoder=Model(input_img,decoder(encoded))

    autoencoder.compile(optimizer='ADAM',loss='mean_squared_error')
    return autoencoder,encoder,decoder
```

Listing B.3: Autoencoder architecture with latent space size 21600

```
def autoencoder_21600():
    input_img=Input(shape=(360,480,3))
    x=Conv2D(8,(2,2),activation='relu',padding='same')(input_img)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    encoded=Conv2D(8,(3,3),activation='relu',padding='same')(x)

    direct_input=Input(shape=(45,60,8))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(direct_input)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)
    decoded=Conv2D(3,(1,1),activation='sigmoid',padding='same')(x)

    encoder=Model(input_img,encoded)
    decoder=Model(direct_input,decoded)
    autoencoder=Model(input_img,decoder(encoded))

    autoencoder.compile(optimizer='ADAM',loss='mean_squared_error')
    return autoencoder,encoder,decoder
```

Listing B.4: Autoencoder architecture with latent space size 86400

```
def autoencoder_86400():
    input_img=Input(shape=(360,480,3))
    x=Conv2D(8,(2,2),activation='relu',padding='same')(input_img)
    x=MaxPooling2D((2,2),padding='same')(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=MaxPooling2D((2,2),padding='same')(x)
    encoded=Conv2D(8,(3,3),activation='relu',padding='same')(x)

    direct_input=Input(shape=(90,120,8))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(direct_input)
    x=UpSampling2D((2,2))(x)
    x=Conv2D(8,(3,3),activation='relu',padding='same')(x)
    x=UpSampling2D((2,2))(x)

    decoded=Conv2D(3,(1,1),activation='sigmoid',padding='same')(x)

    encoder=Model(input_img,encoded)
    decoder=Model(direct_input,decoded)
    autoencoder=Model(input_img,decoder(encoded))

    autoencoder.compile(optimizer='ADAM',loss='mean_squared_error')
    return autoencoder,encoder,decoder
```

Listing B.5: Autoencoder architecture with latent space size 345600

```
def autoencoder_345600():
    input_img=Input(shape=(360,480,3))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(input_img)
    x=MaxPooling2D((2,2),padding='same')(x)
    encoded=Conv2D(8,(3,3),activation='relu',padding='same')(x)

    direct_input=Input(shape=(180,240,8))
    x=Conv2D(8,(3,3),activation='relu',padding='same')(direct_input)
    x=UpSampling2D((2,2))(x)
    decoded=Conv2D(3,(1,1),activation='sigmoid',padding='same')(x)

    encoder=Model(input_img,encoded)
    decoder=Model(direct_input,decoded)
    autoencoder=Model(input_img,decoder(encoded))

    autoencoder.compile(optimizer='ADAM',loss='mean_squared_error')
    return autoencoder,encoder,decoder
```

Appendix C

Sequential Data Experiments

C-1 KITTI City Sequence

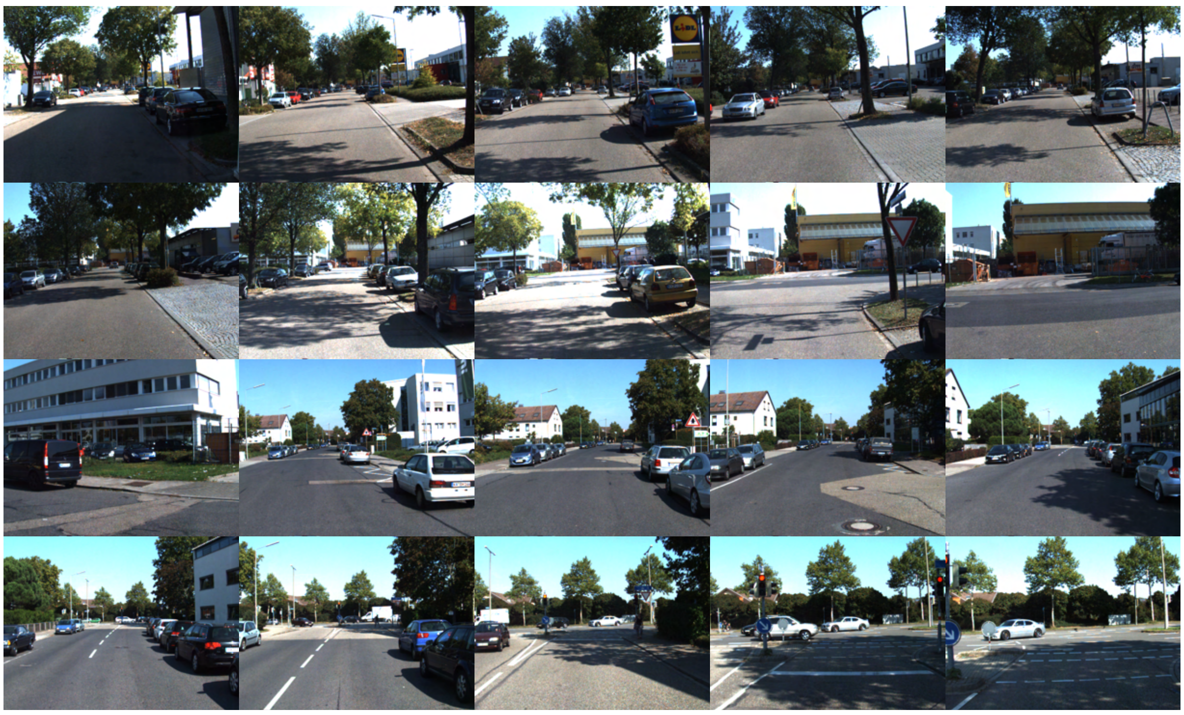


Figure C-1: Every 20'th image from the KITTI City sequence dataset. Images adapted from: [21].

C-2 DeepScene Sequence

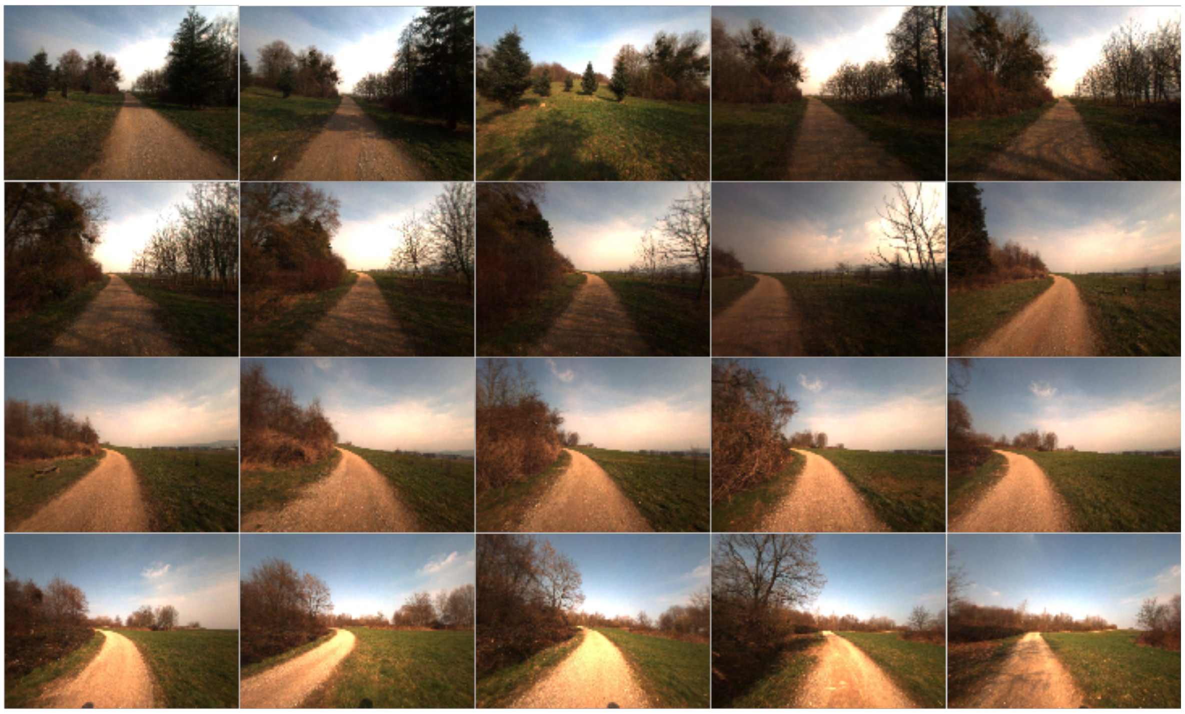


Figure C-2: Every 20'th image from the DeepScene sequence dataset. Images adapted from: [56].

Bibliography

[1] Amazon Web Services (AWS). Bayesian neural network with monte carlo dropout. Retrieved from: https://docs.aws.amazon.com/prescriptive-guidance/latest/ml-quantifying-uncertainty/mc-dropout.html, Accessed: January 2022.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[3] Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Understanding symmetries in deep networks, 2015.

[4] Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 203–210, 2017.

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

[6] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Image classification using random forests and ferns. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[7] Leon Bottou. Stochastic gradient descent trick. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[8] Facundo Bre, Juan M. Gimenez, and Víctor D. Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158:1429–1441, 2018.

[9] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:3213–3223, 2016.

[11] National Research Council. *Technology Development for Army Unmanned Ground Vehicles*. The National Academies Press, Washington DC, 2002.

[12] Travis J. Crayton and Benjamin Mason Meier. Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy. *Journal of Transport Health*, 6:245–252, 2017.

[13] G Cybenko. Approximation by Superpositions of a Sigmodial Function, 1989.

[14] Yinglong Dai and Guojun Wang. Analyzing tongue images using a conceptual alignment deep autoencoder. *IEEE Access*, 6:5962–5972, 2018.

[15] Yann N. Dauphin, Harm De Vries, and Yoshua Bengio. RMSProp and equilibrated adaptive learning rates for non-convex optimization. *Advances in Neural Information Processing Systems*, 2015-Janua(April):1504–1512, 2015.

[16] J. S. Denker and Yann Lecun. Transforming neural-net output levels to probability distributions. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems (NIPS 1990), Denver, CO, April 1991*, volume 3. Morgan Kaufmann, 1991.

[17] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[18] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, 2015.

[19] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[20] D Geçmen. Deep Learning Techniques for Low-Field MRI. 2020.

[21] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. *The International Journal of Robotics Research*, (October):1–6, 2013.

[22] Martin Genzel, Jan Macdonald, and Maximilian März. Solving inverse problems with deep neural networks – robustness included? *arXiv*, pages 1–29, 2020.

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

[24] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[25] Dorde T. Grozdic and Slobodan T. Jovicic. Whispered speech recognition using deep denoising autoencoder and inverse filtering. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(12):2313–2322, dec 2017.

[26] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[28] Jaccard. The distribution of the flora of the alpine zone. In *New Phytologist*, volume 11, pages 37–50, 1912.

[29] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *British Machine Vision Conference 2017, BMVC 2017*, 2017.

[30] Mohamed Kerkech, Adel Hafiane, and R. Canals. Vine disease detection in uav multispectral images using optimized image registration and deep learning segmentation approach. 12 2019.

[31] Dongshin Kim, Jie Sun, Sang Min Oh, James M. Rehg, and Aaron F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006(May):518–525, 2006.

[32] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[33] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010.

[34] Yann Lecun and Yoshua Bengio. *Convolutional networks for images, speech, and time-series.* MIT Press, 1995.

[35] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Johnson. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 152:166–177, 04 2019.

[36] Meng Ma, Chuang Sun, and Xuefeng Chen. Deep coupling autoencoder for fault diagnosis with multimodal sensory data. *IEEE Transactions on Industrial Informatics*, 14(3):1137–1145, 2018.

[37] Travis Manderson, Stefan Wapnick, David Meger, and Gregory Dudek. Learning to Drive off Road on Smooth Terrain in Unstructured Environments Using an On-Board Camera and Sparse Aerial Images. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1263–1269, 2020.

[38] Mathworks. Convolutional neural network for image classification. Retrieved from: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html, Accessed: January 2022.

[39] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-Time Semantic Mapping for Autonomous Off-Road Navigation. pages 335–350, 2018.

[40] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image Segmentation Using Deep Learning: A Survey. *Journal of Physics: Conference Series*, 1712(1):012016, jan 2020.

[41] Jeremy Nixon, Mike Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. *CoRR*, abs/1904.01685, 2019.

[42] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1520–1528, 2015.

[43] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *CoRR*, abs/1711.00614, 2017.

[44] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet : A Deep Neural Network Architecture for Real-Time Semantic Segmentation. pages 1–10, 2016.

[45] Dean A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. San Francisco, CA: Morgan Kaufmann, 1989.

[46] William K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience, USA, 2007.

[47] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[48] Sebastian Ruder. An overview of gradient descent optimization algorithms. pages 1–14, 2016.

[49] Kelley M. Sayler. Artificial Intelligence and National Security. November 2020.

[50] Mennatullah Siam, Sara Elkerdawy, Martin Jagersand, and Senthil Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges, 2017.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[53] Ardi Tampuu, Maksym Semikin, Naveed Muhammad, Dmytro Fishman, and Tambet Matiisen. A survey of end-to-end driving: Architectures and training methods. *arXiv*, 2020.

[54] Yingshui Tan, Baihong Jin, Alexander Nettekoven, Yuxin Chen, Yisong Yue, Ufuk Topcu, and Alberto Sangiovanni-Vincentelli. An encoder-decoder based approach for anomaly detection with application in additive manufacturing. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1008–1015, 2019.

[55] A.C. Turner. Trailblazers of unmanned ground vehicles. *Military Review*, 99.

[56] Abhinav Valada, Gabriel L. Oliveira, Thomas Brox, and Wolfram Burgard. Deep Multispectral Semantic Scene Understanding of Forested Environments Using Multimodal Fusion. pages 465–477, 2017.

[57] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[58] Zhou Wang and Alan Bovik. Bovik, a.c.: Mean squared error: love it or leave it? - a new look at signal fidelity measures. ieee sig. process. mag. 26, 98-117. *Signal Processing Magazine, IEEE*, 26:98 – 117, 02 2009.

[59] Andrew Williams. *Autonomous Systems: Issues for Defence Policymakers.* 10 2015.

[60] Scott Cheng-Hsin Yang, Wai Vong, Ravi Sojitra, Tomas Folke, and Patrick Shafto. Mitigating belief projection in explainable artificial intelligence via bayesian teaching. *Scientific Reports*, 11, 05 2021.

[61] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.

[62] Jun Yu, Chaoqun Hong, Yong Rui, and Dacheng Tao. Multitask autoencoder model for recovering human poses. *IEEE Transactions on Industrial Electronics*, 65(6):5060–5068, 2018.

[63] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8:58443–58469, 2020.

[64] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11207 LNCS:418–434, 2018.

[65] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:6230–6239, 2017.