Position-Aided Road Auto-labeling with Self-supervised features in winter driving conditions

Eerik Alamikkotervo

School of Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 31.07.2023

Supervisor

Prof. Kari Tammi

Advisor

M.Sc. (Tech.) Alvari Seppänen

Aalto University
School of Engineering

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Eerik Alamikkotervo

Title Position-Aided Road Auto-labeling with Self-supervised features in winter driving conditions

Master Programme Mechanical Engineering **Code** ENG25

Supervisor Prof. Kari Tammi

Advisor M.Sc. (Tech.) Alvari Seppänen

Date 31.07.2023 **Number of pages** 54 **Language** English

Abstract

Road segmentation is a critical task for enabling the safe operation of autonomous cars. Currently, most road segmentation models rely on manually labeled data (supervised learning), making the training process very resource heavy. Also, there is a lack of data in adverse conditions like winter and supervised models generalize poorly to domains they have not seen during training.

In this thesis, a road segmentation model that requires no manual labeling (self-supervised learning) is presented. Training and testing are conducted in challenging winter driving conditions but the method can be adapted to any domain with no modifications. The proposed method includes two parts: Position-Aided Road Auto-labeling with Self-supervised features (PARAS) and learning from the PARAS auto-labels. In PARAS auto-labeling, the driven area is extracted based on Global Navigation Satellite System (GNSS) poses, and then the rest of the road is collected by comparing the mean similarity to the driven area with a pre-trained self-supervised feature extractor. Then a segmentation model is trained with the autogenerated labels using a custom loss function. The proposed method improves the current state of the art of self-supervised road segmentation in the winter driving domain (74.8 IoU vs 73.0 IoU) but can't yet compete with supervised methods. Most of the error in our method is caused by the inability to collect all road pixels by comparing feature similarity with the driven area. Performance could be increased by using a more accurate feature extractor or more advanced similarity metric than the simple mean that is used here.

The scalability of our proposed model is excellent as only GNSS and camera sensors are required and it avoids the label-assigning problem that is present in other approaches that utilize self-supervised features. In the current work, labels are assigned by simply clustering similar features together or using manually labeled data to learn projection from features to classes.

Keywords Autonomous driving, Computer vision, Road segmentation, Self-supervised learning, Winter driving conditions

Aalto University
School of Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Eerik Alamikkotervo

Työn nimi Paikkatieto-ohjattu tien automaattinen segmentointi käyttäen itseohjattuja piirteitä talviolosuhteissa

Maisteriohjelma Mechanical Engineering **Koodi** ENG25

Työn valvoja Prof. Kari Tammi

Työn ohjaaja DI Alvari Seppänen

Päivämäärä 31.07.2023 **Sivumäärä** 54 **Kieli** Englanti

Tiivistelmä

Tien tunnistus pikselitasolla, eli segmentointi, kaikissa olosuhteissa on edellytys turvalliselle autonomiselle ajamiselle. Tällä hetkellä suurin osa tien tunnistusmenetelmistä nojaa manuaalisesti merkattuun koulutusdataan (ohjattu oppiminen) ja datan merkkaaminen on hyvin aikaavievää prosessi. Lisäksi koulutusdataa ei löydy juurikaan poikkeavissa olosuhteissa, kuten talvella, ja mallit toimivat heikosti olosuhteissa, joissa niitä ei ole koulutettu.

Tässä työssä esitetään tien tunnistusmenetelmä, joka ei tarvitse lainkaan manuaalisia merkintöjä(itse-ohjattu oppiminen). Koulutus ja testaus suoritetaan vaativissa talviolosuhteissa, mutta menetelmää voidaan soveltaa mihin tahansa olosuhteisiin ilman muutoksia. Menetelmässä on kaksi osaa: automaattinen tiemerkintöjen luonti itsekehitetyllä PARAS (Position-Aided Road Auto-labeling with Self-supervised features)-menetelmällä ja tiensegmentointimallin kouluttaminen näillä merkinnöillä. PARAS-menetelmällä tiemerkinnät luodaan automaattisesti erottelemalla ajettu alue sateellittipaikannauksen (GNSS) avulla ja vertaamalla muiden kuva-alueiden samankaltaisuutta ajettuun alueeseen itseohjatulla piirteiden erotus mallilla. Tien tunnistusmalli voidaan sitten kouluttaa tällä autogeneroidulla datalla. Autogeneroitujen merkintöjen heikkouksia kompensoidaan itsekehitellyllä hukkafunktiolla. Menetelmä parantaa nykyistä itseohjatun tientunnistuksen tasoa talviolosuhteissa (74.8 IoU vs 73.0 IoU) mutta ei vielä pärjää ohjatuille menetelmille. Vaihe, jossa koko tie etsitään vertaamalla muiden alueiden piirteitä ajetun alueen piirteisiin, on menetelmän merkittävin virhelähde. Tarkkuuta voidaan parantaa käyttämällä parempaa piirteiden erotus mallia tai vertaamalla samankaltaisuutta kehittyneemmällä tavalla kuin keskiarvo, jota on käytetty tässä työssä.

Menetelmän skaalautuvuus on erinomainen, sillä vaatimuksena on ainoastaan kamera ja GNSS-paikannusanturi. Lisäksi muissa itseohjatuissa malleissa esiintyvä ongelma luoda tiemerkinnät piirteiden perusteella ratkaistaan käyttämällä hyväksi GNNS-paikkatietoa. Olemassa olevissa ratkaisuissa tiemerkinnät täytyy luoda joko yksinkertaisesti ryhmittämällä samankaltaiset piirteet yhteen tai kouluttamalla manuaalisesti merkatulla datalla projektio piirteistä luokkamerkintöihin.

Avainsanat Autonomominen ajaminen, Konenäkö, Tien segmentointi, Itseohjattu oppiminen, Talviajo-olosuhteet

Preface

This master's thesis was written as a part of the "Automated Vehicle Operation in Winter Conditions" -project funded by Henry Ford Foundation Finland. I wish to express my sincere gratitude to them for supporting this work.

I would like to thank Prof. Kari Tammi for supervising this thesis. His expertise and guidance have improved the thesis and most importantly he has created a positive and motivating working atmosphere.

I also give my thanks to Mr. Alvari Seppänen, Mr. Risto Ojala, and Mr. Jesse Pirhonen for making days in the office enjoyable and giving valuable advice. A special thanks goes to Mr. Seppänen for advising this thesis. His knowledge and enthusiasm in the field have been an immeasurable help when writing the thesis.

Espoo, 31.07.2023

Eerik Alamikkotervo

Contents

Terms

Convolutional Neural Network(CNN): Neural network architecture utilizing 2D kernels. Performs well in vision tasks because images contain 2D information that is hard to process in 1D form.

Dense features: Spatially distributed features (for example one feature for each 8x8 image patch) instead of one feature per image. Enable downstream tasks like segmentation.

East-North-Up (ENU): Cartesian coordinate system where a tangent plane is placed on the surface of the earth and poses are presented in reference to that so that the positive axis point to the east, north, and up. Errors are small close to the reference point while coordinates are easier to process in this frame.

Gaussian Mixture Model (GMM): Combines multiple Gaussian distributions to better model real-life properties.

Global Navigation Satellite System(GNSS): Global positioning system that is based on timed radio signals sent by satellites orbiting Earth. Typical accuracy is in the meter scale. The most common GNNS systems are GPS, Galileo, and GLONASS.

Ground Truth: The correct output, usually labeled by a human. Neural networks learn by adjusting their predictions toward Ground Truth.

Intersection over Union(IoU): Most common metric for segmentation accuracy. The shared area of the predicted segment and the ground truth segment is divided by the united area. $IoU = \frac{Pred \wedge GT}{Pred \vee GT}$.

Loss function: Function that neural network tries to minimize. Commonly, just the difference between predicted output and manually labeled Ground Truth.

Self-supervised learning: Learning without manually labeled data. Usually relies on data augmentations by learning to pull augmentations of the same image together and to push augmentations of different images apart.

Semantic segmentation: Task of assigning each pixel in an image into the correct class. Typical classes in traffic scenes are pedestrian, car, road, sidewalk, and building.

Supervised learning: Learning from examples, requires manually labeled data.

Visual Transformer(ViT): Transformer architecture adapted for vision tasks. Widely used in self-supervised vision models as the backbone.

Abbreviations

D	Mahalanobis distance (point to distribution)
L	loss function
s	scaling factor
u	image pixel coordinate in width direction
v	image pixel coordinate in height direction
X_W	world x-coordinate
Y_W	world y-coordinate
Z_W	world z-coordinate
$\mathbf{A}$	attention tensor
$\mathbf{C}$	camera matrix
$\boldsymbol{f}$	feature vector
$\boldsymbol{F}$	feature correspondence between feature vectors
$\mathbf{H}$	homography matrix
$\mathbf{P}$	projection matrix from world coordinates to image coordinates
$\mathbf{Q}$	query tensor
$\mathbf{R}$	rotation matrix
$\mathbf{T}$	translation matrix
$\mathbf{V}$	value tensor
$\boldsymbol{\mu}$	distribution mean
$\boldsymbol{\Sigma}$	distribution covariance

1 Introduction

1.1 Autonomous driving

Autonomous road vehicles can provide many benefits. They remove human error from driving and there is also a need for fewer cars and parking spaces as one vehicle can serve more people through car sharing [1]. Autonomous cars enable providing flexible transportation as a service that is accessible to everyone, compared to the current situation where a car is an expensive investment that only part of the population can afford. Autonomous operation and car sharing also support the transition to electric vehicles by lowering the production numbers required to full fill all transportation needs.

In order to do autonomous driving safely, three subtasks must be performed successfully: situational and environmental awareness, navigation and path planning, and maneuver control [2]. Situational and environmental awareness includes tracking other road users most importantly cars and pedestrians, self-localization, and detecting the road and lane the car is currently driving. Navigation and path planning include finding a global path from the start point to the destination that is locally adjusted to obstacles and conditions that have been detected. Maneuver control includes the control of the actuators to follow the planned path smoothly and safely. All the steps must be performed in real-time to allow safe operation in high-speed traffic.

The task of situational and environmental awareness is the most challenging out of the three. The most important awareness tasks are object detection and semantic segmentation. In object detection desired objects are located with rectangular bounding boxes and classified. In autonomous driving, the most important objects that must be detected are other road vehicles and pedestrians. In semantic segmentation, classification is done at the pixel level. The most important semantic segmentation classes in autonomous driving are road and sidewalk.

1.2 Neural networks for vision

After graphical processing units became viable for deep learning, neural networks have been dominating the field of visual detection tasks including object detection and semantic segmentation. Neural networks are inspired by the mechanism information is processed in biological neural networks (i.e. brains) and they are able to process complex data like images, text, and speech significantly better compared to classical methods. Convolutional Neural Network (CNN) is a type of neural network designed to operate on images with convolution operations. Before CNN:s images had to be processed as 1D arrays before feeding them to the neural network which effectively lost most of the contextual information of each pixel. CNN:s, on the other hand, use 2D kernels that are able to retain the 2D context of the images and they presented huge improvements over the previous state of the art. Most vision neural networks still utilize CNNs in the core while the high-level architecture has improved significantly from the first CNN implementations.

While neural networks offer better performance over classical methods in vision

tasks they also have drawbacks that are important to understand. Feng et al. [3] presented three major challenges for vision neural networks

1. Scalability: neural networks use a high number of parameters that require high computational resources both when training and testing which makes cost-effective scaling difficult. One solution to this problem is knowledge distillation where the knowledge learned by a larger model is attached to a smaller network while maintaining most of the generalization.

2. Robustness: neural networks are sensitive to small changes caused by noise, occlusion, and weather effects. Sensitivity can be decreased by aiming for high variation in the train set by collecting more data in different conditions. However, neural networks usually learn from examples, meaning the desired output must be marked in the training data, which can be very time-consuming. An emerging alternative to this is self-supervised learning where labels are not required.

3. Interpretability: Even though neural network architectures are handcrafted with a general idea of what each part of the network should do there is very little certainty about what is exactly happening inside. This can be addressed by simplifying the architecture and by visualizing outputs in the hidden layers of the network to gain a better understanding of the network.

All of these factors should be carefully considered when developing new neural network-based vision models.

1.3 Semantic road segmentation

Under the general task of traffic segmentation road segmentation is a popular topic of research where the goal is to detect all pixels that are part of the drivable area. This is a crucial task for autonomous driving. The road segmentation networks are usually trained and tested on a single dataset like KITTI [4] or Cityscapes[5] that lack variability, especially in weather conditions. While these models perform well in the domain they have been trained on they are not likely to perform well outside the dataset. It is evident that these models can't be used in real-life autonomous driving where robustness to different conditions is a requirement.

A trivial solution for this issue would be to use more data in different driving conditions, but in the case of road segmentation labeled data is very limited because segmentation is one of the most demanding labeling tasks. Due to the lack of labeled data, it is obvious that if a robust model is desired it is not feasible to just use human-annotated data sets. Fortunately, a significant development in the field of self-supervised segmentation has been made recently that enables learning without hand-labeled datasets.

1.4 Road segmentation in winter conditions

Winter driving conditions differ drastically from summer driving conditions: there is little contrast in color between road and roadside areas, the edge between road and roadside is not clearly defined and the road appearance can change from non-snow covered to fully snow-covered. These factors make segmentation networks trained on summer data infeasible to use in winter conditions. The same factors combined with slippery road surfaces make winter conditions demanding to human drivers as well. Snowy conditions are reported to increase the crash rate by 870% compared to dry conditions [6]. In many countries, including Finland, winter conditions dominate half of the year and in many other countries experience at least short periods of snowfall. In order to implement autonomous driving in these regions perception algorithms, including road segmentation, must adapt to winter conditions.

In this thesis, a self-supervised model for road segmentation in winter conditions is developed to address this issue. The proposed method consists of two parts: Position-Aided Road Auto-labeling with Self-supervised features (PARAS) and learning from the PARAS auto-labeled data. In the PARAS auto-labeling process driven area is extracted based on Global Navigation Satellite System (GNSS) poses and the similarity of other image areas is compared to the extracted area to collect the rest of the road. The performance of a self-supervised feature extractor is evaluated against classic methods for determining the similarity. In the second part, a road segmentation model is trained with the labels generated by PARAS, and a suitable loss function is developed to increase performance.

1.5 Scope

Autonomous cars use mostly lidar, stereo cameras, and monocular cameras for segmentation tasks. In this thesis, only one forward-facing monocular RGB camera and GNSSIMU position sensor are used on train time and only one forward-facing RGB camera is used on test time. In many approaches depth information from lidar or stereo camera is used for the generation of training labels or it is fused with the camera also on test time [7, 8]. Some models use labeled data from another domain or extend labels in time [9, 10, 11, 12]. In this thesis, the model will not see any labels at any point in the training procedure. The approach is completely self-supervised. Even though our proposed method can be used in any conditions, in this thesis training, and testing is conducted with daylight winter driving data. Similarly, benchmarking against other methods is conducted with this data. Testing in other conditions is not in the scope of this study.

1.6 Objectives

Must have:

1. Develop a road segmentation model that can be trained without labels in a self-supervised manner
2. Model works in winter conditions and can be easily adapted to any adverse conditions
3. Beats supervised methods trained with summer data
4. Model has novelty. Not just a fine-tuned version of existing work.

Nice to have:

1. Beats current self-supervised state-of-the-art
2. Beats supervised methods trained with winter data

2 State of the art

The state of the art of road segmentation can be divided into four subcategories: classic methods, supervised neural networks, semi-supervised neural networks, and self-supervised neural networks. The properties of each of them are briefly shown in Table 2.1 and a detailed background study for each of them is presented below.

Table 2.1: Properties of different road segmentation methods.

Method	Accuracy	Labeling	Feasible scenarios
Classic	OK	Not required	Simple environments
Supervised	Excellent	Required	Environments present in trainset
Semi-supervised	Good	Partially required	Environments present in trainset
Self-supervised	OK	Not required	Environments present in trainset

2.1 Classic methods

Classic methods try to segment the road based on manually defined features. The most commonly used features are road color distribution, road edges, the road vanishing point, and image horizon. Most approaches use a combination of multiple features. He et al. [13] utilize a combination of multivariate Gaussian distribution of road color and horizon, vanishing point, and edge detection to segment the road from an image. This method fails in areas where road color is not distinguishable from the background or road edges are not clear.

Almazan et al. [14] rely on probabilistic prior of average road location, road location registered to the vanishing point, and road location registered to the horizon location that are computed from a small dataset. To make a prediction, the horizon line and vanishing point are detected and probability distributions are anchored to these locations. Finally, these distributions are combined with the initial distribution before producing a prediction. The prediction accuracy is strongly dependent on the accuracy of the vanishing point and the road geometry. An ill-defined vanishing point will produce very poor results and even if a vanishing point is defined accurately the results hold only in straight road segments. Junction areas and curving roads are handled very poorly with this method.

Lieb et al. [15] present an adaptive segmentation algorithm that predicts future road segments based on the road appearance in the recent past. The road appearance at a given distance is estimated by doing reverse optical flow until the region of interest (ROI) is at the desired distance. Then features are extracted from this area and the best match is searched in front of the vehicle using template matching. The best match is chosen as the prediction for road position on that distance. Horizontal ROI is matched at multiple distances to provide a rough outline for the road segment and these Horizontal ROIs are then connected to produce a complete segmentation output. This approach needs to only assume that the road looks similar to the recent past and the road width is constant. The model is tested on data collected from the Mojave desert that has similar aspects to winter driving conditions: the

road edges are vague and there is little difference between the road and environment color. Despite these challenges, the model performs well on the test set. However, in urban conditions, a constant width of the road cannot be assumed which makes this approach inapplicable to autonomous driving on its own.

Ozguna et al. [16] utilize depth information captured with a stereo camera to segment road areas using disparity mapping. The performance on its own is quite poor but this method is later used to autogenerate labels for a neural network with better performance.

Liu et al. [17] fuse lidar with a monocular camera to segment the road. Lidar points are projected to the image and hand-crafted features are created based on the color and the point cloud information of the pixel. Classification of each pixel is performed with a Markov network with belief propagation.

Li et al. [18] extract features based on the intensity, color, and texture of each pixel and its close proximity with hand-crafted extractors. Road pixels are then clustered together using Gaussian Mixture Model and separated from non-road pixels. In this approach, satellite data is used and some processing steps require an assumption of a bird's eye view perspective to be made.

2.2 Supervised neural network models

Due to the shortcomings of classic methods, most road segmentation research in the last 10 years has concentrated on neural networks. They can learn more complex features compared to classic methods. The traditional approach is to use supervised learning where the model learns from examples. The model is provided with input images and the correct segmentation output (ground truth), where each pixel that is part of the road is separated from the rest of the image. Based on the difference between desired and model output parameters are tuned using backpropagation. With supervised learning, high accuracy can be achieved even in complex tasks if the training dataset is large and of good quality. However, it is time-consuming to produce training data for segmentation because each pixel has to be annotated.

The top 1 published method [19] on popular Cityscapes dataset [5] benchmark relies on deformable convolutions applied to vision transformer (ViT) like architecture. Previously ViTs outperformed CNNs in large-scale vision tasks, but the presented architecture brings CNNs back to the competition. The scope of the model is much broader than traffic scene segmentation as it can be utilized in a similar fashion to ViTs. These possibilities will be explored later in this thesis.

The Cityscapes top2 published method [20] implements additional boundary loss term to refine the performance in segmentation boundary areas. The boundary loss is computed by comparing the difference between predicted segmentation boundaries and ground truth segmentation boundaries. The model backbone is based on the architecture presented in [21] (HRNet) which is currently a common CNN-based backbone in segmentation tasks.

The top1 published segmentation model in winter conditions is presented by Vachmanus et al. [22]. The training and testing dataset is self-captured by the researchers with a total size of 1200 images. The model backbone is Resnet50 [23]

with ASPP module and pyramid supervision. The proposed model achieves the new state-of-the-art on winter road scene segmentation with road class IoU of 95.6 %.

Supervised models can be combined with classic color models to improve the performance of the model. Color models are not capable of road segmentation on their own but add value when used as a post-processing method. Yadav et al. [24] present a method where a CNN segmentation network produces an initial segmentation output for the road. Then a color line model is fitted to the background and road pixels and the probability of each point belonging to the road and background is computed. The final prediction is produced by feeding the probabilities to Conditional Random Field (CRF) model.

Alvarez et al. [25] present a hand-crafted color texture-based descriptor that can separate road areas significantly better compared to previous approaches. The novelty is to find a new color presentation that minimizes the variation inside an image patch as this presentation seems to separate different image areas better. The descriptor is combined with a segmentation network to refine the output of the neural network.

2.3 Semi-supervised neural network models

Semi-supervised methods require ground truth data but can extend the labels to different time instants or weather conditions. This lowers the required labeling effort but does not eliminate it. It should be also noted that the further the auto-labeled data is from the original domain the larger the error usually is.

The top1 method [9] in KITTI[4] dataset benchmark is a semi-supervised method that can auto-label frames close to a manually labeled reference point. The KITTI benchmark provides only 200 train and 200 test images so evidently ability to produce more training data will improve the accuracy of the neural network model. Here semantic cues and motion cues are combined to predict the labels near the ground truth reference point. The backbone is HRNet in this approach as well.

Another semi-supervised approach is to use scheduled learning where the difficulty of the training data is increased sequentially. Kothandaram et al. [26] present an approach where the model is first trained with data from good driving conditions including ground truth labels. Then unlabeled training data is divided into different difficulty categories and the model is trained sequentially with each category starting from the easiest category (closest to the original labeled dataset). The labels are generated online based on the model trained on the previous category using entropy minimization loss. This procedure is repeated until the target domain is achieved. Finally, the model is fine-tuned using a small set of labeled data from the target domain.

GAN-based data augmentation and neural style transfer allow the simulation of adverse driving conditions to already labeled images from a different domain. Generative Adversarial Networks (GANs) can generate whole new samples from the source data set utilizing generator-discriminator architecture. The generator network tries to produce new samples that the discriminator can't separate from the original data sample and the discriminator tries to separate all generated samples from real

ones, resulting in realistic fake samples. Neural style transfer augmentation takes style from one source image and content from another source image and combines them. This enables the transformation of images into a new domain that is not present in the original data set, summer to winter or day to night being obvious examples.

Mușat et al. [10] present an architecture that utilizes GAN modules to simulate adverse driving conditions into real images. Training with the generated data provides improved performance both on synthetic and real test data with adverse weather conditions compared to baseline but performance is not on bar with supervised methods. Choi et al. [27] present a GAN-based architecture where contents from virtual environment street scene images are mapped into representations of real-life images to generate new data samples. GAN-based style transfer architecture can even be used to change images from the summer domain to the winter domain [11] or from daylight style to low-light style [12] to enable training networks in conditions that are not present in the original dataset.

2.4 Self-supervised neural network methods

Fully self-supervised methods don't require any manual labeling, labels are either generated automatically or not used at all. Mayr et al. [7] and Wang et al. [8] auto-label segmentation data based on a disparity map generated with a stereo camera. Road pixels are assumed to locate on the same plane so they will share the same depth value in any given horizontal pixel row (disparity mapping). All pixels that belong to the same plane with some threshold are labeled as road. These labels are incomplete but the segmentation network trained with these labels produces predictions that are significantly better than the labels that it is learning from. This suggests that it is possible to train a good segmentation model with labels that only segment part of the road in each image.

Laddha et al. [28] utilize a combination of location data and available street maps for auto-labeling. With calibrated camera and location information, the road extracted from a map can be projected to the image and used as a label for road segmentation. A segmentation network can then be trained with these labels. This approach relies strongly on the accuracy of the street maps.

One option is to use class activation mapping, where parts of the images that have the highest effect on the predicted label are extracted. This idea was originally presented by [29] and later improved in [30]. However neural networks don't generally use all pixels part of the object for classification so CAMs don't produce accurate segmentation outputs. The performance is even lower for complex classes like roads that don't clearly stand out from the background.

Later it was discovered that the poor segmentation performance of CAMs could be improved by combining visual transformers with self-supervised training. Caron et al. [31] presented a feature extractor that is able to create meaningful dense feature representation of an image, completely self-supervised. The architecture consists of a student and teacher network with a visual transformer backbone that get a different transformation of the same image as an input. The student tries to

produce the same feature representation as the teacher despite having different input image transformations. The architecture is called DINO as it is doing knowledge DIstillation with NO labels.

Hamilton et al.[32] continued the work presented by [31] by adding a segmentation network to the feature extractor to produce a full self-supervised semantic segmentation model. Loss is calculated based on correspondence with three different image pairs: image with itself, image with its KNN nearest neighbor (chosen in the feature space), and image with another random image. Loss is minimized when the correspondence in the feature space is similar to the correspondence in the segmentation space. Essentially, this process is distilling correspondence from feature space to segmentation space. This model is called Self-supervised Transformer with Energy-based Graph Optimization (STEGO). The vanilla STEGO architecture has good performance on the Cityscapes dataset that consists of summer data but can't yet compete with supervised methods. The supervised state-of-the-art has 87.0 mIoU on Cityscapes and STEGO achieves 21.0 mIoU. Despite the drastic difference in mIoU STEGO can segment road areas reasonably well. The standard Cityscape benchmark includes multiple classes in addition to road and mIoU is calculated over all of them. When published, STEGO achieved the new state-of-the-art in self-supervised segmentation in the Cityscapes benchmark.

The field of deep learning moves quickly so STEGO didn't hold the first place for long. Recently after STEGO was published, the new state-of-the-art was gained by a model utilizing Visual Concept Embeddings (ViCE) [33] with mIoU of 25.2. The major difference to STEGO is that superpixel transform is used instead of using a pre-trained DINO backbone that has grid-like feature mapping.

Recently, also a new version of DINO has been published, named DINOv2 [34] where the loss mechanism presented in DINO with the loss mechanism of iBOT [35] and centering mechanism of SwAV [36]. The implementation is also optimized which allows training on larger datasets. All of these factors together result in better quality dense and image-level self-supervised features. Fully self-supervised segmentation benchmarks have not been provided but linear probe evaluation where a linear classifier is trained on top of the features DINOv2 achieves significantly higher mIoU compared to DINO in all benchmarks including Cityscapes (66.2 mIoU vs. 81.0 mIoU).

2.5 Research gap

The background study reviled that currently, the best-performing road segmentation models are dependent on labeled data. Most approaches use the Cityscapes dataset for training and testing which hides the shortcomings of the models in adverse conditions. Some efforts have been made to collect and label winter driving data for training suitable models for winter driving conditions but these datasets are small and non-public. Self-supervised approaches have been presented to address the issue of manual labeling but the accuracy leaves room for improvement.

Our PARAS auto-labeling combines recently published self-supervised dense feature extraction models with GNSS pose data to present a novel self-supervised

road segmentation architecture that aims to improve on the current state-of-the-art in self-supervised road segmentation. At the same time, the lack of research in adverse conditions is addressed by doing training and testing in varying winter conditions. Winter conditions are very challenging for the current road segmentation models as they have mostly been trained in summer conditions and neural networks generalize poorly to conditions that they have not been trained on. On the other hand, existing winter driving models use such small datasets that likely, they don't generalize well to different snow, light, and environmental conditions.

3 PARAS auto-labeling

In this section, the label autogeneration process of our proposed method PARAS is presented in detail, including the data used, the driven area extraction process, and the method for comparing similarity with the driven area. The performance of self-supervised features is tested against a classic GMM-based similarity metric.

3.1 Datasets

For our method, a dataset with the following properties is required:

1. time stamped or synchronized GNSS pose-image pairs
2. calibration parameters for projection from the ground plane to the image frame
3. winter driving conditions

The number of data sets that have all these properties is very limited even though labels are not required here. The only larger dataset with these properties is the Canadian Adverse Driving Conditions Dataset (CADCD) [37]. The performance of neural networks is strongly correlated with the amount and quality of the data so there is clearly a need for more raw winter driving data. With this motivation, a new unlabeled winter driving dataset is collected. The advantages of self-supervised learning become evident here: lack of data in the desired environment can be easily tackled with new data because labeling is not required.

CADCD
Even though the raw size of CADCD is relatively large, 263 637 images, it has been captured on a route of just 20 km with a high frame rate (10 Hz) and 8 different cameras meaning that only 1/8th of the images are forward facing and the variation between consecutive frames is small. Fortunately, a labeled (object level) subset of the dataset samples approximately every fifth frame of the raw data and covers most of the whole route. The labeled subset also includes synchronized GNSS poses. The labeled dataset includes 7000 forward-facing images distributed to three different driving days. On only one of the driving days, there is snow on the road, leaving 5430 images with desired properties.

In CADCD the extrinsic and intrinsic calibration parameters are provided. Intrinsic parameters define the projection from camera coordinates to the image frame and extrinsic parameters define the transformation between the camera frame and the vehicle frame.

Self-collected dataset
Our dataset is collected in the Finland Espoo area on two different days in February 2023. Driving scenes include different snow conditions: full snow cover, partial snow cover, and no snow cover on the road and different driving environments: suburban, highway, and countryside roads. All data collection has been done during day time. The specialty of the dataset is fully snow-covered countryside roads with very little

color contrast between road and roadside areas. This kind of data is not present in the CADCD dataset. The driving scenes include suburbs, highways, countryside, and intersections (Figure 3.1). A wide variety of scenes were included so that a model trained with this data would work in any winter driving scenario.

Figure 3.1: Driving scenes from the self-collected dataset.

Data collection was conducted with Henry research car platform donated by Henry Ford Foundation Finland. The platform is a 2019 Ford Focus retrofitted with a high-quality sensor setup and an onboard computer that enables data collection tasks. The sensor setup used in this dataset is:

1. Novatel PWRPAK 7DE GNSS with dual antenna and INS unit
2. FLIR blackfly S 2448x2048 forward-facing machine vision camera

The Novatel GNSS system is connected to Robot Operating System (ROS) that is running on the onboard computer. The camera runs on its own program independent from ROS but every time a frame is captured ROS timestamp is saved to allow synchronization. Thus raw data is distributed to three different files: the GNSS pose data is saved into a rosbag file, the camera data is saved to an AVI video file and the image frame timestamps in ROS time are saved to a text file.

The raw data contains around 184 000 frames captured with 20Hz frequency meaning there is little difference between consecutive frames. Thus it is justified to downsample the data by only taking every fifth frame before any further processing yielding 36 738 downsampled frames. The GNSS translations are extracted from Novatel/oem7/bestpos topic which is published at 10 Hz frequency and the rotation is extracted from Novatel/oem7/inspva topic which is published at 50 Hz frequency.

Each image can be paired with the closest GNSS pose simply by comparing the timestamp of the image to all GNSS pose timestamps and then choosing the closest one. 2D bounding boxes for cars are detected with pre-trained Yolo v5 [38] and saved

to a text file. The data is now distributed to three different file types: image folder, closest GNSS poses for each frame in a text file, and 2D bounding boxes for each frame in a text file. This format is adapted from the CADCD dataset [37].

In order to project pixels to the world coordinate frame transformation between these frames needs to be determined. Instead of defining the translation matrix, rotation matrix, and camera matrix separately, the desired projection can be computed from a single measurement, because the transform between world coordinates and image pixel coordinates is a homography transform. By definition, homography transform maps points from one plane to another. Now, the source plane is the ground and the destination plane is the image. Homography transform should not be confused with a camera matrix that maps 3D world coordinates to the image frame such that the world coordinates frame origin is in the focal point and in the same orientation as the camera.

By measuring a minimum of four point pairs of form $[(u, v), (X_W, Y_W)]$, where (u, v) is the image coordinate and (X_W, Y_W) is the world coordinate, the homography transform can be determined. The world coordinate-locations of the sample points are measured with a laser measuring tool as illustrated in Figure 3.2. After measurements, the optimal homography matrix is calculated with the OpenCV findHomography method so that the least squares back projection error is minimized.

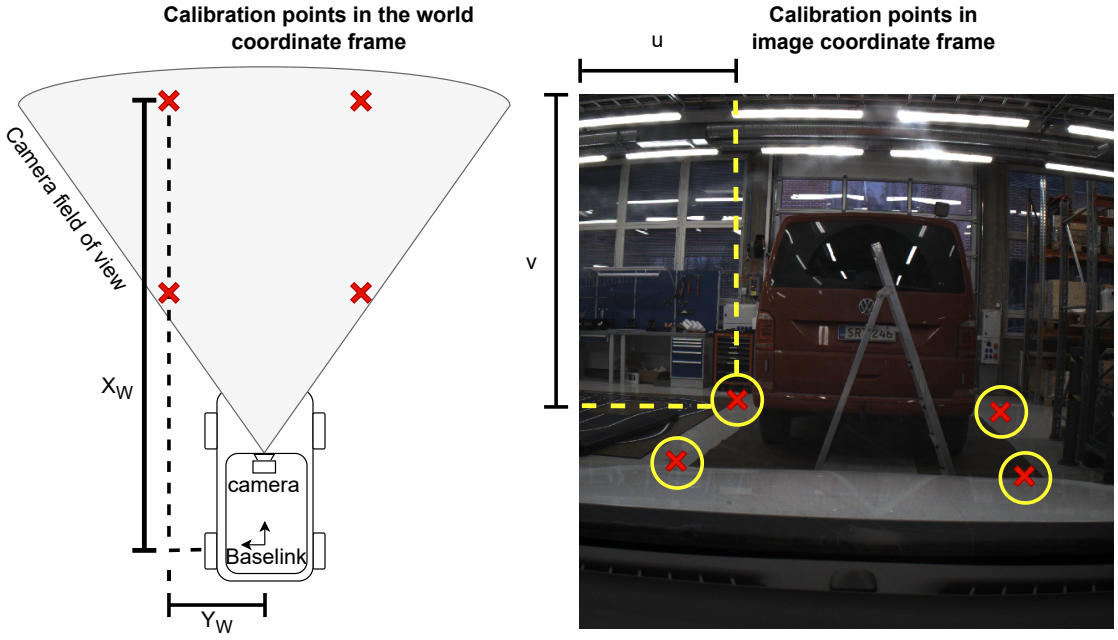


Figure 3.2: Camera calibration. The position of each point is measured in world coordinates and image coordinates and then the homography transform between these planes can be defined.

3.2 Image to ENU projection

In this section, the necessary formulas are presented for projecting image coordinates to world coordinates. Here we have two different cases. The calibration process of our self-collected data set yields a homography matrix directly, but for CADCD the homography matrix needs to be derived from the camera matrix, rotation matrix, and translation matrix. For both the assumptions are the same:

1. The world coordinate origin is in the GNSS receiver location
2. The ground is assumed to be planar, meaning that 2D mapping from image coordinates to the ground plane is desired. The accuracy of the mapping depends on the distance between the ideal 2D road plane and the real road.

If the camera matrix is known the projection from camera coordinates to image coordinates is defined as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underset{3\times3}{\boldsymbol{C}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{3.1}$$

where s denotes the scaling factor and u and v are the image coordinates in pixel units. $\boldsymbol{C}$ defines the projection from the camera frame to the image frame, also called the camera matrix. x,y z are coordinates in the camera frame. The transformation from the world coordinate frame to the camera frame is given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \underset{3\times3}{\boldsymbol{R}} & \underset{3\times1}{\boldsymbol{T}} \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}, \tag{3.2}$$

where $\boldsymbol{R}$ defines the rotation matrix from the world coordinate frame to the camera frame and $\boldsymbol{T}$ defines the translations from the world coordinate frame to the camera frame. Together they form the transformation matrix that defines the full transformation from the world coordinate frame to the camera frame. X_W, Y_W, and Z_W are the resulting coordinates in the world coordinate frame. Combining the rotation matrix, translation matrix, and camera matrix yields

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underset{3\times3}{\boldsymbol{C}} \begin{bmatrix} \underset{3\times3}{\boldsymbol{R}} & \underset{3\times1}{\boldsymbol{T}} \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \underset{3\times4}{\boldsymbol{P}} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}, \tag{3.3}$$

where $\boldsymbol{P}$ is the projection matrix from world coordinates to image coordinates. As we assume all world coordinates to be located in the ground plane we can remove the Z_W from the input coordinates and remove the corresponding column from the projection matrix, which yields the homography matrix representation:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underset{3\times3}{\boldsymbol{H}} \begin{bmatrix} X_W \\ Y_W \\ 1 \end{bmatrix}. \tag{3.4}$$

However, we need to project image coordinates to world coordinates, meaning that we need to solve X_W and Y_W from Equation 3.4 when u, v and $\boldsymbol{H}$ are known:

$$X_W(u,v) = \frac{H_{12}H_{23} - H_{13}H_{22} + H_{22}H_{33}u - H_{23}H_{32}u - H_{12}H_{33}v + H_{13}H_{32}v}{H_{11}H_{22} - H_{12}H_{21} + H_{21}H_{32}u - H_{22}H_{31}u - H_{11}H_{32}v + H_{12}H_{31}v} \tag{3.5}$$

$$Y_W(u,v) = \frac{H_{11}H_{23} - H_{13}H_{21} + H_{21}H_{33}u - H_{23}H_{31}u - H_{11}H_{33}v + H_{13}H_{31}v}{H_{11}H_{22} - H_{12}H_{21} + H_{21}H_{32}u - H_{22}H_{31}u - H_{11}H_{32}v + H_{12}H_{31}v}. \tag{3.6}$$

The relationships between the world, camera, and image coordinates are illustrated in Figure 3.3.

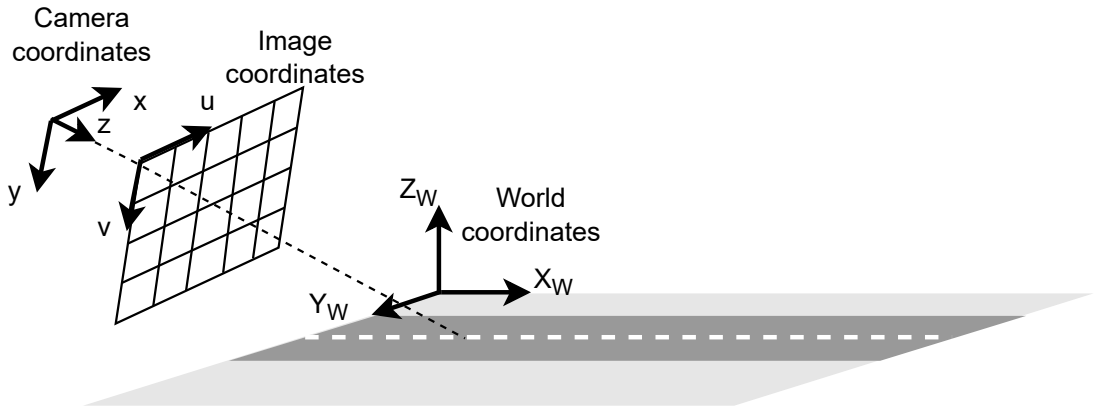


Figure 3.3: World coordinates, camera coordinates and image coordinates visualized. The camera coordinates origin is located in the focal point of the camera.

3.3 Driven area extraction

The first stage of the proposed method is to map the route that the car drives in the immediate future to the image and extract all pixels close to that route. The driven area extraction includes the following steps:

1. Transform GNSS poses to ENU frame
2. Transform image coordinates to ENU frame
3. Collect pixels close to the driven route
4. Remove vehicles

Each of the steps are presented in detail below and the full process is illustrated in Figure 3.4.

Transform GNSS poses to ENU frame
East-North-Up(ENU) frame mounted on the starting position of the car is chosen as the global coordinate frame so the image must be converted to this frame. ENU is a local tangent plane frame that uses a cartesian coordinate system that is mounted to a reference point on the surface of the earth where the positive x-axis points to the east, the positive y-axis to the north, and the positive z-axis up. ENU coordinates are easier to process compared to the latitude-longitude system and the projection error is small if distances from the origin point are small. GNSS poses are first converted to cartesian Universal Transverse Mercator (UTM) frame using the utm python library and then to ENU by subtracting the original position from the UTM coordinates.

Transform image pixels to ENU-world coordinate frame
In the previous section mapping from image coordinates to the world coordinate frame is solved. In this mapping, the origin is located in the GNSS receiver location. However, the mapping from the image to the ENU frame can be easily solved by multiplying the local world coordinates, with the ENU transformation matrix $\boldsymbol{T}_{ENU}$ that transforms local world coordinates to the global ENU frame:

$$\begin{bmatrix} X_{ENU}(u,v) \\ Y_{ENU}(u,v) \end{bmatrix} = \underset{2\times2}{\boldsymbol{T}_{ENU}} \begin{bmatrix} X_W(u,v) \\ Y_W(u,v) \end{bmatrix}. \tag{3.7}$$

Collect points in the driven route
Now the image is projected to ENU coordinates as well as the GNNS poses. The next step is to collect all poses starting from the moment the image was taken until the accumulated Euclidian distance between the poses reaches 50 m. This yields a set of poses that present the vehicle's route in the immediate future. It is desired to somehow collect all pixels that are close to this path. Here, a least square error circle is fitted to the poses with Scipy optimize package [39] and then all pixels closer than 1m to this circle are collected. A circle provides a good fit for poses both in straight and curving segments. The output of this process is a mask that is set to True for pixels that are part of the driven route. The pseudocode for the operation is presented in Algorithm 1.

Remove vehicles
In some cases, there can be cars in the area where the car drives during the next 50 m. Cars have to be removed from the mask because it is not desirable to include other cars in the drivable area. In most data sets bounding boxes for cars are provided and if not other cars can be detected using an object detection algorithm. In this thesis, pre-trained Yolo v5 is used [38].

Algorithm 1 Collect points in the driven route

for imgIndex = 0 to numImgs **do** ▷ Go through all images
  $drivenDist \leftarrow 0$ ▷ Set driven distance to zero
  $i \leftarrow imgIndex$ ▷ Set starting index to the current image
  while driven < 50 **do** ▷ Repeat until 50m has been driven
    $i \leftarrow i + 1$ ▷ Go forward one pose
    $drivenDist \leftarrow drivenDist + ||poses[i], poses[i-1]||_2$ ▷ Current distance
  end while
  $next50m \leftarrow poses[imgIndex\ to\ i]$ ▷ Collect poses in the next 50m
  $x_c, y_c, r, maxAngle \leftarrow fitCircle(next50m)$ ▷ Fit circle to the poses
  for u = 0 to imgWidth **do** ▷ Go through each pixel
    for v = 0 to imgHeight **do**
      $currentPose \leftarrow [X_{ENU}[u,v], Y_{ENU}[u,v]]$ ▷ Current pose
      $firstPose \leftarrow poses[imgIndex]$ ▷ Starting pose
      $dist \leftarrow |||[currentPose, [x_c, y_c]|||_2$ ▷ Distance from circle center point
      $angle \leftarrow \angle(currentPose, firstPose)$ ▷ Angle between poses
      if $(r-1) < dist < (r+1)$ & $angle < maxAngle$ **then**
        $mask[u,v] \leftarrow True$ ▷ True if distance and angle condition met
      end if
    end for
  end for
  save mask ▷ Save mask for current image
end for

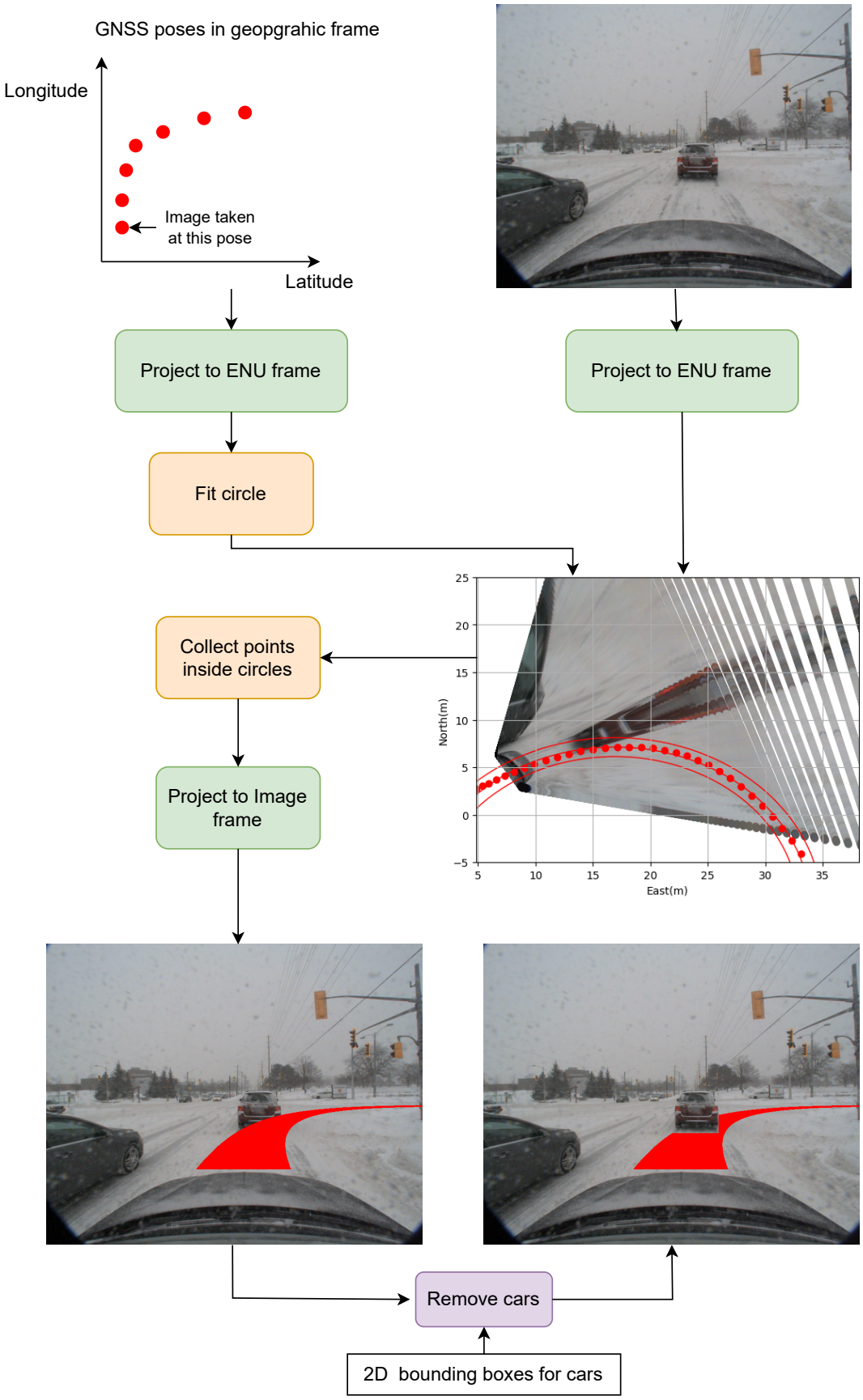


Figure 3.4: Driven area extraction process.

3.4 Similarity comparison with Gaussian mixture model

Many real-life properties can be modeled as Gaussian distribution. The usability of Gaussian distributions is inherited from the central limit theorem which states that when adding together multiple independent observations of a random variable the resulting distribution converges into Gaussian distribution [40]. However, sometimes a single Gaussian distribution can't model the problem accurately, so a combination of multiple Gaussian distributions must be used. A combination of multiple Gaussian distributions is called a Gaussian Mixture Model (GMM)[41]. The density function f of a GMM is defined as

$$f = \sum_{i=1}^{K} \phi_i N(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}), \tag{3.8}$$

where ϕ_i is the weight of i:th Gaussian component and $N(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$ is i:th multivariate Gaussian distribution defined by mean vector $\boldsymbol{\mu_i}$ and covariance matrix $\boldsymbol{\Sigma_i}$.

It is justified to model the road with GMM instead of using one Gaussian distribution because snowy roads usually have multiple distinctive parts that require their own Gaussian components, like pure snow-covered parts and clear asphalt parts. The suitability of GMM is tested with histogram visualization of the image where the GNSS mask and the rest of the image are separated. Then GMM is fitted to the GNSS mask to see the correspondence between the histogram and the fitted GMM model. The result is presented in Figure 3.5.

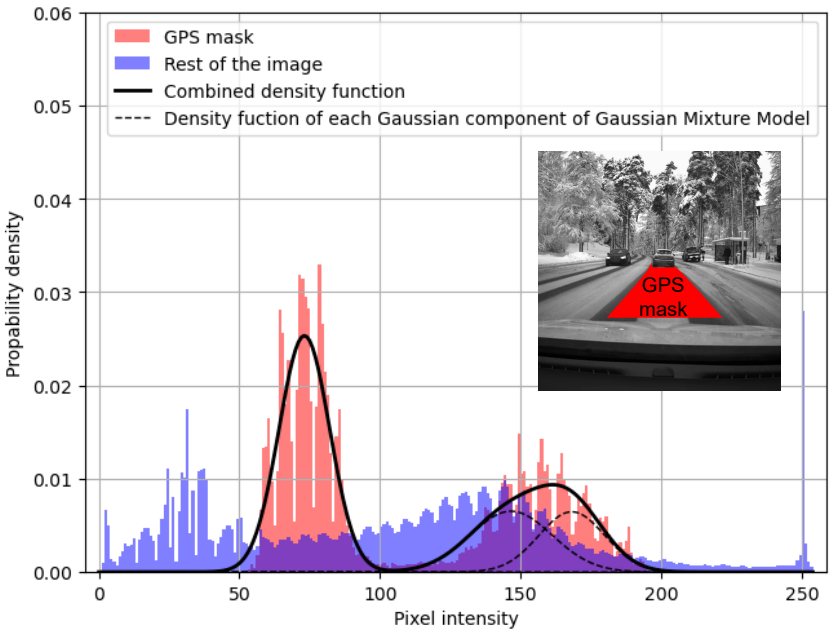


Figure 3.5: Distribution of driven area (GPS mask) compared to the rest of the image and fitted GMM.

Similarity with single pixel

After modeling the driven area as GMM, another critical task is to find the similarity between a given image area and the fitted GMM model. Essentially we are looking for a metric for measuring the distance between a point and a probability distribution. For this purpose, Mahalanobis distance is well suited. It describes the distance between a distribution and a point, taking into account both the mean and covariance of the distribution. Mahalonobis distance is defined as

$$mahal(\boldsymbol{x}, N(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}, \tag{3.9}$$

where d is the Mahalanobis distance, x is the point vector, S is the covariance matrix of the distribution and μ is the mean vector of the distribution. Mahalanobis distance measures the distance between a point and a distribution in N-dimensions. Here three dimensions are used, one for each color channel (R, G, B). Mahalanobis distance is separately computed for each component of GMM and multiplied with the corresponding GMM weight ϕ_i to formulate a combined distance that represents the similarity between a given point and the GMM:

$$D = \sum_{i=1}^{K} \phi_i mahal(\boldsymbol{x}, N(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})), \tag{3.10}$$

where D is the combined Mahalanobis distance between point x and GMM consisting of Gaussian components $i = 1...K$. However, this metric is not normalized to any fixed scale. The minimum and maximum values change in each image, which is not desired for a similarity metric that is used across images. Instead, it is desired to produce similarity values that are always scaled between 0-1 so that the point with the highest similarity in each image has a value of 1. This can be achieved by taking the shortest Mahalonobis distance D_{min} and dividing it by each of the distances D which yields normalized values D_{norm} in the range of 0-1:

$$D_{norm} = \frac{D}{D_{min}}. \tag{3.11}$$

Average similarity with image patch

It can be quite challenging to evaluate similarity based on a single point because essentially the only available information is the color of the point. To get more accurate results, a larger sample area is required. One option would be to just divide the image into square patch samples but this would cause problems in the road edge areas as the road geometry doesn't agree with simple square patches. However, knowledge of the road shape is available in the GNSS mask and it can be utilized here to form more accurately shaped sample patches. In the same way that the GNSS mask is computed, we can collect all pixels that belong to a specific strip on either side of the GNSS mask to form a collection of strips that are of the same shape as the GNSS mask by redefining the radius parameter. As the GNSS mask follows the road curvature, the generated patches will also follow the road curvature. The image is distributed to 0.5m wide segments in the world coordinates based on the GNSS mask and the average Mahalanobis distance is then computed for these

segments. A low average distance suggests that the segment is a close match with the driven area.

3.5 Similarity comparison with self-supervised feature extractor

Neural networks can extract features from images that are more complex compared to classic methods. The difficulty is training the network to create meaningful features. Supervised segmentation networks usually utilize architecture, where images are first transformed to feature space, and then segmentation labels are defined based on those features. The working principle can be simplified as first creating meaningful features for each image area and then grouping areas with similar features together. However, this approach requires labeled data.

As briefly presented in the state-of-the-art section promising self-supervised feature extractor architecture (DINO) has been published in [31] and the features have been found suitable for segmentation (STEGO) in [32]. The results presented with STEGO are promising but the accuracy has room for improvement. However, in this study, there are two main advantages compared to that approach:

1. Only the road needs to be segmented. Other classes are not required. This means all effort can be concentrated on separating road pixels and non-road pixels.

2. Part of the road is extracted based on the GNSS poses and is available during training.

In this context, the target is to create labels for road utilizing the GNSS mask. The approach is similar to the GMM but instead of measuring the distance to the GMM, the similarity of each image area is evaluated by comparing the DINO features in the GNSS mask and each image area. DINO is chosen as the feature extractor based on its performance and it has already been used for segmentation in STEGO.

Arcitechture

First, it is important to understand the architecture and training procedure used for creating DINO as labels are directly derived from the features produced by DINO. The backbone of DINO is a Visual Transformer (ViT). Transformers were originally presented in the context of natural language processing [42]. Transformer takes a sample text as input and in addition to each word being encoded to a vector, each word has also positional encoding related to it. This way the model knows the positional relationship between the words which is crucial when processing natural language. The positional encoding is used for self-attention: when translating each word the model can decide which words it should attend to instead of translating word by word. Thanks to the self-attention mechanism and easily parallelizable training that enable bigger training datasets transformer set the new state-of-the-art in translation accuracy. After the success in the language domain, transformers

have been implemented in the image processing context as well [43, 44, 45] yielding so-called Visual Transformers (ViT). Visual transformers give positional encoding to image patches instead of words but otherwise, the architecture is very similar. The architecture of a Visual Transformer as used in DINO is presented in Figure 3.6.

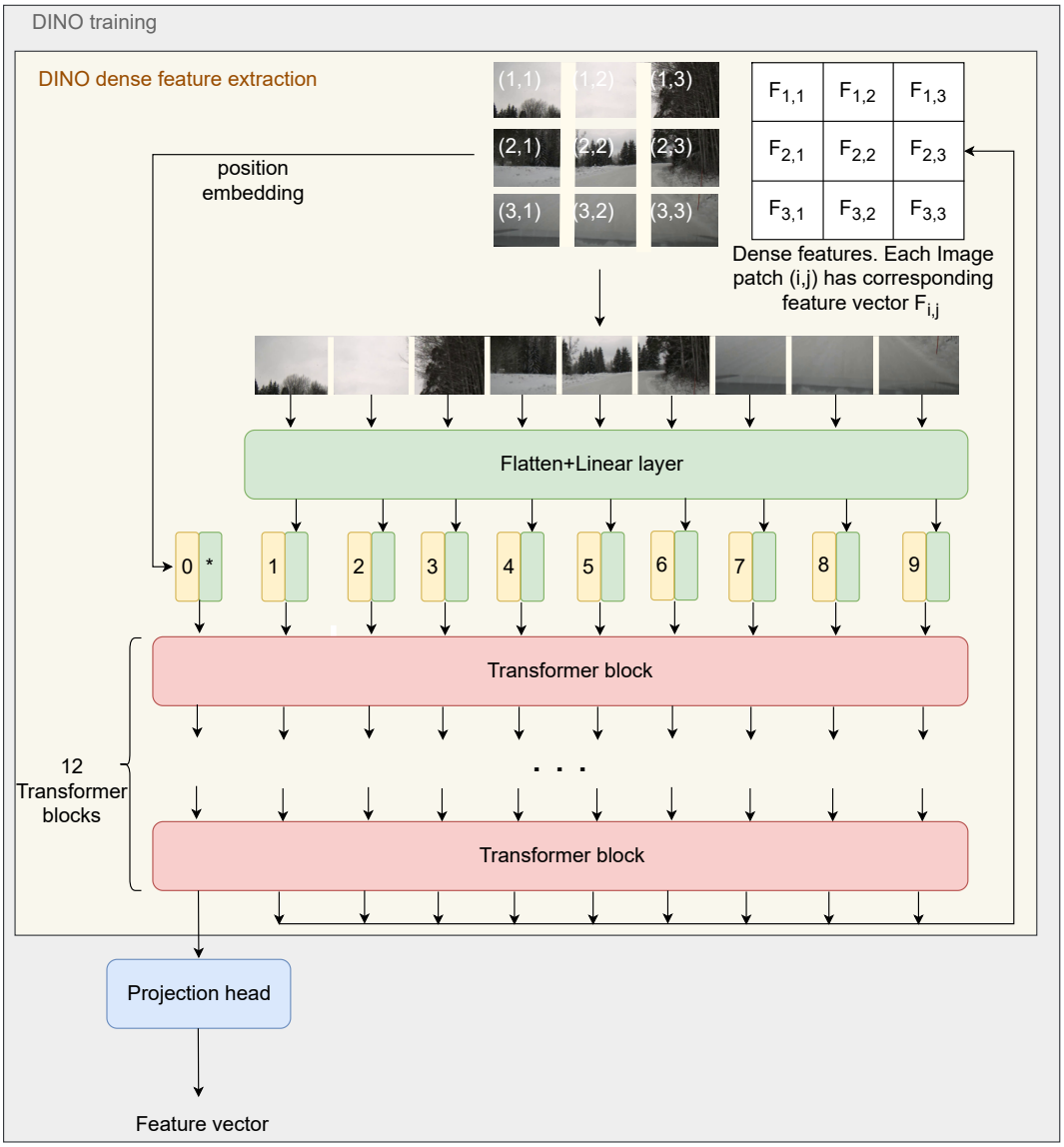


Figure 3.6: Visual transformer architecture. DINO only uses the class token value fed through the projection head during train time but can still produce meaningful dense features. Visualization inspired by [43].

In ViTs input image is first divided into image patches. Each of these image patches is projected to a 1D vector by a linear layer. To capture the positional information, the location of the image patch is also embedded in a 1D vector of the same size as the patch vector. These two vectors are then summed together to produce a complete embedding of the image patch. This embedding contains information about both

the location and content of the image patch. These embeddings are generally called tokens in the context of transformers Additionally, one extra embedding, called a class token is added to the sequence. Its embedding is not connected to an image patch like the other tokens. Instead, its task is to extract general classification information about the image by attending all the patch tokens as the image is classified just based on this token. All tokens, including the class token, are fed together to a series of transformer encoder blocks. In each transformer encoder block, there are the following operations: Layer norm, Multi-Head Attention, and MLP (Figure 3.7). The layer norms are just regularization operations and the Multi-Head Attention and MLP are the parts that are actively learning.

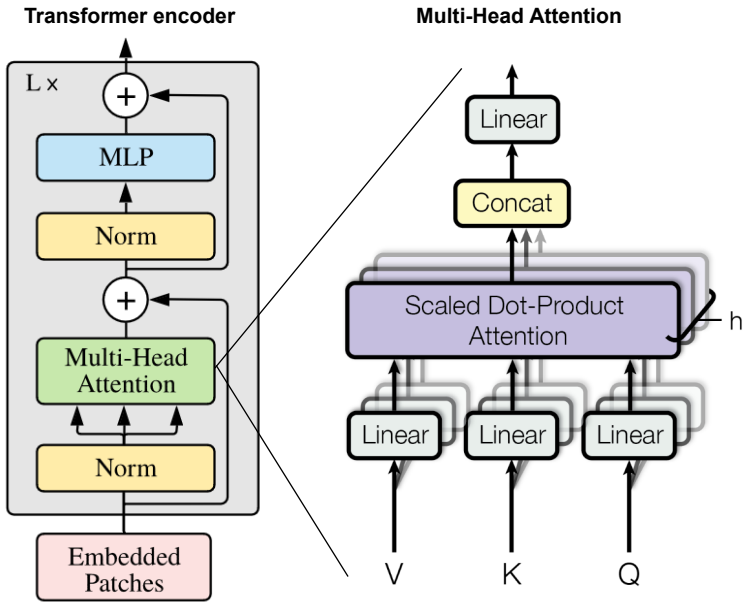


Figure 3.7: Arcitechture of a transformer encoder block used in visual transformers. Adapted from [42, 43].

The Attention block allows the model to attend to all tokens with different weights to produce a new output token. First, the input is fed through three parallel linear layers that are independent of each other. This outputs three different mappings of the input that are called Value($\boldsymbol{V}$), Key($\boldsymbol{K}$), and Query($\boldsymbol{Q}$) tensors respectively. Then the $\boldsymbol{V}$ tensor is scaled based on the scaled Dot-Product Attention that is defined as:

$$\boldsymbol{A} = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}, \tag{3.12}$$

where $\boldsymbol{A}$ is the resulting attention tensor and d_k is the dimension of the $\boldsymbol{Q}$ and $\boldsymbol{K}$ tensor. The idea of the attention mechanism is to compute the new value vector of each token as a weighted sum of the value vectors of all tokens. The weight for each value pair is computed by taking the dot product with the query vector and the key vector, dividing it with scaling factor d_k, and finally normalizing with Softmax.

Here Multi-Head Attention is used which means that there are multiple Attention heads in parallel. These heads are completely independent of each other so their advantage over single-head attention is that each head can learn a different mapping for the $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ tensors and thus each head can attend to different parts of the image. This behavior was confirmed by the authors of DINO who discovered that different heads attended to subparts of an object accurately. The new value vector from each of the heads is concatenated together and fed through a linear layer to produce the final output of the Multi-Head Attention module. Finally, the output of the Multi-Head Attention module is refined by an MLP to produce the output of the transformer encoder block. There are skip connections around the Attention module and the MLP inspired by residual networks [23].

Training procedure

When training ViTs with labels the performance has been barely better than with CNNs and the attention maps are too sparse for segmentation [43]. The major innovation in the DINO paper is to train these ViTs in a self-supervised manner which produces good results in classification tasks while retaining spatially accurate attention maps even though the model is not explicitly trained for that.

DINO training procedure starts by computing two random transformations for an image and one of them is fed to the teacher network and one of them is fed to the student network. The teacher and student networks share the same architecture, where a Visual Transformer (ViT) backbone is combined with a projection head. The projection head is a simple 3-layer multi-layer perceptron with a hidden layer size of 2048 that is followed by a Euclidian norm and weight normalized fully connected layer. The projection head takes only the class token of the transformer as its input.

The output of the teacher network is centered by subtracting the mean over the batch. Centering is followed by temperature softmax. Temperature softmax is an operation where the output is randomized to a desired degree before feeding it to the softmax. In practice randomizing can be done by diving the logits with a constant, the larger the constant the more randomized the output is. If the constant is smaller than 1 the operation sharpens the output between dimensions and if the constant is larger than 1 the operation smooths the output between dimensions. In DINO implementation temperature is set to 0.1 meaning that the output is sharpened before softmax pushing dimensions further apart. The output of the student network is not centered but temperature softmax is applied to it. The full training setup is presented in Figure 3.8.

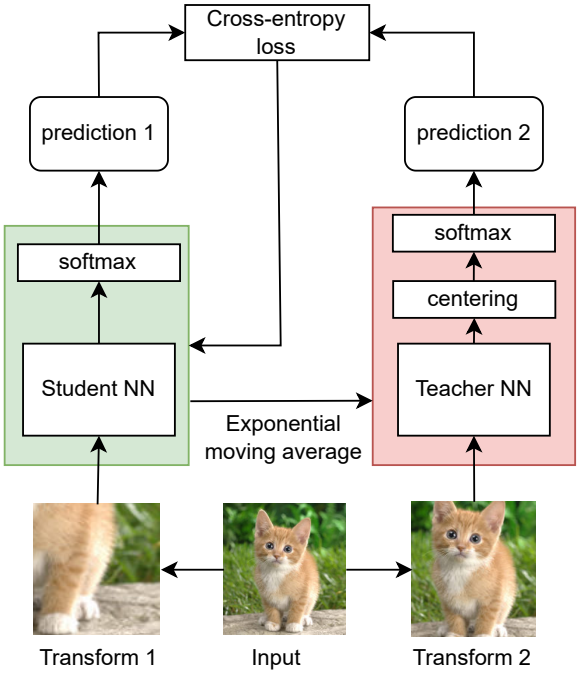


Figure 3.8: DINO self-supervised training procedure. The model encourages a dense understanding by feeding smaller local crops to the student while giving larger crops to the teacher.

After temperature softmax, cross entropy loss is computed between the student and the teacher. Backpropagation is only performed on the student network and gradient flow is prevented to the teacher network. Teacher network weights are updated by taking the exponential moving average of the student network weights. Temperature softmax and centering were found to be necessary to avoid collapse. Centering pushes the network to use more dimensions but at the same creates pressure to output uniform distribution. This pressure is balanced with temperature softmax with a low temperature that pushes dimensions further apart from each other. In addition to augmentations of the same image, the network is given augmentations of different images, and the loss is increased if similar output is given to different images. This prevents the model from collapsing to a solution where all images are given the same output.

Developing dense semantic understanding

The original scope of DINO is to do classification types of tasks where each image is assigned a single feature vector so DINO can't be used for segmentation as it is. However, already in the original publication, the authors of DINO presented visualizations of the transformer attention maps that were spatially very accurate. Interestingly the DINO architecture seems to attend to the complete objects instead of just small parts like supervised classifiers. DINO's ability to attend to complete objects is very counter-intuitive because it is trained just for classifications, not for

an accurate semantic understanding of the image. The authors propose that this behavior is enabled by the self-supervised training procedure.

Random transformations used in the self-supervised learning procedure are likely a key aspect for developing semantic understanding. The transformations include crops and distortions of the image. The teacher is only given global crops that include more than 50% of the image area, but the student is also given substantially smaller local crops. To minimize the loss the student must produce the same feature representation despite only getting a small part of the image. The only way to do this is to develop a complete understanding of the objects so that the model knows to produce the same output despite only getting a random subpart of the object. For example, if the teacher is given an image of a human and the student a crop of just a hand, a leg, or a head the student should learn to produce the same output in all of these cases that at the same time agrees with the teacher's output. The local-global crop augmentation on its own doesn't breach the gap between the classification domain and the segmentation domain. For that purpose, this training procedure must be combined with a ViT backbone that by definition has a spatial understanding of the image. It is possible to extract dense information from CNNs as well but it is more complicated compared to transformers.

Hamilton et al. [32] presented that the dense understanding of DINO doesn't end to accurate attention maps, but it also can produce similar features for similar image areas. The ability to create dense meaningful feature representation of an image self-supervised is an important finding for segmentation. It enables self-supervised segmentation by clustering similar features together. The ability to produce similar features for similar image areas is tested by taking a query point from different parts of the image and then finding the feature similarity across the image itself and with one of its nearest neighbors. The result is presented in Figure 3.9. The feature correspondences for road and roadside areas seem to be quite accurate inside the image itself and only lose little accuracy when compared to the nearest neighbor.

Figure 3.9: Feature correspondence between DINO features in winter driving scenario. Each heatmap color represents correspondence with the sample location of the same color. The plotting tool is adapted from [32].

Feature correspondences

The Python implementation for the feature extraction with pre-trained DINO is borrowed from the public source code of STEGO [32]. The pre-trained DINO backbone is used with the parameters specified in Table 3.1.

Table 3.1: Feature extraction parameters.

Backbone	ViT-small
Trainset	ImageNet[46]
Patch size	8x8 pixels
Patch feature vector size	1x384
Image input resolution	640x640
Feature space resolution	80x80 (640/8)
Output label resolution	640x640 (interpolated)

The frozen DINO image patch features are extracted after the last transformer block as illustrated in Figure 3.6. These features are unnormalized as they don't pass through the projection head that includes the softmax normalizing like the class token feature. Because our approach relies on feature correspondences like STEGO, using the same kind of processing is justified. The feature correspondences are computed using cosine similarity. Cosine similarity for unnormalized features is defined as

$$\boldsymbol{F}_{hwij} = \sum_{c} \frac{\boldsymbol{f}_{chw}\boldsymbol{f}_{cij}}{\|\boldsymbol{f}_{chw}\|\|\boldsymbol{f}_{cij}\|}, \tag{3.13}$$

where $\boldsymbol{F}_{hwij}$ is the correspondence between image patch (i, j) and (h, w). $\boldsymbol{f}_{cij}$ and $\boldsymbol{f}_{chw}$ describe the value for channel c of the image patch feature vector for image patch (i, j) and (h, w).

The authors of STEGO found that the model's performance improved when zero clamping and spatial centering is applied to the feature correspondences. In spatial centering mean over all image patch correspondences is computed and subtracted from the feature correspondence tensor:

$$\boldsymbol{F}^{SC}_{hwij} = \frac{1}{IJ}\sum_{i'j'} \boldsymbol{F}_{hwi'j'}, \tag{3.14}$$

where $\boldsymbol{F}^{SC}_{hwij}$ is the spatially centered feature correspondence and (IJ) is the number of image patches. In zero clamping all feature vector values that are negative are set to zero so that all feature vector values are non-negative. Zero clamping can be expressed as:

$$\boldsymbol{F}^{SC,Clamp}_{hwij} = max(0, \boldsymbol{F}^{SC}_{hwij}), \tag{3.15}$$

where $\boldsymbol{F}^{SC,Clamp}_{hwij}$ is the zero clamped feature vector and max takes maximum value for each element of the vector.

Similarity with GNSS mask

The feature correspondence describes the similarity between two image patches but it still needs to be defined what patches should be compared to produce useful results. A simple solution would be comparing each image area with a single query point that is sampled from a fixed location in front of the vehicle as there is usually road. However, the road is a complex area with variations in color and texture so presenting the whole road with a single feature vector sampled in front of the vehicle is not a satisfactory method. Additionally in some cases when the vehicle is turning it is possible that this query point is not part of the road at all, leading to completely incorrect results.

At this stage the GNSS mask becomes useful. Instead of just computing the correspondence between a single point in front of the vehicle and each image area, average correspondence with each image area and the whole GNSS mask is computed (Figure 3.10). The GNSS mask is likely to contain a good representation of the features present in the complete road area so any area that is part of the road should produce high average correspondence. The average GNSS mask correspondence can be defined as:

$$\boldsymbol{F}_{hw}^{SC,Clamp,GNSS} = \frac{1}{N_{patch}^{mask}} \sum_{i,j \in mask} \boldsymbol{F}_{hwij}^{SC,Clamp}, \tag{3.16}$$

where N_{patch}^{mask} is the number of image patches in the GNSS mask and the sum is only calculated over indices (i, j) that are part of the GNSS mask. This correspondence is finally normalized to range 0-1 by dividing with the maximum correspondence value:

$$\boldsymbol{F}_{hw}^{SC,Clamp,GNSS,norm} = \frac{\boldsymbol{F}_{hw}^{SC,Clamp,GNSS}}{\boldsymbol{F}_{max}^{SC,Clamp,GNSS}} \tag{3.17}$$

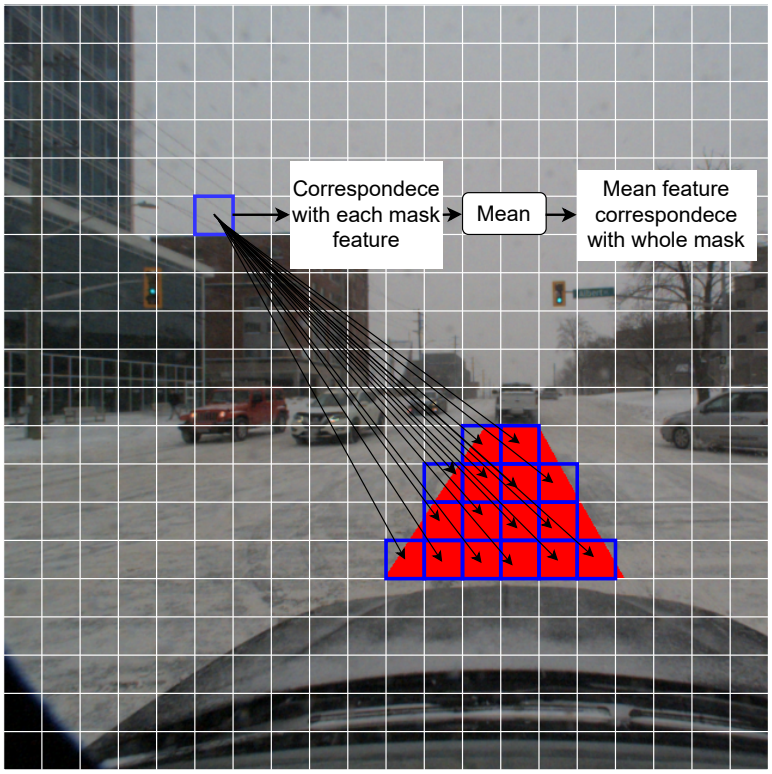


Figure 3.10: Computing average correspondence with GNSS mask. This process is repeated for each image patch.

3.6 Label evaluation

Labels are autogenerated by comparing similarity with the driven area with point level GMM, patch level GMM, and DINO self-supervised feature extractor. These labels are visualized in Figure 3.11. DINO features seem to be more robust than GMM features. Because DINO uses a visual transformer as its backbone, features can encode information about their location inside the image in addition to the color and texture type information. This helps the model filter out areas that have similar color but locate further away from the road. Based on this test, DINO features are chosen for the auto-labeling process.

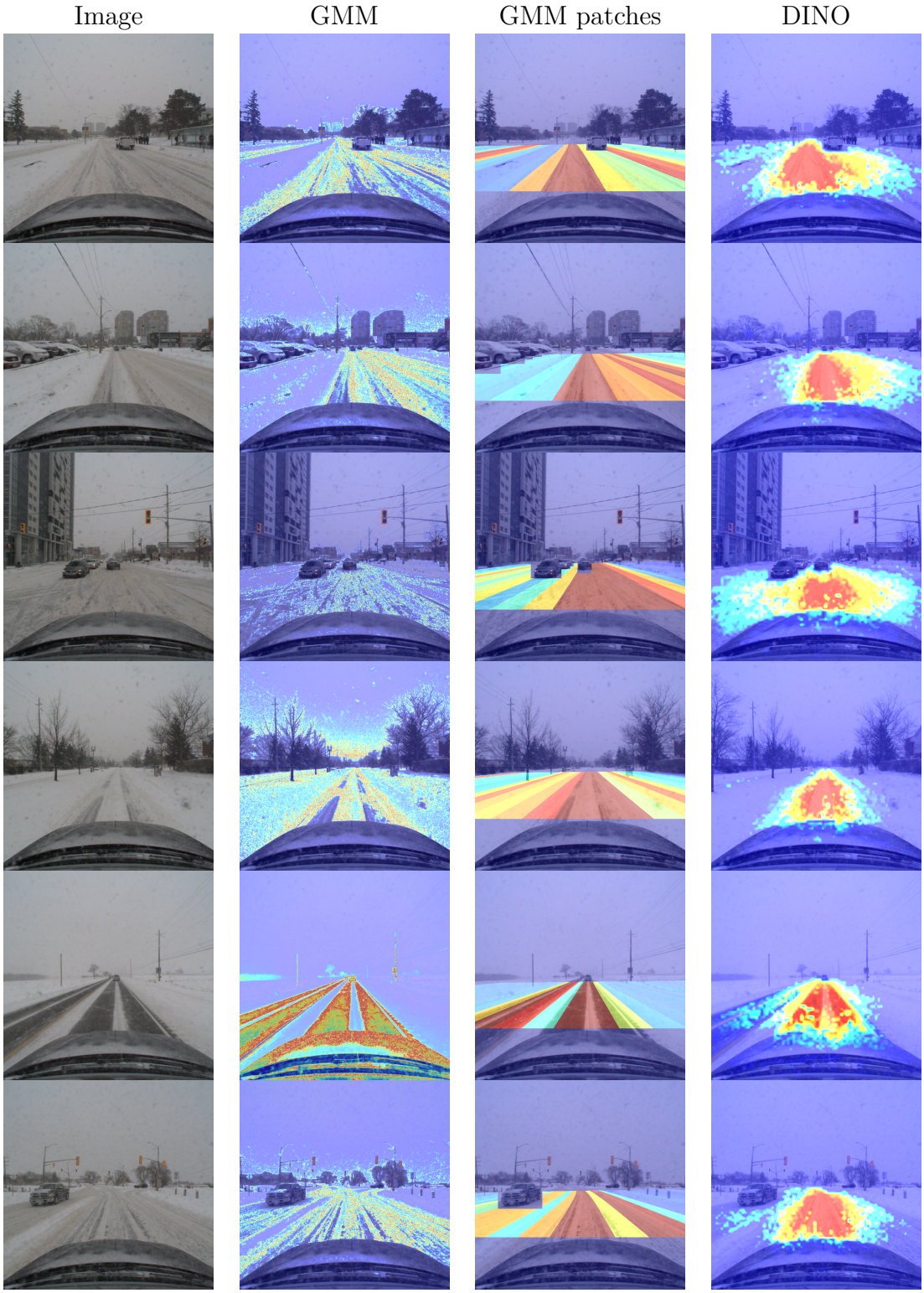


Figure 3.11: Comparison of labels autogenerated with different methods. DINO feature correspondence is a more robust similarity metric compared to the GMM-based methods.

4 Learning from PARAS auto-labels

There is a clear difference between human-annotated ground truth and PARAS labels that are automatically generated based on the average GNSS mask feature correspondence. Human-annotated ground truth is binary (road or no road) and in each frame, all road areas are segmented with little errors. PARAS labels on the other hand are continuous in the range 0-1, 1 meaning high correspondence with GNSS mask and 0 meaning low correspondence with GNSS mask and there might be a significant amount of false positive and false negative areas in each label. False negative means areas where our model has misclassified a road pixel to not road class. False positive means areas where the model has misclassified a not road pixel to road class.

The most prominent challenges here are capturing road areas with differences in color and texture or spatial location compared to the road in the GNSS mask. This follows directly from the process of how the PARAS labels are generated: only similar road areas that are present in the GNSS mask have a high confidence value. If the GNSS mask is dominated by snow-covered road then correspondence with clean road areas is low and if the GNSS mask is dominated by mostly clean road areas, snow-covered road areas will have low correspondence. Because the DINO featurizer uses a visual transformer as its backbone spatial attention is embedded into the architecture. This means that the features are not only determined by their local features, like color and texture but also their location in the image and global context. In road segmentation, this is in most cases desired behavior as roads have a distinctive spatially connected structure where areas close to other road areas are more likely to be road as well. However, in the exception cases like cross-sections and separated road sections, this leads to large false-negative areas where road areas have low label values. Example cases for poor PARAS label quality caused by road color variance and road spatial variance are illustrated in Figures 4.2 and 4.1.

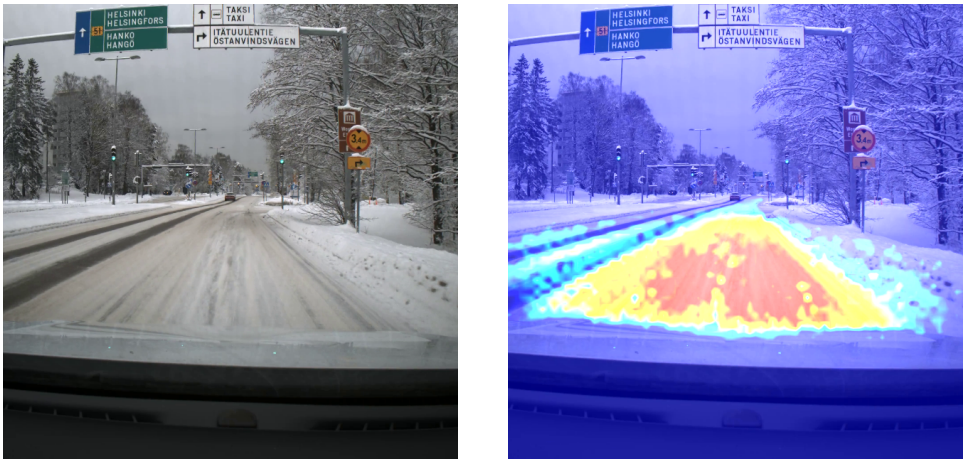


Figure 4.1: Poor detection of the adjacent lane due to different colors.

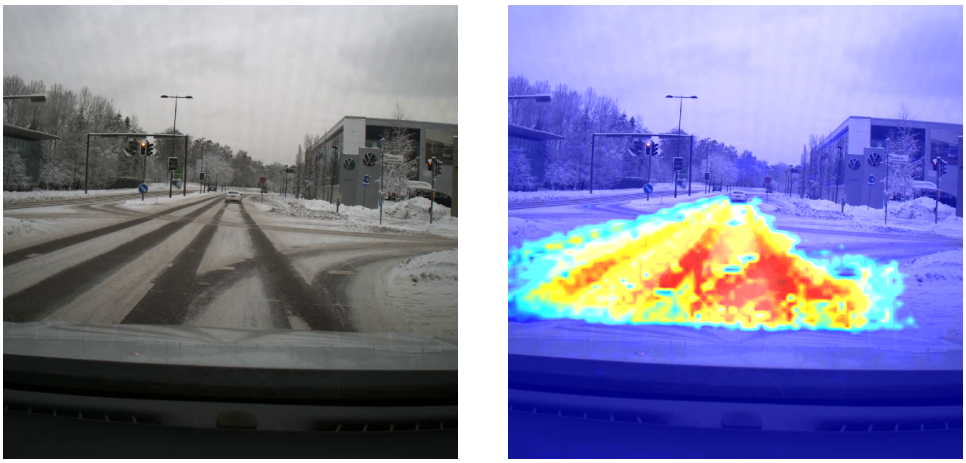


Figure 4.2: Poor detection of the road areas far away from own lane.

There are three major ways to affect the performance of the segmentation, given the quality of the labels can't be improved: data augmentation, model architecture, and loss function. In this section, all of these factors are discussed to develop a suitable configuration.

4.1 Model architecture

In this thesis, a well-established segmentation model called UNET [47] is used. UNET architecture is named based on its schematic appearance that resembles the letter U (Figure 4.3). UNET has two main components: downward pass where image size decreases and channels increase and upward pass where image size increases and channels decrease. The downward pass is a series of blocks that include a max-pooling operation and two convolution operations. In each block, the first convolution increases the number of channels and the max pooling operation downscales the image size. The downward blocks are followed by an equal amount of upward blocks that include an upscaling operation and two convolution operations. Upsampling can be done with bilinear interpolation or an extra convolutional layer. The first convolutional layer decreases the number of channels and the upscaling operation increases the image size. There is also an input block with two convolutions that maps the image from the input channel size to desired channel size for the first downward block and an output block with a single convolution that maps the output channels to the desired number of classes.

Additionally, there is a skip connection at each level connecting the downward and upward blocks. The skip connection copies the output of the downward block and concatenates it to the same level upward block skipping all blocks below that level. This skip connection enables to combine deep contextual knowledge with high-resolution spatial knowledge to produce high-resolution segmentation labels.

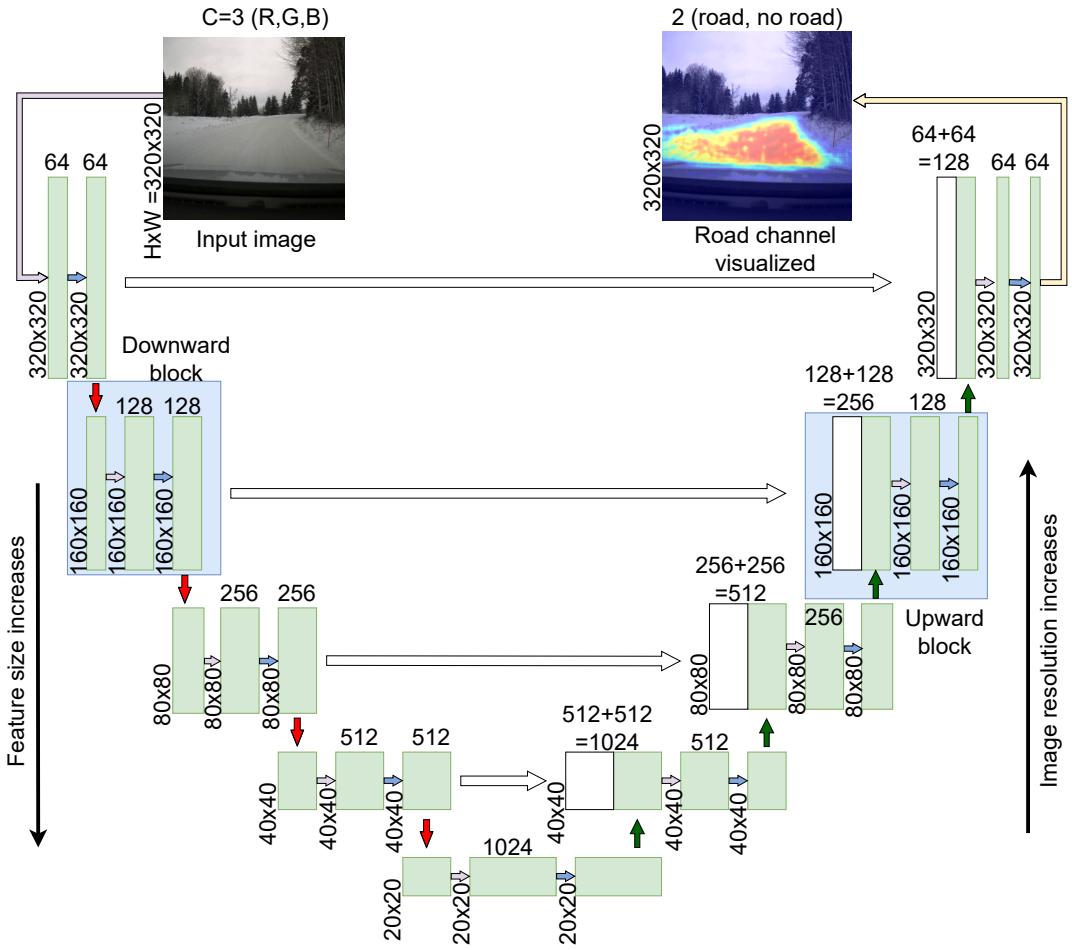


Figure 4.3: UNET schematic view in the configuration it is used here. The number of channels is at the top of the tensor and the resolution of the tensor is on the left side. Inspired by original publication [47].

In our use case, it is desired to have a model that is translationally equivariant meaning that if an object is translated in the image the output activations will remain equivalent but they are just translated the same amount. Normal convolutional layers are translationally equivariant but subsampling operations like pooling and

striding operations do not necessarily [48, 49]. Ideally, the model should have a minimal amount of subsampling to retain translational equivariance. UNET includes subsampling operations that potentially hurt the translational equivariance but it is compensated with skip connections that feed features that are not as deep but at the same time also have less sub-sampling applied to them. It is justified to assume that even though UNET is not perfectly translationally equivariant it is at least very close to that.

4.2 Data augmentation

Data augmentation is a widely used technique in deep learning where training samples are augmented before feeding them to the model to increase the generalizability of the model. Geometric and color transformations and kernel filters are the most common augmentations in road segmentation tasks. Combination of affine transformations, perspective transformations, mirroring, cropping, distortions, blur, noise, and color changes improved segmentation F-score up to 2% with KITTI dataset [50]. Abderrahim et al. [51] utilized inversion, distortion, rotations, and elastic transformations when augmenting data for road detection in a satellite image perspective. Oliveira et al. [52] used only scaling and color space transformations in Hue-Saturation-Lightness (HSV) color space. However, the improvements are quite small compared to classification tasks. Taylor et al. [53] compared the effectiveness of different data augmentations for classification and cropping was found to have the highest improvement out of geometric methods (+14%) and color jittering highest improvement out of color space transformations (+1.5%).

It seems that data augmentation yields only small improvements in segmentation tasks. Nevertheless, perspective transform, cropping, and color jitter were tested but none of them improved the performance in our case. The only augmentation used in the final tests is a random horizontal flip. The probable cause for the ineffectiveness of augmentations is the large size of the data set and model architecture that is close to translational equivariance. The main advantage of data augmentation is to prevent overfitting by artificially generating new samples that are not present in the raw data. Here the risk of overfitting the shape and location of the road label is the most critical as the labels have clear spatial bias. However, this risk is not realized as our model is close to translationally equivariant and it is resilient to spacial bias by nature.

4.3 Loss function

When choosing a loss function it is important to understand that neural networks converge towards weights that minimize the loss function expected value for the whole trainset. Although there are false positive and false negative areas in the PARAS labels the effect of these errors in training is smaller assuming the errors are close to randomly distributed across the train set. However, this is a simplification, and with any real data, the errors in the labels are not randomly distributed which is the case here also. The PARAS labels are more prone to false negatives than false

positives and they tend to fail in specific conditions.

There is a selection of well-established loss functions for different purposes that produce good learning outcomes. The common factor for all loss functions is that the loss function should decrease when the error between the prediction and ground truth decreases to provide a meaningful learning signal. The most common loss functions are Mean Squared Error (MSE) loss and Cross Entropy loss. MSE loss is best suited for regression problems where the goal is to find a relationship between input and output variables. Cross Entropy loss is best suited for classification where inputs are assigned to a set of classes. Cross Entropy loss is the most common loss function also in segmentation tasks as they can be interpreted as pixel-level classification.

Cross entropy loss is defined as

$$L_{ce} = -\sum_{batch}\sum_{pixels}\sum_{c=1}^{C} log\frac{\exp(x_c)}{\sum_{i=1}^{C}\exp(x_i)}y_c, \tag{4.1}$$

where c denotes class, x_c denotes predicted logit for class c, x_i denotes predicted logit for class i and y_c denotes ground truth probability for class c. In the numerator, the predicted logit for class c and the ground truth probability for class c are multiplied, which yields the highest values over all classes when the predicted logits and ground truth have similar distributions over all classes. The predicted logits x don't need to be normalized to the range of 0-1 as there is a normalizing term in the denominator. Before multiplication logits are divided by the sum of logit values over all classes, which yields outputs that by definition sum to 1 over all classes.

Generally, the PARAS labels segment too little proportion of the image as a road, and if just cross-entropy loss is used the model is not encouraged to expand its predictions outside the PARAS label domain meaning that the predictions will only include the most certain road areas. To accommodate this issue a balancing loss term is added that encourages the model to predict a larger proportion of the image as a road. This loss term is defined as:

$$L_b = 1 - \sum_{batch}\sum_{pixels} softmax(x_{background}) \tag{4.2}$$

The loss term is the sum of softmax normalized predictions that each pixel is road. If all pixels are predicted to be road with 100 % certainty this term has a value of 1. And if all pixels are predicted to not be road with 100 % certainty this term has a value of 0. The loss term is subtracted from 1 to keep the loss positive in all conditions to avoid collapse. Otherwise, the total loss could collapse to zero when the balancing term is equal but negative to the cross entropy loss. Another positive quality is that the balancing is done over a batch and not over a single image. Thus the loss will encourage finding the best road pixel candidates over the whole batch instead of forcing the prediction to be expanded equally in every image.

The total loss is then defined as:

$$L = L_b + L_{ce} \tag{4.3}$$

An important note is that there are no tuned parameters in this loss term, unlike other self-supervised approaches [32], making the training procedure more robust and

easier to implement. The functionality of this loss term comes from the relationship between the cross-entropy loss and the balancing loss. The model is encouraged to increase the predicted probability of a pixel if the balancing term decreases more than the cross-entropy loss increases. Formally, the balancing term increases the value of road prediction if $-\Delta L_b > \Delta L_{ce}$ holds for this change.

4.4 Full training procedure

Now the full training procedure can be constructed after suitable data augmentations (or justified lack of them), and the segmentation model and the loss function have been defined. The training starts by running the auto-labeling process to produce labels for the segmentation model. Then UNET is trained with our PARAS labels using the custom loss function. The full training process is presented in Figure 4.4.

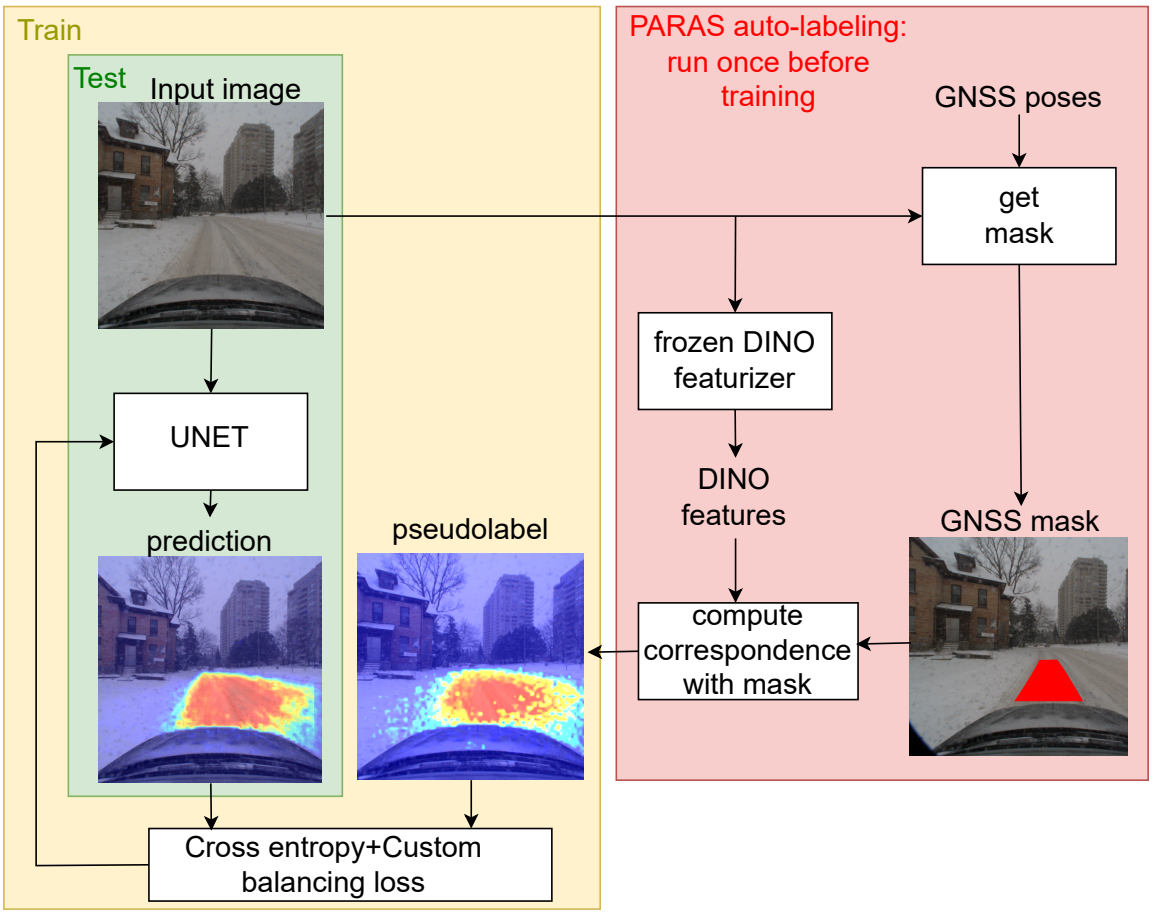


Figure 4.4: Our proposed method for self-supervised road segmentation utilizing GNSS poses and frozen DINO backbone for auto labeling training data.

5 Results and Discussion

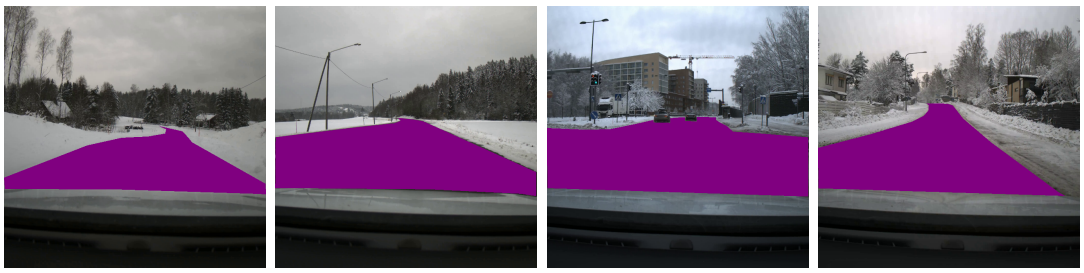
The quantitative and qualitative results of our model are presented here. A detailed description of the datasets used and benchmarked model implementations are also included. [54]

5.1 Data sets

The training set and test set used for each of the models when benchmarking are specified in Table 5.1 and a detailed description for each of the datasets is presented below.

Ours unlabeled
Combination of CADCD winter driving images and our self-collected winter driving images. Scenes include suburbs, countryside, highways, and intersections. GNSS poses are available but our model is the only one capable to utilize them.

Ours labeled
Currently, there are no published datasets that are recorded in winter conditions and include semantic segmentation labels for roads. Thus a random split of 200 images is separated from our dataset and manually labeled using the MATLAB polygon labeling tool to allow accurate benchmarking. No shared images with the unlabeled train set. Test sets of the same size are identical and can be directly compared against each other. The labels include four different scenes. The driving scenes are suburb, highway, countryside and intersection(Figure 5.1). Scene definitions enable comparison between different conditions and finding the strong and weak areas of the model.



(a) countryside (b) highway (c) intersection (d) suburb

Figure 5.1: Example of a labeled frame from each of the driving scenes used in the test set.

Cityscapes labeled
The Cityscapes dataset includes 5000 frames with fine semantic annotations and 20 000 frames with coarse semantic annotations. Cityscapes includes 30 classes in total but only road class predictions are used here. It is mostly recorded in cities in summer conditions.

Table 5.1: Dataset used for training and testing for benchmarked models.

Type	Model	Training set(N images)	Test set(N images)
Supervised	Cityscapes #3	Cityscapes labeled(25 000)	ours labeled(150)
	UNET (150)	ours labeled(150)	ours labeled(35)
	UNET (100)	ours labeled(100)	ours labeled(35)
	UNET (50)	ours labeled(50)	ours labeled(35)
	UNET (20)	ours labeled(20)	ours labeled(35)
Self-supervised	STEGO	ours unlabeled(33 000)	ours labeled(150)
	Ours	ours unlabeled(33 000)	ours labeled(150)

5.2 Implementation

OURS=PARAS+UNET+Custom Loss
Labels are created with the PARAS auto-labeling method and UNET is trained with these labels using the presented custom loss. Hyperparameters used are Adam optimizer, batch size of 10 and learning rate of 1e-4. Training is stopped when IoU with the validation set plateaus. The test score is computed from a separate test set that the model has never seen, not even in validation, to follow the standard practice of benchmarking. The validation set includes 50 images and the test set 150 images. The same test-validation split is used with Cityscapes and STEGO to allow direct comparison between these models.

STEGO
STEGO is chosen as the state-of-the-art self-supervised benchmark as it utilizes the same DINO features as our model and its performance is #1 in the COCO benchmark [55] and #2 in the Cityscapes benchmark [5]. STEGO is first tested with ViT-small backbone with both 5 and 10 clusters and with and without cropping and CRF post-processing. The cluster corresponding to the road is extracted for evaluation. Then the best-performing setup is tested again with a ViT-base backbone. The results for all of these setups are presented in Table 5.2. The CRF parameters are the same as presented in the paper: 10 iterations with parameters $(a = 4, b = 3, \theta_\alpha = 67, \theta_\beta = 3, \theta_\gamma = 1)$ and $\boldsymbol{BGR}_{mean} = (104.008, 116.669, 122.675)$ precomputed from ImageNet. The hyperparameters are the same as used with the Cityscapes data set in the paper $(\lambda_{rand} = 0.9, \lambda_{knn} = 0.58, \lambda_{self} = 1.0, b_{rand} = 0.31, b_{knn} = 0.18, b_{self} = 0.46)$. Training is stopped when the validation IoU plateaus and the model is then evaluated with the test set.

Table 5.2: STEGO parameters study. Best performance is achieved with ViT-small backbone, with 5 clusters and without cropping and CRF.

Backbone	Crop	Clusters	CRF	IoU
ViT-small		10		63.0
	✓	10		62.0
		10	✓	67.6
	✓	10	✓	64.9
		5		**73.0**
		5	✓	72.8
	✓	5		61.4
	✓	5	✓	64.8
ViT-base		5	✓	62.4
		5		61.6

The best-performing setup is ViT-small-backbone with 5 clusters without crop and CRF. The same setup with ViT-base-backbone achieves lower performance. Surprisingly the parameters are almost the opposite compared to the optimal parameters presented in the paper, where cropping, CRF, and ViT-base-backbone all increased the performance over the baseline. However, the main advantage was increased segmentation performance of fine details, which is not needed here as the road usually covers quite large areas. Our parameter study also underlines the importance of choosing the correct parameters for each use case. Variation between different parameter combinations is quite large here.

Cityscapes #3

The Cityscapes supervised #3 model [56] is chosen to represent the current state of the art of supervised traffic scene segmentation as it is easy to implement for images of arbitrary size and its mIoU for all classes is only 0.7% lower than the #1 model [19] and the IoU for road class is 0.1 % higher than the #1 model. For reference, it achieves 85.4 mIoU for all classes and 99.0 IoU for the road class. The model runs inference on multiple scales and then combines a final prediction from them. The model backbone is an HRNet with OCR-block.

Pre-trained weights are downloaded for the #3 model that is trained with the Cityscapes dataset. No modifications are done for the weights or the architecture. Predictions include all Cityscapes classes but only the road class is extracted for evaluation with the test set. The validation set is not used here for stopping the training as the weights are pre-trained.

UNET

The same UNET model architecture and hyperparameters are used here as when learning from the PARAS labels but cross-entropy loss is used instead of the custom loss. The idea is to compare the performance between manually created labels and PARAS labels. UNET is trained with 150,100,50 and 20 manually labeled images. The validation set is the same 15 images and the test set is the same 35 images for

all train sizes to enable direct comparison between different train set sizes. As the supervised UNET models have smaller test sets they can't be directly compared to the other models but they still give a good indication of the performance that supervised methods achieve.

5.3 Quantative results

Our model achieves 1.8% higher IoU than the self-supervised benchmark STEGO and 41.6 % higher IoU than the supervised benchmark trained with Cityscapes summer data (Table 5.3). This means that our model has the highest IoU out of the methods that don't have access to manually labeled winter driving data.

Table 5.3: Benchmarking against supervised and self-supervised models.

Type	Model	Suburb	Highway	Country	Intersection	total
Supervised	Cityscapes #3	32.3	36.0	29.3	36.7	33.2
	UNET (150)	**87.2**	**89.6**	72.4	**88.4**	**83.9**
	UNET (100)	87.2	89.4	**73.2**	85.1	83.6
	UNET (50)	83.5	86.0	69.2	81.9	79.9
	UNET (20)	79.3	78.0	59.8	71.7	72.4
Self-supervised	STEGO	73.6	**75.4**	68.3	**73.4**	73.0
	Ours	**75.1**	75.1	**75.3**	72.8	**74.8**

However, when trained with the manually labeled winter driving data, supervised models achieve higher IoU, depending on the size of the trainset. The supervised model trained with 20 images has a lower IoU than our method, but supervised models trained with 50 or more images have a higher IoU than our model. It should be noted that even the highest-performing supervised model that is trained with 150 labeled images only achieves an IoU of 83.9 %. For reference, the best-performing supervised segmentation models in the Cityscapes benchmark achieve 99.0 % IoU for road class with a dataset of 25 000 images and the best-performing supervised model trained in winter conditions achieves 95.6 % IoU for road class with a dataset of 1200 images. There is a clear correlation that a larger dataset is required for better performance.

The current trend in deep learning is to continuously increase the size of the model to achieve ever-decreasing improvements over the previous state of the art. In this study, a simple model is chosen on purpose to decrease the risk of overfitting and lower the required computation. The number of parameters for benchmarked models are presented in Figure 5.2. Supervised UNETs are only presented with train set sizes of 20 and 150 for clarity.

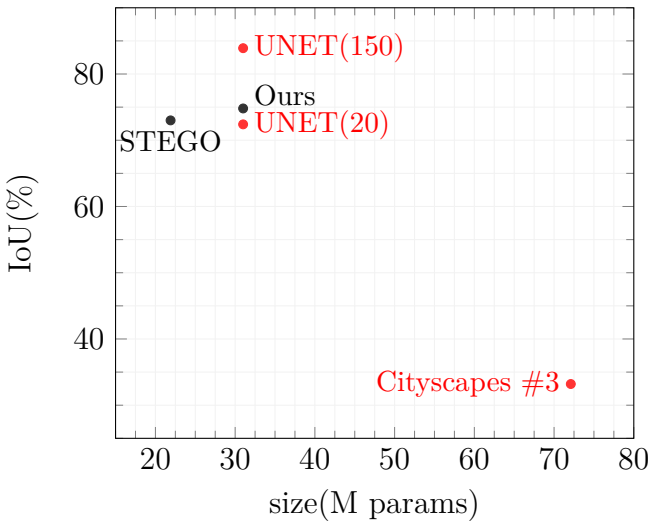


Figure 5.2: Size of the models and achieved IoU. Supervised models are marked with red color and self-supervised with black color.

Current self-supervised segmentation models utilize frozen self-supervised featurizers as their backbone and these kinds of backbones are generally quite large. With that kind of architecture, there is no way to make the model any smaller than the backbone. Our approach is to run the featurizer only once through the dataset to autogenerate labels. In a sense, the knowledge of the large backbone is distilled to these labels, and then any model can be trained with these labels. In this study, UNET is used, but any other model could have been chosen. It is likely that a more advanced segmentation network could achieve higher accuracy than presented here with UNET, but before all this study is a proof of concept for this kind of training method.

5.4 Qualitative results

Qualitative evaluation is conducted by presenting predictions from different methods in each of the driving scenes to allow direct visual comparison. Results are presented in Figure 5.3. More visual results for specifically our model are presented in Figure 5.4. Based on visual evaluation our model can't segment areas that are far away or disconnected from the road where the car currently is driving as accurately as the supervised methods but otherwise, the performance is quite similar. As presented in the quantitative results, our model has very stable performance across scenes where other models have higher variation. This is prominent in the visual results as well.

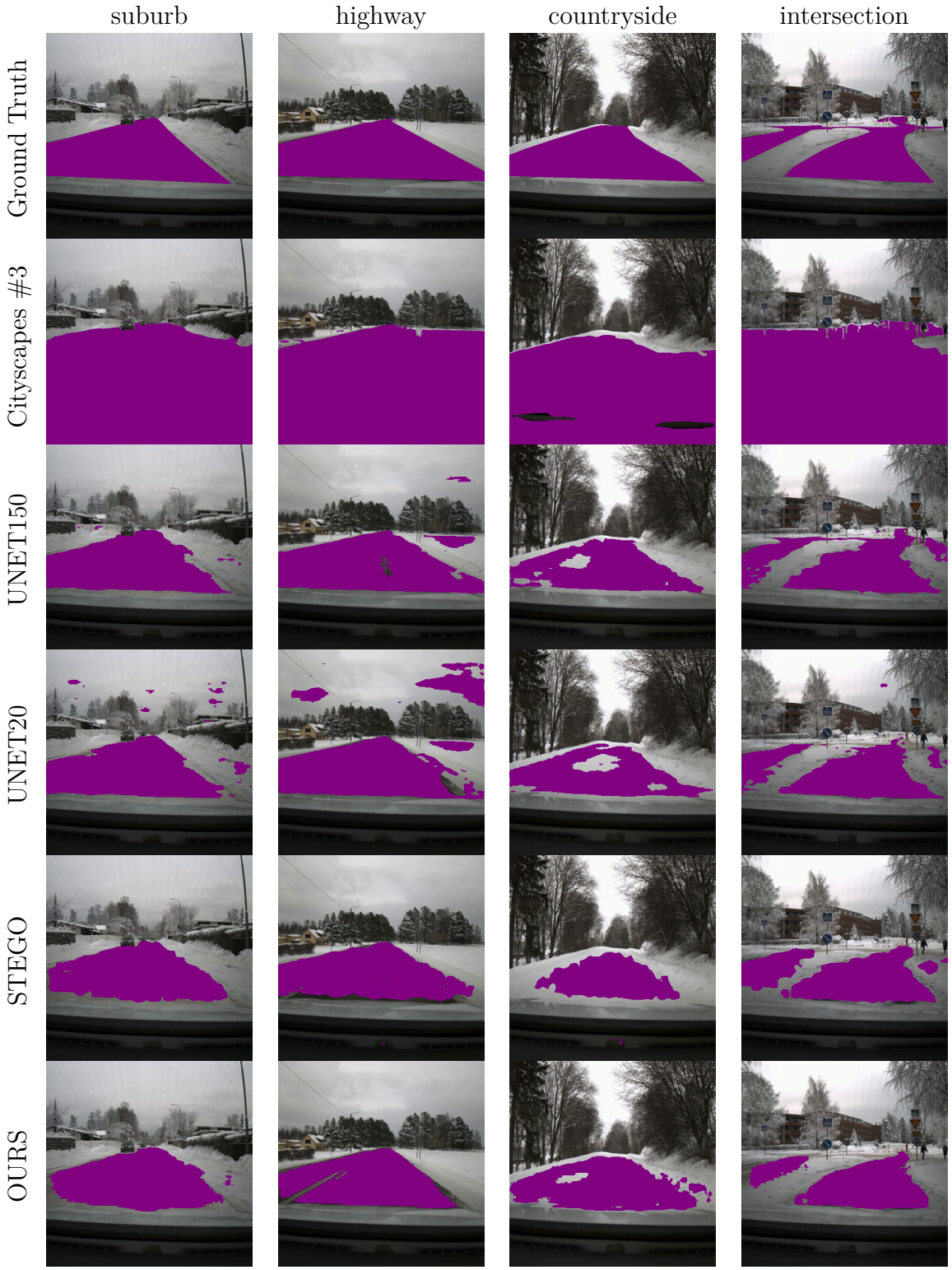


Figure 5.3: Qualitative results against benchmarked models. Models are defined in the horizontal axis and scenes in the vertical axis

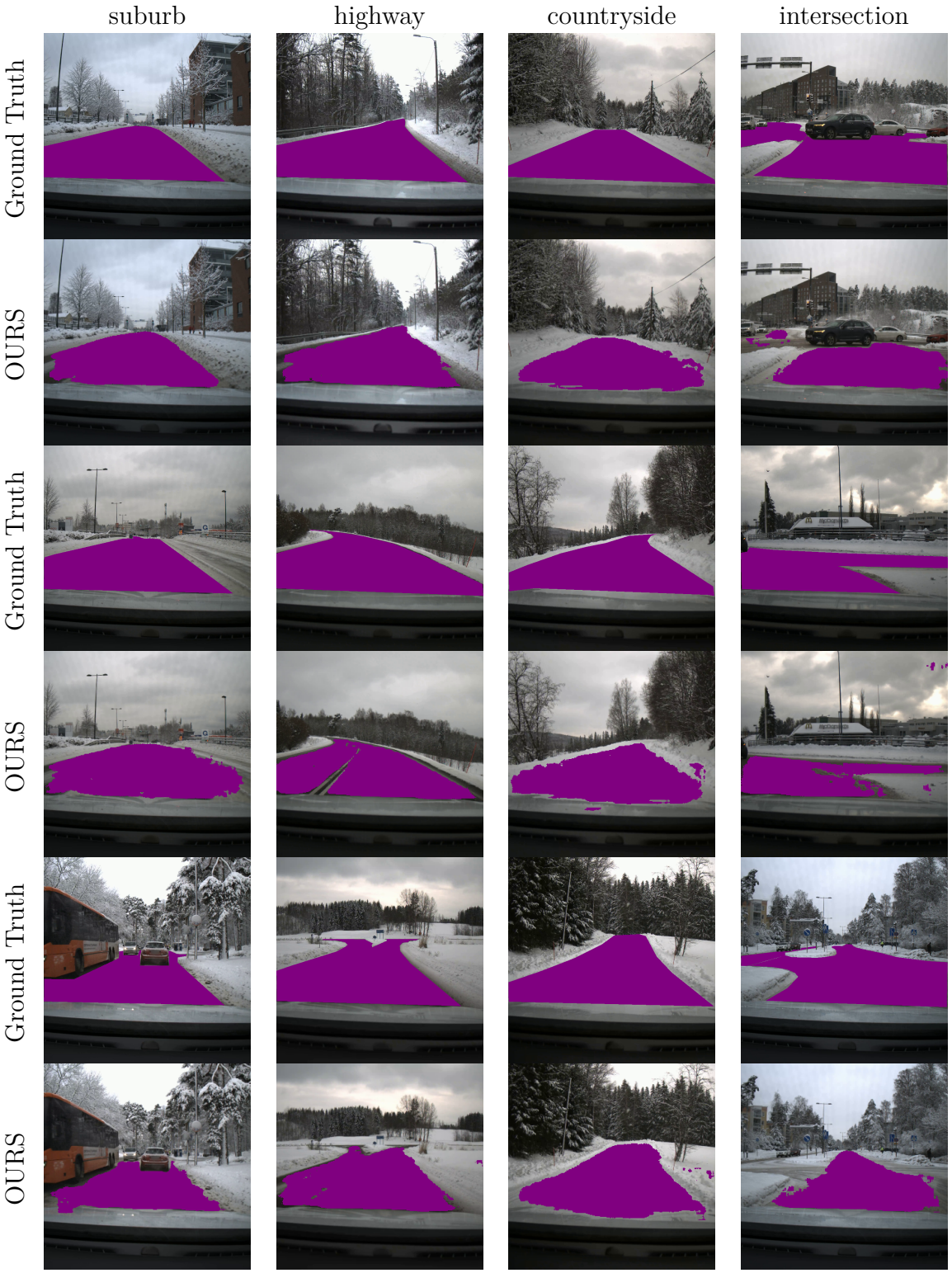


Figure 5.4: More qualitative results for our model.

The performance of our model can be further evaluated by visualizing the distribution of the false negative and false positive areas in different driving scenes (Figure 5.5). The common factor for all cases is that the road edge areas seem to be hard to classify correctly. Most of both false positives and false negatives are located close to the road edge area. Also, false negatives seem to be more common than false positives, which in a sense is desired behavior as false positives can lead to dangerous situations in autonomous driving. On the other hand in a pure benchmarking context, it indicates that the model seems to be unbalanced and it should be encouraged to expand the predictions even more.

There are also distinguishable differences for each scene. In suburb scenes false positives are mostly on the road edge areas, which can be partially caused by cases where sidewalks are classified as road. However, false negatives are more common and mostly located on the sides close to the vehicle. Interestingly there seem to be more false positives on the left side and more false positives on the right side. The hypothesis for this behavior is that data has been collected mostly while driving on the right side of the road, which means that the GNSS mask includes more samples from the right lane and the labels have been generated based on the similarity with the GNSS mask. Highway scenes include very few false positives. The false negatives are mostly located on the road edge areas where the texture of the highway usually changes from asphalt to snow-covered and thus they are difficult to detect. The countryside scene is the only scene where false positives and false negatives are distributed quite evenly. False positives are mostly located on the road edges like in other scenes but they are distributed to a larger area. This is mostly caused by the curvier road profile and unclear road boundary areas. False negatives are mostly located close to the horizon, where the road is hard to separate from the roadside areas because there is very little contrast between them. In intersection scenes false negatives are much more common than false positives. False negatives are located on the sides of the image and close to the horizon. These areas are hard to capture by the auto-labeling process as they are usually far away from the GNSS mask.

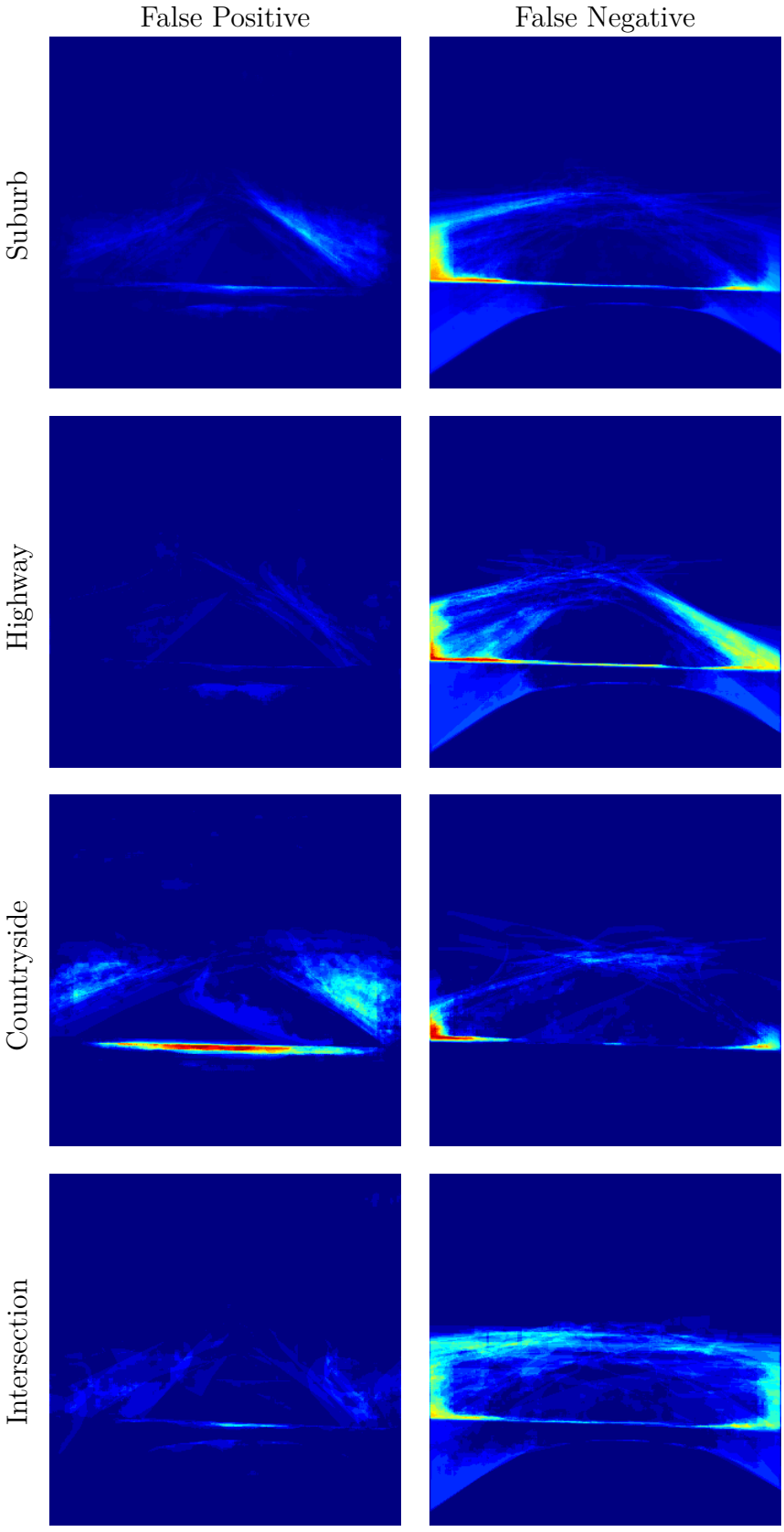


Figure 5.5: False negative and false positive distribution for each scene. Red areas have a high number of false detections and blue areas have a low number of false detections.

5.5 Future improvements

None of the benchmarked models achieve the required performance for real autonomous driving applications so the important question is if our model has the highest potential out of the tested methods to justify further development towards a more robust version.

Improving a supervised model is very straightforward. First, a more sophisticated segmentation model can be chosen just like with our model. Secondly, more data can be labeled. Based on previous work labeling 1000 images and choosing a suitable segmentation model could yield over 95 % IoU. However, this is only in the exact conditions where data has been collected. Even small changes from the current domain most likely cause the performance of the model to drop and more data should be labeled again from the new domain to retain the performance.

The important factor here is the number of different conditions the labeling has to be repeated in. In Finland and any other country where there are four seasons, labeling just summer and winter data most likely is not enough. There are multiple unique driving conditions present in each of the seasons. And on top of that general adverse conditions like fog and rain can be present in most of the seasons. In that context, manual labeling becomes a very challenging and resource-heavy way to teach the model. On the other hand, self-supervised models can be taught with no human labeling effort, making them more scalable for different driving conditions. Any improvements in the self-supervised models automatically apply to any driving conditions. In that context, self-supervised models have more future potential than supervised methods and it is justified to keep developing them.

Currently, best-performing self-supervised feature extractor networks use training procedures where different data augmentations of the same image are pushed to have similar features, and augmentations of different images are pushed to have different features. However, there is a gap between the task of creating semantically meaningful dense features and producing pixel-level semantic segmentation. This gap is difficult to close, and currently, the most common solutions are clustering and training a linear projection.

In clustering, features are grouped together based on their feature similarity but this method comes with many problems. The number of clusters must be decided beforehand and the segmentation class for each pixel is assigned just based on the feature similarity. This may lead to behavior, where for example the wheels of all vehicles are clustered together and the remaining parts of the vehicles are clustered together, instead of clustering the same types of vehicles together. This kind of behavior was observed also when training STEGO with winter driving data. The model wasn't able to cluster different kinds of road segments together because the feature representation of the road was so different across different images. It is obvious that clustering is not a satisfactory method to assign segmentation labels based on the dense features, but at the moment there are no better ways to do it if no labels are available.

Another common way to approach the problem is to train a one-layer linear pixel-level classifier that learns to connect features to a segmentation class. This approach

has significantly better accuracy but labeled data is required in order to train the linear classifier, so these kinds of models are not considered fully self-supervised. However, this is a good way to benchmark the quality of the self-supervised features because the linear classifier has to classify each pixel only based on the feature vector without looking at any other information. The best-performing self-supervised dense feature extractors combined with linear supervised feature classifiers are already getting close to the traditional completely supervised state-of-the-art in mIoU benchmarks. The usefulness of such a setup might seem questionable as labeled data is required but the performance is not quite on bar with the supervised state of the art. However, there are some undeniable advantages to this kind of approach. First, the model architecture is very simple as only the pre-trained feature extractor and one linear layer are required. Second, training this kind of setup for a variety of custom tasks is extremely fast and requires less data as only the linear layer needs to be trained.

The contribution of this thesis is to present a better way to utilize these self-supervised features for road segmentation without having access to labels. Essentially the GNSS information enables to segment the road more accurately than using simple clustering while no manual labels are required like with the linear projection network. The different ways to utilize self-supervised features are presented in Figure 5.6.

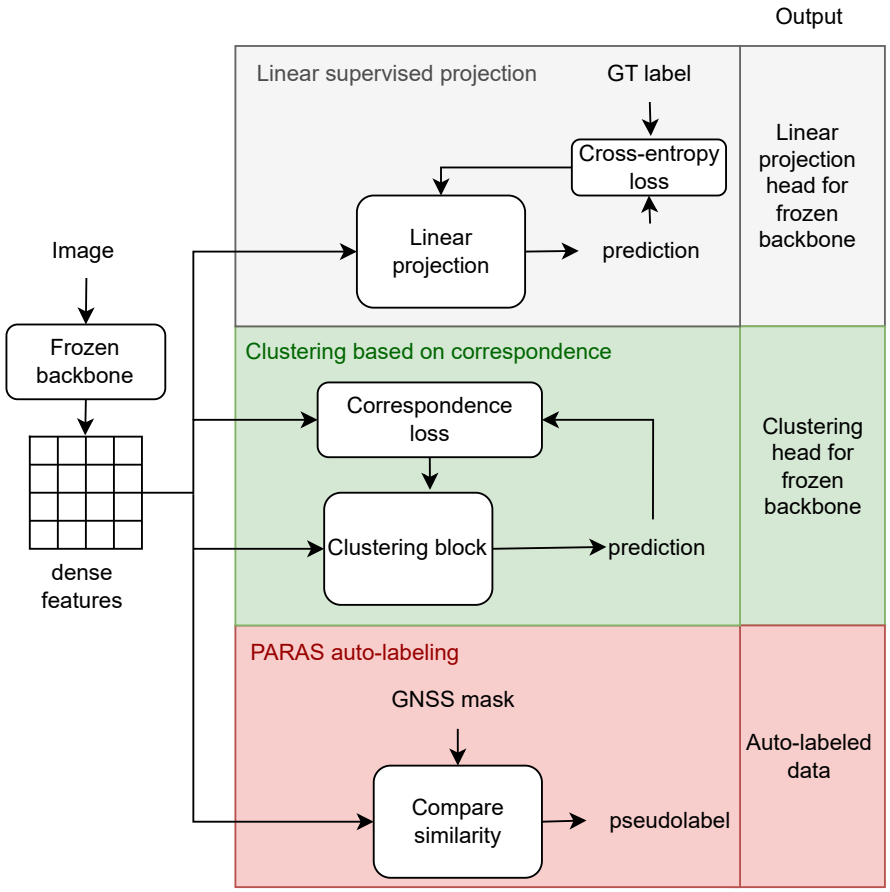


Figure 5.6: Using self-supervised features with clustering, linear projection layer, and GNSS-based auto-labeling (PARAS).

There are two main development areas for our model: improving PARAS auto-labels and improving the segmentation model trained with the PARAS auto-labels. The auto-labeling process can be improved by selecting a backbone that has the ability to detect road areas more accurately when compared to the GNSS mask. One potential backbone choice would be the updated version of DINO, DINOv2. Additionally, the comparison to the GNSS mask can be done in a more advanced way. Currently, the comparison is done simply by comparing correspondence with a GNSS mask inside a single image without knowledge of road representation in any other image, which leads to underestimated road labels. Alternatively, a bag of features kind of approach could be implemented, where a larger database of known road features is constructed from multiple images and then the correspondence of each image patch is computed against this bag of features. The challenge here is that the computational complexity increases when each image patch is compared against a larger database of features.

The second development area is choosing a more advanced segmentation model. UNET is a quite simple model and it is lacking behind the current state-of-art in segmentation. It was chosen for this thesis based on its low parameter count and long history. There are two possible directions to go with the segmentation model. The first option is to use the same frozen backbone as in the auto-labeling process and simply train a projection head on top of the backbone. Another option is to choose a more sophisticated CNN architecture than the traditional UNET. However, it is important to remember that neural networks learn from data so most likely improvements in the auto-labeling process have a higher impact than changing the segmentation model.

It is important to understand that all of these self-supervised methods rely on the pre-trained feature extractor which brings its own challenges. The self-supervised training of the dense feature extractor enables the usage of huge datasets which leads to a model that can generalize to almost any domain. This pre-trained general-purpose model can then be used in a variety of downstream tasks without modifications. In practice, this means that the self-supervised dense feature extractor is once trained with an extensive amount of computing resources and then the pre-trained model is distributed for further use. For supervised models, the limiting factor is mostly labeled data, for current self-supervised models the limiting factor is mostly computing resources.

This leads to a situation where all users have to use the same general-purpose pre-trained model instead of training a model with different data for different use cases. This can lead to suboptimal performance in some conditions. While the train sets are very large with millions and millions of images they still can't include all kinds of scenes. In this thesis, the model was found to generalize quite well to winter driving conditions but even more extreme conditions can be present if the model is used for example in the forest or mining industry.

6 Conclusion

Road segmentation is a crucial task for autonomous mobility. Currently best performing road segmentation models rely on manually labeled data, which makes adaption to new domains, like winter driving, difficult. Self-supervised feature learning offers a way to learn image representation, without any labels. The quality of these features is very promising but methods to utilize these features in downstream tasks are still in the early stages. In this thesis, a road segmentation model utilizing these self-supervised features is presented. The main contributions are the PARAS auto-labeling process that creates road labels by comparing feature similarity to the driven area extracted based on GNSS and the custom loss function that enables better learning from the auto-labels. Training and testing are conducted in winter conditions to underline the suitability to any domain.

Our proposed method has promising performance and improved the current state of the art in self-supervised road segmentation in the winter driving domain (74.8 IoU vs 73.0 IoU). While supervised models trained in the winter domain outperformed our method, supervised methods trained in the summer domain performed very poorly when tested in the winter domain, demonstrating the challenges of supervised models. While state-of-the-art results are presented the full potential of our method is not explored here. PARAS auto-labeling method can be improved by using a more advanced method for comparing feature similarity with the driven area. A bag of features type approach would allow cumulating features from multiple frames when doing the comparison and could reduce the false negatives in the auto-labeling process. Additionally, more advanced feature extractors have been published that would likely increase the accuracy of the auto-labeling process. On the other hand, the UNET model trained with the PARAS labels, is a very basic segmentation model and more advanced models are available. Upgrading the segmentation model could yield improvements in performance. However, auto-labels are imperfect and models that perform well when trained on manually labeled data might not yield expected results when trained with auto-labels.

Most importantly, this thesis presents a solution to the scalability problem of road segmentation. Robust road segmentation models need huge amounts of data, rendering manual labeling an infeasible method of training application-ready solutions. Inevitably, the focus of road segmentation research will shift from improving supervised methods to self-supervised learning and auto-labeling, and foremost, the results presented in this thesis should encourage further research in this field.

References

[1] R. Hussain, J. Lee, and S. Zeadally, "Autonomous cars: Social and economic implications," *IT professional*, vol. 20, no. 6, pp. 70–77, 2018.

[2] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[3] P. Feng and Z. Tang, "A survey of visual neural networks: current trends, challenges and opportunities," *Multimedia Systems*, pp. 1–32, 2022.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[6] C. K. Strong, Z. Ye, and X. Shi, "Safety effects of winter weather: the state of knowledge and remaining challenges," *Transport reviews*, vol. 30, no. 6, pp. 677–699, 2010.

[7] J. Mayr, C. Unger, and F. Tombari, "Self-supervised learning of the drivable area for autonomous vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 362–369.

[8] H. Wang, Y. Sun, and M. Liu, "Self-supervised drivable area and road anomaly segmentation using rgb-d data for robotic wheelchairs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4386–4393, 2019.

[9] A. Ganeshan, A. Vallet, Y. Kudo, S.-i. Maeda, T. Kerola, R. Ambrus, D. Park, and A. Gaidon, "Warp-refine propagation: Semi-supervised auto-labeling via cycle-consistency," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 499–15 509.

[10] V. Muşat, I. Fursa, P. Newman, F. Cuzzolin, and A. Bradley, "Multi-weather city: Adverse weather stacking for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2906–2915.

[11] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *Advances in neural information processing systems*, vol. 30, 2017.

[12] T. Liu, Z. Chen, Y. Yang, Z. Wu, and H. Li, "Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer," in *2020 IEEE intelligent vehicles symposium (IV)*. IEEE, 2020, pp. 1394–1399.

[13] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 309–318, 2004.

[14] E. J. Almazan, Y. Qian, and J. H. Elder, "Road segmentation for classification of road weather conditions," in *European Conference on Computer Vision*. Springer, 2016, pp. 96–108.

[15] D. Lieb, A. Lookingbill, and S. Thrun, "Adaptive road following using self-supervised learning and reverse optical flow." in *Robotics: science and systems*, vol. 1, 2005, pp. 273–280.

[16] U. Ozgunalp, X. Ai, and N. Dahnoun, "Stereo vision-based road estimation assisted by efficient planar patch calculation," *Signal, Image and Video Processing*, vol. 10, pp. 1127–1134, 2016.

[17] Z. Liu, S. Yu, X. Wang, and N. Zheng, "Detecting drivable area for self-driving cars: An unsupervised approach," *arXiv preprint arXiv:1705.00451*, 2017.

[18] J. Li, Q. Hu, and M. Ai, "Unsupervised road extraction via a gaussian mixture model with object-based features," *International Journal of Remote Sensing*, vol. 39, no. 8, pp. 2421–2440, 2018.

[19] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, X. Wang, and Y. Qiao, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," 2023.

[20] S. Borse, Y. Wang, Y. Zhang, and F. Porikli, "Inverseform: A loss function for structured boundary-aware segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5901–5911.

[21] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[22] S. Vachmanus, A. A. Ravankar, T. Emaru, and Y. Kobayashi, "Semantic segmentation for road surface detection in snowy environment," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2020, pp. 1381–1386.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] S. Yadav, S. Patra, C. Arora, and S. Banerjee, "Deep cnn with color lines model for unmarked road segmentation," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 585–589.

[25] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez, "Road scene segmentation from a single image," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII 12*. Springer, 2012, pp. 376–389.

[26] D. Kothandaraman, R. Chandra, and D. Manocha, "Ss-sfda: Self-supervised source-free domain adaptation for road segmentation in hazardous environments," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3049–3059.

[27] J. Choi, T. Kim, and C. Kim, "Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6830–6840.

[28] A. Laddha, M. K. Kocamaz, L. E. Navarro-Serment, and M. Hebert, "Map-supervised road detection," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016, pp. 118–123.

[29] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[30] Y. Wang, J. Zhang, M. Kan, S. Shan, and X. Chen, "Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 275–12 284.

[31] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[32] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman, "Unsupervised semantic segmentation by distilling feature correspondences," *arXiv preprint arXiv:2203.08414*, 2022.

[33] R. Karlsson, T. Hayashi, K. Fujii, A. Carballo, K. Ohtani, and K. Takeda, "Vice: Improving dense representation learning by superpixelization and contrasting cluster assignment," *arXiv e-prints*, pp. arXiv–2111, 2021.

[34] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023.

[35] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong, "ibot: Image bert pre-training with online tokenizer," *arXiv preprint arXiv:2111.07832*, 2021.

[36] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2021.

[37] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, "Canadian adverse driving conditions dataset," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 681–690, 2021.

[38] G. Jocher, L. Changyu, A. Hogan, L. Y. , changyu98, P. Rai, and T. Sullivan, "ultralytics/yolov5: Initial release," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3908560

[39] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[40] P. S. marquis de Laplace, *Théorie analytique des probabilités*. Courcier, 1820, vol. 7.

[41] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[44] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[45] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 076–10 085.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[47] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[48] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *arXiv preprint arXiv:1805.12177*, 2018.

[49] R. Zhang, "Making convolutional networks shift-invariant again," in *International conference on machine learning*. PMLR, 2019, pp. 7324–7334.

[50] J. Munoz-Bulnes, C. Fernandez, I. Parra, D. Fernández-Llorca, and M. A. Sotelo, "Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 366–371.

[51] N. Y. Q. Abderrahim, S. Abderrahim, and A. Rida, "Road segmentation using u-net architecture," in *2020 IEEE International conference of Moroccan Geomatics (Morgeo)*. IEEE, 2020, pp. 1–4.

[52] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4885–4891.

[53] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2018, pp. 1542–1547.

[54] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[55] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[56] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *arXiv preprint arXiv:2005.10821*, 2020.