BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

SEMANTIC SEGMENTATION OF IMAGES FROM OFF-ROAD ENVIRONMENT

SÉMANTICKÁ SEGMENTACE SNÍMKŮ Z TERÉNNÍHO PROSTŘEDÍ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bára Spilková

SUPERVISOR
VEDOUCÍ PRÁCE

Ing. Roman Adámek

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Bachelor's Thesis

Institut:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Bára Spilková**
Degree programm:	Mechatronics
Branch:	no specialisation
Supervisor:	**Ing. Roman Adámek**
Academic year:	2023/24

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Bachelor's Thesis:

Semantic segmentation of images from off–road environment

Brief Description:

Semantic segmentation of image data is used to divide the individual parts of an image into predefined classes based on their meaning. Nowadays, it is mainly encountered in conjunction with convolutional neural networks.

In the case of semantic segmentation of images of the terrain environment, we are concerned with the division of individual parts of the image into categories corresponding to, for example, a road, a puddle, grass, bushes, a tree, etc. The goal of this process is to extract semantic information about the environment in which, for example, a mobile robot is moving and use it for motion planning.

Bachelor's Thesis goals:

1) Perform a survey of the current state of the art in semantic segmentation of terrain images. Include different methods and available datasets of annotated images in the research.

2) Based on the search, select at least two pre–trained best performing methods and compare them against each other on different available image datasets. Select methods suitable for real–time image processing. Compare the methods based on classification accuracy and computational power. Use images of the terrain environment taken at different landscape types, seasons, weather, and times of day.

3) Conduct experiments in which you train existing neural networks based on images with different numbers of classification categories and again compare the methods.

4) Test the most appropriate neural network type and number of categories on your own set of images.

Recommended bibliography:

[1] CHOLLET, François. 2019. Deep learning v jazyku Python: knihovny Keras, Tensorflow. Přeložil Rudolf PECINOVSKÝ. Praha: Grada Publishing. Knihovna programátora (Grada).

[2] De Marchi, Leonardo, and Laura Mitchell. 2019. Hands-On Neural Networks: Learn How to Build and Train Your First Neural Network Model Using Python. Birmingham, England: Packt Publishing.

[3] Michelucci, Umberto. 2019. Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection. 1st ed. Berlin, Germany: APress.

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2023/24

In Brno,

L. S.

prof. Ing. Jindřich Petruška, CSc. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

Abstrakt

Hlavním cílem bakalářské práce je prozukoumání různých metod sémantické segmentace snímků z off-road terénu. V rešeršní části jsou popsány základní principy sémantické segmentace, různé přístupy k tomuto problému, metody sémantické segmentace a různé datové sady. Dále je popsán proces evaluace a trénování několika modelů s rozdílnými parametry a vytvoření nového evaluačního datasetu. Získané výsledky jsou porovnány s výsledky z rešeršní části a jsou navrhnuty další kroky pro zvýšení přesnosti modelů.

Summary

The main focus of the bachelor's thesis is to explore different semantic segmentation methods of off-road terrain images. In the theoretical survey are described the basic principles of semantic segmentation, multiple approaches to the problem, the methods of semantic segmentation and different datasets. The process of evaluating and training multiple models with various parameters and the creation of a new evaluating dataset are described next. The attained results are compared to the results from the theoretical survey and the next steps for improving the accuracy of the models are proposed.

Klíčová slova

sémantická segmentace, rozpoznání obrazu, neuronové sítě, off-road terén

Keywords

semantic segmentation, image recognition, neural networks, off-road terrain

Bibliographic citation

SPILKOVÁ, B. *Semantic segmentation of images from off-road environment.* Brno: Brno University of Technology, Faculty of Mechanical Engineering, 2024. 44 pages, Thesis supervisor: Ing. Roman Adámek.

I declare that I have created the bachelor's thesis on the topic **Semantic segmentation of images from off-road environment** independently with the use of scientific and technical literature and sources listed in the appendix to this thesis.

Bára Spilková

Brno . . . .

I would like to thank my thesis supervisor Ing. Roman Adámek, for his help in the preparation of my thesis and his willingness to answer all my questions. Next, I would like to thank my friends and family, especially my father, for their support and technical advice.

Bára Spilková

Contents

1 Introduction

Semantic segmentation is a process used in image recognition where the objects in an image are classified pixel-wise. It is used mostly in autonomous driving and medicine. The state-of-the-art of semantic segmentation is advanced in the field of autonomous driving, however, most of the existing models and datasets are created and specifically tailored for urban environments.

This work includes a theoretical survey which introduces the concept of semantic segmentation and its differences from other types of image recognition. The next two parts of the survey are focused on the two different approaches used in this field - the traditional approach, which takes advantage of the architectures using machine learning and the neural network approach, where the architectures of segmentation models depend on various types of neural networks. Next are introduced common key performance indicators used for evaluation based on the performance of the models, existing models and methods of semantic segmentation from off-road terrain and various datasets which differ in captured environments and number of classes.

The experiments were performed based on the findings from the theoretical survey, where two different pretrained models were chosen for further evaluation on three datasets, also chosen based on the theoretical survey. It is described how the datasets had to be adjusted for proper evaluation.

Next, six models with various training parameters - different number of epochs, batch sizes and backbone architectures - were chosen, again based on the theoretical survey, and trained. Some of these parameters were chosen as close to those of the models described in the research as possible for proper comparison of the results. All models are compared based on their speed of segmentation of one image (also referred as latency) and their accuracy of the segmentation.

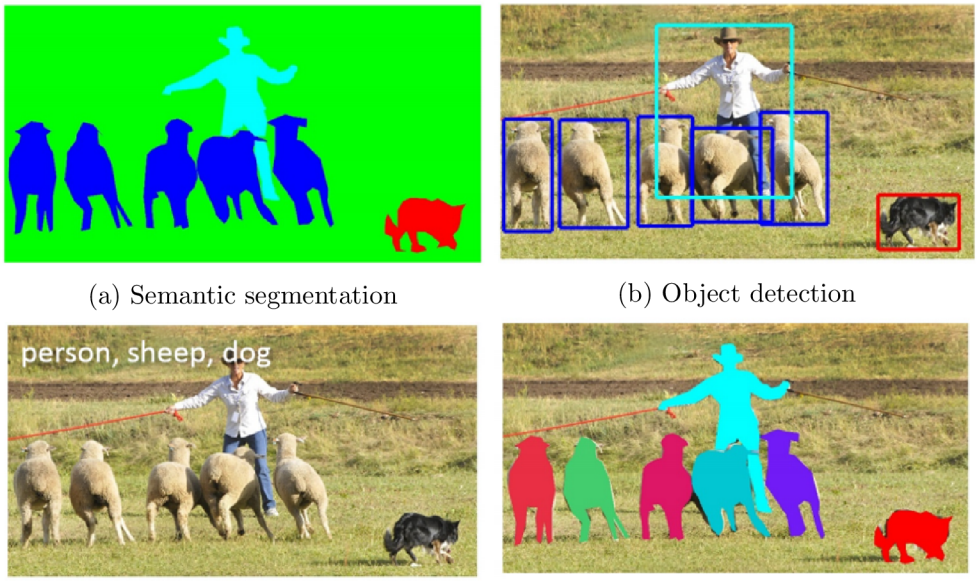
Finally, a new evaluation dataset was created and the model with one of the lowest values of latency and highest values of mean accuracy was used to perform semantic segmentation on this dataset. All of the findings and results were then analyzed and further steps for improvement were proposed.

2 Theoretical survey

The theoretical survey is focused on the basics of semantic segmentation and introducing important concepts in this field (chapter 2.1). Semantic segmentation with the use of the traditional approach is described in chapter 2.2, neural network approach in chapter 2.3. Another part of the research is on key performance indicators (chapter 2.4), various datasets with off-road terrain images (chapter 2.5) and different methods and models (chapter 2.6)

2.1 Semantic segmentation

Image recognition is a process of identifying objects in an image. There are many types of image recognition, but the most common are semantic segmentation, object detection, image classification and instance segmentation. Each type of image recognition can be used for different purposes, for example, object detection is commonly used for autonomous driving [1], instance segmentation in medicine. The main focus of this work is on semantic segmentation, which is typically used for terrain mapping, either from terrain or from satellite images. A comparison of semantic segmentation, object detection, image classification and instance segmentation is shown in Figure 2.1.



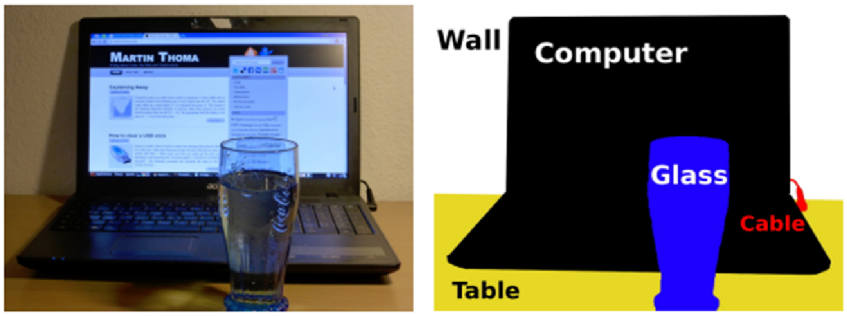
(a) Semantic segmentation (b) Object detection

(c) Image classification (d) Instance segmentation

Figure 2.1: Comparison of semantic segmentation, object detection, image classification, instance segmentation [2]

Semantic segmentation is a task where an image is divided into areas that can be assigned different object classes based on certain criteria. In semantic segmentation, the

whole image is classified, as opposed to instance segmentation, where only some objects are segmented. It is a pixel-wise operation, the neighbouring pixels are grouped and are given a class. These classes are usually known beforehand. A simple example of semantic segmentation can be seen in Fig. 2.2.



(a) Original image (b) Segmented image

Figure 2.2: Example of semantic segmentation

A computer with a glass of water in front of it can be seen in the original image (Figure 2.2a). Figure 2.2b shows the original image segmented into areas that have assigned classes. A dataset containing pairs of these original images and labelled images is required for the training and evaluation of semantic segmentation models.

2.2 Traditional approach

Semantic segmentation algorithms applying traditional methods are machine learning (ML) algorithms. These methods can be used individually or in combination [1]. Nowadays are not used for tasks requiring fast segmentation, only smaller applications, but these principles are still used as a part of more complicated algorithms using neural networks. It is important to note that algorithms using a traditional approach can be faster than neural networks, depending on the given task.

2.2.1 Sliding Window approach

Semantic segmentation mostly applies a Sliding Window pipeline (Fig. 2.3). The classifier is trained on images that are all the same size. It is given rectangle-shaped patches of images that are called windows. This process can classify only the central pixel [1].

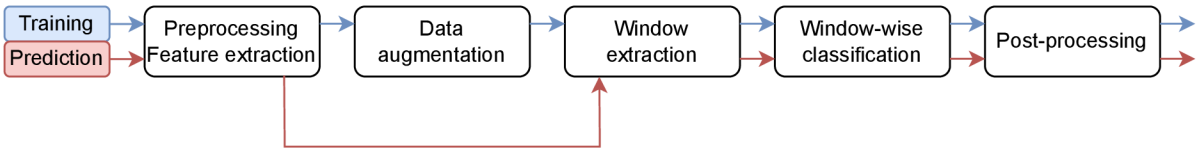


Figure 2.3: Sliding window approach

2.2.2 Markov Random Fields

Markov Random Fields (MRFs) is an undirected probabilistic model that assigns a random variable to every pixel of an image and every feature. For example, an image with the size of 224px $\times$ 224px that gets its variables according to RGB has $224 \cdot 224 \cdot 3 + 224 \cdot 224 =$

200704 random variables. MRFs assume that the probability of a variable depends only on its neighbouring variables. This method requires a large amount of computation due to a complicated structure [1, 3].

2.2.3 Conditional Random Fields

Conditional Random Fields (CRFs) is a machine learning method created for labelling and segmentation, it is an undirected discriminative graphic model. This method focuses not only on the distribution of the data but also on possible label sequence [1, 3]. CRF with 4-neighbourhood where each node x_i represents a pixel and node y_i represents a label can be seen in Fig. 2.4.

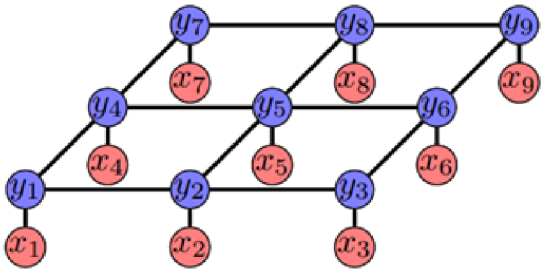


Figure 2.4: CRF with 4-neighbourhood [1]

2.2.4 Random decision forests

Random decision forests (RDFs) apply the ensemble learning technique, meaning that more than one class is trained and their hypotheses are combined. Every class is trained on a random subspace and the training is done on a random subset of training data. The classifier is called a decision tree. Every node uses one or more features to decide which branch of the algorithm to pass on (Figure 2.5). Nodes are added to the decision tree until every part of the tree consists only of nodes of the same class or until the branch can't be separated further. Training and predictions with RDFs are faster compared to previous methods [1].

2.2.5 Support Vector Machines

Support Vector Machines (SVMs) are binary classifiers. The data is described as (x_i, y_i), where x_i is the feature vector and $y_i \; \epsilon \; \{-1, 1\}$ is the binary label. SVMs separate data linearly in *feature space*, if the dataset is not linearly separable it is transformed into a higher dimensional space. The detailed principles of SVMs are explained in [4].

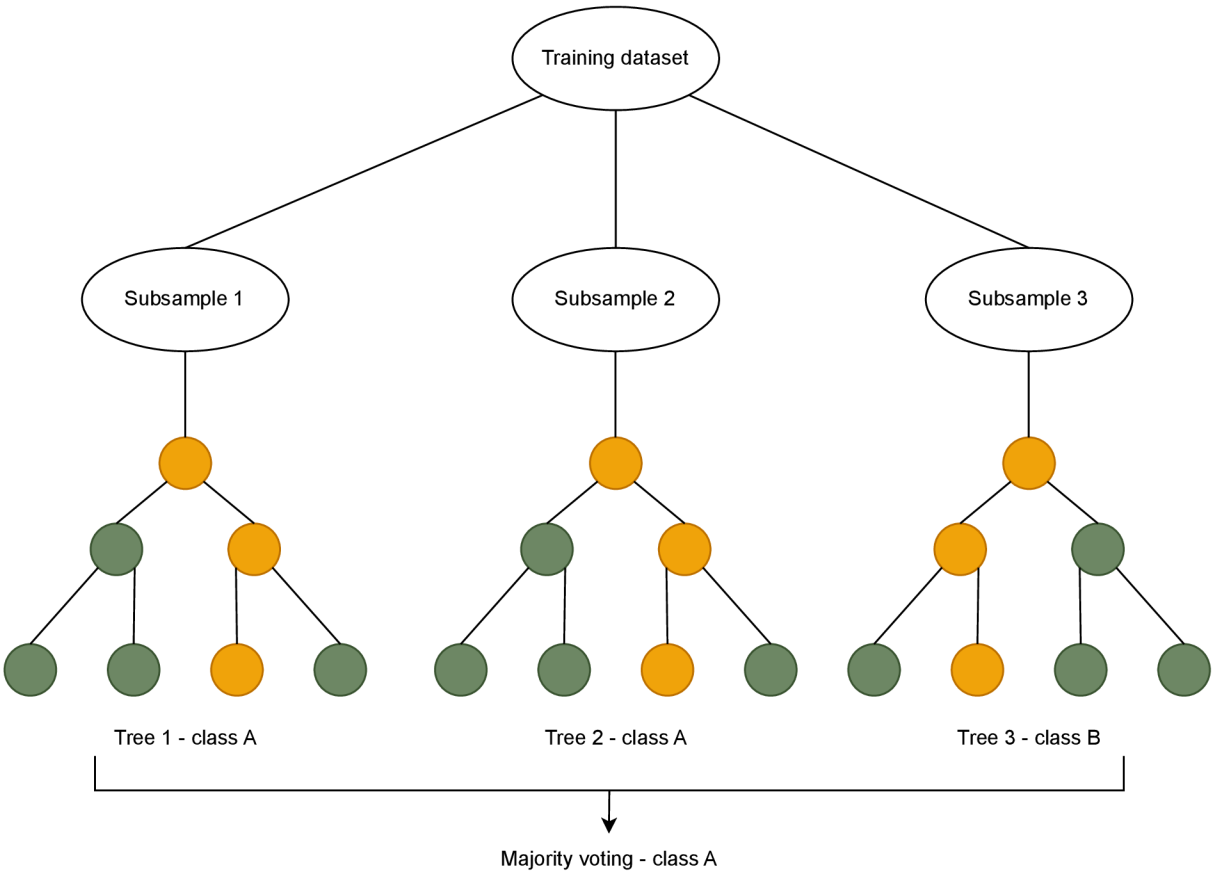


Figure 2.5: Example of simple RDF

2.3 Neural network approach

Neural networks are classifiers inspired by biological neurons. A weighted input is created and then goes through an activation function. Then the input is processed and we get an output that is compared with true values. The loss score is calculated, which represents how much was the prediction accurate compared to the true values. This value of the loss score is put through an optimizer, input weights are updated and the process starts again. The algorithm iterates until the loss score is minimized enough [5, 6, 7].

In semantic segmentation, the input is the original image and its corresponding mask or label and the output is a segmented image.

2.3.1 Artificial Neural Network

Artificial Neural Network (ANN), also known as *Feed-Forward Neural Network* is a learning machine which processes data only forward. It is a very simple algorithm, but it is dependent on hardware and computational power. ANN has its given topology based on the connections between its elements and it has to have at least one start element (SE), one end element (EE) and one processing element (EE). ANN with only one PE is called a *Perceptron*. Elements are sometimes referred to as *nodes* [5, 7]. An example architecture of ANN can be seen in Fig 2.6.

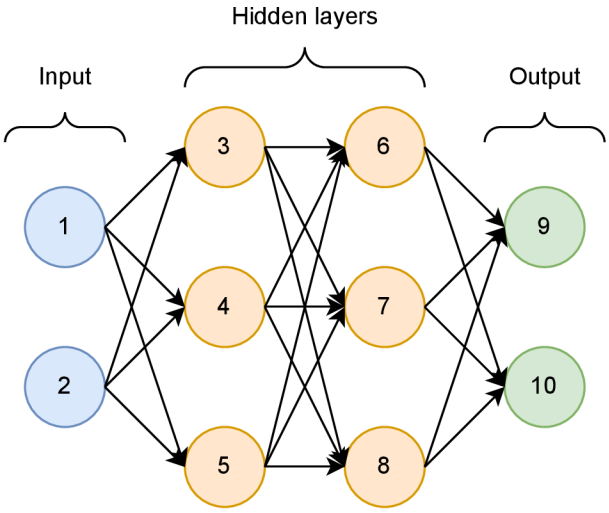


Figure 2.6: ANN with 2 hidden layers - 2 SE, 6 PE, 2 EE, 10 nodes in total

2.3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of neural network commonly used for image recognition. CNNs input signal only forward, which in most cases is an image, but has a hierarchical architecture. It consists of one or more convolutional layers where feature maps are created for each convolution and then subsampled. This process is repeated as many times as needed, and then a full connection of all feature maps is connected into a flattened layer (Fig. 2.7). This element goes through a classification process and the output is a probabilistic distribution (e.g. probability that this pixel is black is 90%, probability that it is white is 10%). CNNs have a disadvantage due to their requirement for a large amount of training data. [8, 9]

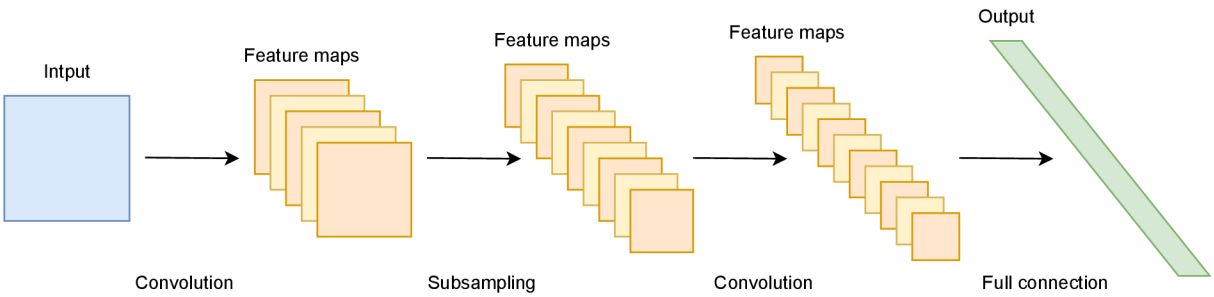


Figure 2.7: Example of a simple CNN

2.3.3 Recurrent Neural Network

Recurrent Neural Network (RNN) works the same as ANN. The difference is that RNN saves the output of processing elements (nodes) and "feeds" the output back to the model. Every node serves as a memory cell, if a prediction is not correct, the model learns the right output and continues in its iterations. It doesn't operate only in one direction, this process is called *recurrence* (Fig. 2.8).

RNN is very hard to train. If the algorithm is given badly dimensioned true values, it

can cause a failure in the training process and incorrect predictions. It is mostly used in text-to-speech conversions.

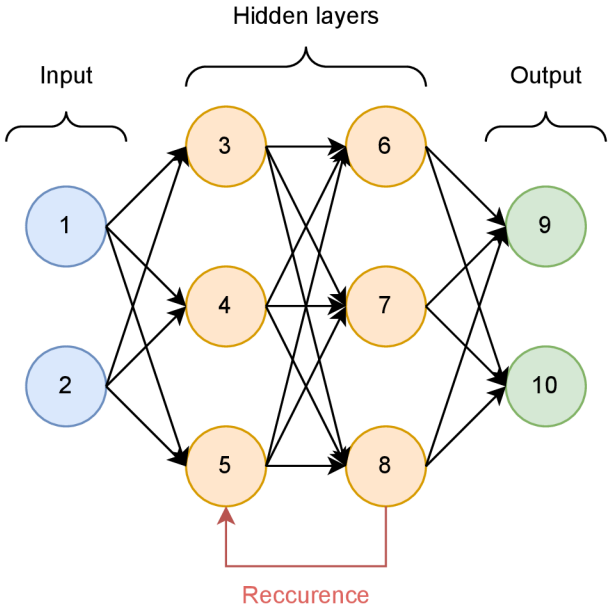


Figure 2.8: Example of RNN architecture

2.3.4 Most common types of architectures

There are many different types of architecture used for semantic segmentation. Below will be mentioned three backbone architectures - ResNet, U-Net and DeepLabV3+. All of these are a CNN-type architecture or are based on CNN.

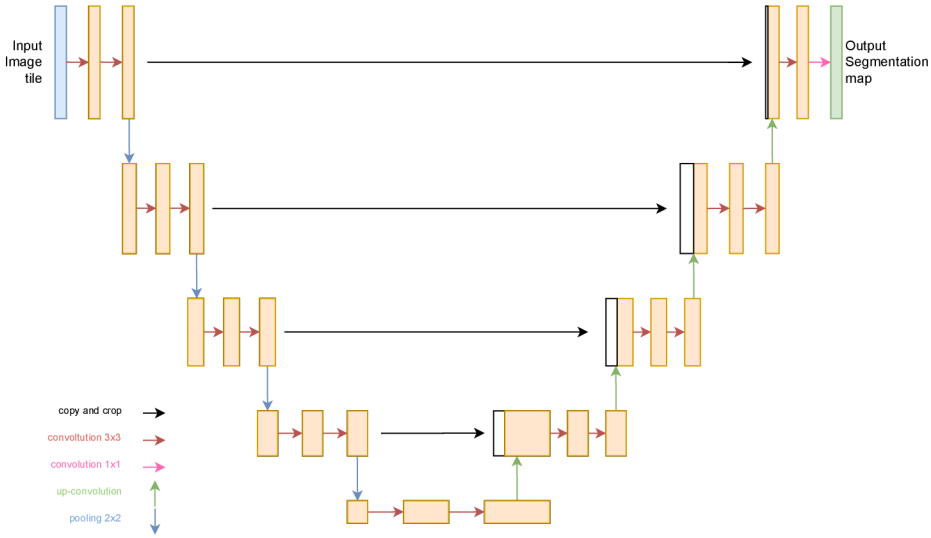


Figure 2.9: U-Net architecture

U-Net is an encoder-decoder architecture. First, the image goes through convolutions and pooling (left side of Fig. 2.9). The image is downsampled and feature maps are created. This part is called *contracting*. Then the feature maps are again upsampled

(up-convolutions), and they go through convolutions again. This is called the *expansion* (right side of Fig. 2.9). Different operations are made for every pooling and corresponding up-convolution of a feature map. That helps preserve spatial information.

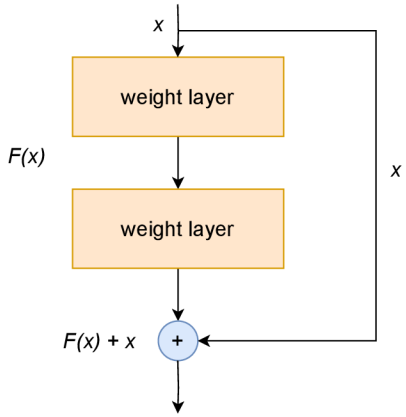


Figure 2.10: ResNet building block

ResNet [10] is used for training neural networks with many layers (deep neural networks). It was created to address the problem of vanishing gradients - the values used to update the weights during training can be very small. It adapts residual learning to every few stacked layers, *residual blocks*, meaning it has shortcut connections between them (see Figure 2.10) where the input to the residual block is added to the output. The residual block then learns the residual mapping $F(x) = H(x) - x$, where $H(x)$ is the actual mapping, and the output is $F(x) + x$. Adding these skip connections ensures that the gradient goes through the connections between layers without lowering its value.

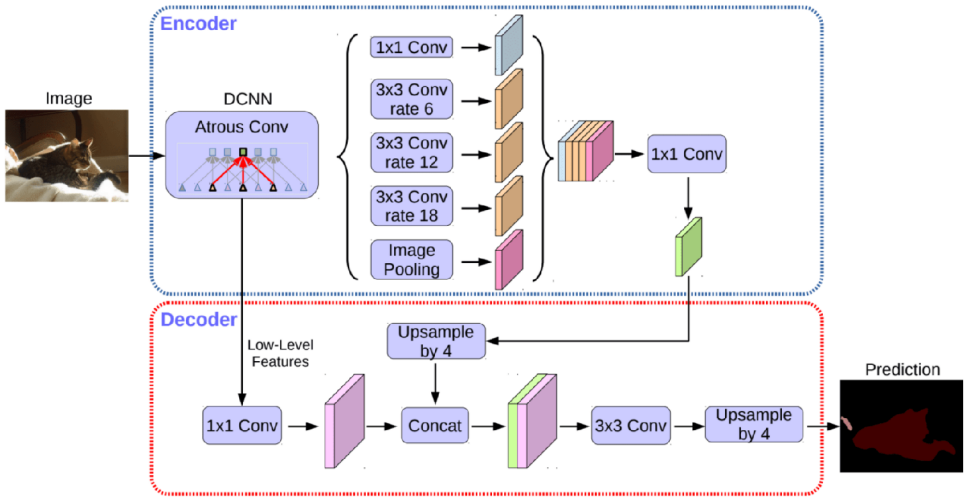


Figure 2.11: DeepLabv3+ architecture [11]

DeepLabV3+ [11] is a CNN, usually with the ResNet backbone and it was created specifically for semantic segmentation. It uses an encoder-decoder architecture, where the encoder is used to gain contextual information from the image. The decoder module finds the boundaries of objects. Simplified architecture is shown in Figure 2.11.

2.4 Key performance indicators

Segmentation models are evaluated based on Key Performance Indicators (KPIs). Those can be measures of accuracy, speed of segmenting a single image (latency) or memory usage. Below are the most used KPIs for evaluating how precise the segmentation is and their calculations [1].

1. **Per pixel rate**

$$PPR = \frac{\sum_{i=1}^{k} n_{ii}}{\sum_{i=1}^{k} t_i}, \tag{2.1}$$

where k is the number of classes, n_{ii} is the number of predicted pixels of one class and t_i is the number of pixels that belong to this class.

2. **Mean accuracy**

$$MA = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{n_{ii}}{t_i}, \tag{2.2}$$

mean accuracy shows how many of the pixels belonging to each class in the predicted image were labelled correctly.

3. **MIoU (Mean Intersection over Union)**

MIoU is calculated as the area where predicted pixels of one class overlap with the ground truth label divided by the union of predicted pixels and ground truth label. A combination of mean accuracy and MIoU is common for the evaluation of semantic segmentation models. Their values should be close to each other, if for example, the value of mean accuracy is much higher than MIoU, it gives us a warning, that the segmentation is probably done wrong.

$$MIoU = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{n_{ii}}{t_i - n_{ii} + \sum_{i=1}^{k} n_{ji}}. \tag{2.3}$$

Other variations of MIoU are shown in the equations 2.4 and 2.5.

4. **Frequency weighted MIoU**

$$FWMIoU = (\sum_{i=1}^{k} t_i)^{-1} \cdot \sum_{i=1}^{k} t_i \cdot \frac{n_{ii}}{t_i - n_{ii} + \sum_{i=1}^{k} n_{ji}}. \tag{2.4}$$

5. **F-measure**

$$F_\beta = (1 + \beta)^2 \cdot \frac{TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP} \tag{2.5}$$

where TP is true positive, FN false negative, FP false positive. $\beta = 1$ is chosen in most cases.

2.5 Datasets

The correct choice of the dataset used for training a semantic segmentation model is essential for creating a reliable model. In the case of segmentation of off-road terrain, it is important to use a dataset with images from the right environment, with variable weather and light conditions to achieve the best results.

Commonly used annotation formats are RGB masks, JSON (JavaScript Object Notation) or COCO (Common Objects in Context) format labels. JSON and COCO annotations typically contain information about the segmented objects, e.g. their class labels and the coordinates of their boundaries. COCO annotation format is JSON-based and is designed for image recognition [12]. RGB masks are mostly used for semantic segmentation.

Every dataset should be split into three parts - *train*, *validation* and *test*. The ratio for the split is most commonly 70-80% for training, 10-15% for validation and 10-15% for testing. Each set contains pairs of corresponding images and their annotations. *Train* and *validation* sets are used when training a model, *test* for predictions and final evaluation of the model.

Below are mentioned three datasets created specifically for semantic segmentation of off-road terrain. All of them have RGB mask annotations but vary in the number of classes and the location where the photos were taken.

2.5.1 Rellis3D

Rellis3D dataset [13] is created by researchers from Texas A&M University. Annotations consist of ground truth RGB labels for images and point-wise labelled masks for LiDAR scans. There are 20 different classes - void, dirt, grass, tree, pole, water, sky, vehicle, object, asphalt, building, log, person, fence, bush, concrete, barrier, puddle, mud and rubble. All of the images were taken at the Ground Research facility on the Rellis Campus of Texas A&M University.

2.5.2 RUGD

Images in RUGD dataset [14] have only ground truth RGB masks with 24 classes - dirt, sand, grass, tree, pole, water, sky, vehicle, container, asphalt, gravel, mulch, rockbed, log, bicycle, person, fence, bush, sign, rock, bridge, concrete, table, building.

2.5.3 Yamaha-CMU Off-Road Dataset (YCOR)

The YCOR dataset [15] contains 1076 images split only into train and validation sets with a ratio of 931 images for training and 145 images for validation. Photos were taken in Western Pensylvania and Ohio during three different seasons. Images were labelled with polygons and then converted to RGB masks with 8 classes - sky, rough trail, smooth trail, traversable grass, high vegetation, low vegetation, non-traversable, obstacle.

2.6 Methods

2.6.1 GA-Nav

GA-Nav models [16, 17] are created with a Mixed Transformers backbone, which serves as an alternative to ResNet architecture. GA-Nav architecture is shown in Fig. 2.12. The model is created for navigation in off-road terrain and is trained on RUGD and Rellis3D datasets. GA-Nav models use coarse-grained labels and compute binary masks for each group. The authors provide pretrained models on both datasets. GA-Nav is heavily based on MMSegmentation [17], which is a semantic segmentation toolbox based on the PyTorch library.

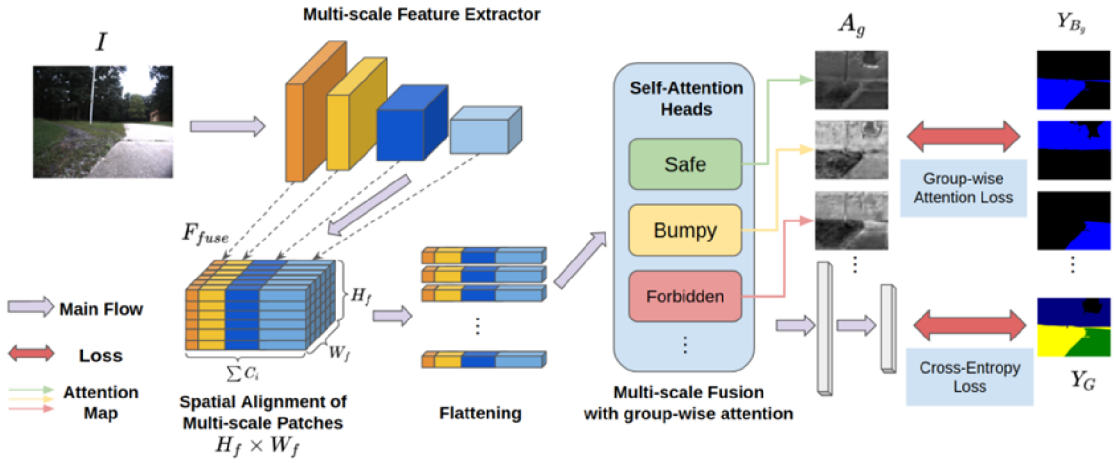


Figure 2.12: GA-Nav architecture [17]

2.6.2 YOLOv8 (You Only Look Once)

Ultralytics YOLOv8 segmentation models [18] are pretrained on COCO dataset [10] and its architecture is inspired by GoogLeNet (Fig. 2.13). YOLO label format is required for training and prediction on different datasets. It is important to note, that these models are for instance segmentation and not for semantic segmentation.

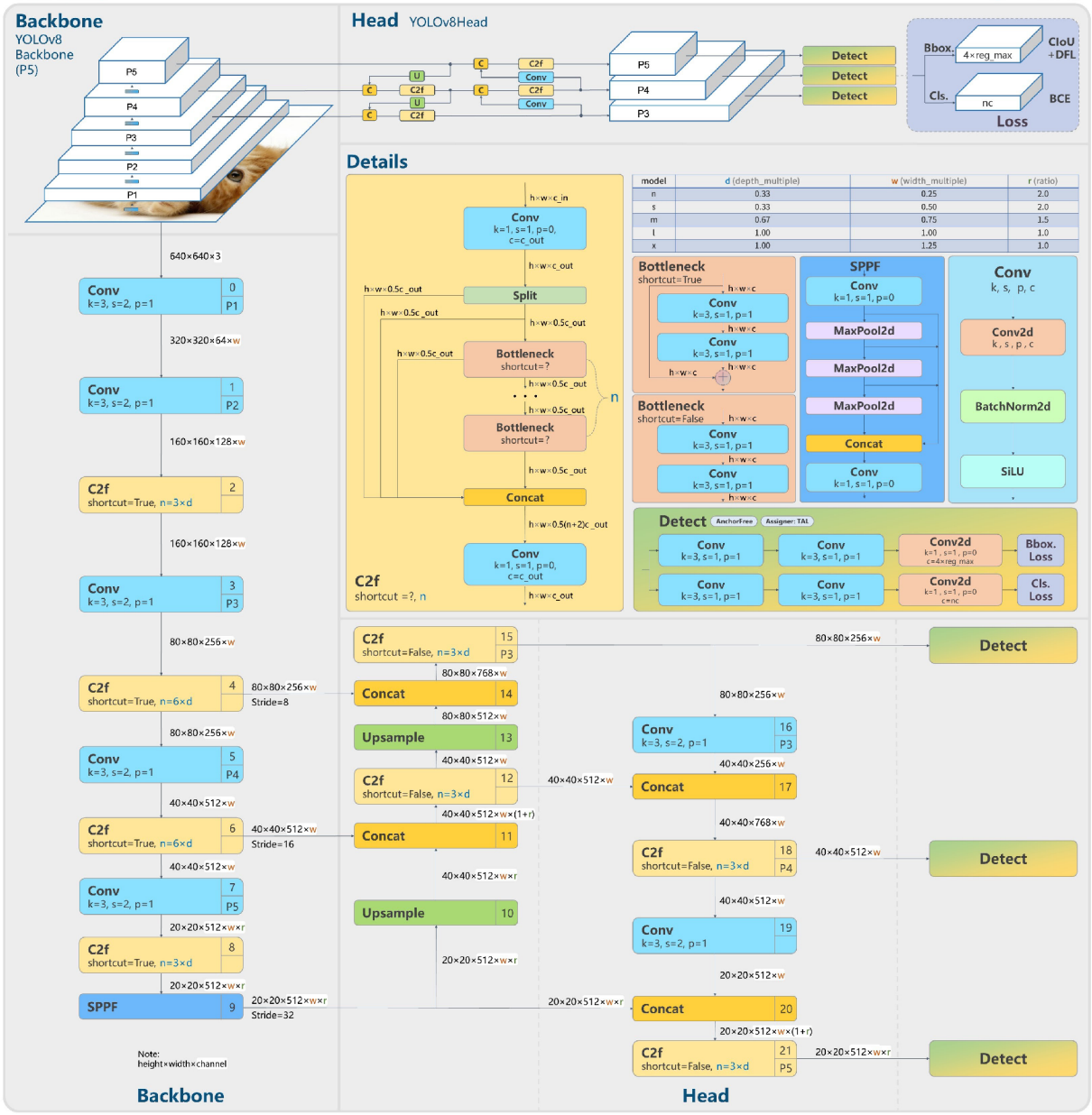


Figure 2.13: YOLOv8 detailed architecture [19]

2.6.3 Semantic segmentation using DeepLabv3+ by Haddad and Mulay

Semantic segmentation models by N. M. Haddad and A. V. Mulay [20] depend on PyTorch libraries utilizing DeepLabv3+ [11]. Provided pretrained model is trained on COCO and Yamaha-CMU Off-road dataset with MobileNetV3-Large backbone [21]. It is possible to train and finetune models using three different backbones - ResNet50, ResNet101 and MobileNetV3-Large.

The authors have created three models with the backbones ResNet101, ResNet50 and MobileNetV3-Large, trained on 50 epochs with a batch size of 8. These models were trained on the YCOR dataset. Table 2.1 shows the results from training the models and examples of their experiments are shown in Figure 2.14.

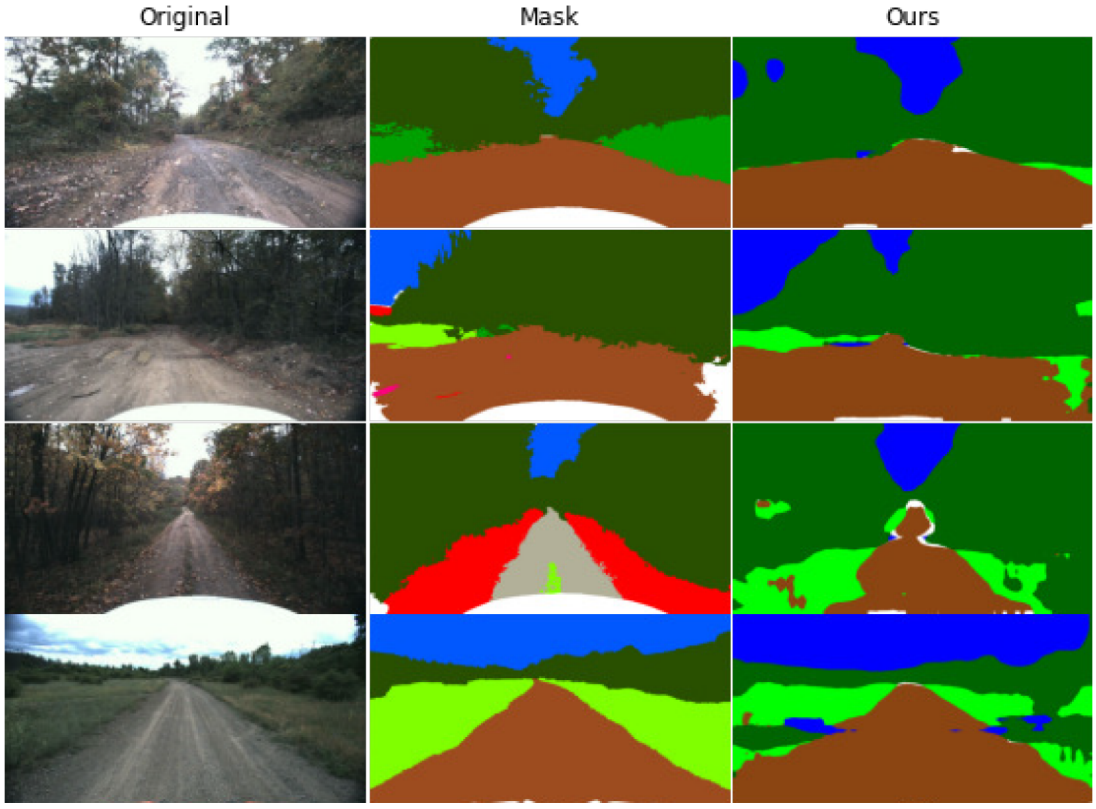


Figure 2.14: Example of the authors output from MobileNetV3-Large after 10 epochs [20]

Model number	Backbone architecture	Training time [HH:MM:SS]	Best validation MIoU
1	MobileNetV3-Large	01:02:01	85.53%
2	ResNet50	03:09:18	84.81%
3	ResNet101	04:12:56	85.81%

Table 2.1: Experiments with DeepLabv3+ network performed by Haddad and Mulay

3 Experiments

Four tasks needed to be done during the experiments. The first was to test all chosen pretrained models on different datasets which is described in chapter 3.2. The second task was to train different models with various training parameters and again test all models on different datasets (chapter 3.3). The third part of the experiments was to create a dataset (chapter 3.4) and test the best model on it (chapter 3.5). The final results of the experiments are described in chapters 3.3.7 and 3.6.

3.1 Preparations

Semantic segmentation requires high computation power, especially GPU, and the results of training and prediction are highly hardware-dependent. It was necessary to install NVIDIA CUDA Toolkit [22], which enables the creation of high-performance, GPU-accelerated applications. Nvidia CUDA 12.2 and Nvidia CUDA Toolkit 11.5 were used. All of the experiments were done on a computer with the following parameters:

Operation system	Linux, Ubuntu 22.4
CPU	13th Gen Intel(R) Core(TM) i9-13900HX
GPU	NVIDIA GeForce RTX4080

Table 3.1: System and hardware parameters

All of the code used and written was in Python 3.11.5, with the use of *Conda* [23] or *VSCode* [24] environments. *Pytorch*, *numpy*, *Pillow*, *matplotlib* and *os* were libraries that were used in almost every part of the experiments.

Methods and datasets used for experiments were chosen in the final part of the preparations. The methods chosen were GA-Nav and models by Haddad and Mulay. Both of these are tailored for semantic segmentation of off-road terrain, compared to YOLO segmentation models, they have RGB mask labels as an input, which is the most common annotation format for semantic segmentation. The datasets were selected to match the ones used to train the models - RUGD, Rellis3D and YCOR.

3.2 Testing pretrained models

3.2.1 GA-Nav

The first step when testing GA-Nav pretrained models was to prepare all the datasets. Rellis3D and RUGD datasets were downloaded and put in a repository structure according to the instructions provided by the authors [17].

Then it was necessary to install the required libraries and packages. There were issues when installing the requirements files. The provided installation files were not up to date and changes in package and library versions had to be made during the installation

process. The biggest issue occurred when installing the *MMCV* toolbox, which at the time went through a version update and GA-Nav models have not yet been adapted to it.

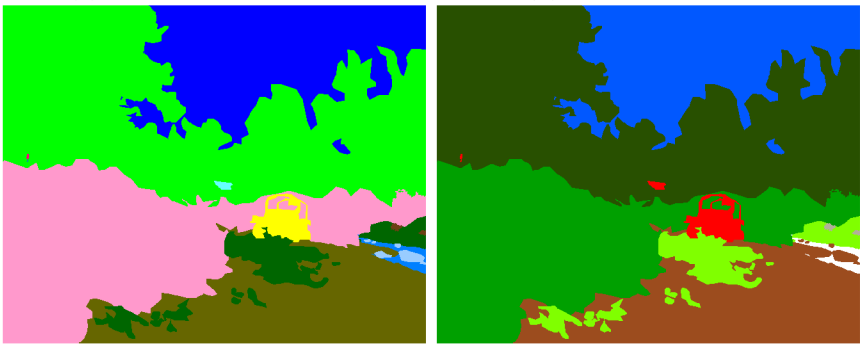
The next step was dataset processing, where the ground truth labels were processed and grouped labels were created. Other problems have occurred while running training and prediction of the models. Errors were mostly related to the MMSegmentation toolbox. It was decided that GA-Nav models are inconvenient to use in their current state. Progress with GA-Nav can be made when the authors update their work according to new library and package versions.

3.2.2 Models by Haddad and Mulay

The model was trained with Mnv3 backbone and 50 epochs on the YCOR dataset. It was not possible to evaluate the model on the YCOR dataset due to its structure. As was mentioned before (2.5.3), the YCOR dataset is split only in *train* and *validation* sets, predictions on the dataset as it is would not be relevant. For this reason, the model was tested only on the RUGD and Rellis3D datasets.

The RUGD and Rellis3D datasets had to be relabeled corresponding to the classes used in YCOR. Class mapping was created (table 3.2) and used on *test* sets of both datasets. Visualization of the relabeling can be seen in Figures 3.1 and.

(a) Rellis3D original mask (b) Rellis3D relabeled



(c) RUGD original mask (d) RUGD relabeled

Figure 3.1: Example of mask relabeling on RUGD and Rellis3D datasets

The authors of the model created only a function for calculating MIoU, the mean accuracy function was added to the code. Images used for testing were chosen based on list files provided with both datasets, their file structure had not been changed. This pretrained model has performed with MIoU and mean accuracy over 70% and latency 11 ms, exact results can be seen in Table 3.3. These results can not be compared with the authors' results, because they did not test their models on other datasets.

class ID	YCOR class	RUGD class	Rellis3D class
0	low vegetation	bush	bush
1	sky	sky	sky void
2	high vegetation	tree	tree
3	rough trail	mulch rock-bed rock	puddle rubble
4	non-traversable	void water	water
5	traversable grass	grass	grass
6	smooth trail	dirt sand asphalt gravel concrete	dirt asphalt concrete mud
7	obstacle	pole vehicle object building log bicycle person fence sign bridge picnic-table	pole vehicle object building log person fence barrier

Table 3.2: Class mapping between YCOR, RUGD and Rellis3D datasets

Problems occurred with segmenting buildings and objects that were further away in the images. Most of these objects were classified as low or high vegetation. Reflection of the sky in a body of water was segmented as the sky and not water, which is a very common issue for many semantic segmentation models.

The example of the model's predictions on the Rellis3D dataset is shown in the first row in Figure 3.2. It recognized rough terrain but sometimes classified it as high vegetation. An example of predictions on the RUGD dataset is shown in Figure 3.2.

Dataset	MIoU	Mean accuracy	Latency
RUGD	72.2983%	72.1856%	0.0110 s
Rellis3D	71.3578%	71.2378%	

Table 3.3: Evaluation of the *Mnv3-50-pretrained* model

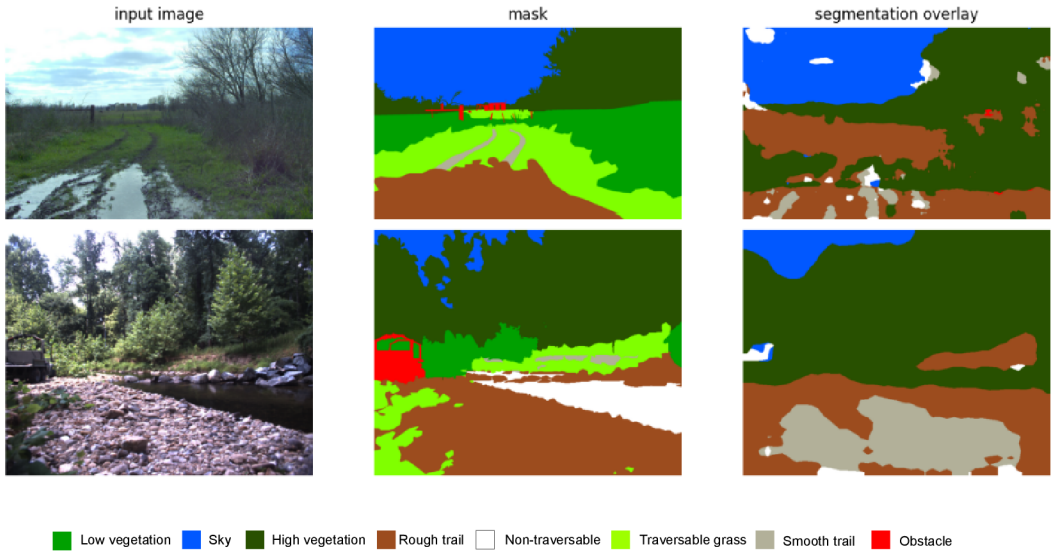


Figure 3.2: Predictions with *Mnv3-pretrained* model

3.3 Training models

Six models with different backbones and other parameters were trained and tested. The parameters were chosen corresponding to the previous work of Haddad and Mulay [20].

Our parameters for training were chosen as close to the original parameters provided by the authors as possible. Three models with ResNet101, ResNet50 and MobileNetV3-Large backbones were trained on 50 epochs. The batch size differs (see chapters 3.3.4, 3.3.3 for further details). Additionally, three other models were trained on 10 epochs for comparison with the models trained on 50 epochs.

All of our models have been trained on the restructured version of the YCOR dataset which has been changed so that models could be properly evaluated. A test set has been added and a new split was created with the ratio 2629 *train*, 435 *validation* and 165 *test*. Images for the new split were chosen randomly.

All models were then tested on *test* sets of restructured YCOR dataset, RUGD and Rellis3D datasets. Relabeled masks of images from RUGD and Rellis datasets were used again. Models are named as follows: *[backbone]-[number of epochs]*. All training parameters are summarized in Table 3.4.

Model	Epochs	Backbone	Batch size	Training time [HH:MM:SS]	Best MIoU
Mnv3-10	10	MobileNetV3-Large	8	00:06:56	86.01%
Mnv3-50	50	MobileNetV3-Large	8	00:34:47	83.35%
Res101-10	10	ResNet101	2	00:45:26	86.77%
Res101-50	50	ResNet101	2	03:49:21	86.78%
Res50-10	10	ResNet50	3	00:30:54	87.15%
Res50-50	50	ResNet50	3	02:34:56	86.90%

Table 3.4: Training parameters of all models

3.3.1 Mnv3-50

The first model was trained with the MobileNetv3Large backbone on 50 epochs and batch size 8. The training was finished in 10 minutes. All training parameters are shown in the Table 3.4.

Predictions on the YCOR dataset had reached the value of mean accuracy below 62%, on the Rellis3D and RUGD datasets around 68%. The latency of this model was 12.4 ms. Exact results are shown in Table 3.5. The model had problems segmenting objects, sand roads and recognizing bushes as low vegetation. Examples of predictions on all datasets are shown below in Figure 3.3.

Dataset	MIoU	Mean accuracy	Latency
RUGD	68.6628%	68.2317%	0.0124 s
Rellis3D	69.2031%	67.6891%	
YCOR	61.5430%	61.3518%	

Table 3.5: Evaluation of the *Mnv3-50* model

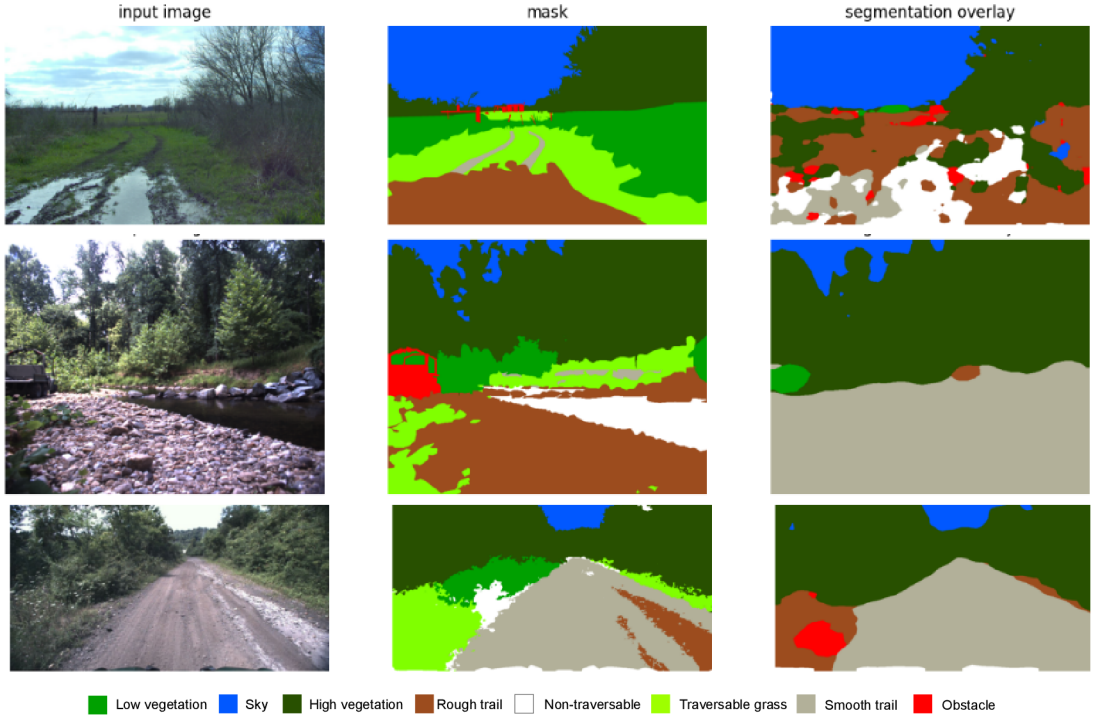


Figure 3.3: Testing *Mnv3-50* model

The first row shows predictions in the Rellis3D dataset. This model could segment sky and high vegetation correctly on the Rellis3D dataset but was not able to segment the rough trail well. The second row of the image shows predictions on the RUGD dataset where the model classified the rock bed as a smooth trail and not a rough trail. The third row shows predictions on the YCOR dataset.

3.3.2 Mnv3-10

The *Mnv3-10* model was trained with 10 epochs. The batch size had to be lowered to 2 due to the incapability of the GPU to process the training with a higher value. Training took 45 minutes with the best validation MIoU of 86.77%.

The model performed similarly on the YCOR dataset as the *Mnv3-50* model. The best mean accuracy has been reached when testing the model on Rellis3D with a value of almost 71%. The latency of this model was 10.6 ms. All results can be found in Table 3.6 below.

Dataset	MIoU	Mean accuracy	Latency
RUGD	68.4727%	68.2543%	0.0106 s
Rellis3D	71.1870%	70.9124%	
YCOR	62.1549%	61.9316%	

Table 3.6: Evaluation of the *Mnv3-10* model

An example of predictions on the Rellis3D dataset can be found in the first row in Figure 3.4, on the RUGD dataset in the second row, and on the YCOR dataset in the third row. The model classified most of the high vegetation correctly but was not able to classify low vegetation or traversable grass.

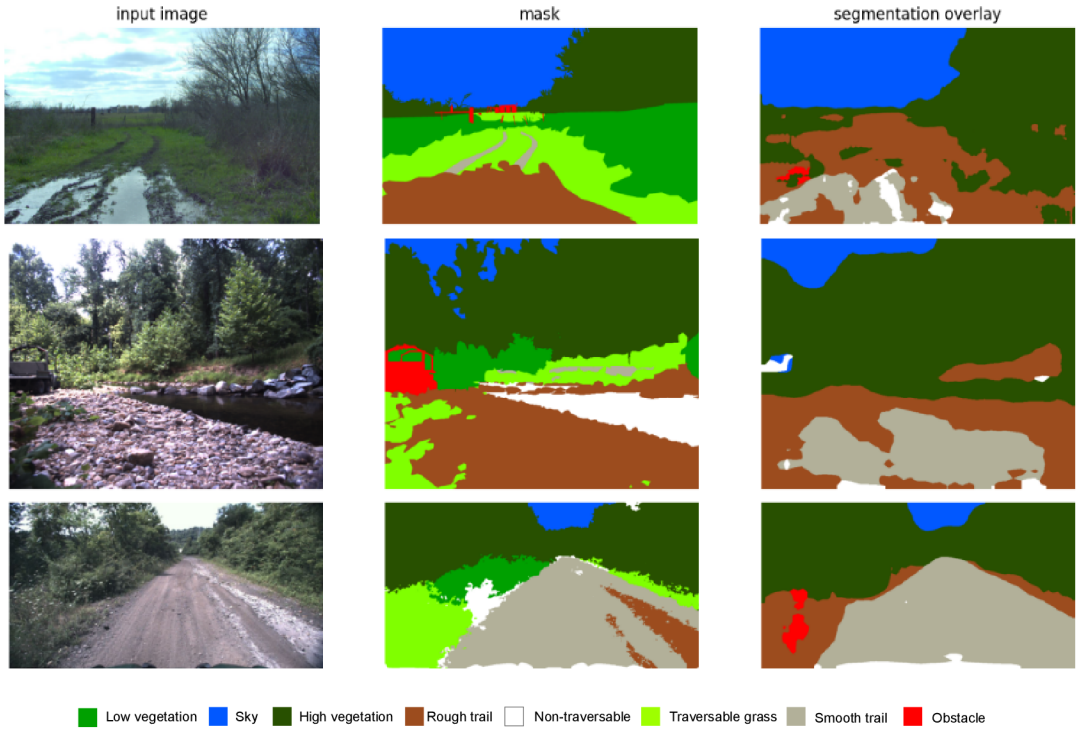


Figure 3.4: Testing *Mnv3-10* model

3.3.3 Res101-50

The third model was trained with the variation of ResNet architecture Resnet101. This model was trained on 50 epochs with batch size 2. Due to insufficient GPU computational power, the batch size had to be lowered again. The training took nearly 4 hours with the best validation MIoU of 86.78% (see Table 3.4).

The model could not classify non-traversable terrain correctly and had issues segmenting parts of low vegetation and traversable terrain as an obstacle. It also classified parts of the smooth trail as non-traversable when evaluated on the YCOR dataset. Examples of predictions are shown in 3.5, where each consecutive row represents an example from Rellis3D, RUGD and YCOR datasets. Predictions on the YCOR dataset had the worst results of all models with little over 61%. Contrary to that, the predictions on the Rellis3D dataset reached the best values of mean accuracy, nearly 72%. This model had the highest latency of 86.5 ms. The exact values are shown in Table 3.7 below.

Dataset	MIoU	Mean accuracy	Latency
RUGD	69.4481%	69.3576%	0.0865 s
Rellis3D	72.0036%	71.8967%	
YCOR	61.2914%	61.1644%	

Table 3.7: Evaluation of the *Res101-50* model

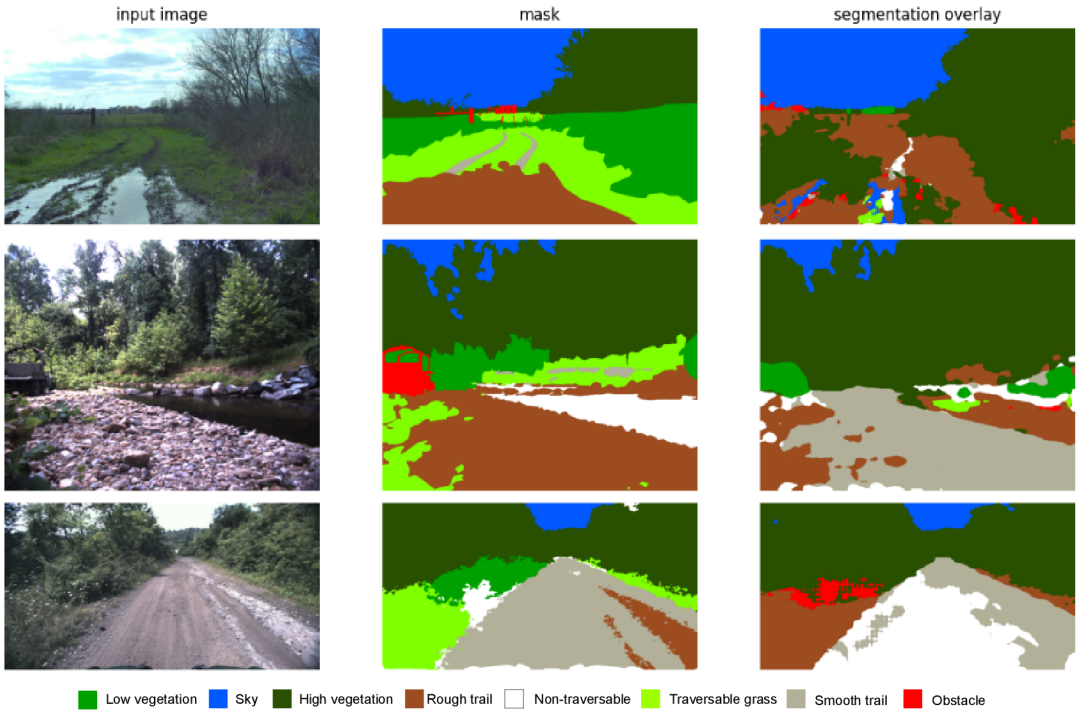


Figure 3.5: Testing *Res101-50* model

3.3.4 Res101-10

This model was trained on 10 epochs with a batch size of 2, ResNet101 was used as a backbone. The training was finished in 45 minutes. For all training parameters see Table 3.4.

Predictions of rough terrain and traversable grass in the RUGD dataset were inaccurate (first row in Figure 3.6). The *Res101-10* model had issues when tested on the Rellis3D dataset and achieved a mean accuracy of 68%. It was not able to correctly classify bodies of water, which were predicted as rough terrain (see the second row in Figure 3.6), and rough terrain as smooth terrain. When tested on the YCOR dataset the model segmented parts of the terrain, such as traversable grass or low vegetation, as rough terrain (third row in Figure 3.6). Predictions on the YCOR dataset reached the best values of all models, all results are summarized in Table 3.8.

Dataset	MIoU	Mean accuracy	Latency
RUGD	69.5141%	69.3339%	0.0863 s
Rellis3D	69.3168%	67.9990%	
YCOR	65.0408%	64.9108%	

Table 3.8: Evaluation of the *Res101-10* model

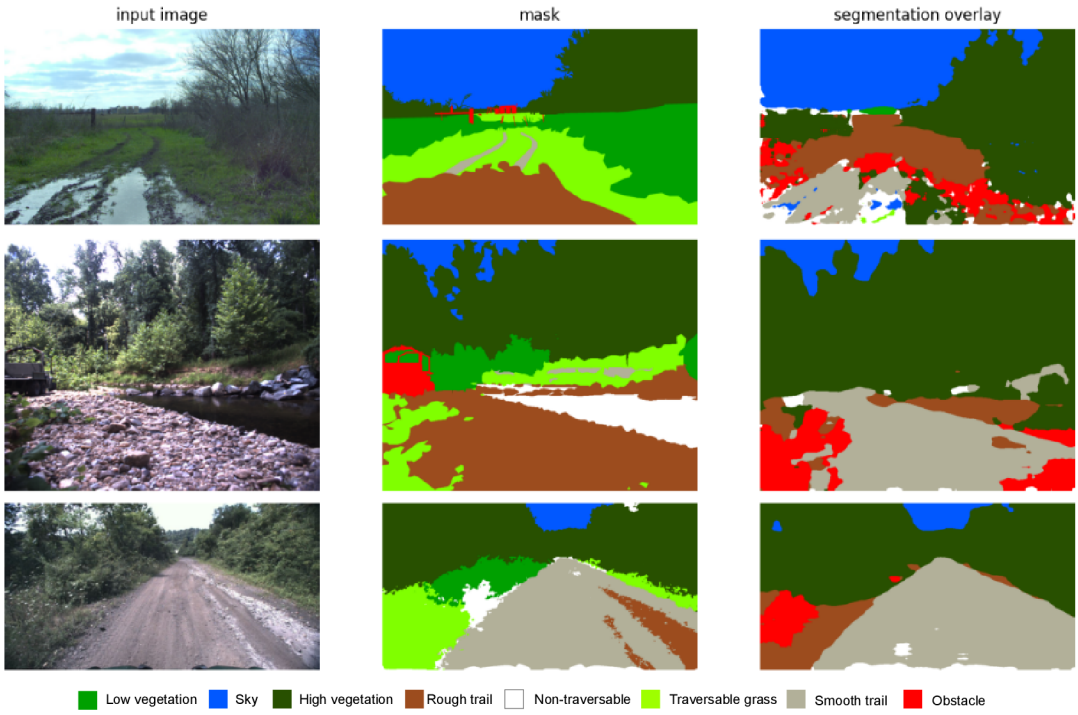


Figure 3.6: Testing *Res101-10* model

3.3.5 Res50-50

The fifth model was trained with the backbone ResNet50, which is another variation of ResNet architecture. The GPU was able to perform the training of the model with a batch size of 3 of 50 epochs. Training of the model was finished in over two and a half hours with the best validation MIoU of 86.8%.

Predictions on the Rellis3D dataset reached a mean accuracy of 71.5%, on the RUGD dataset 70.5% and the YCOR dataset 62.1% with a latency of 57 ms. The model had issues segmenting obstacles, which were classified as low vegetation. Examples of predictions are shown in Figure 3.7, where the first row represents evaluation on the Rellis3D dataset, the second row on the RUGD dataset, and the third row on the YCOR dataset. Exact training parameters are in Table 3.4 and results of testing the model are in Table 3.9.

Dataset	MIoU	Mean accuracy	Latency
RUGD	70.6128%	70.5438%	0.0570 s
Rellis3D	71.5969%	71.4655%	
YCOR	62.2076%	62.1174%	

Table 3.9: Evaluation of the *Res50-50* model

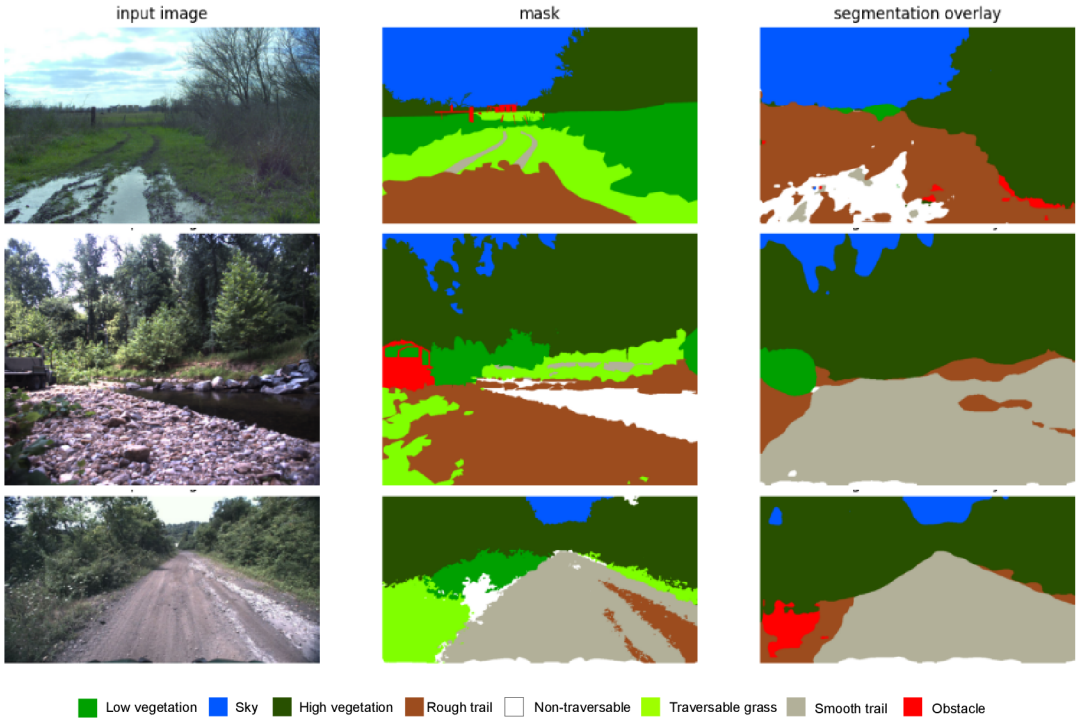


Figure 3.7: Testing *Res50-50* model

3.3.6 Res50-10

The last model was trained with the backbone ResNet50 on 10 epochs with a batch size of 3. The training was performed in 30 minutes with the best validation MIoU of 87.15%. All training parameters are specified in Table 3.4.

The model performed the most accurately on the RUGD dataset with a value of mean accuracy of 71.6%, but it classified parts of rough terrain and traversable grass as high vegetation (see the first row in Figure 3.8). Issues occurred when segmenting objects and the Res-50-10 model classified parts of traversable terrain as obstacles (second row in Figure 3.8). When tested on the YCOR dataset, the model was inaccurate in segmenting traversable grass and low vegetation (see the third row in Figure 3.8). Results of testing are shown in Table 3.10.

Dataset	MIoU	Mean accuracy	Latency
RUGD	71.6493%	71.6129%	0.0546 s
Rellis3D	71.0637%	71.0120%	
YCOR	63.2410%	63.3029%	

Table 3.10: Evaluation of the *Res50-10* model

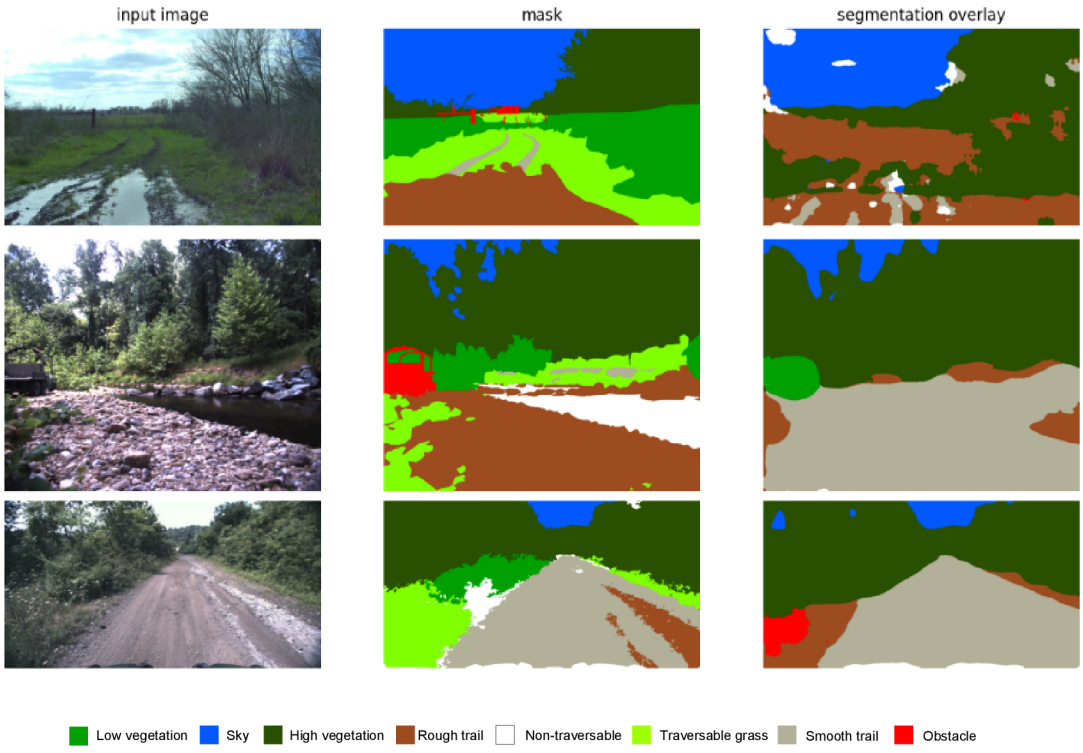


Figure 3.8: Testing *Res50-10* model

3.3.7 Final results

All results of testing the semantic segmentation models are summarized in Table 3.11 below. Comparison of the models based on accuracy can be seen in Figure 3.9, on MIoU in Figure 3.10 and on latency in Figure 3.11.

Model	Dataset	MIoU	Mean accuracy	Latency
Mnv3-preterained	RUGD	**72.2983%**	**72.1856%**	0.0110 s
	Rellis3D	71.3578%	71.2378%	
Mnv3-50	RUGD	68.6628%	68.2317%	0.0124 s
	Rellis3D	69.2031%	67.6891%	
	YCOR	61.5430%	61.3518%	
Mnv3-10	RUGD	68.4727%	68.2543%	**0.0106 s**
	Rellis3D	71.1870%	70.9124%	
	YCOR	62.1549%	61.9316%	
Res101-50	RUGD	69.4481%	69.3576%	0.0865 s
	Rellis3D	**72.0036%**	**71.8967%**	
	YCOR	61.2914%	61.1644%	
Res101-10	RUGD	69.5141%	69.3339%	0.0863 s
	Rellis3D	69.3168%	67.9990%	
	YCOR	**65.0408%**	**64.9108%**	
Res50-50	RUGD	70.6128%	70.5438%	0.0570 s
	Rellis3D	71.5969%	71.4655%	
	YCOR	62.2076%	62.1174%	
Res50-10	RUGD	71.6493%	71.6129%	0.0546 s
	Rellis3D	71.0637%	71.0120%	
	YCOR	63.2410%	63.3029%	

Table 3.11: Final results of testing all models

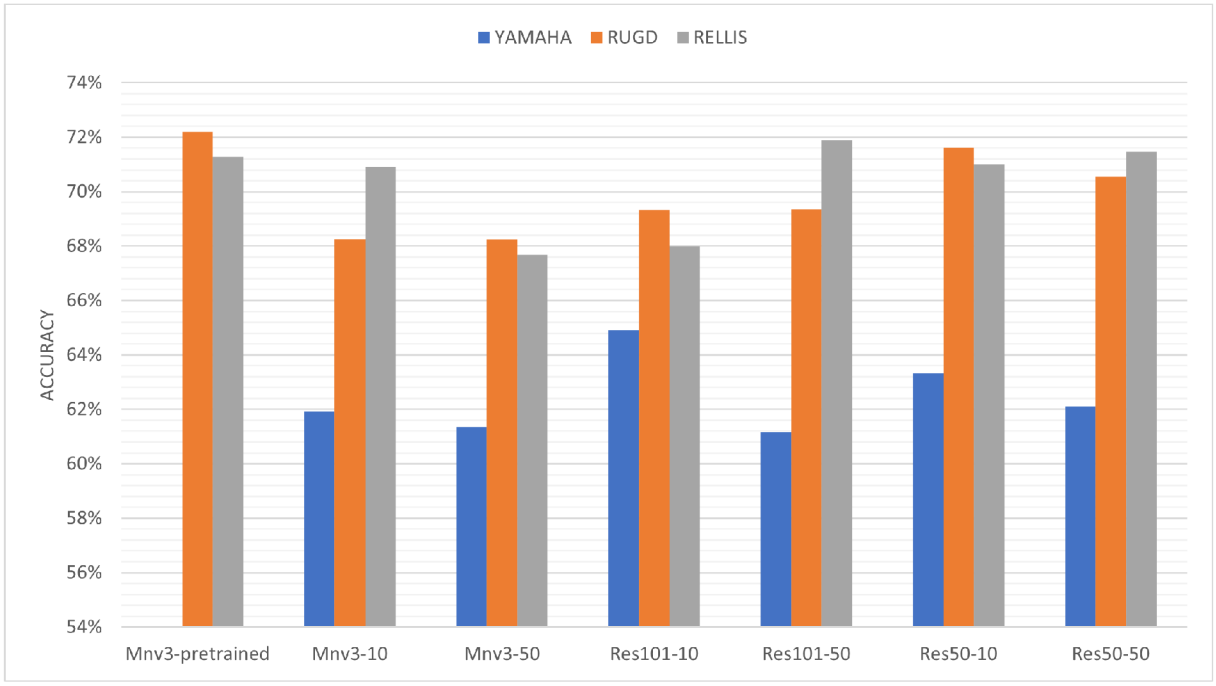


Figure 3.9: Comparison of the models based on accuracy

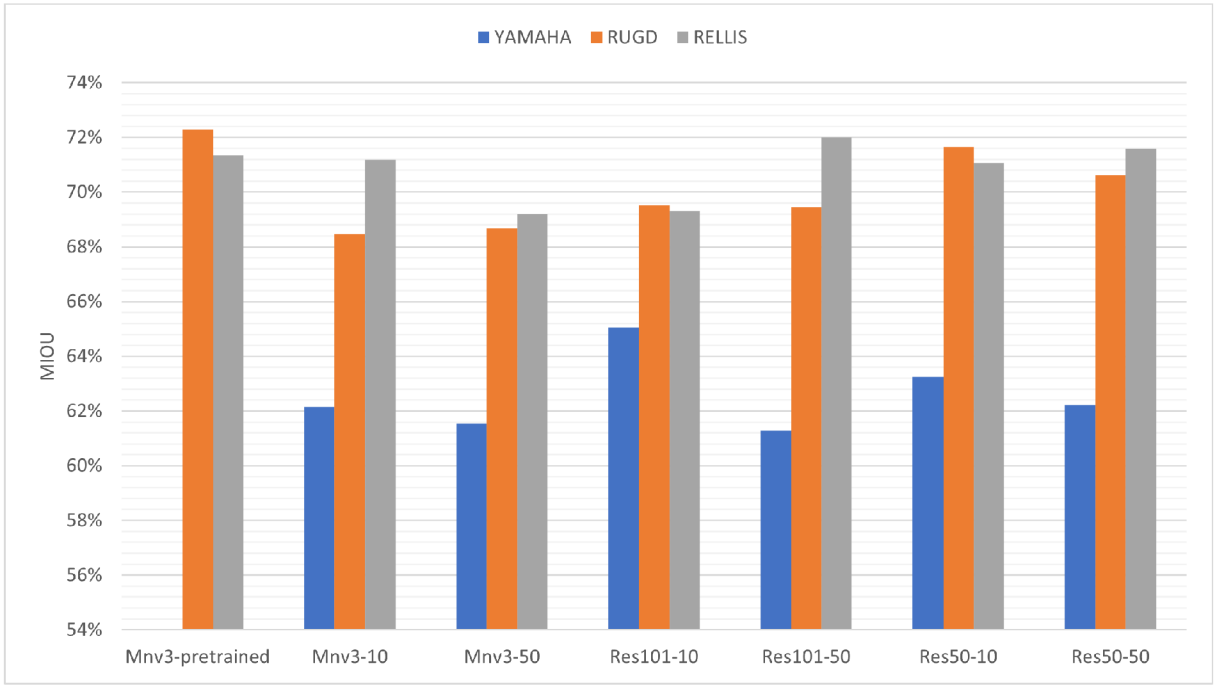


Figure 3.10: Comparison of the models based on MIoU

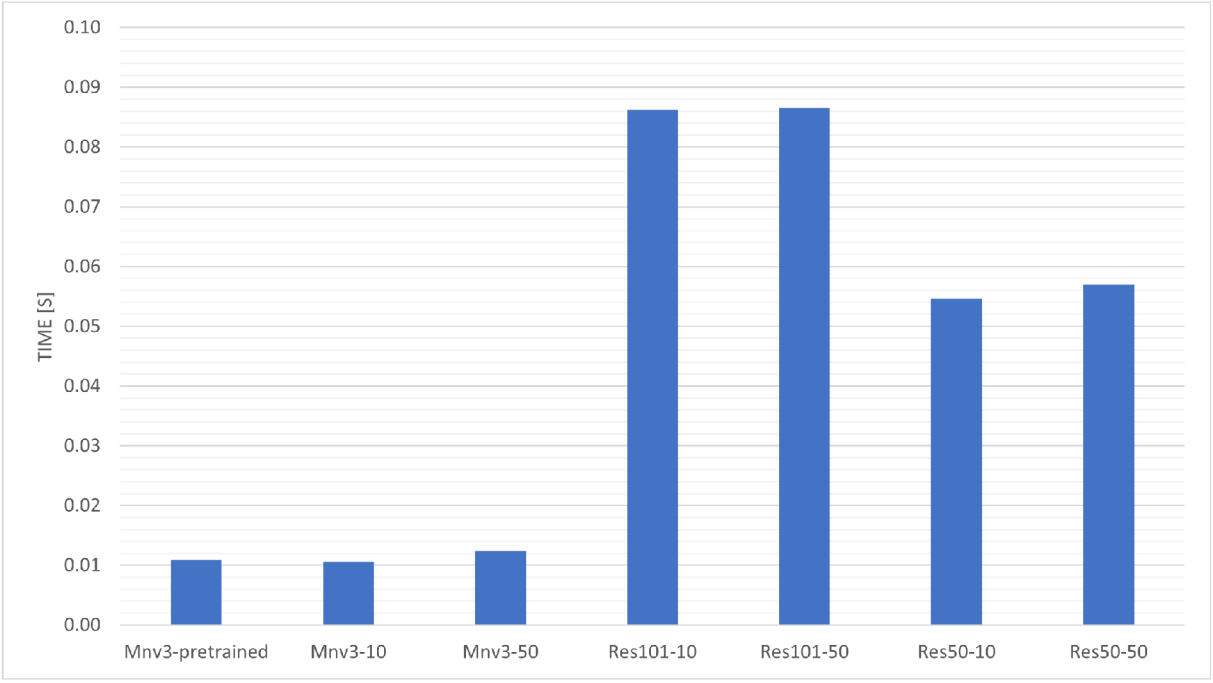


Figure 3.11: Comparison of the models based on latency

3.4 Creating an evaluation dataset

The third part of the experiments was to create a dataset. This dataset was further used for testing the best semantic segmentation model. Collecting much data was unnecessary because the images were not meant for training the models, only to show how the best model performed in various situations.

The dataset consists of 205 images in total. The photographs in this dataset were selected to include as diverse weather conditions, seasons, and types of environments as possible. The main focus was to create a set that best represents the environment in which the semantic segmentation models will be used.

The first part of the dataset named *moravia* contains 33 images taken in South Moravia during spring. The images were taken on a GoPro camera with good visibility and light conditions. High vegetation, objects, smooth trail, rough trail and non-traversable terrain can be seen in all images.

The second part, *bikepark*, is divided into four sections based on the location and season when the photos were taken. Section *vid1* contains 37 images from the bike park Dolní Morava taken in summer, section *vid2* 72 images from the same location in spring. Sections *vid3* and *vid4* include images from the bike park Haypark Tošovice taken in winter and autumn. Both sections contain 48 images each. All photos were taken on a GoPro camera positioned on the rider's helmet. Rough trail, low and high vegetation and obstacles can be seen in the images from different angles.

3.5 Testing the best model

The final part of the experiments was to choose the model that performed the best based on its accuracy and speed of the segmentation and test it on the created dataset. The *Mnv3-pretrained* model performed the best on the RUGD dataset and well on the Rellis3D

dataset. The best value of mean accuracy of the prediction on the Rellis3D dataset was achieved by model *Res101-50*.

The best MIoU value when testing the models on the RUGD dataset was reached again by the *Mnv3-pretrained* model, on the Rellis3D dataset by the *Res101-50*. A comparison of all results can be seen in Figure 3.9 and 3.10.

All models achieved mean accuracy and MIoU of 65% and lower when tested on the restructured YCOR dataset, which is noticeably lower than their performance with other datasets.

Models were also compared based on their latency. The *Mnv3-pretrained*, *Mnv3-50* and *Mnv3-10* models have achieved the fastest latency under 15 ms. The rest of the models were noticeably slower with the latency around 86 ms of the *Res101-10* and *Res101-50* models and 56 ms of the *Res50-10* and *Res50-50* models. Models with ResNet backbone were slower because this type of architecture is more complicated than MobileNetV3-Large. A comparison of all models based on their speed is visualized in Figure 3.11. All results are summarized in Table 3.11, chapter 3.3.7.

Mnv3-pretrained was chosen as the best model. It performed well when predicting on both Rellis3D and RUGD datasets. Its latency was the second lowest with 11 ms.

The last task was to test the best model. Predictions on the *moravia* set of images were sufficient. The model was able to segment roads, high vegetation and obstacles well. There were issues with segmenting parts of rough terrain and objects further away. Examples of predictions on the *moravia* set are in Figure 3.13. The model had issues segmenting the *bikepark* set, where it could correctly segment only parts of roads and high vegetation, segmentation of the images taken in winter was not inadequate (Figure 3.12).

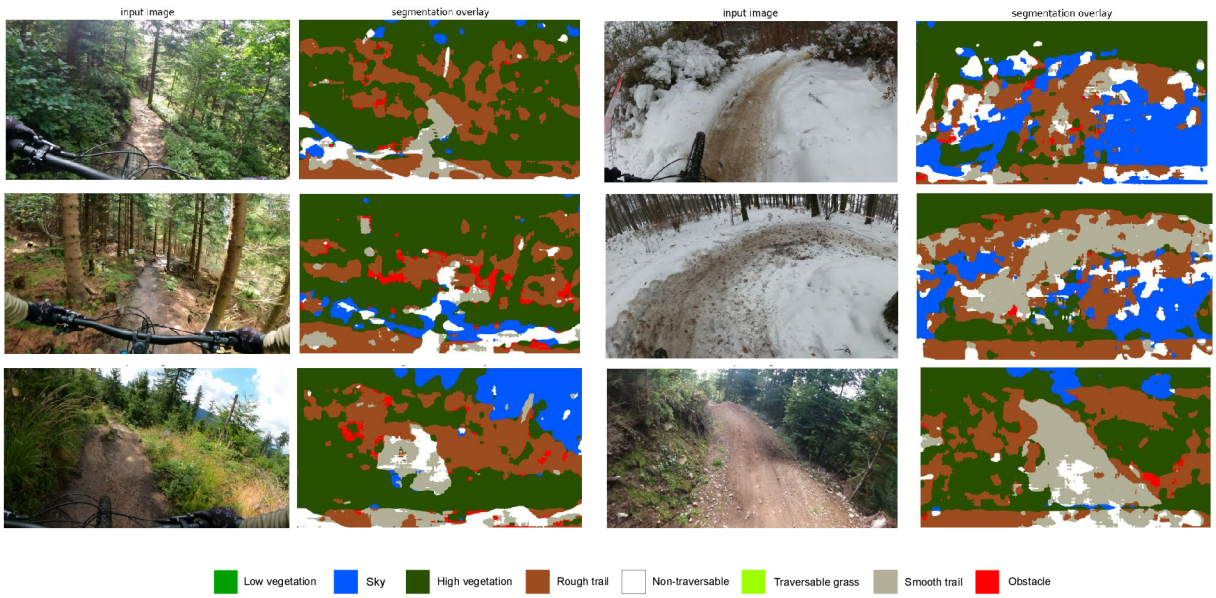


Figure 3.12: Predictions on the *bikepark* part of the dataset

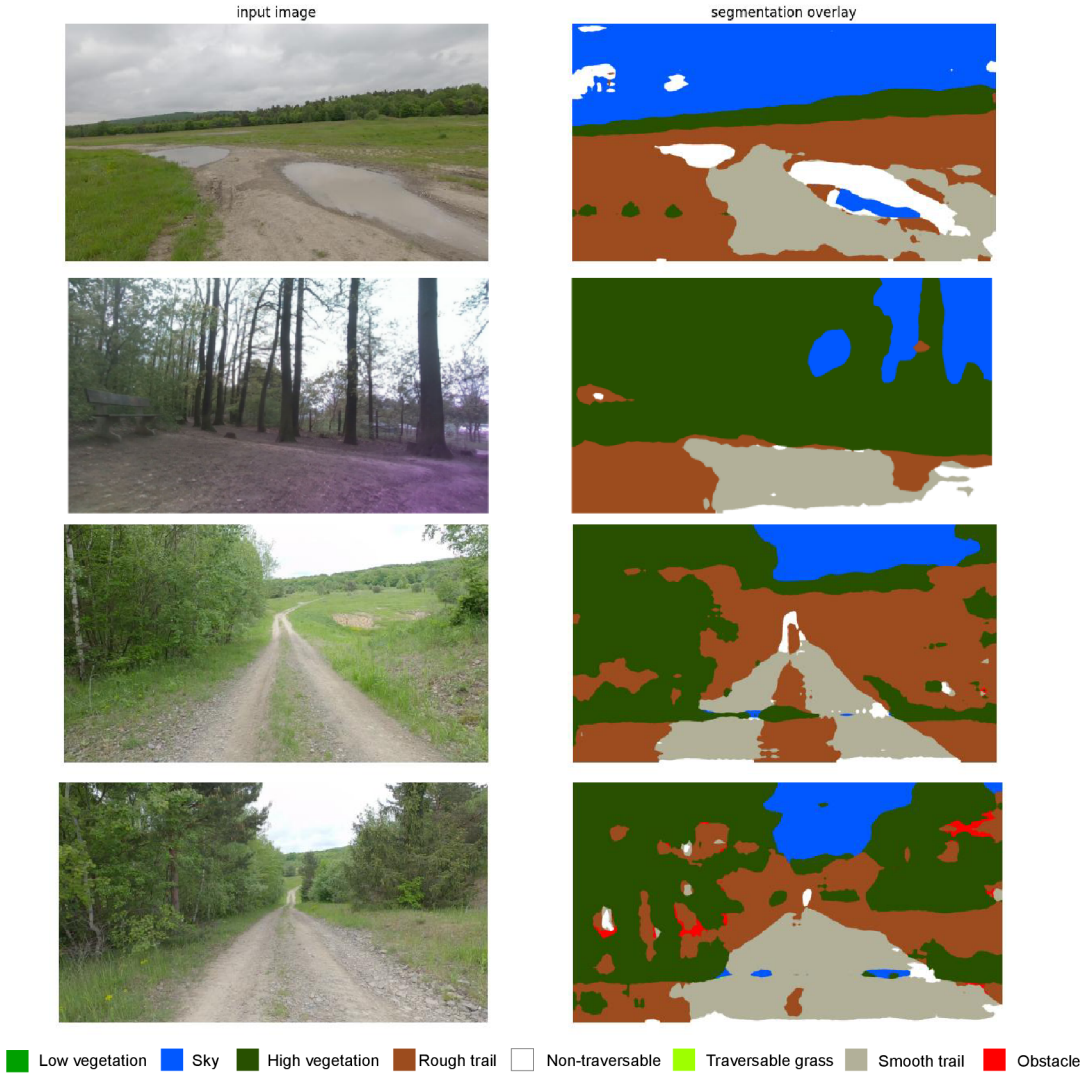


Figure 3.13: Predictions on the *moravia* part of the dataset

3.6 Results analysis, discussion

All models had issues segmenting dirt or sand roads when tested on the Yamaha-CMU Off-Road dataset. After reviewing the images in *train*, *validation* and *test* sets, it was noted that the split of the images was done poorly. There were a few images containing dirt roads in *train* set and most of them were put in the *test* set. This caused the model to perform the prediction of said dirt roads inaccurately. Because of that and the relatively small size of the dataset, the mean values were much lower. Rellis3D and RUGD datasets do not include many images with this type of terrain, their mean values were higher and the predictions were more accurate.

The predictions on the RUGD dataset were performed the best by the *Mnv3-pretrained* model which had the highest mean accuracy and MIoU score. The most precise predictions on the Rellis3D dataset were achieved with the *Res101-50* model and on the YCOR dataset with *Res101-10*. The fastest predictions were performed by the *Mnv3-10* model.

Most models could not segment bodies of water and objects such as buildings correctly. The bodies of water were usually segmented as sky due to the reflection of it in the water.

Objects and buildings were segmented as low vegetation. This was probably caused by the size of the training dataset (YCOR) and its insufficient variability of environments.

The *Mnv3-pretrained* was chosen as the best model overall. Its latency reached only 16 ms and the model performed well on the RUGD and Rellis3D datasets. When tested on the *moravia* part of the created dataset, it managed to predict smooth trails, high vegetation and rough trails. It had issues segmenting parts of low vegetation and rough trails and instead predicted these areas as high vegetation. When tested on the *bikepark* set, it could only segment images where the camera was on the rider's head facing the horizon. Images where the rider was looking down were not segmented well. The predictions were also performed poorly on images from dense forests and on images taken in winter.

4 Conclusion

This work was focused on semantic segmentation of off-road terrain which is a task of classifying areas of images. The first goal was to perform a theoretical survey on semantic segmentation. The basics of semantic segmentation have been summarized. After that were explored algorithms using traditional approaches and algorithms using neural networks, basic principles of neural networks and their different architectures were explained. The examples of the most common key performance indicators for evaluating semantic segmentation models were given, and their calculation was included. At the end of the theoretical survey, the possibilities of different data used for training and the various methods themselves were explored.

The second goal was to perform experiments. Testing of pretrained segmentation models on various datasets was the first task. The model by Haddad and Mulay was evaluated on two different datasets. The second task was to train segmentation models with diverse parameters and test them again on multiple datasets. Six models with three different backbone architectures and various training parameters were trained and then evaluated on three datasets. The evaluations of all models were compared and the pretrained model with *MobileNetV3-Large* backbone was chosen as the best according to its precision and speed. The other models had higher values of the Mean Intersection over Union in the training and validation phase, but their mean accuracy and Mean Intersection over Union values were lower when tested on different datasets.

The third task of the experiments was to create an evaluation dataset. Photos from off-road terrain in South Moravia and two different bike parks in the Czech Republic were collected. These images were sorted in two parts according to the location where the photos were taken, and a dataset consisting of 205 images was created. The best model that was chosen was then used for predictions on this dataset.

The model performed well on the images taken in South Moravia, it was capable of segmenting high vegetation, roads and surrounding terrain, but had issues segmenting buildings and obstacles. It was not able to perform sufficiently on the images taken in the bike parks. The images taken in spring and summer were classified only partly correctly - the model was capable of recognizing parts of smooth trails, sky and parts of high vegetation. The images taken in winter were not sufficiently segmented. This was caused by the different environment compared to the training dataset and the fact, that the images from bike parks were taken on a GoPro camera located on the driver's helmet - the images were taken from various angles.

The next steps for improving this work would be creating a training dataset with various environments in different weather conditions and fine-tuning the training parameters to gain better results. The possibilities of other architectures could be explored and a new semantic segmentation model created.

List of Abbreviations

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CRFs	Conditional Random Fields
EE	End Element
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
MIoU	Mean Intersection over Union
MRFs	Markov Random Fields
RDFs	Random Decision Forests
RGB	Red Green Blue
RNN	Recurrent Neural Network
SE	Start Element
SVMs	Support Vector Machines
YCOR	Yamaha-CMU Off-Road
YOLO	You Only Look Once

List of Figures

References

[1] THOMA, Martin. A Survey of Semantic Segmentation [online]. 2016 [visited on 2024-05-20]. Available from DOI: https://doi.org/10.48550/arXiv.1602.06541.

[2] LIU, Li; OUYANG, Wanli; WANG, Xiaogang; FIEGUTH, Paul; CHEN, Jie; LIU, Xinwang; PIETIKÄINEN, Matti. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision* [online]. 2019, vol. 128, no. 2020, pp. 261–318 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.1007/s11263-019-01247-4.

[3] LI, Yixin; LI, Chen; LI, Xiaoyan; WANG, Kai; RAHAMAN, Md Mamunur; SUN, Changhao; CHEN, Hao; WU, Xinran; ZHANG, Hong; WANG, Qian. A Comprehensive Review of Markov Random Field and Conditional Random Field Approaches in Pathology Image Analysis. *Arch Computat Methods Eng* [online]. 2021, vol. 29, pp. 609–639 [visited on 2024-05-20]. Available from DOI: https://doi.org/10.1007/s11831-021-09591-w.

[4] HEARST, Marti A. Support vector machines. *IEEE Intelligent Systems* [online]. July-Aug. 1998, vol. 13, no. 4, pp. 18–28 [visited on 2024-05-21]. Available from DOI: 10.1109/5254.708428.

[5] AGGRAWAL, Charu C. *Neural Networks and Deep Learning*. 2nd ed. Springer, 2023. ISBN 978-3-031-29641-3.

[6] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. Praha: Grada Publishing. Knihovna programátora (Grada), 2019. ISBN 978-80-247-3100-1.

[7] GURESEN, Erkam; KAYAKUTLU, Gulgun. Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science* [online]. 2011, vol. 3, pp. 426–433 [visited on 2024-05-21]. ISSN 1877-0509. Available from DOI: doi:10.1016/j.procs.2010.12.071.

[8] KOUSHIK, Jayanth. Understanding Convolutional Neural Networks [online]. 2016 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.48550/arXiv.1605.09081.

[9] MALLAT, Stéphane. Understanding Deep Convolutional Networks [online]. 2016 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.48550/arXiv.1601.04920.

[10] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition [online]. 2015 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.48550/arXiv.1512.03385.

[11] CHEN, Liang-Chieh; ZHU, Yukun; PAPANDREOU, George; SCHROFF, Florian; ADAM, Hartwig. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. 2018. Available from DOI: https://doi.org/10.48550/arXiv.1802.02611.

[12] LIN, Tsung-Yi; MAIRE, Michael; BELONGIE, Serge; BOURDEV, Lubomir; GIRSHICK, Ross; HAYS, James; PERONA, Pietro; RAMANAN, Deva; ZITNICK, C. Lawrence; DOLLÁR, Piotr. Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014* [online]. 2014, vol. 8693, pp. 740–755 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.1007/978-3-319-10602-1_48.

[13] JIANG, Peng; OSTEEN, Philip; WIGNESS, Maggie; SARIPALLI, Srikanth. RELLIS-3D Dataset: Data, Benchmarks and Analysis [online]. 2022 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.48550/arXiv.2011.12954.

[14] WIGNESS, Maggie; EUM, Sungmin; III, John G. Rogers; HAN, David; KWON, Heesung. A RUGD Dataset for Autonomous Navigation and Visual Perception in Unstructured Outdoor Environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* [online]. 2019, pp. 5000–5007 [visited on 2024-05-21]. Available from DOI: 10.1109/IROS40897.2019.8968283.

[15] MATURANA, Daniel; CHOUR, Po-Wei; UENOYAMA, Masashi; SCHERE, Sebastian. Real-time Semantic Mapping for Autonomous Off-Road Navigation. *Proceed-*

ings of 11th International Conference on Field and Service Robotics (FSR '17). 2017, pp. 335–350.

[16] GUAN, Tianrui; KOTHANDARAMAN, Divya; CHANDRA, Rohan; SATHYAMOORTHY, Adarsh Jagan; WEERAKOON, Kasun; MANOCHA, Dinesh. GA-Nav: Efficient Terrain Segmentation for Robot Navigation in Unstructured Outdoor Environments. *IEEE Robotics and Automation Letters* [online]. 2022, vol. 7, no. 3, pp. 8138–8145 [visited on 2024-05-21]. Available from DOI: 10.1109/LRA.2022.3187278.

[17] *GANav-offroad* [online]. 2022. [visited on 2024-05-21]. Available from: https://github.com/rayguan97/GANav-offroad.

[18] DIWAN, Tausif; ANIRUDH, G.; TEMBHURNE, Jitendra V. Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications* [online]. 2023, vol. 82, pp. 9243–9275 [visited on 2024-05-21]. Available from DOI: https://doi.org/10.1007/s11042-022-13644-y.

[19] *Yolov8 Architecture vs Yolov5*. 2023. Available also from: https://medium.com/@EG_Johnson/yolov8-architecture-vs-yolov5-49d23b462ea6.

[20] HADDAD, Nathaniel M.; MULAY, Amit Vijaykumar. *Semantic segmentation of off-road images using transfer learning and DeepLabv3+* [online]. 2022. [visited on 2024-05-21]. Available from: https://github.com/nmhaddad/semantic-segmentation/tree/master.

[21] HOWARD, Andrew; SANDLER, Mark; CHU, Grace; CHEN, Liang-Chieh; CHEN, Bo; TAN, Mingxing; WANG, Weijun; ZHU, Yukun; PANG, Ruoming; VASUDEVAN, Vijay; LE, Quoc V.; ADAM, Hartwig. Searching for MobileNetV3. *ICCV*. 2019. Available from DOI: https://doi.org/10.48550/arXiv.1905.02244.

[22] *CUDA Toolkit* [online]. 2024. [visited on 2024-05-21]. Available from: https://developer.nvidia.com/cuda-toolkit.

[23] *CONDA* [online]. 2017. [visited on 2024-05-21]. Available from: https://docs.conda.io/en/latest/%5C#.

[24] *Visual Studio Code* [online]. 2024. [visited on 2024-05-21]. Available from: https://code.visualstudio.com/.

Appendix

final	Folder with example output images from the final testing
README	ReadMe file with additional information
conf-rellis	An example of a configuration file for testing on the Rellis3D dataset
rellis-relabel	Code for relabeling the datasets
test-yamaha-v2	Example code of testing the models
training-demo	Code used for training the models