Road condition classification from CCTV images using machine learning

Master's Thesis in Engineering Mathematics and Computational Science

Ludvig Askbom

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Road condition classification from CCTV images using machine learning

Ludvig Askbom

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2023

Road condition classification from CCTV images using machine learning
Ludvig Askbom

Supervisor: Daniel Persson, Department of Mathematical Sciences
Examiner: Daniel Persson, Department of Mathematical Sciences

Supervisors at Klimator AB: Gustaf Gulliksson and Jesper Landmér Pedersen

Master's Thesis 2023
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Road condition classification from CCTV images using machine learning

Ludvig Askbom
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Understanding and categorizing road conditions is crucial for driver safety and road maintenance. This research explores practical approaches to classify road conditions using images from CCTV stations. Two classification challenges are addressed: distinguishing between snowy and non-snowy conditions and between snowy, wet, and dry conditions.

The thesis evaluates various machine learning methods for road condition classification on multiple CCTV stations, including established and novel approaches. Established methods involve feature extraction through texture analysis and fine-tuning convolutional neural networks and vision transformers. Novel contributions include training an image segmentation model for road segmentation and utilizing persistent homology for feature extraction. Notably, this thesis sets itself apart by separating data into training and test sets based on CCTV stations. This is important to evaluate the methods' and models' abilities to generalize to new CCTV stations.

The best-performing model, a fine-tuned vision transformer, achieved accuracies of 87.9% and 75.3% for classifying snow/no snow and snow/wet/dry, respectively. These results underscore the complexity of the classification problem and highlight the effectiveness of deep learning models for large-scale road condition classification based on images.

Keywords: road condition, classification, machine learning, deep learning, feature extraction, vision transformer

Acknowledgements

I want to thank my family; without your love and support, I would not have made it through the toughest days. I want to thank my friends; without your friendship, I would have dropped out of Chalmers a long time ago. I want to thank my supervisors for helping make this thesis all that it could be. And finally, I want to thank Klimator AB for giving me the opportunity to investigate an interesting and meaningful problem.

Ludvig Askbom, Gothenburg, November 2023

Contents

1
Introduction

It is estimated that 21% of all road accidents are weather-related, and in the U.S., approximately 5,000 people are killed, and 418,000 people are injured each year in weather-related accidents [1]. Knowledge about road conditions and the ability to predict the weather helps the maintenance of roads and improves safety.

A. Kuehnle and W. Burghout [2] produced the first study that used machine learning to classify road conditions based on images alone. By segmenting the road from the rest of the image and extracting statistical features based on filters and color distributions, they achieved an accuracy of $40 - 50\%$ in classifying dry, wet, snowy, ice, and snowy with tracks. P. Jonsson [3] used images and weather data to classify the same conditions. Using image analysis techniques such as run length, edge detection, and the difference in grey level to nearby pixels, P. Jonsson showed that image features are as valuable as weather data in classifying road conditions. P. Jonsson achieved an accuracy of $91 - 100\%$ on all classes except snowy with tracks where 74% accuracy was achieved. Z. Sun and K. Jia [4] used another texture analysis technique, co-occurrence matrices, to extract features from road images. They classified the road condition as either dry, mild snow coverage, or heavy snow coverage with an accuracy of 96.3%.

Both P. Jonsson [3] and A. Kuehnle and W. Burghout[2] used relatively small, fully connected neural networks in their studies. Studies using neural network architectures designed to classify images have also achieved good results. K. Ozcan et al. [5] used a convolutional neural network (CNN) to classify road images taken from CCTV and mobile-based cameras to classify dry, wet, and snowy road conditions. They achieved accuracies of 98.57% and 77.32% for CCTV and mobile images, respectively. In 2022, A. Abdelraouf et al. [6] presented a vision transformer with extra spatial information to predict wet conditions with a $\sim 98\%$ recall and precision.

In recent years, persistent homology has emerged as a promising tool for extracting valuable topological features from high-dimensional data [7]. It has been demonstrated that persistent homology can generate features beneficial for various machine-learning tasks, including image analysis and computer vision. Persistent homology treats images as discrete surfaces that are cross-sectioned at different levels. At each level, holes of different dimensions are recorded, and by storing the level at which holes appear and merge with each other, it is possible to capture the topology and geometry of the surface.

The distinction between snowy and non-snowy road conditions can often be attributed to variations in the grey level of the road, which is reflected in the geometry and topology of the image. Therefore, persistent homology may generate features

capable of distinguishing between different road condition classes.

This thesis explores and evaluates various methods for two classifications. The first classification task is to distinguish between roads with snow on them and roads with no snow. This task will be referred to as snow/no snow. The second classification task is distinguishing between snowy, wet, and dry roads. This task will be referred to as snow/wet/dry.

Several methods are evaluated for classifying road conditions, some novel and some previously tested. The new methods explored comprise the training of an image segmentation model for segmenting roads and the use of persistent homology for feature extraction. The previously tested methods consist of using texture analysis for feature extraction and fine-tuning a convolutional neural network and a vision transformer. This thesis also sets itself apart from previous work through the evaluation technique of models. The aim is to find a method and model capable of processing images from a large number of CCTV stations. Therefore, the images will be divided into training and test data based on the CCTV station. All images from a CCTV station will be present in either the training data or the test data. This helps evaluate the models' ability to generalize to new CCTV stations.

The methodology is divided into two parts: the conventional method and the deep learning method. The conventional method comprises segmenting roads and extracting features using texture analysis and persistent homology. The deep learning method consists of fine-tuning a convolutional neural network and a vision transformer.

Each method and model will be judged on several criteria. Firstly, the mean accuracy achieved on images from unseen stations. Secondly, the distributions of the predicted probabilities. In practical implementations, predictions with a low predicted probability are useless. The third criterion is the practicality of implementation. The aim is to find a model able to classify road conditions accurately for varying types of road layouts.

The thesis begins with an introduction to machine learning and its application to classification tasks. Some commonly used conventional classifiers are presented. This is followed by an introduction to deep learning and fully connected neural networks. Convolutional neural networks, vision transformers, and image segmentation are then presented. This is followed by methods for evaluating machine learning models and ranking the importance of features.

Texture analysis and grey level run-length matrices are introduced along with the statistical features that can be extracted. This is followed by a short introduction to persistent homology with the aim of providing a basic and intuitive understanding of how features are extracted.

The implementation details of the conventional and deep learning methods are presented. This section comprises descriptions of datasets, models, training of models, and evaluation metrics.

The results of the different methods are then presented. Mean accuracies for each model and method are presented along with the confusion matrices of the best-performing models. This is complemented by the distributions of the predicted probabilities. For deep learning methods, the evolution of the losses on training and test data during training are shown.

The results are then discussed along with the main advantages and disadvantages of each method. The practicality and generalization abilities of the methods are also discussed, and the discussion ends with cases where each method might be beneficial to implement.

The thesis ends with a short conclusion related to the aim and a discussion about future research. Here, several relationships in the data that might help with predicting road conditions are suggested.

2
Machine Learning

Machine learning aims to provide algorithms and statistical methods enabling computers to perform a specific task without being specifically programmed to do so [8]. By being fed data related to the given task, the models find patterns that allow for inference on similar but previously unseen data. The ability of these models to find patterns in high-dimensional data and generalize to new data has made machine learning a useful tool applicable in many fields.

Machine learning is separated into supervised, unsupervised, and reinforcement learning. Supervised learning consists of tasks with a correct answer, and the models are trained with guidance to learn the patterns of the data representing the correct answer. Unsupervised learning handles data in which there is no correct answer. The model aims to find interesting patterns and structures within the data. Once the inherent patterns and structures have been found, the model can use these to infer information about new, previously unseen data. Reinforcement learning consists of teaching an agent to make intelligent decisions. The agent interacts with an environment and gets different rewards depending on the actions taken. The agent wants to maximize the rewards received and needs to learn a strategy that produces good rewards.

This chapter aims to give a basic understanding of machine learning for classification tasks and some prominent algorithms for classification problems. It begins by introducing classification and conventional classifiers. This is followed by mathematical backgrounds for four commonly used conventional classifiers. Deep learning for classification is then introduced through fully connected neural networks along with the mathematics behind the learning process of deep learning models. The chapter then presents convolutional neural networks and vision transformers for image classification. Image segmentation is introduced as a pixel-level classification problem, and the U-net model is introduced. Fine-tuning is then introduced before the chapter finishes with evaluation methods, model selection, and feature importance.

2.1 Classification

Classification tasks fall under the category of supervised learning [9]. Classification tasks consist of inputting data with associated true labels. The data can, for instance, consist of images, text, or numerical features (pp. 1-6) [10]. Examples of classification tasks are "Is this an image of a cat or a dog?" or "Is the review negative or positive?". In these examples, the input data would be in the form of images and texts, respectively. Regardless of input data, all classification tasks can be seen as

creating a boundary between distributions. This boundary is called the decision boundary.

There are many different algorithms and models available for classification [8]. A. Kuehnle and W. Burghout [2], and P. Jonsson [3] used fully connected neural networks. Z. Sun and K. Jia [4] evaluated the performance of naive Bayes classifiers, support vector machines, K-nearest neighbors, and fully connected neural networks. K. Ozcan et al. [5] and A. Abdelraouf et al. [6] used more complex deep learning models for classifying, convolutional neural networks and vision transformers, respectively. The choice of classifier can be difficult and is important to evaluate.

The terminology used in machine learning, statistical learning, and pattern recognition vary. This thesis will use the terms *input*, *predictors*, and *features* interchangeably to describe the data. The terms *output*, *labels*, and *responses* will also be used interchangeably. All input sets will be referred to as X, and single observations will be referred to as x. The outputs will be denoted as y. Inputs will be assumed to have the shape $n \times m$ where n refers to the number of observations and m refers to the number of features. Consequently, x has the shape $1 \times m$ and y has the shape $n \times 1$. The available classes will be referred to as C and a single class c.

2.2 Conventional Classifiers

A common misconception is that machine learning is synonymous with deep neural networks. There is a vast amount of diverse classifiers which do not make use of any neural networks. These classifiers are often referred to as conventional classifiers. This thesis introduces the mathematics of three different conventional classifiers. These classifiers are used for evaluating different datasets. Details about the evaluation can be found in Section 4.1.

2.2.1 Naive Bayes Classifiers

The naive Bayes classifier is based on the Bayes rule, defined as

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)},$$

where c is the class label and x is the feature vector of an observation [4]. The terms are the posterior probability, $p(c|x)$, the likelihood, $p(x|c)$, the prior probability, $p(c)$, and the marginal likelihood, $p(x)$. Naive Bayes assumes that all features in X are independent. This allows for the following rewriting.

$$p(x|c) = \prod_{i=1}^{n} p(x_i|c),$$

where x_i is the i:th feature of the feature vector. We then assume that the likelihood follows some distribution. A common choice is the Gaussian distribution, and these classifiers are often called Gaussian naive Bayes classifiers. Assuming a Gaussian distribution allows us to estimate the variance and mean from our input data X.

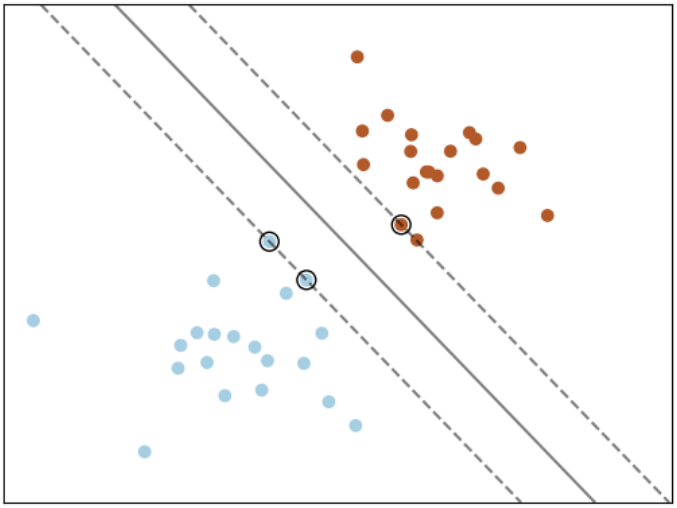

Figure 2.1: A fitted support vector machine with the optimal separating plane marked as a solid grey line. The distance between the dotted lines is the maximal margin.

In simple terms, the Gaussian naive Bayes classifier fits a multivariate Gaussian to each class, and the independent feature assumption results in the covariance matrix being diagonal. We then predict labels for our test data by choosing the most significant $p(c_i|x)$, $c_i \in C$.

2.2.2 Support Vector Machines

Support vector machine classifiers (SVM) optimize the position of hyperplanes to classify data [4]. The goal of SVM is to maximize the margin between the classes. The margin is the distance between the classes. By maximizing this, we position the separating hyperplane as far away from the separated classes as possible. Figure 2.1 shows an example of a fitted support vector machine with the optimal separating plane and maximal margin marked.

Define a hyperplane as

$$\{x : f(x) = x^T\beta + \beta_0\},$$

where β is a unit vector (pp. 417-426) [10]. We can now define a classifying rule as

$$G(x) = sign(x^t\beta + \beta_0).$$

It can be shown that $f(x)$ calculates the signed distance from a point x to the hyperplane $f(x) = 0$. Let $y_i \in \{-1, 1\}$ be the class label of observation x_i, then we can find a function $f(x)$ such that $y_i f(x_i) > 0$, $\forall i$. Let M be the margin and define the following optimization problem.

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$
$$\text{s.t. } y_i\left(x_i^T\beta + \beta_0\right) \geq M, i = 1, \ldots, n$$

The optimization problem only applies if the classes are linearly separable. In the cases where the classes are not linearly separable, we can introduce slack variables $\xi = (\xi_0, \xi_1, ..., \xi_n)$. The slack variables are the proportional amounts by which the predictions fall on the wrong side of the hyperplane. The introduction of ξ gives the following maximization problem.

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$
$$\text{s.t. } y_i \left(x_i^T \beta + \beta_0\right) \geq M(1 - \xi_i), i = 1, \ldots, n$$
$$\sum_{1}^{n} \xi_i < C$$
$$\xi_i > 0, \forall i$$

By bounding $\sum_{i=1}^{n} \xi_i$, we bound the total proportion of predictions falling on the wrong side of the hyperplane. By solving the optimization problem, we estimate the equation for the hyperplanes which separate the classes. Once we have the equations of the hyperplanes, we can make predictions on previously unseen data.

The model presented so far only produces linear boundary decisions. To create nonlinear decision boundaries, we can use kernels. Kernels replace 'regular' inner products with a more complex inner product. Let $h(x)$ define a transformation of x, we then define the transformed inner-product as

$$K(x, x') = \langle h(x), h(x') \rangle.$$

We can now define $K(x, x')$. Some popular choices include:

- dth-Degree polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$
- Radial basis: $K(x, x') = \exp\left(-\gamma |x - x'|^2\right)$.

By introducing kernels, we have augmented the feature space. The SVM classifier then creates hyperplane decision boundaries in the transformed feature space.

2.2.3 Random Forests

Random forest classifiers consist of an ensemble of de-correlated decision trees. Decision trees are the most straightforward and intuitive form of classifiers (pp. 305-312) [10]. The decision boundary is determined by asking yes or no questions about the data. These questions are in the form of "Is the value of feature z larger than 0.5?". This divides the feature space into hyperrectangles, and classification can be performed based on which hyperrectangle the observation lies in. The process can also be seen as a tree where each node corresponds to a question, and the leaves are classification labels. Figure 2.2 shows an example of the decision-making process of a decision tree.

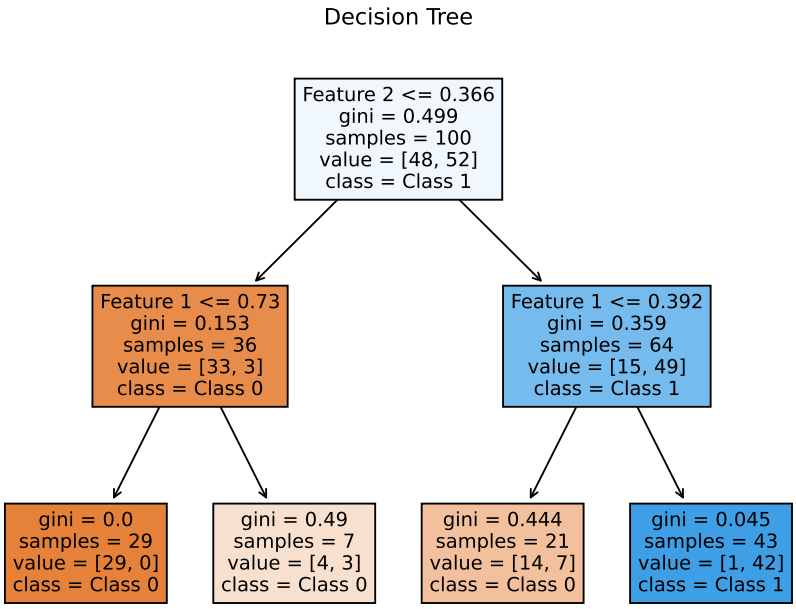

Figure 2.2: An illustration of a decision tree and the questions asked about the data in the decision process. Each node contains a question, the Gini impurity, how many samples are present, the class distribution, and what class is dominant. The color of each node represents the majority class in the node.

Let v define a node that will be split, R_v be the region represented by v, and N_v be the number of observations in R_v. Define the proportion of class c observations in v as

$$\hat{p}_{vc} = \frac{1}{N_v} \sum_{xi \in R_v} I(y_i = c).$$

Classification in node v can then be defined as $c(v) = \arg\max_c \hat{p}_{vc}$.

There are several impurity measures to evaluate the optimal split of a node. Impurity can be seen as how the samples are distributed in a node. For example, consider a node and n samples belonging to two different classes. If the node only contains samples from one class, then this node is considered pure. If the node includes samples from both classes, then it is considered impure.

The most common ways of calculating the impurity of the node include:

- Missclassification error: $\frac{1}{N_v} \sum_{i \in R_v} I(y_i \neq c(v)) = 1 - \hat{p}_{vc(v)}$
- Gini index: $\sum_{c \neq c'} \hat{p}_{vc} \hat{p}_{vc'} = \sum_{k=1}^{c} \hat{p}_{vc}(1 - \hat{p}_{vc})$
- Cross-entropy: $-\sum_{c=1}^{C} \hat{p}_{vc} \log(\hat{p}_{vc})$

Although all measures are similar, using either the Gini index or cross-entropy is preferable since these are differentiable and better suited for optimization. The Gini index and cross-entropy are also more sensitive to changes in the node probabilities.

The hierarchical structure of trees leads to susceptibility to high variance. If noise or other minor changes affect the initial splits of the tree, this change will propagate down through the tree and have a large impact on the resulting tree. A solution to this is bagging or bootstrap aggregation. Bagging consists of drawing B samples with replacements from the training data and fitting a decision tree to each. The final prediction generated is the average of all trees (pp. 587-602) [10]. This procedure works especially well for trees since they are approximately unbiased and have a high variance. An ensemble of trees will have the same bias as each tree. The only way to improve the ensemble's performance is to reduce the variance.

The correlation between trees is the main limitation of bagging. Consider a set of B identically distributed trees, let σ^2 be the variance of each tree, and let ρ be the positive pairwise correlation. The variance of the average becomes

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

As B increases, we can see that correlation becomes the limiting factor for the benefits of averaging. Random forests solve this issue by randomly sampling a number of the input features before splitting each node. This results in an ensemble of de-correlated trees, lowering the prediction's variance.

2.3 Deep Learning for Classification Tasks

Deep learning is a subset of machine learning that employs neural networks with multiple layers of interconnected nodes to approximate functions by adjusting the weights and biases during a training process [11]. These structures are loosely based on the structure of a human brain and are, therefore, referred to as neural networks. Today, there are many variations of neural network architectures, each designed with a specific task and data type in mind. The most general kind is the fully connected neural network. These networks were the first to be proposed and are not specialized in any specific data type. Other important network designs include convolutional neural networks, specialized in image data, and recurrent neural networks, designed with sequential data in mind.

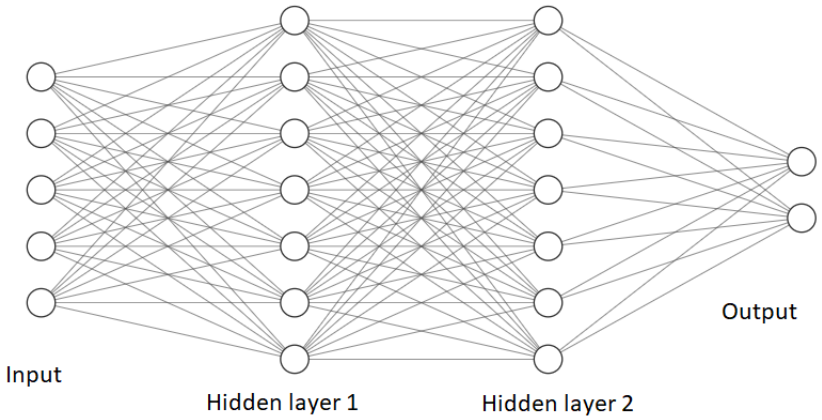

Figure 2.3: An example of a fully connected neural network for binary classification. Neurons are marked as rings, and the weights are represented by lines connecting neurons. The network takes five features as input. These features are processed in two hidden layers with seven neurons each, and finally, the output consists of two neurons for classification.

2.3.1 Fully Connected Neural Networks

Fully connected networks (FCNs) are the most basic kinds of neural networks (pp. 389-404) [10]. They consist of layers of neurons, and every neuron is connected to every neuron in the next layer. These networks were the first to be proposed and make no assumptions on the input data. This makes them excellent generalists able to perform well on various tasks. Figure 2.3 shows an FCN for binary classification.

The basic idea of all neural networks is to model the output as a nonlinear combination of the input. In essence, what all networks are is layers of linear regressors that take the output-vector from the previous layer and perform the standard calculation $y = w^T x + b$ where x is the input, w contains the weights and b is the bias. For the linear combination to become nonlinear, each layer includes an activation function applied to y [11]. The output of the j:th neuron in a layer can be described as

$$y_j = \sum_{i=0}^{n} w_{ji} x_i + b_j$$
$$x_j = f(y_j),$$

where x_i is the i:th element in the input, w_{ij} are the weights, b_j is the bias, f is the activation function, and x_j is the output and will act as input to the next layer.

Some commonly used activation functions are:

- Sigmoid: $f(y) = \dfrac{1}{1+e^{-y}}$
- ReLU: $f(y) = \max(0, y)$
- Softplus: $f(y) = \log(1+e^{y})$
- Tanh: $f(y) = \dfrac{e^{y}-e^{-y}}{e^{y}+e^{-y}}$.

All activation functions are designed such that their derivative is a function of the original function. The application of the nonlinear activation function is what makes sufficiently large neural networks able to approximate any function. The universal approximation theorem states that a network with just two layers, the first nonlinear and the second linear, can approximate any continuous function to an arbitrary degree of accuracy.

Applying the softmax activation function to the output is common in classification tasks. The softmax function for a K-class problem is defined as

$$f_i(y) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}},$$

where f_i refers to applying the softmax function to the i:th entry in y (pp. 389-404) [10]. The role of the softmax function is to transform the output into probabilities for each class.

The training process for neural networks consists of tuning the weights and biases in every layer [11]. For this to be possible, it is first necessary to have a ground truth, i.e., for a classification dataset, a correct class label must be assigned to each observation. This makes it possible to calculate the error of the network's predictions.

During the training of the network, the error of the output is propagated backward through the network; this is called backpropagation. The first step of backpropagation is calculating the error relative to a ground truth. The error measurement is referred to as the loss and is calculated using a loss function. There are many loss functions, each one suitable for different situations. In the case of classification, the most commonly used loss function is the cross-entropy, defined as

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij}\log(p_{ij}),$$

where y_{ij} is whether sample x_i belongs to class j, and p_{ij} is the predicted probability of sample x_i belonging to class j. The end goal of the training process is to find the minimum of the loss function.

The next step is determining how much each weight and bias in the network affects the loss. It might sound complicated, but all that is needed is the chain rule. The network can be seen as a bunch of connected parts. For example, the loss depends on what comes out of the last part of the network and how it is transformed by an activation function. But this output also depends on the weights and biases used in

the previous part of the network. To calculate how changes in weights and biases impact the loss, we need to follow these connections backward through the network, step by step. This way, we can determine how much each part contributes to the overall loss.

Let $\mathcal{L}$ be the loss, f be the activation function, $\hat{y}$ be the final prediction, y be the output before applying the activation function, and let w_i be the weight connecting the i:th neuron to the output layer. The gradient of the loss with respect to w_i becomes

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial y} \times \frac{\partial y}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times f'(y) \times x_i.$$

We now have an expression that contains only terms that are known. This method allows us to determine the gradient of the loss with respect to all parameters in the network. We use these gradients to minimize the loss through gradient descent.

Gradient descent is an iterative process of 'walking' closer and closer to the optimum. The general idea is relatively straightforward: we first choose a step size (often called the learning rate). We then calculate the gradient of the loss with respect to the weights and bias. A step is then taken in the direction of the gradient. Let $\mathcal{L}$ be the error, α be the step size, w be weight, and b be a bias. The update rule is then defined as

$$w \leftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$
$$b \leftarrow b - \alpha \frac{\partial \mathcal{L}}{\partial b}.$$

When the datasets are huge, it becomes computationally heavy to calculate the gradient using all n samples. The solution is to approximate the actual gradient by using a randomly selected sub-sample of the data. This is called stochastic gradient descent (SGD). In SGD, all observations are divided into batches. The gradient is then approximated, and the weights and biases are updated using one batch at a time. Once all batches have been processed, one epoch has passed.

There are several variations of the standard SGD algorithm; the most prominent are Adam and RMSprop. These algorithms use concepts like momentum and moving averages to generate updates. For more information, the interested reader is referred to the original paper for the Adam-optimizer and RMSprop lectures, [12] and [13], respectively.

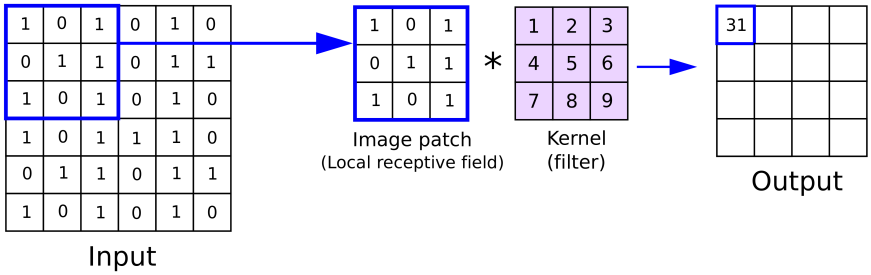

Figure 2.4: Example of applying a 3×3 kernel to one patch of an image. The kernel is elementwise multiplied by the patch, and the result is summed. Image taken from [14].

To conclude, one step of the training process for all neural networks can be broken down into the following steps:

1. Input a batch of data and get the predictions
2. Calculate the loss w.r.t. the ground truth
3. Perform backpropagation to calculate all gradients
4. Perform one step of gradient descent

2.3.2 Convolutional Neural Networks

For an FCN to process an image, it must first be flattened into a one-dimensional vector [11]. Let the image be an $n \times n$ RGB image. The input layer of the FCN would then be of size $3 \times n^2$, and each of these neurons would be connected to each neuron in the next layer. Even for smaller images, the parameter space becomes too big to be practical. Flattening images also removes the spatial relationship between pixels and is equivalent to looking at an image where the pixels have randomly switched places with each other.

Convolutional neural networks (CNNs) are networks that are specifically designed to process images effectively and maintain spatial relationships between pixels. This is achieved through kernels. Kernels are smaller patches of fixed weights that move along the image with a fixed stride. In each step, the kernel and the pixels it covers are multiplied elementwise and then summed. The process of applying the kernel to each image patch is called a convolution. Figure 2.4 shows the process of one step in the convolution.

The convolution can be seen as an FCN, except that not all neurons are connected. The only connected neurons correspond to pixels close to each other in the image. The benefit of this is the reduction in the number of parameters in the network and the preservation of the spatial relationship between pixels.

The reason for using constant weights in the kernel is to preserve local spatial invariance. Local spatial invariance refers to similar regions of the image being

processed similarly. For example, if the image contains two chairs in different regions of the image, then we would like these regions to produce a similar reaction from the kernel.

The output from the convolution is often referred to as a feature map [15]. As an image progresses through the layers of a CNN, the encoding becomes more and more complex. In the initial layers, the network tends to detect basic features like edges. In the following layers, it combines these edges to identify more complex patterns and features, such as textures, shapes, and object parts. The final feature map contains highly abstract and intricate image representations.

The shape of the output from a convolutional layer is determined by the chosen stride, number of kernels, kernel size, and padding [11]. The stride refers to how many pixels the kernel is moved at each step in the convolution. Padding refers to adding rows and columns to the image edges; this can be useful in manipulating the output image shape and helping the kernel process the image in more detail. Stride, padding, and kernel size affect the height and width of the output, and the number of kernels affects the depth of the output, often referred to as the number of channels. For example, let the input be a $n \times n \times 1$ feature map and let h and w be the height and width of the output. If we apply k kernels to the input, then the shape of the output will be $h \times w \times k$.

It is common to place pooling layers in between convolutional layers [16]. Pooling layers replace a neighborhood of a feature map with a summary statistic. These layers can also be seen as employing a convolution, except the kernel is not being learned but simply summarising the patch it covers. Pooling layers make the feature map more robust to small translations. This can make the feature map better if the presence of a feature is more important than the exact location. Common summary statistics are maximum and average.

Combinations of multiple convolutional layers and pooling layers are often referred to as convolutional blocks. A CNN designed for image classification typically consists of several convolutional blocks that learn to extract abstract representations of images. These convolutional blocks are then followed by an FCN that takes the flattened representation as input and learns to transform these into class predictions.

The VGG16 network is an example of a deep CNN [17]. It consists of 16 convolutional and fully connected layers. An illustration of the architecture can be seen in Figure 2.5.

An implementation of a VGG16 is used for classification later in this thesis. The details regarding the implementation can be found in Section 4.2.2.

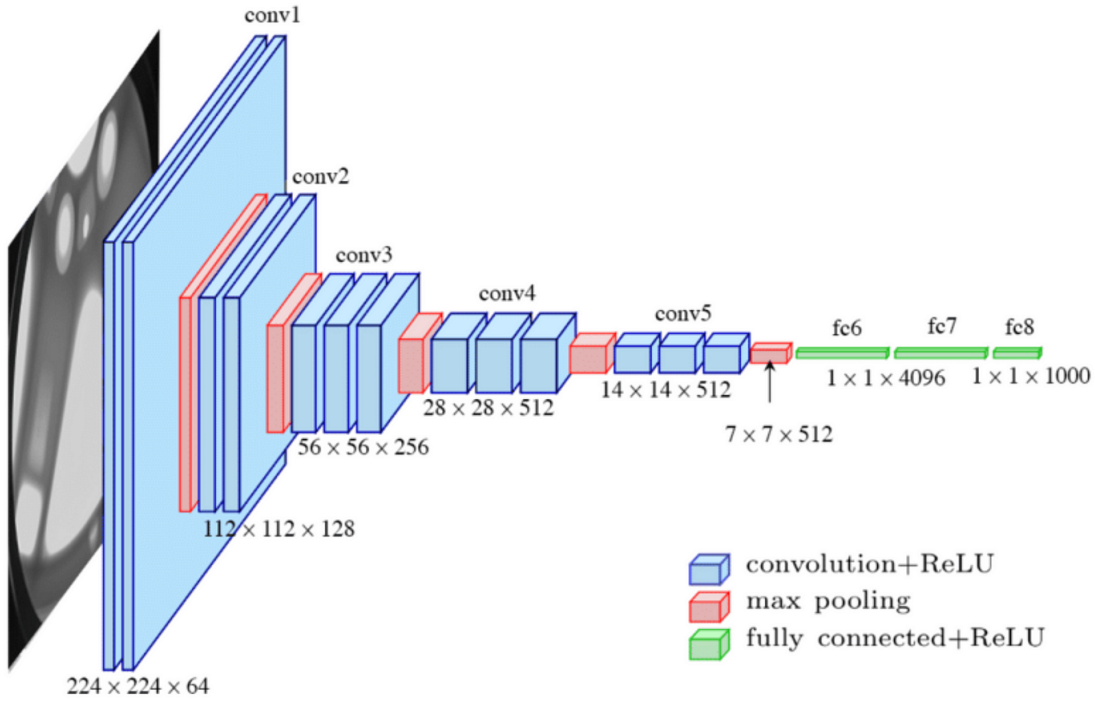

Figure 2.5: The VGG16 CNN architecture. The network consists of blocks of convolutional layers with max-pooling layers between them. Finally, a fully connected network performs the classification for the 1000 classes in the classification task. Image taken from [18].

2.3.3 Vision Transformers

Transformers were first introduced in 2017 as a solution to natural language processing problems [19]. Transformers proved to be remarkably efficient at solving natural language processing problems, and many recent developments in machine learning can be attributed to transformers (Ch. 11) [20]. After the success of transformers, many variations were proposed. These were designed to adapt transformers to other kinds of tasks. One such variation is the vision transformer [21]. Vision transformers have become a powerful tool for image classification, segmentation, and object detection (Ch. 11) [20]. An implementation of a vision transformer is investigated as a model for road condition classification in this thesis. The implementation details can be found in Section 4.2.3.

Transformers process sequential data through a learned attention mechanism. Attention can be seen as a measure of importance. The attention mechanism is comparable to a database. Databases consist of keys matched to values. A query is entered into the database and is matched to a key. Once the key has been identified (given that the key exists in the database), the matching value is returned as output. We can say that the inner workings of the database place attention on a value based on the query and the keys. This can be described mathematically. Let $\mathbf{q}$ be a query, $\mathbf{k}$ be the set of keys, and $\mathbf{v}$ be the set of values.

Attention is defined as

$$\text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

where $\alpha(\mathbf{q}, \mathbf{k}_i) \in [0, 1]$ are scalar attention weights. In a traditional database, exactly one of the weights is 1, and the rest is 0. In transformers, this does not have to be the case.

The weights $\alpha(\mathbf{q}, \mathbf{k}_i)$ are computed through what are called attention scoring functions. There are several variations, but the most commonly used is the scaled dot product attention [19]. Suppose we have n queries and m key-value pairs. Let the queries and keys be vectors of length d, and values are of length v. Store all queries, keys, and values in the matrices $\mathbf{Q} \in \mathbb{R}^{n\times d}$, $\mathbf{K} \in \mathbb{R}^{m\times d}$, and $\mathbf{V} \in \mathbb{R}^{m\times v}$ respectively. The scaled dot product attention is then defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}.$$

To motivate the equation, consider the case where n and m equals 1. Let $\mathbf{q}$ be a query and $\mathbf{k}_i$ be a key. Additionally, let both the query and the key be independently and identically drawn random variables with mean 0 and variance 1. The dot product $\mathbf{q}^T\mathbf{k}_i$ measures how similar the query and the key are. The problem is that the variance of the dot product is equal to d. To solve this, we scale with $1/\sqrt{d}$. The softmax function is then applied to limit the output to the range $[0, 1]$ and for the sum of all weights to be equal to 1. We now have an expression for the weight $\alpha(\mathbf{q}, \mathbf{k}_i)$, which is based on the similarity of the query and the key

$$\alpha\left(\mathbf{q}, \mathbf{k}_i\right) = \text{softmax}(\frac{\mathbf{q}^\top\mathbf{k}_i}{\sqrt{d}})$$

The keys and values are already established as concrete pairs in a traditional database. In transformers, we need the keys and values to be more flexible and not limited to what has already been put into the database. To achieve this, we train fully connected neural networks to create representations of our input sequence. The FCNs used commonly only have one hidden layer and no activation function applied. This results in linear representations of the input. These representations are then used as queries, keys, and values. Let $\mathbf{W}^Q, \mathbf{W}^K$, and $\mathbf{W}^V$ be the learnable parameters of three different FCNs, and let X be an input sequence. The queries, keys, and values are as follows.

$$\mathbf{Q} = \mathbf{W}^Q X$$
$$\mathbf{K} = \mathbf{W}^K X$$
$$\mathbf{V} = \mathbf{W}^V X$$

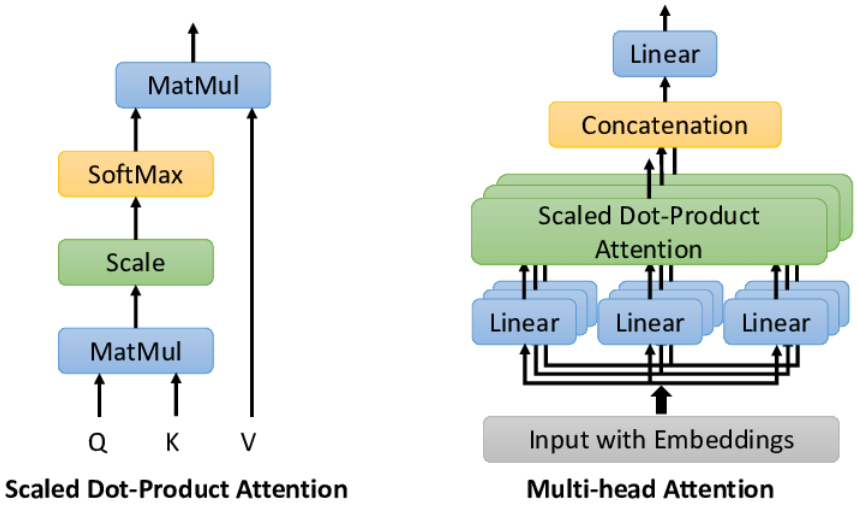

Figure 2.6: A schematic of scaled dot product attention (left) and multi-headed attention (right). In the multi-headed attention several attentions are calculated in parallel based on queries, keys, and values produced through different learned linear transformations. These attentions are then concatenated and transformed linearly again. Image taken from [22].

These matrices are then input into the attention scoring function, and an attention matrix with the same shape as X is returned. This procedure is referred to as self-attention. The entire chain of inputting a sequence, computing queries, keys, and values, and computing attention is called an attention head.

Many attention heads can be used in parallel to gain additional information, this is called multi-headed attention[19]. When training a neural network from scratch, the resulting parameters rely on their initialization (Ch. 5.4) [20]. There are many schemes for initializing parameters, but all of them result in sampling from some probability distribution. Training two neural networks with the same architecture and with the same data will result in two slightly different final models because of the randomness in the parameter initialization.

Multi-headed attention uses several attention heads to gain insight into the queries, keys, and values from many different FCNs [19]. The resulting attentions from each attention head are concatenated and processed once more in another FCN to output the final attention. Figure 2.6 shows schematics of scaled dot product attention to the left and multi-headed attention to the right.

The original transformer architecture consists of an encoder and a decoder. The decoder is, however, not needed in vision transformers [21]. Readers interested in the decoder used in the original transformer are referred to [19].

The encoder consists of a multi-headed attention layer, an MLP (multi-layered perception, an FCN with multiple hidden layers), residual connections, and layer normalization [19]. Transformer encoders can use the output of another encoder as input. This allows the encoders to be stacked, creating deeper and more complex transformers. Figure 2.7 shows a schematic of a transformer encoder.

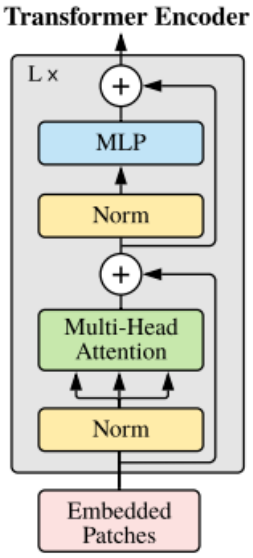

Figure 2.7: A schematic over a transformer encoder. The encoder takes embedded image patches and normalizes them before sending them to a multi-headed attention layer. Afterward, the original patches are added, called a residual connection. The output is then normalized again before it is sent to an MLP. After that, another residual connection adds the output from the last residual connection. Encoders can be stacked multiple times, making the model more complex. Image taken from [21].

Residual connections is a technique commonly used for training very deep neural networks (Ch. 8.6) [20]. All deep learning models are trained through backpropagation, which requires the multiplication of many gradients. Deeper models run the risk of gradients either vanishing or exploding. Residual connections counteract this by simply allowing the model to 'skip' a block of layers that does not improve the output.

Transformers operate by processing data in the form of sequential embeddings [19]. In vision tasks, such as image classification, vision transformers break down images into fixed-size patches, treating them as sequential data [21]. These patches are converted into one-dimensional vectors and linearly projected to a dimension D. Subsequently, a positional embedding is incorporated, enabling the model to preserve spatial information about each patch's location within the image, which aids in capturing global context. A learnable class token, denoted as $\mathbf{x}_{class}$ and featuring a positional embedding of 0, is also introduced. Both the embedded patches and the class token undergo processing by the encoders. Finally, the processed class token serves as input to an MLP to generate the final classification outcome. For a visual representation of this architecture, see Figure 2.8.

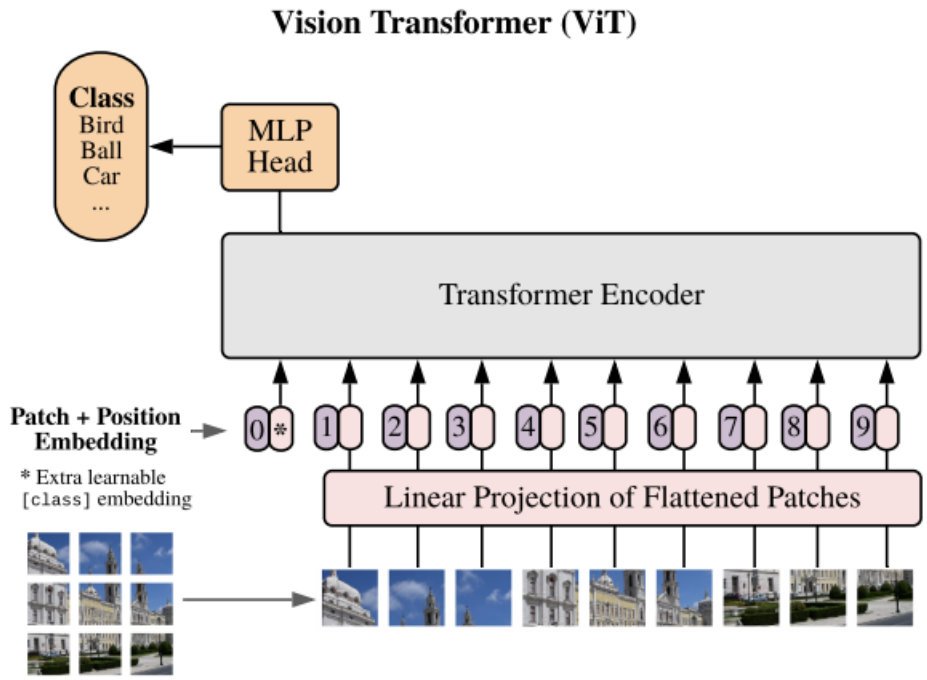

Figure 2.8: The architecture of a vision transformer. The input image is divided into patches, which are subsequently flattened and linearly projected. An additional learnable embedding is added, called a class token. Position embeddings are then applied to each patch before encoders process them. The resulting class token is then propagated through an MLP to get a final classification. Image taken from [21].

2.3.4 Image Segmentation

Image segmentation is the problem of classifying each pixel in an image instead of the image as a whole [23]. Segmentation has many important applications, such as medical image analysis, robotic perception, and image compression. The rise of deep learning has encouraged researchers to develop deep learning models to solve the problem. There are many different models in deep learning to segment images, and the models are not restricted to using convolutional layers, but because of convolutional layers' efficiency in processing image data, most models do. This paper focuses on segmentation models that use convolutional layers.

In image segmentation, there are three main tasks: semantic segmentation, instance segmentation, and panoptic segmentation. Semantic segmentation is concerned with labeling all the 'stuff' in an image, such as marking cars as 'car' and everything else as 'not car.' Instance segmentation is concerned with counting all the 'things' in an image, like 'car 1' and 'car 2.' Panoptic segmentation is a mix of both – it classifies all pixels like semantic segmentation but also gives each object a unique ID.

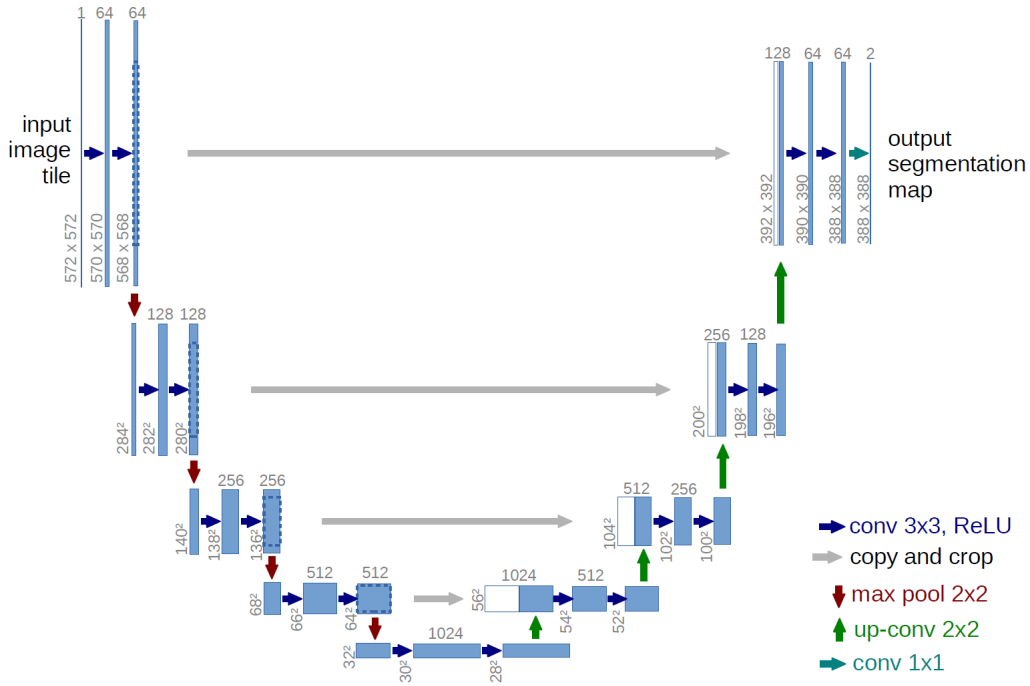

Figure 2.9: The U-net architecture. The image is contracted into an abstract representation using convolutional blocks. The representation is then expanded using transposed convolutional layers. The output of each contracting block is cropped and concatenated to the output of each expanding block to transfer knowledge of features of different complexity found during contraction. Image taken from [25].

There are different metrics for measuring the performance of segmentation models. A commonly used metric is the intersection over union (IoU), defined as

$$\text{IoU} = J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

where A is the prediction segmentation map, and B is the true segmentation map. The IoU can take values between 0 and 1, where 1 is a perfect segmentation performance.

As previously mentioned, convolutional layers can create abstract representations of the objects in images [24]. These feature maps can be used as the foundation for locating and identifying the different objects in the image. Once convolutional blocks have created an abstract representation, we must recreate the original image but with a class label assigned to each pixel. There are many interesting and creative solutions to this, and surveys of different methods can be found in [23] and [24].

The U-net model is an image segmentation model initially proposed in 2015 and was designed to segment neural structures in electron microscopic stacks [25]. This thesis implements a U-net for the segmentation of roads. Details regarding the implementation can be found in Section 4.1.1.

The U-net architecture consists of a contracting and expanding part (encoder and decoder). An illustration can be seen in Figure 2.9.

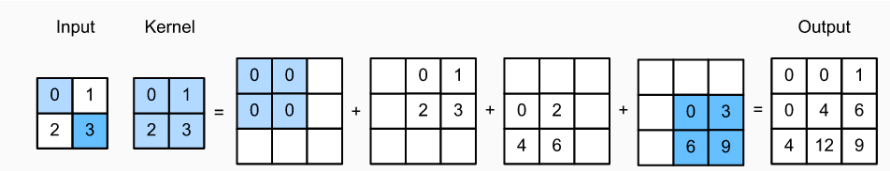

Figure 2.10: An example of a transpose convolution where a 2×2 kernel is applied to a 2×2 image. The kernel is multiplied by each pixel in the image, and the image is expanded. Image taken from [26].

The U-net makes use of convolutional blocks to create an abstract representation. Once the representation has been created, it is expanded using transpose convolutional layers. In each convolutional block in the contraction, features relevant to the composition are extracted. To make use of this information, it is cropped and concatenated to the corresponding transpose convolution block. This allows the model to learn important information about the original image.

Transpose convolutional layers can be seen as the opposite of convolutional layers (Ch. 14.10) [20]. They use learned kernels, convoluted over an image precisely like a regular layer. The key difference is that it does not aggregate the element-wise multiplications. Instead, it expands the border of the image.

Figure 2.10 shows an example of a transposed convolution. The kernel is multiplied by each pixel, and the result is summed up in a new image. Like regular convolutional layers, transposed layers have a stride and kernel size.

2.3.5 Fine-tuning

Transformers require large amounts of data to be trained [21]. The first vision transformer required 14-300 million images (depending on model complexity) to compete with the best CNNs in image classification. Since these data amounts are usually unavailable, most transformers utilize fine-tuning. Fine-tuning consists of copying the parameters of a model that has already been trained on a similar problem and training the model on new data (Ch. 14.2) [20]. There are many possible strategies for fine-tuning. One common strategy is to 'freeze' earlier layers of the model, i.e., to not update the parameters of these layers during training on the new dataset.

For example, consider a model trained to classify if the animal in an image is a dog, cat, or rabbit. We want to solve the problem of classifying whether the images in another dataset contain horses or not. We can take advantage of the patterns and features that the model has already learned by simply replacing the final classifying layer of the model with a new layer suitable for our task. We then freeze all layers except the output layer. The model is then trained on the new dataset to adapt its knowledge to the new task. By fine-tuning, we do not need to use as much data as training from scratch since the model does not have to learn basic features and patterns. Fine-tuning works on all neural networks but is most commonly applied to CNNs and transformers.

2.4 Model Evaluation and Selection

This section presents the challenges of selecting and evaluating machine learning models. It begins by discussing model complexity and common techniques for evaluating model performance. Then, the use of these evaluation methods for selecting the best model is discussed. The section ends with the importance of selecting relevant features and presents filtering methods for feature selection.

2.4.1 Model Complexity and Overfitting

The goal is the same in all classification tasks regardless of the chosen model [27]. We want to find the decision boundary that best separates the classes. The best model creates a decision boundary that can accurately predict previously unseen data. If a model performs well on data already seen but fails at making accurate predictions on new data, then the model is overfitting. More complex models can create more complex decision boundaries and precisely model the data. If the model is more complex than the data, this might lead to overfitting. Choosing the right complexity of a model is often called the bias-variance tradeoff.

To evaluate a model's performance, it is common to separate the dataset into two independent datasets. One dataset is used for training, and the other for testing. The model does not 'see' the test data during training. This method measures the model's ability to generalize to new data. The downside of this method is that the performance on the test data relies on the choice of test data.

Another common way to evaluate a model's performance is K-fold cross-validation (pp. 241-249) [10]. In cross-validation, the dataset is divided into K folds that are iteratively used as training and test data. In each iteration, $K - 1$ folds are used to train the model, and the K:th fold is used as test data. In the next iteration, a different fold is used for test data, and the remaining folds are used for training. The procedure continues until every fold has been used for testing once. This gives a good estimation of the model's performance on the entire dataset.

The choice of K is important in cross-validation. An extreme choice is $K = n$, in which one sample is left out in each iteration. This gives a very accurate representation of the performance of our model but might lead to a high variance because of potential similarities between the training data and test data. Another downside is that this choice can become computationally heavy for large datasets and complex models. Common choices for K are 5 or 10.

Using cross-validation is the ideal way to evaluate a model, but training a complex model over and over again on large datasets is computationally heavy. When evaluating deep learning methods, it is more common to split the data once into training and test data.

Cross-validation will not prevent a model from overfitting on data, but it will estimate a model's ability to generalize. If a model is overfitting, it is necessary to make it less complex [27]. One way to battle overfitting is through regularization. Regularization refers to techniques that essentially make the model less complex during training. Not all techniques are applicable to all models. For neural networks, one regularization method is called weight decay (pp. 389-404) [10]. This method adds

a penalty proportional to the sum of all weights and biases (the tunable biases of neural networks) to the loss. In the optimization process, this will force the weights and biases towards zero and, therefore, decrease the complexity of the model. Another common method is drop-out (Ch. 5.6) [20]. In drop-out, weights are randomly set to zero during each training iteration, killing the neuron. This elimination of neurons forces the model to not rely too much on any given neuron and spread the weights between all neurons more evenly. During testing, drop-out is not activated.

Another technique, especially useful with image data, is data augmentation (Ch. 14.1) [20]. By augmenting images from the dataset, we can increase the dataset's size and improve our model's generalization. By introducing altered images, we teach the model to be invariant to the changes introduced, leading to better generalization.

2.4.2 Model Selection

Each of the models described has tuning parameters. Naive Bayes requires the choice of assumed likelihood distribution and prior distribution [4]. SVM can be tuned through the slack variables and choice of kernel (pp. 417-426) [10]. Random forests and neural networks have a significantly higher number of hyperparameters and methods to choose between (pp. 587-602) [10] (pp. 389-404) [10].

Hyperparameters and models can be evaluated through cross-validation or by splitting the data into training and test data (pp. 241-249) [10]. Let $f(x, \alpha_i), \alpha_i \in \alpha$ be a set of models with different tuning parameters α_i. By evaluating each model's performance, an estimation of the test error curve is acquired. The model which minimizes this curve is the model which performs best.

2.4.3 Feature Importance and Selection

All classification algorithms need meaningful features as input [28]. In many applications, not all features are correlated with the classes and, therefore, act as noise. More features also lead to a higher computational cost, and higher dimensional feature spaces can lead to difficulty in making predictions. The ability to measure the importance of features and only select the most important ones is crucial and can lead to improvements in performance.

Feature selection methods are categorized into filtering, wrapper, and embedded methods. Filtering methods rank the importance of each feature using some statistical measurement. Once the features are ranked, the best ones can be selected. Wrapper methods evaluate subsets of features to find the one that maximizes the model's performance. Embedded methods take the feature importance into account during the training of the model. Embedded and wrapping methods are not relevant to this thesis and will not be explained further. Interested readers are referred to [28].

To apply a filtering method, it is important to choose relevant statistical criteria to rank the feature importance [28]. Three examples are variance, ANOVA F-statistic, and mutual information. Variance is the most straightforward choice and is easily motivated. Features that do not vary across samples are unlikely to contribute information related to class labels.

ANOVA stands for analysis of variance, and the F-statistic is the ratio of two variances [29]. The null hypothesis for ANOVA tests is that all group means are equal, and the alternative hypothesis is that the means are not all equal.

$$H_0 : \mu_1 = \mu_2 = ... = \mu_K$$
$$H_1 : \text{The means are not all equal}$$

In the context of feature ranking, one feature is selected at a time, and the groups are the classes. The feature is then split up among the classes, and the null hypothesis is that the mean of the feature across all classes is the same. The test-statistic is

$$F = \frac{\text{Variance between groups}}{\text{Variance within groups}} = \frac{\sum_{i=1}^{K} n_i \left(\bar{X}_i - \bar{X}^2\right)/(K-1)}{\sum_{i=1}^{K} \sum_{j=1}^{n_j} \left(X_{ij} - \bar{X}_i\right)^2/(n-K)},$$

where n_i is the sample size of group i, $\bar{X}_i$ is the mean of group i, $\bar{X}$ is the overall mean, X_{ij} is the j:th observation in group i, K is the number of groups and n is the number of samples in total, i.e. $n = \sum_j n_j$. A high F-statistic indicates a higher class separability and, therefore, a more important feature.

Mutual information is a measure of dependency between two variables based on entropy [28]. Let Y be a discrete random variable with an associated probability distribution $p(Y)$. We define entropy for an output Y as follows.

$$H(Y) = -\sum_{y \in \mathcal{Y}} p(y)\log(p(y)),$$

where $\mathcal{Y}$ are all values that Y can take. $H(Y)$ represents the uncertainty in output Y. Suppose we observe a variable X, we can then define the conditional entropy as

$$H(Y|X) = -\sum_{y \in \mathcal{Y}} p(y,x)\log(p(y|x)).$$

This definition is for discrete distributions but can easily be extended to continuous distributions by replacing the summations with integrals. We can now define mutual information as

$$I(Y,X) = H(Y) - H(Y|X).$$

In classification problems, one distribution might be continuous and the other discrete. Calculating the mutual information in this case becomes more complicated, and the procedure will not be explained here; for more details on the calculations, see [30].

The advantage of filtering methods is that they are computationally light [28]. The overall simplicity of feature rankings does come with some flaws. One major flaw is that features that do not separate classes well on their own might work well in combination with other features. These features are likely to be removed, which can result in a sub-optimal selection. Another flaw is that the classification algorithm is not taken into account. This can make it hard to find the optimal algorithm to use.

Filtering methods also require an input of how many features to select, and there is no optimal way to find this number.

3
Feature Extraction

Feature extraction constitutes a foundational step in data analysis and machine learning, serving the purpose of converting raw data into a more concise and informative representation. Across diverse domains, including computer vision, natural language processing, and signal processing, feature extraction assumes a pivotal role in simplifying intricate data while accentuating pertinent patterns and attributes.

Through the extraction of meaningful features from the original data, practitioners can effectively reduce the dimensionality of datasets, enhancing their manageability and efficiency for subsequent analysis or modeling. The selection or engineering of these extracted features hinges on domain expertise, statistical methodologies, or machine learning techniques, depending on the specific problem at hand.

This chapter introduces ways to extract features from images. The concept of texture analysis is introduced along with a common texture analysis technique, grey level run-length matrices. Then, persistent homology is introduced as a feature extraction technique.

3.1 Texture Analysis

Texture analysis is a broad field consisting of different methods to extract meaningful features that describe textures in images [31]. There is no agreed-upon definition of texture, but it can be considered to measure coarseness, contrast, directionality, line-likeness, regularity, and roughness. It can also be considered to measure similarity groupings or semi-repetitive arrangements of pixels. Despite lacking an agreed-upon definition, texture analysis has proven successful in various fields, such as texture classification, segmentation, and pattern recognition. P. Jonsson [3] and Z. Sun and K. Jia [4] showed that features extracted using texture analysis are useful in classifying road conditions.

Many techniques for extracting texture features are available and fall into many different categories, such as statistical approaches, structural approaches, transform-based approaches, etc. [31]. This thesis evaluates the statistical method grey level run-length matrices, which has been shown to perform well for road condition classification [3].

The concept of run-lengths was originally introduced by M. Galloway in 1975 [32]. This method involves examining pixels with varying grey levels while tracking how many pixels one can traverse in a particular direction before encountering a change in grey level. The results of this analysis are organized into a run-length matrices.

To perform this analysis, the image is typically scanned in four distinct directions:

0, 45, 90, and 135. However, if we were to preserve all 255 grey levels for each pixel, it would be computationally heavy. Consequently, it is common practice to reduce the number of grey levels in the image through quantization.

Each element, $p(i, j)$, of the run-length matrix shows how many times pixels of grey level i had a run of length j. M. Galloway [32] originally proposed five statistical measures based on the run-length matrix. In 1989, A. Chu et al. [33] introduced two additional statistics which were later generalized into four statistics [34]. Let the run-length matrix have the shape $I \times J$ and P be the number of pixels in the image. These eleven features are defined as:

1. Short run emphasis: $\text{SRE} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)}{j^2}$
2. Long run emphasis: $\text{LRE} = \sum_{i=1}^{I}\sum_{j=1}^{J} j^2 p(i,j)$
3. Grey level nonuniformity: $\text{GLNU} = \sum_{i=1}^{I}\left(\sum_{j=1}^{J} p(i,j)\right)^2$
4. Run length nonuniformity: $\text{RLNU} = \sum_{j=1}^{J}\left(\sum_{i=1}^{I} p(i,j)\right)^2$
5. Run percentage: $\text{RPC} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)}{P}$
6. Low grey level run emphasis: $\text{LGRE} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)}{(i+j)^2}$
7. High grey level run emphasis: $\text{HGRE} = \sum_{i=1}^{I}\sum_{j=1}^{J} p(i,j)(i+j)^2$
8. Short run, low grey level emphasis: $\text{SRLGE} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)}{j^2(i+j)^2}$
9. Long run, high grey level emphasis: $\text{LRHGE} = \sum_{i=1}^{I}\sum_{j=1}^{J} p(i,j)j^2(i+j)^2$
10. Short-run, high grey level emphasis: $\text{SRHGE} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)(i+j)^2}{j^2}$
11. Long-run, low grey level emphasis: $\text{LRLGE} = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{p(i,j)j^2}{(i+j)^2}$

These features capture overall tendencies in the run-length matrix. The short run emphasis, for instance, is large if the image is dominated by short runs, and high grey level run emphasis is large if the image is dominated by runs of higher grey levels. All eleven statistics in all four standard directions are extracted for road condition classification in Section 4.1.3.

3.2 Topological Data Analysis and Persistent Homology

Machine learning models often face challenges when dealing with high-dimensional data due to a phenomenon known as the 'curse of dimensionality' (pp. 22-33) [10]. The 'curse of dimensionality' refers to the distortion of distance measurements as the number of dimensions increases. Persistent homology offers a solution for reducing the dimensionality of high-dimensional data by extracting meaningful features that capture the data's geometric and topological properties [7]. Although the underlying mathematics is interesting, it is not necessary to understand the general concept. Interested readers are referred to Appendix C.

Topological data analysis (TDA) consists of a set of methods to find structures in data by analyzing the data's topology [35]. TDA uses concepts such as connectivity and shape to allow for visualization and analysis of high-dimensional and complex datasets. Persistent homology (PH) is one of the more widely used methods of TDA.

To extract features from images, PH treats the image as a discrete surface [36]. A cross-section is then applied to the surface at different levels. At each level, homology groups are recorded; this process is called filtration. Homology groups can be viewed as a way to measure the number of holes of different dimensions in a structure. The 0th homology group keeps track of connected components, the 1st homology group looks at loops, the 2nd group looks at enclosed volumes, etc.

During the filtration of images, we are concerned with the parts of the surface that lie below the cross-section at each level. Connected components, in this context, refer to points on the surface that are either alone or neighbors of other points below the cross-section. By recording at which filtration level a connected component appears and which level it merges with another component, we can track the birth time and death time of a homology group of dimension 0. All homology groups' birth and death times can be summarised in a persistence diagram (PD) [7]. A PD is a scatterplot where birth times are on the x-axis and death times are on the y-axis. The PD is sometimes transformed into a rotated PD. This is done through the transformation $(x, y - x)$, where $y - x$ is the persistence of a homology group.

A PD cannot be directly input into a machine learning model, but extracting useful features is possible. One way of doing this is by computing persistent images [37]. By assigning a weighted Gaussian to each point in the PD, we create a persistent surface. The surface is then divided into 'pixels' through a grid, and finally, we integrate each cell in the grid. This creates an image of the PD, which can be used as input into a machine learning model.

4

Methodology and Datasets

The methods explored and evaluated in this thesis can be divided into two sections. The first method is inspired by the methods presented by A. Kuehnle and W. Burghout [2], P. Jonsson [3], and Z. Sun and K. Jia [4]. These papers focused on extracting features from the image based on texture analysis, color distributions, and grey levels. The extracted datasets were evaluated using conventional classifiers and small, fully connected neural networks. This method will be referred to as the 'Conventional method'.

The second method explores deep learning methods inspired by K. Ozcan et al. [5] and A. Abdelraouf et al. [6]. This method involves fine-tuning a pre-trained VGG16 convolutional network and a pre-trained vision transformer. This method will be referred to as the 'Deep learning method'.

This chapter presents the details of the conventional and deep learning methods. The chapter begins by explaining the details of the conventional method. This includes information about the image data, the training of a U-net, features extracted from the roads, evaluated classifiers, and feature selection techniques used. This is followed by the details regarding the deep learning method. This includes information about the image data, the fine-tuning of a VGG16 network, and the fine-tuning of a vision transformer. The chapter ends with the metrics used for evaluating the methods and models.

4.1 Conventional Method

This section describes the conventional method. It gives details about the images used in the dataset. It then presents the details of training the U-net for image segmentation. Then, the image dataset and the distributions of the classes are presented. The features extracted from the images are then presented. This is followed by the classifiers used for evaluation and the procedure of selecting features.

4.1.1 Training the U-net for Road Segmentation

First, the road was segmented from the background of the image. This was done to exclude noise from pixels unrelated to the road. For this purpose, a U-net was trained on a dataset comprising 264 images of roads from different cameras. Details about the theory of the U-net can be found in Section 2.3.4. The U-net depicted in Figure 2.9 uses the encoder and decoder proposed in the original paper. A U-net can, however, use many different kinds of encoders and decoders. The encoder and decoder used in this thesis are based on the EfficientNet architecture. The EfficientNet was chosen through the evaluation of different options. The EfficientNet is not explained in this thesis. Interested readers are referred to [38].

All images were preprocessed before being sent to the U-net. They were first cropped by 80% in width and height. They were then resized to 544×544 pixels. The channels were normalized with means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$ for the red, green, and blue channels, respectively.

The data was split into a 70/30 training data and test data split. The Adam optimizer with weight decay was used with a learning rate of 10^{-4}. The batch size was set to 2, and the model was trained for 20 epochs. The loss function was binary cross entropy.

The U-net architecture was chosen for its ability to learn from small sets of images, as presented by Ronneberger et al. [25]. Through heavy image augmentation, they managed to produce a good image segmentation model using only 30 images as training data.

The training used batch-wise data augmentation. Each batch had a 10% chance to be augmented. If the batch was augmented, then all images in the batch were subject to one of the augmentations found in Table 4.1.

Once the model was trained, the road from each CCTV station was segmented, and the pixels were stored. When features were extracted from images, only these pixels were used.

Type	Range
Rotation	[0, 179] (degrees)
Jitter	Brightness: [0.5, 1.5] Contrast: [0.5, 1.5] Saturation: [0, 1.5]
Affine Transform	Rotation: [-180,179] (degrees) Translation: ([0,99],[0,99]) (pixels) Shear: ([-180,179],[-180,179]) (degrees)
Crop	[100,300] (square cropped image size, pixels)
Flip	[Horizontal, Vertical] (axis)

Table 4.1: The augmentations applied to the batches when training the U-net. The left column describes what type of augmentation was applied. The right column shows what range a random value was sampled from. If an augmentation comprises several transformations, each is listed with the associated sample range. If the augmentation changed the image size or distribution of pixel values, the augmented images were resized and normalized to preserve the original image shape and pixel value distribution.

4.1.2 Image Data

For this method, 2450 images were collected from ca. 30 different CCTV stations placed throughout Sweden. The class balance of the dataset was 707/839/904 for snow/wet/dry. For the classification of snow/no snow, the 'no snow' samples were reduced to 50% of the original size through random sampling, resulting in a snow/no snow ratio of 707/872. This was done to maintain class balance.

The datasets consist of 2450 images from ca. 30 different stations. This can result in several images being almost identical and, thus, an overlap between the training and test data. To properly evaluate the generalization of a model, it is important that training and test data are split based on the stations, i.e., images from one station cannot occur during training and testing. The cross-validation used to evaluate the classifiers splits the data approximately 70/30 for training/test data based on CCTV stations.

Feature
Mean Birth Time
Mean Death Time
Mean Persistence
Median Birth Time
Median Death Time
Median Persistence
Variance Birth Time
Variance Death Time
Variance Persistence
Minimum Death Time
Minimum Persistence

Table 4.2: List of features extracted from the persistence diagrams for the 0th and 1st homology groups.

4.1.3 Feature Extraction

Three datasets were extracted from the images with the road segmented. These datasets will be referred to as follows:

1. GLRLM (grey level run length matrix)
2. PD basic (persistence diagram basic)
3. Persistent images

Introductions to the theory behind the features contained in the datasets can be found in Section 3.

The first dataset, GLRLM, consisted of all features extractable through grey level run-length matrices, as described in Section 3.1. These 11 features were extracted in all four standard directions. In addition to these features, a binary feature for day or night was added. This resulted in 45 features.

The second dataset, PD basic, consists of simple features extracted from persistence diagrams. These features were extracted for both the 0th homology group and the 1st homology group. To these features, a Day/Night indicator feature was added. This resulted in 24 features. The features extracted from the 0th and 1st homology groups' PDs can be found in Table 4.2.

The third dataset, Persistence images, comprises features extracted through persistent images of the 0th and 1st homology groups. The persistent images consisted of 400 pixels, where the Gaussians had a bandwidth of 15, and the associated weight was the same for all distributions. Most features extracted this way had a variance of ~ 0. Therefore, only the features with a variance higher than 2.5 were selected. To these features, a Day/Night indicator was added. This resulted in 63 features.

4.1.4 Classifiers

The datasets were evaluated using four different classifiers: a Gaussian naive Bayes classifier, a support vector machine, a random forest, and a fully connected neural network. All classifiers, except the naive Bayes classifier, were tuned using cross-validation. The details of the tuned parameters can be found in Appendix A. For the mathematical theory behind the classifiers, see Sections 2.2 and 2.3.1.

4.1.5 Feature Selection

Two metrics were used to rank the feature importance: ANOVA-F and mutual information. The optimal number of features was determined for each ranking by incrementally adding one feature at a time in order of rank and evaluating the tuned models each time through cross-validation. ANOVA-F and mutual information are introduced in Section 2.4.3.

4.2 Deep Learning Method

This section presents the details of the deep learning method. It presents the image datasets, data pre-processing, and data augmentations. The fine-tuning of a VGG16 network is then presented. The fine-tuning of a vision transformer follows this.

4.2.1 Image Data

Training deep learning models generally requires more data than training conventional models. For fine-tuning to classify snow or other, a dataset consisted of 8959 images from ca. 80 CCTV stations from Sweden and Finland. For test data, images from 15 stations were used. This resulted in a training/test split of 7219/1740. The training and test data were both relatively balanced between the classes snow and no snow. The ratios of snow/no snow were 3266/3953 and 864/876 for training and test data, respectively.

For the classification of snow, dry, or wet, the dataset consisted of 7693 images with a training/test split of 5953/1316. The ratio of snow/wet/dry for training and test data were 2000/2202/1751 and 440/453/423, respectively.

All images were cropped by 20% percent on each side, 10% at the bottom, and 20% at the top. This was done to remove as much background and surrounding landscapes as possible and force the model to rely more on the road for classification.

Data augmentation was applied to the training datasets. Every image in the training data was subject to the augmentations in Table 4.3.

Type	Range
Rotation	[0, 20] (degrees)
Flip	[Horizontal, Vertical] (axis)

Table 4.3: The augmentations applied to each image in the training data before training the model. The left column describes what type of augmentation was applied. The right column shows what range a random value was sampled from.

4.2.2 Fine-tuning VGG16

Ozclan et al. [5] showed that fine-tuning a VGG16 network resulted in promising performance for classifying road conditions. The network was pre-trained on the Places365 dataset. Places365 is a scene recognition dataset comprising 10 million images of scenes from 434 classes [39].

VGG16 is a convolutional neural network with 16 layers (not including pooling layers). A detailed explanation of each layer can be found in Table 4.4, and a visual explanation can be seen in Figure 2.5.

Block	**Layers**	**Frozen**
Input	224x224x3 (RGB Image)	-
Block 1	conv3-64+ReLU	Frozen
	conv3-64+ReLU	Frozen
	2x2 Max Pooling, Stride 2	-
Block 2	conv3-128+ReLU	Frozen
	conv3-128+ReLU	Frozen
	2x2 Max Pooling, Stride 2	-
Block 3	conv3-256+ReLU	Frozen
	conv3-256+ReLU	Frozen
	conv3-256+ReLU	Frozen
	2x2 Max Pooling, Stride 2	-
Block 4	conv3-512+ReLU	Frozen
	conv3-512+ReLU	Frozen
	conv3-512+ReLU	Frozen
	2x2 Max Pooling, Stride 2	-
Block 5	conv3-512+ReLU	Frozen
	conv3-512+ReLU	Frozen
	conv3-512+ReLU	Frozen
	2x2 Max Pooling, Stride 2	-
Flatten	-	-
Fully Connected 1	4096 neurons, ReLU, Drop-out	Not Frozen
Fully Connected 2	4096 neurons, ReLU, Drop-out	Not Frozen
Output	C neurons, Softmax	Not Frozen

Table 4.4: The VGG16 architecture used for classification. The network has five convolutional blocks and an FCN with two hidden layers. The FCN has been randomly initiated and contains no pre-trained weights. The convolutional layers are denoted as 'conv⟨kernel size⟩−⟨number of kernels⟩'. The activation function for each layer is ReLU. The fully connected layers are regularized through drop-out with a probability depending on snow/no snow or snow/wet/dry classification. The input is an RGB image with size $224 \times 224 \times 3$, and the output is C probabilities, where C is the number of classes. The column to the right shows which layers are frozen during fine-tuning.

The network comprises five convolutional blocks and an FCN with two hidden layers. Each convolutional block contains convolutional layers with ReLu functions and a max pooling layer. The fully connected layers are not pre-trained and are the only layers that are not frozen. These layers are regularized with drop-out, and the weights are restricted to having a norm smaller than 4. The drop-out rate was set to 80% for snow/no snow classification and 70% for snow/wet/dry classification.

The images were pre-processed by being resized to $224 \times 224 \times 3$ and changed from RGB to BGR. The images were then normalized with mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$ for each channel, respectively.

The model was trained for 20 epochs with a batch size of 32. The optimizer used was Adam, with a learning rate 10^{-3}.

4.2.3 Fine-tuning Vision Transformer

A. Abdelraouf et al. [6] demonstrated the effectiveness of fine-tuning a vision transformer as a robust road condition classifier. The network used in this thesis underwent the same pre-training as the one used by Abdelraouf et al. The network underwent a two-step pre-training process, starting with pre-training on the ImageNet-21K dataset, encompassing a vast collection of 14 million images spanning 21,000 classes [40]. Subsequently, fine-tuning was performed on the ImageNet 2012 dataset, consisting of 1 million images distributed across 1,000 classes.

In the preprocessing stage, the images were resized to $224 \times 224 \times 3$ pixels and normalized with mean values of $(0.5, 0.5, 0.5)$ and standard deviations of $(0.5, 0.5, 0.5)$. The model processes a sequence of patches, each measuring $16 \times 16 \times 3$ pixels.

To facilitate the processing of these patches, a convolutional layer was employed, equipped with a kernel size of 16×16, a stride of 16, and 768 kernels. Each patch was linearly embedded into a one-dimensional vector of size 768, and this dimensionality was preserved throughout the encoder.

The encoder comprised 12 layers, each featuring a multi-headed attention mechanism of 12 attention heads. The fully connected networks (FCNs) responsible for generating queries, keys, and values had 768 neurons each. Attention was computed using a scaled dot-product mechanism. Within each encoder, an MLP (Multi-Layer Perceptron) was implemented, housing one hidden layer with 3072 neurons and utilizing the GELU activation function (a 'smoother' ReLU). The output layer of the MLP comprised 768 neurons.

For access to pre-trained weights and detailed implementation specifications, see [41]. The encoder's output was subsequently channeled through a single linear, fully connected layer responsible for classification.

In the fine-tuning stage, only the output layer underwent training, while the remainder of the model remained frozen. The training was conducted over 5 epochs, employing a batch size of 32. The Adam optimizer, configured with weight decay and a learning rate of 10^{-3}, was utilized for training.

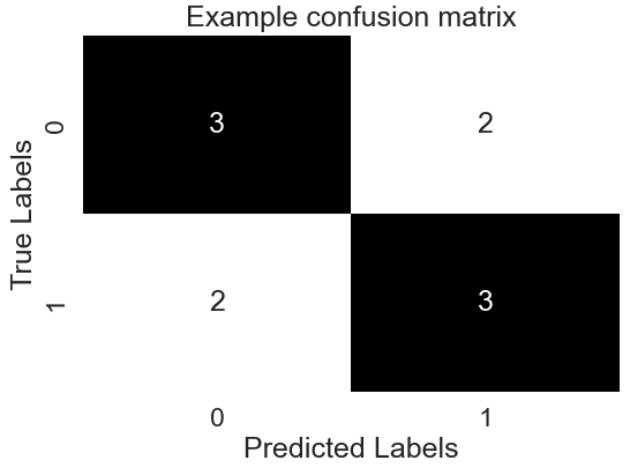

Figure 4.1: Example of a confusion matrix. In the example, there are two classes and 10 samples. The classifier has correctly classified 6 samples and mistakenly classified 2 instances of class '0' as class '1' and vice versa.

4.3 Evaluation of Performance

For the U-net, the evaluation metric used was the average intersection over union (IoU) achieved over all images in the test data. The performance of deep learning classifiers was measured in the mean accuracy on the test data. For the conventional method, a cross-validation scheme is used. This allows for both a mean accuracy over all data and a standard deviation across the folds to be estimated.

In addition to the mean accuracies, confusion matrices were also evaluated. Confusion matrices visualize the performance of a model. The matrix shows how often a model has incorrectly classified each class and common mistakes made. Figure 4.1 shows an example of a confusion matrix.

All classifiers evaluated can produce a probability for the predicted label. This is interesting to investigate since a model that achieves a high mean accuracy but predicts with a high probability when the prediction is wrong is inherently unreliable. A prediction with low probability is also not useful. A prediction of 'snow' with a probability 50% when classifying snow/no snow would be equivalent to the model guessing. A good model would predict high probabilities when predicting correctly and low probabilities when predicting incorrectly. To evaluate this, the distributions of the predicted probabilities for correct and incorrect predictions are investigated.

5
Results

This chapter presents the results of classifying snow/no snow and snow/dry/wet. The chapter is divided into two parts: results from the conventional method and results from deep learning method. The conventional results include image segmentation metrics, the performance of the classifiers on the extracted datasets, and results from feature importance. The deep learning results include the performance of the fine-tuned VGG16 network and vision transformer.

5.1 Conventional Method

This section presents the results of the conventional method. It begins with metrics regarding the U-net's performance. This is followed by the results from classifying snow/no snow and snow/wet/dry.

5.1.1 Road Segmentation

Figure 5.1 shows the loss on the training and test data during training. The loss of both steadily decreases, indicating no overfitting. The training loss is more unstable than the test loss. This is an expected result when applying random batch-wise data augmentations. Since the training data is not the same for each epoch, there is a chance that one epoch can be more difficult to classify than the next. The test loss reaches around 0.08.

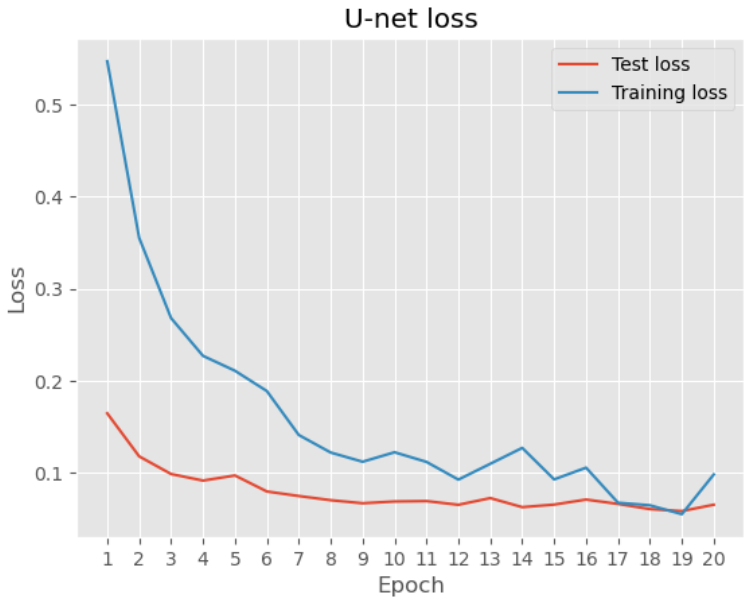

Figure 5.1: The evolution of the loss during training of the U-net. The training and test losses are shown for each epoch of training.

The U-net achieved an IoU of 0.9132 on average for the images in the test data. An example of a segmented road can be seen in Figure 5.2.

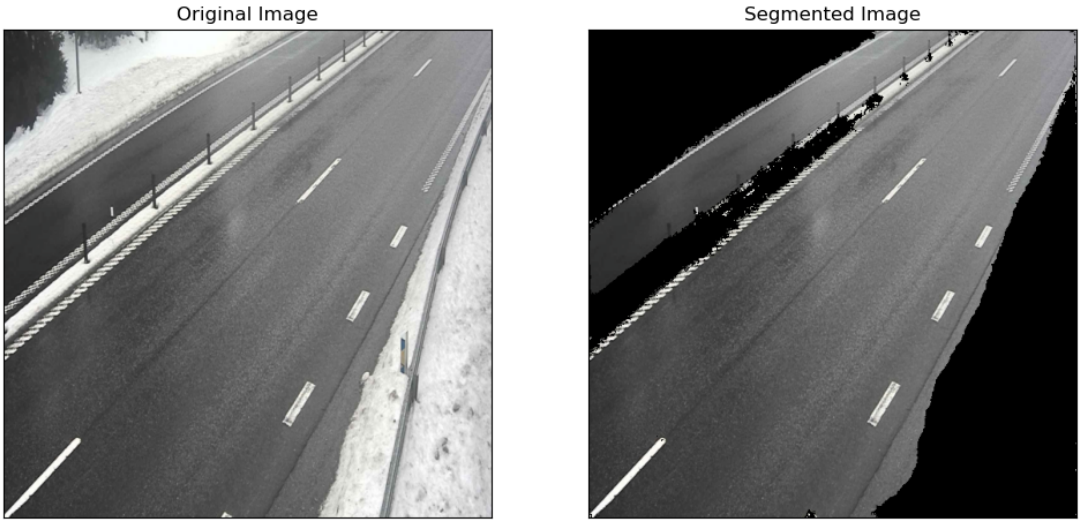

Figure 5.2: Example of the segmentation of a road. All pixels not part of the road are set to -1 so as not to be confused with parts of the road.

5.1.2 Snow/No Snow

This section presents results from classifying snow/no snow using the conventional method. Table 5.1 shows the mean accuracy and standard deviation of all classifiers on all datasets. The best-performing classifier for each dataset is marked in bold.

Classifier	dataset		
	GLRLM	PD basic	Persistent images
FCN	$76.1\% \pm 5.7$	$76.3\% \pm 4.9$	$\mathbf{73.4\% \pm 3.7}$
SVM	$\mathbf{77.6\% \pm 3.7}$	$\mathbf{76.3\% \pm 3.4}$	$67.5\% \pm 3.2$
Naive Bayes	$52.7\% \pm 9.6$	$49.1\% \pm 7.6$	$58.7\% \pm 7.2$
Random Forest	$61.6\% \pm 5.9$	$66.3\% \pm 4.7$	$71.4\% \pm 1.9$

Table 5.1: The classifier's performance on each dataset for classifying snow/no snow. The mean accuracy is presented with the standard deviation. The performance was evaluated using cross-validation with the data split according to CCTV stations. The performance of the best classifier on each dataset is marked with bold characters.

The best performance is applying a support vector machine to the GLRLM dataset. The achieved accuracy is 77.6% with a standard deviation 3.7. The PD basic is a close second with an accuracy of 76.3%.

The two datasets extracted through persistent homology perform well, especially the PD basic dataset combined with a support vector machine. This indicates that persistent homology produces features relevant to the classification.

Figure 5.3 shows the confusion matrix of the SVM's predictions on the GLRLM dataset. The model tends to misclassify 'snow' images more often than 'no snow'.

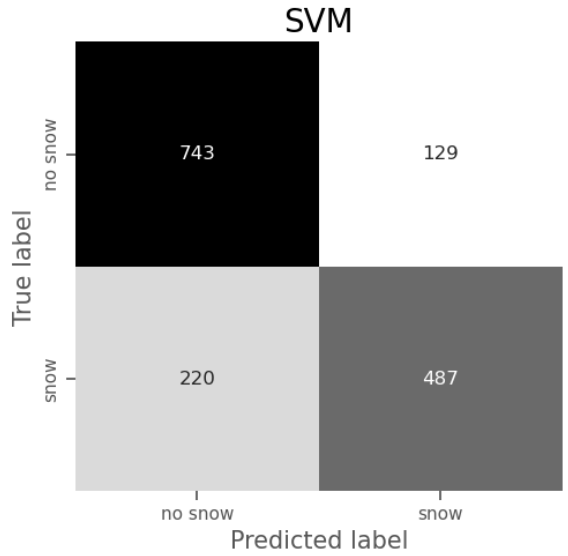

Figure 5.3: The confusion matrix for the support vector machine on the GLRLM dataset.

Figure 5.4 shows the distribution of predicted probabilities for the SVM and the GLRLM dataset. To the left are the probabilities when the model is correct, and to the left are the probabilities when predicting the wrong class.

When the SVM is correct, it tends to be relatively sure, and when it is wrong, it is not always certain that it is correct. The distributions are similar for both classes, but as noted in Figure 5.3 the SVM struggles more with classifying 'snow' than 'no snow'.

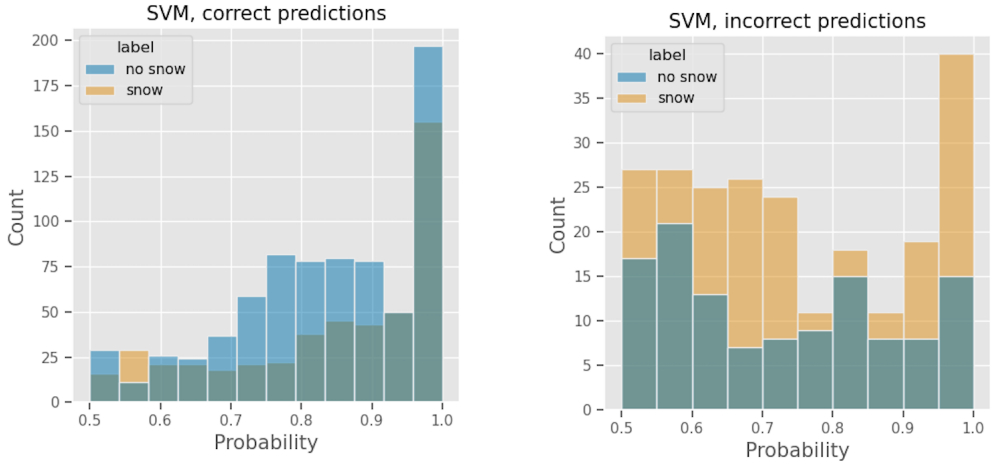

Figure 5.4: The probabilities predicted by the support vector machine for each class. To the right is shown the distribution when the SVM predicts correctly, and to the left is for incorrect predictions. The labels refer to the true label.

Some of the datasets contain many and often correlated features. This is especially true for the GLRLM and Persistent images datasets. The GLRLM contains, for instance, the short run and long run emphasis, these are naturally highly correlated. This means there is a redundancy among the features, and reducing the number of features might be beneficial. Table 5.2 shows the optimal number of features and the achieved accuracy according to the ANOVA-F and mutual information metrics.

There are no significant increases in accuracy from reducing the number of features, but reducing the number of features can result in similar performance and a lighter computational cost. For example, choosing the 13 most important features of the PD basic dataset according to ANOVA-F results in a similar performance as using all 45 features from the GLRLM dataset.

ANOVA-F			
Classifier	**dataset**		
	GLRLM	PD basic	Persistent images
FCN	77.2% ± 4.8 (41/45)	76.6% ± 4.5 (15/23)	**73.4% ± 3.7 (63/63)**
SVM	**77.6% ± 3.7 (45/45)**	**77.0% ± 2.7 (13/23)**	67.5% ± 3.2 (63/63)
Naive Bayes	60.0% ± 6.1 (3/45)	60.8% ± 6.1 (9/23)	58.9% ± 7.0 (53/63)
Random Forest	61.7% ± 5.9 (40/45)	67.6% ± 3.2 (13/23)	71.5% ± 1.9 (63/63)

Mutual Information			
Classifier	**dataset**		
	GLRLM	PD basic	Persistent images
FCN	77.1% ± 4.0 (28/45)	76.3% ± 4.9 (23/23)	**73.4% ± 3.7 (63/63)**
SVM	**77.6% ± 3.7(45/45)**	**76.3% ± 3.4 (23/23)**	67.5% ± 3.2 (63/63)
Naive Bayes	58.2% ± 6.0 (1/45)	60.3% ± 3.6 (9/23)	60.0% ± 6.5 (21/63)
Random Forest	63.2% ± 5.7 (21/45)	70.4% ± 5.2 (3/23)	71.9% ± 2.5 (56/63)

Table 5.2: Tables showing the best number of features according to the ANOVA-F and mutual information metrics when classifying snow/no snow. The performance is evaluated using cross-validation. The top table shows the optimal number of features according to ANOVA-F and the bottom according to mutual information. Presented in the tables is the mean accuracy and the standard deviation for each dataset and classifier. The number of features is shown as (Number of features selected/Total number of features). The best-performing dataset is marked with bold characters.

5.1.3 Snow/Wet/Dry

This section presents results from classifying snow/wet/dry using the conventional method. Table 5.3 shows the mean accuracy and standard deviation of all classifiers on all datasets. The best-performing classifier for each dataset is marked in bold.

Classifier	dataset		
	GLRLM	PD basic	Persistent images
FCN	$65.9\% \pm 5.5$	$\mathbf{67.4\% \pm 3.6}$	$\mathbf{64.2\% \pm 2.3}$
SVM	$\mathbf{66.1\% \pm 5.9}$	$65.3\% \pm 3.5$	$59.8\% \pm 2.8$
Naive Bayes	$53.4\% \pm 3.5$	$45.4\% \pm 5.1$	$49.1\% \pm 4.6$
Random Forest	$56.1\% \pm 4.6$	$59.9\% \pm 4.4$	$60.4\% \pm 2.8$

Table 5.3: The classifiers' performance on each dataset for classifying snow/wet/dry. The mean accuracy is presented with the standard deviation. The performance was evaluated using cross-validation with the data split according to CCTV stations. The performance of the best classifier on each dataset is marked with bold characters.

The best performance is achieved using a fully connected neural network on the PD basic dataset. The confusion matrix from this case can be seen in Figure 5.5.

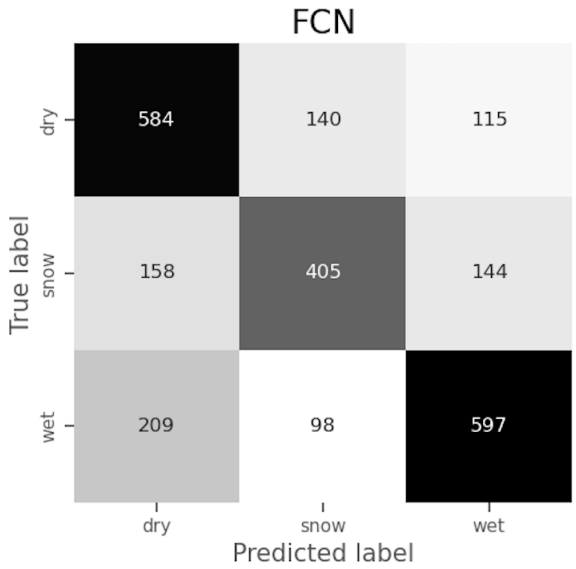

Figure 5.5: The confusion matrix for the fully connected neural network on the PD basic dataset.

Figure 5.6 shows the distributions of the predicted probabilities over each class. The distributions come from the FCN's predictions. It is important to note that there are some differences in the sizes of the classes. Therefore, it is only relevant to look at the overall shape of the distributions for each class and not pay too much attention to the differences in the heights of specific stacks. That said, the FCN is more confident when predicting wet conditions correctly, compared to other classes. When the FCN makes incorrect predictions, it is generally less confident. This is important since this makes the model reliable.

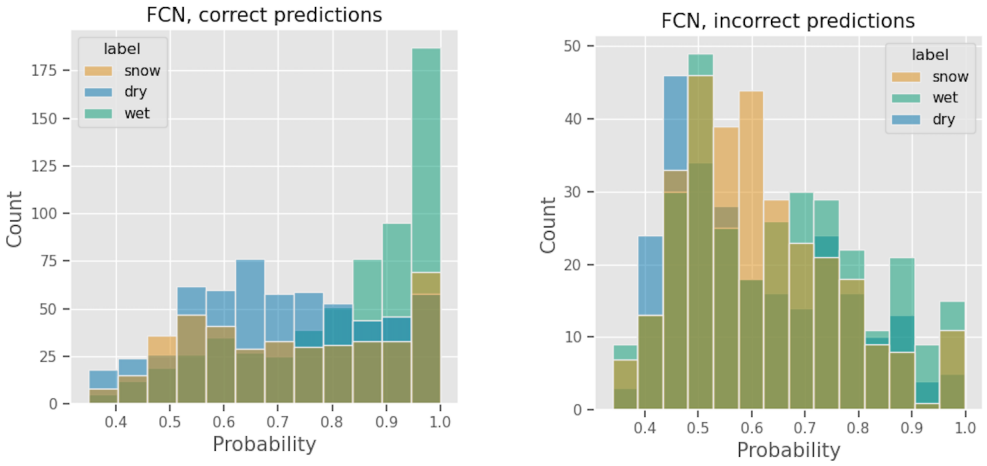

Figure 5.6: The probabilities predicted by the support vector machine for each class. To the right is shown the distribution when the FCN predicts correctly, and to the left is for incorrect predictions. The label refers to the true label.

Table 5.4 shows the optimal number of features for each classifier and dataset according to ANOVA-F and mutual information rankings. There are no significant increases in performance. The best option is still the FCN combined with the PD basic. Other combinations, such as FCN with the GLRLM dataset, perform similarly, but these require more features and are, therefore, more computationally heavy.

ANOVA-F			
Classifier	**dataset**		
	GLRLM	PD basic	Persistent images
FCN	**67.4% ± 4.4 (35/45)**	**67.4% ± 3.6 (23/23)**	**64.2% ± 2.3 (63/63)**
SVM	66.8% ± 6.1 (33/45)	65.6% ± 2.9 (20/23)	59.8% ± 2.8 (63/63)
Naive Bayes	53.4% ± 3.5 (44/45)	49.4% ± 4.5 (17/23)	51.5% ± 3.0 (8/63)
Random Forest	56.3% ± 4.1 (34/35)	60.8% ± 4.8 (18/23)	60.4% ± 2.8 (63/63)

Mutual Information			
Classifier	**dataset**		
	GLRLM	PD basic	Persistent images
FCN	**67.6% ± 4.4 (42/45)**	**67.4% ± 3.6 (23/23)**	**64.3% ± 1.9 (51/63)**
SVM	66.3% ± 5.0 (41/45)	66.2% ± 2.8 (22/23)	61.1% ± 2.4 (61/63)
Naive Bayes	53.5% ± 3.6 (42/45)	49.4% ± 4.7 (15/23)	51.3% ± 4.3 (45/63)
Random Forest	56.1% ± 4.6 (45/45)	61.2% ± 3.0 (6/23)	60.4% ± 2.8 (63/63)

Table 5.4: Tables showing the best number of features according to the ANOVA-F and mutual information metrics when classifying snow/wet/dry. The performance is evaluated using cross-validation. The top table shows the optimal number of features according to ANOVA-F, and the bottom according to mutual information. Presented in the tables are the mean accuracy and the standard deviation for each dataset and classifier. The number of features is shown as (Number of features selected/Total number of features). The best-performing classifier for each dataset is marked with bold characters.

5.2 Deep Learning Method

This section presents the results from classifying snow/no snow and snow/wet/dry using the deep learning methods.

5.2.1 Snow/No Snow

This section presents the results from classifying snow/no snow using the deep learning method. Figure 5.7 shows the training and test loss for the VGG16 network and vision transformer during fine-tuning. Both models converge steadily, with the vision transformer reaching a lower loss.

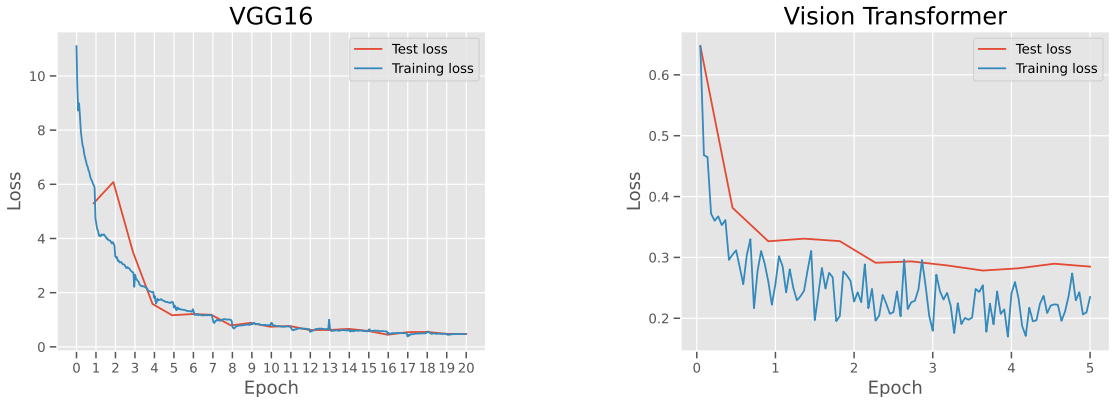

Figure 5.7: The training and test loss during fine-tuning of the VGG16 network and vision transformer for classifying snow/no snow. The VGG16 was trained for 10 epochs, with training loss evaluated every 100th batch and test loss calculated after every epoch. The vision transformer was trained for 5 epochs, with the loss calculated every 10th batch for the training loss, and every 100th batch for the test loss.

Table 5.5 shows the mean accuracy on the test data of the VGG16 network and vision transformer. The vision transformer performs significantly better than the VGG16 with an achieved accuracy of 87.9%.

Model	**Mean Accuracy**
VGG16	81.0%
Vision transformer	87.9%

Table 5.5: Mean accuracy for the fine-tuned VGG16 and vision transformer on the test dataset when classifying snow/no snow.

Figure 5.8 shows the confusion matrices for the VGG16 and vision transformer. Both models tend to misclassify 'snow' images more than 'no snow'.

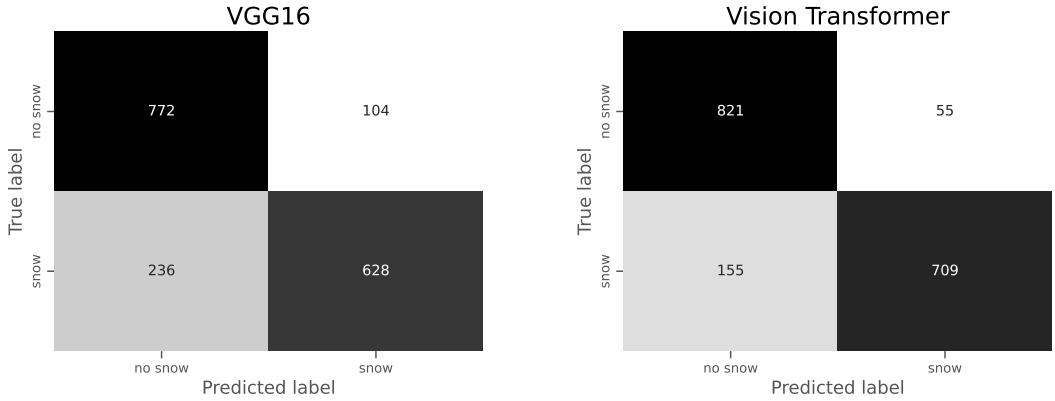

Figure 5.8: Confusion matrices for the VGG16 network and vision transformer when classifying snow/no snow.

Figure 5.9 shows the distributions of the probabilities predicted for each class. The VGG16 network is more confident that images are 'no snow'. When correct, the VGG16 shows an almost uniform probability distribution for 'snow'. This would mean that few 'snow' images would be predicted with a high enough probability to be useful. This is a big problem with the VGG16 since correctly classifying 'snow' with high probability is of more interest than correctly classifying 'no snow'.

The vision transformer shows similar distributions for both classes, but when incorrect, it is more confident when misclassifying 'snow' images.

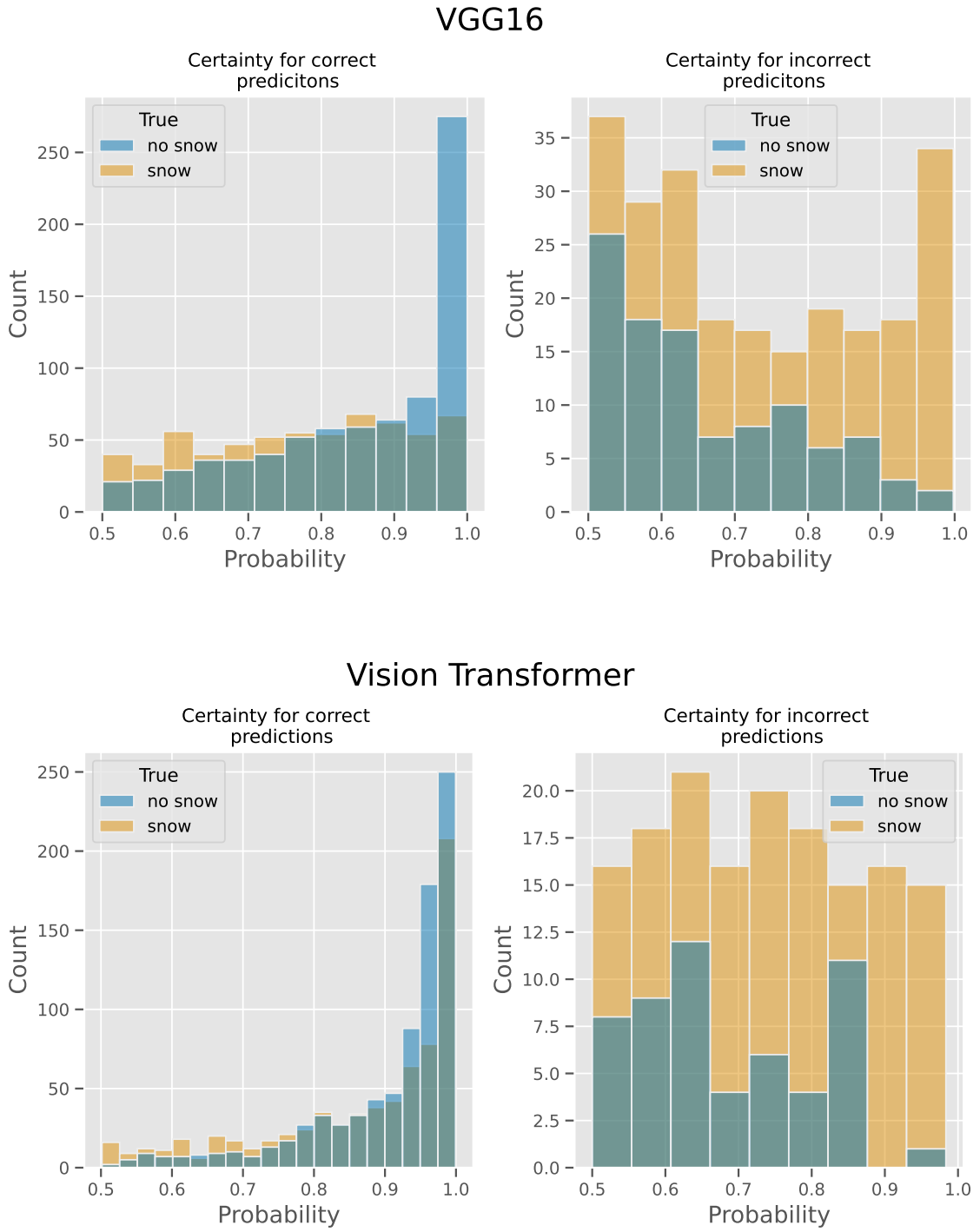

Figure 5.9: Distributions of probabilities of predictions. The top row shows the distributions predicted by the VGG16, and the bottom row is for the vision transformer. The left column is for correct predictions, and the right column is for incorrect predictions. The labels refer to the true labels.

To try and understand the vision transformer better, further investigation was taken into what might cause the vision transformer to predict incorrectly. To do this, the distributions of predicted probabilities over day and night were investigated.

The model did not show any sign of struggling more with predicting images taken during the day or night. The distributions of predicted probabilities can be found in Appendix B.

Next, images, where the vision transformer predicted with a probability higher than 90% but was still wrong, were examined. One way to make the predictions of a vision transformer more interpretable is to look at the attention produced. Specifically, it is of interest to look at the attention generated in the final encoder layer for the classification token. By doing this, it is possible to create a map of what regions of the image are considered more important for the final classification. Some example images can be seen in Figure 5.10.

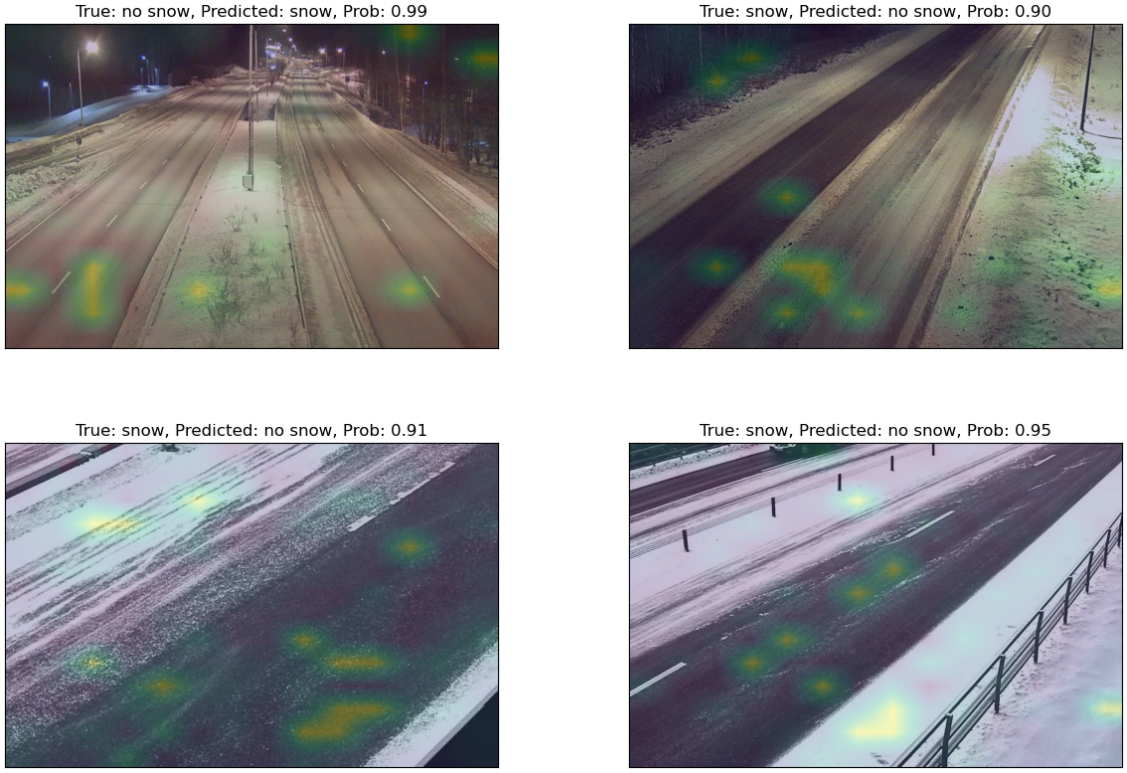

Figure 5.10: Four examples where the vision transformer predicts incorrectly when predicting snow/no snow. The examples shown are cases where the true road condition is clear. The attention heat map produced from the last attention layer is overlaid with the images. Brighter regions indicate regions where the model places larger attention. Each image is labeled with the predicted and true label, and the probability for the prediction.

5.2.2 Snow/Wet/Dry

This section presents the results from classifying snow/wet/dry using the deep learning methods. Figure 5.11 shows the training and test loss during the fine-tuning of the VGG16 network and the vision transformer. Both models converge steadily but the vision transformer reaches a lower loss than the VGG16.

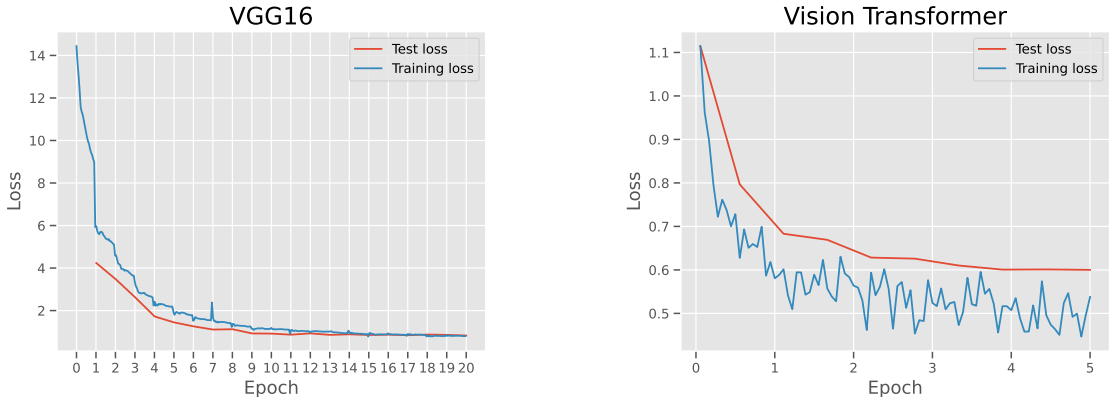

Figure 5.11: The training and test loss during fine-tuning of the VGG16 and vision transformer for classifying snow/wet/dry. The VGG16 was trained for 20 epochs with the training loss being evaluated every 10th batch, and test loss being evaluated at the end of each epoch. The vision transformer was trained for 5 epochs, with the loss calculated every 10th batch for the training loss, and every 100th batch for the test loss.

Table 5.6 shows the mean accuracy of the VGG16 and vision transformer on the test dataset. The vision transformer performs best, achieving a mean accuracy of 75.3%.

Model	**Mean Accuracy**
VGG16	65.7%
Vision Transformer	75.3%

Table 5.6: Mean accuracy for the fine-tuned VGG16 and vision transformer on the test dataset when classifying snow/wet/dry.

Figure 5.12 shows the confusion matrices for the VGG16 and vision transformer. Both models show similar matrices but the VGG16 missclassifies 'dry' images significantly more.

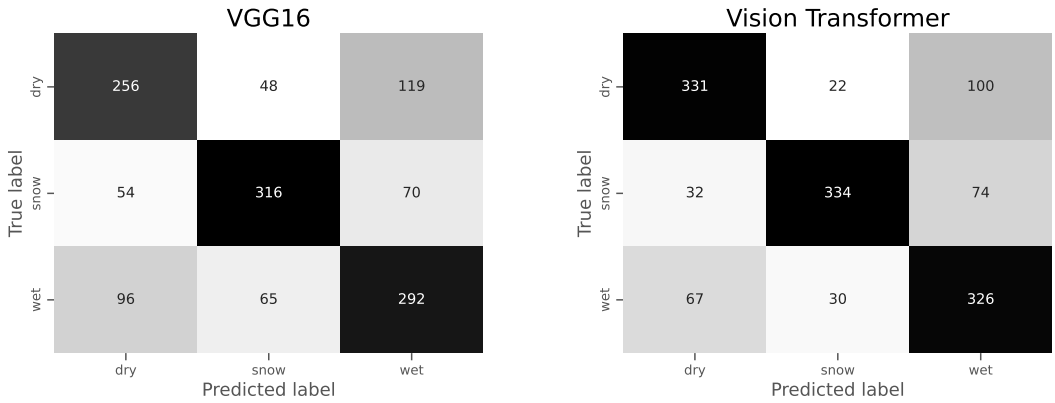

Figure 5.12: Confusion matrices for the VGG16 network and vision transformer when classifying snow/wet/dry.

Figure 5.13 shows the distributions of the predicted probablities for each class. For the VGG16, the distributions for correct and incorrect predictions are very similar. It is generally only about 70% sure when predicting any class. The vision transformer on the other hand is more certain when predicting 'snow' than other classes. It is also less confident when predicting the wrong class than the VGG16.

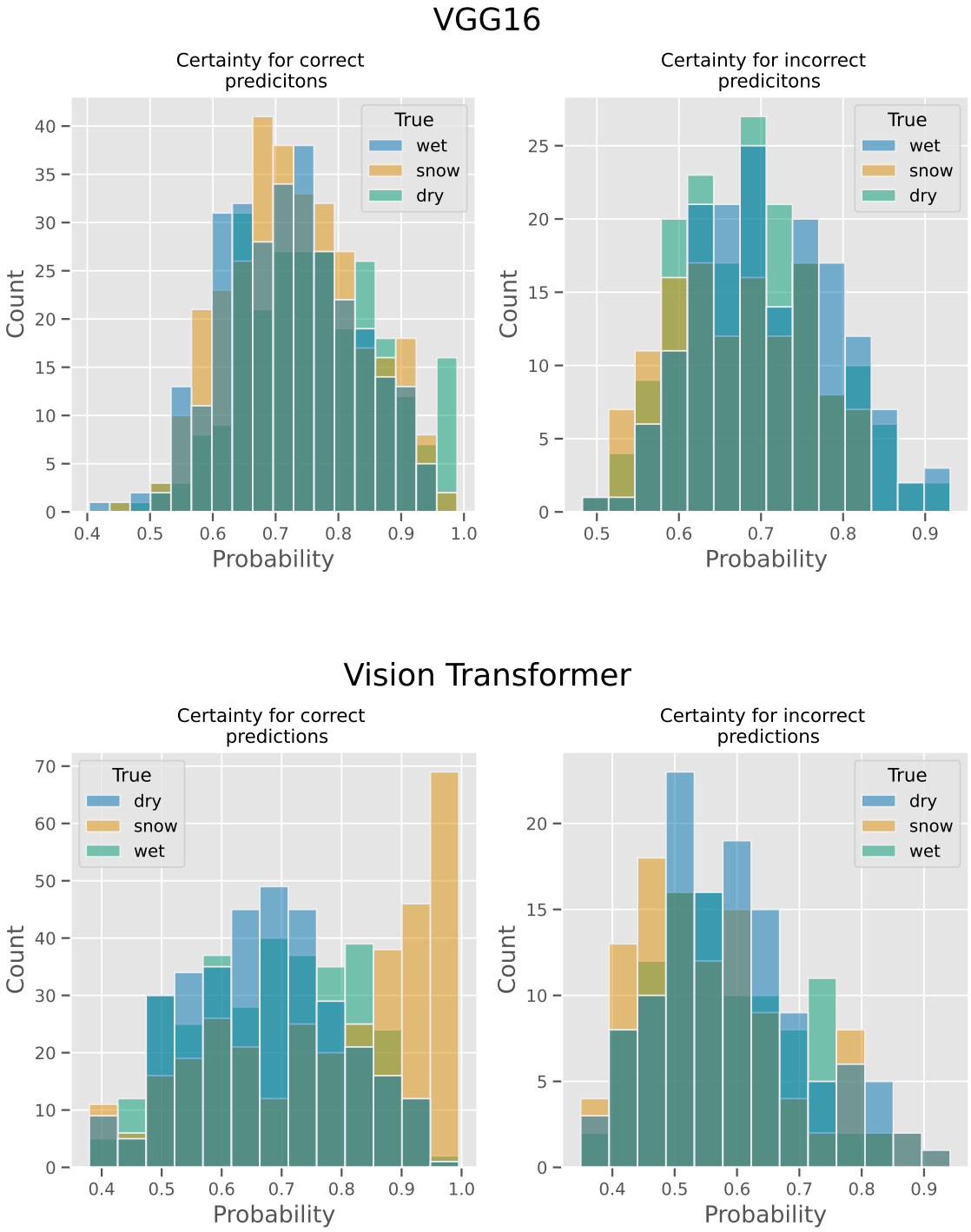

Figure 5.13: Distributions of probabilities of predictions. The top row shows the distributions predicted by the VGG16, and the bottom row is for the vision transformer. The left column is for correct predictions, and the right column is for incorrect predictions. The labels refer to the true labels.

Due to the vision transformer's superior performance, it is of interest to see why it might predict the wrong class for images. To do this, the distributions of predicted probabilities were plotted, and the attention heatmap from the last encoder was overlaid with the original images. The probability distributions did not show any

decrease in certainty depending on the image being taken during the day or night. The distributions are found in Appendix B.

As for the attention heatmap, a clear pattern for confident misclassifications is difficult to find. Some examples of cases where the model predicts incorrectly on instances where the road condition is evident can be found in Figure 5.14.

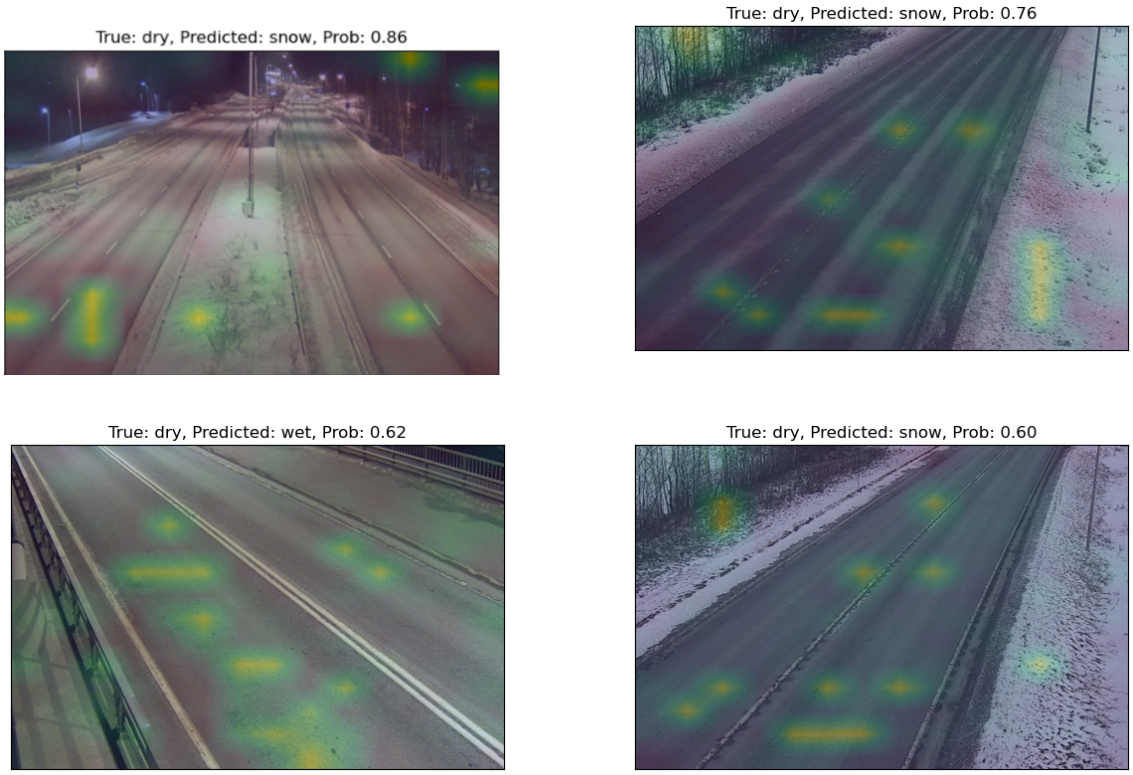

Figure 5.14: Four examples where the vision transformer predicts incorrectly when predicting snow/no snow. The examples shown are cases where the true road condition is clear. The attention heat map produced from the last attention layer is overlaid with the images. Brighter regions indicate regions where the model places larger attention. Each image is labeled with the predicted and true label, and the probability for the prediction.

6
Discussion

This chapter discusses the advantages and disadvantages of the conventional and deep learning methods. It begins by discussing the difficulties of training a reliable segmentation model. It then discusses the conventional method with a separate section for snow/no snow and snow/wet/dry classification. This is followed by a similar section discussing the deep learning method. The chapter ends with a general discussion.

6.1 Conventional Method

This section discusses the results of the conventional method.

6.1.1 Road Segmentation

A key component of the conventional method is a good model for segmenting the road in each image. If the road is not well segmented, the features extracted from the road will naturally suffer in quality and become noisy.

The biggest challenge in training a good model for road segmentation is to collect a dataset with enough variety in road layout. The U-net presented in Section 5.1.1 is trained on images taken from Swedish CCTV stations. Most of these images consist of a road in the center of the image, which runs straight through the image. These are the kinds of images where the U-net performs best. In images where the road is not centered or follows an unusual layout, such as a roundabout, the U-net performs worse. Figure 6.1 shows two examples. The left image is similar to the training data, and the U-net performs well. The U-net struggles with the right image because of the hole between the roads.

The difficulty in training a reliable image segmentation model makes the conventional method difficult to scale and generalize to new CCTV stations. To be sure of the segmentation results, it would be necessary to check the produced segmentation for each new station manually. Expanding the training data and retraining the model when new regions are added might also be necessary. This would, however, eliminate the purpose of having a model. It might just be more practical to segment every road by hand.

Figure 6.1: Example images of where the U-net typically performs well and where it performs worse. To the left is an image where the image typically performs well, and to the right is an image where the U-net struggles. The reason for the worse performance in this case comes from the hole between the roads. This is confused for road and becomes part of the segmented road.

6.1.2 Snow/No Snow

The conventional method was a reliable method of achieving decent performance and generalization ability. The extraction of features using grey level run-length matrices and persistent homology results in similar performance. Table 5.1 shows the mean accuracies and standard deviations of all classifiers on all datasets.

Texture analysis offers various methods for extracting features from images, with grey level run-length being just one. Numerous studies have demonstrated that features extracted using grey level run-length matrices may exhibit inferior performance compared to alternative techniques [31]. In a paper by Z. Sun and K. Jia [4], co-occurrence, another texture analysis technique, was employed with promising results. This suggests the possibility of enhancing the classification performance for distinguishing between snow and no snow through further exploration and refinement of texture analysis methods.

The best-performing persistent homology dataset was the PD basic dataset. There are many potential reasons why this performed better than the persistent images. The first and most likely reason is that the persistent image is meant to be treated as an image. Selecting the pixels with the most variance can lead to a loss of information, especially spatial information about the pixels' relations to each other. The performance of the PD basic dataset, however, suggests that features based on persistent homology are useful and can be a powerful tool in this type of classification.

Figure 5.3 illustrates the confusion matrix, revealing that 'snow' images are generally more challenging to classify accurately. Ideally, the reverse should be true, given the higher stakes associated with misclassifying 'snow' images in road maintenance and safety. The increased difficulty in classifying 'snow' images may arise from close-call instances featuring minimal snow coverage on roads.

Figure 5.4 displays the predicted probabilities. Notably, the model does not assign higher probabilities to any specific class when predicting correctly. This is advantageous as it ensures that no particular class suffers from uncertainty during accurate predictions. In instances of incorrect predictions, the model tends to exhibit greater uncertainty. This characteristic is preferable, as lower probabilities associated with incorrect predictions could lead to a classification of 'unknown' rather than an incorrect label like 'snow'. In practical implementation, this would mean that the model is more likely to withhold predictions with low probabilities, mitigating potential misclassifications.

Table 5.2 presents the outcomes of determining the optimal number of features for each dataset and classifier. However, it is crucial to acknowledge certain limitations in the method employed for feature selection. A key drawback is that the classifiers are fine-tuned based on datasets with all features, resulting in an inherent bias favoring the full dataset's performance over reduced datasets. Additionally, the use of filtering methods for feature selection doesn't consider feature combinations, indicating untapped potential for further feature reduction beyond what has been explored in this thesis. The decision to adopt the method presented here ultimately hinges on computational cost considerations.

Notwithstanding these limitations, the feature selection method demonstrates that leveraging the top 13 features identified by ANOVA-F from the PD basic dataset can yield commendable performance. This finding is noteworthy, as a reduced feature set reduces computational costs during classifier training and inference. Consequently, the reduced PD basic dataset emerges as the preferred option for classifying snow/no snow using the conventional method.

6.1.3 Snow/Wet/Dry

This section addresses the performance of the conventional method for classifying snow/wet/dry and presents the subsequent results. Given the overlap with the classification of snow/no snow, this discussion will be less detailed compared to Section 6.1.2. While the flaws of the feature selection method are not discussed here, they can be found in Section 6.1.2.

Table 5.3 displays the mean accuracy and standard deviation of all classifiers across various datasets. The highest performance is observed with a fully connected neural network on the PD basic dataset. Examining the confusion matrix in Figure 5.5, it is evident that no specific class poses a greater challenge in terms of classification.

Figure 5.6 illustrates the probabilities of predictions. An intriguing result is the elevated probabilities when correctly classifying wet conditions. This outcome aligns with the methodology of persistent homology. Wet road conditions are primarily identified through bright reflections of car lights or street lamps. If we conceptualize these images as surfaces, the light reflections create high peaks on the surface. Persistent homology focuses on these peaks, which are isolated connected components. Their persistence is reflected through the 0th homology group, representing the connected components, while their boundaries, 1-dimensional holes, are captured through the 1st homology group. This capability enables persistent homology to capture highly relevant information about wet road conditions. When it comes

to classifying dry and snowy conditions, the model is less certain. This might be because these conditions create more homogeneous images and surfaces.

6.2 Deep Learning Method

This section discusses the implementation and results of the deep learning method.

6.2.1 Snow/No Snow

The deep learning models show promising results when classifying snow/no snow. The vision transformer performs significantly better than the VGG16 network. This is in accordance with the results presented by A. Abdelraouf et al. [6].

Several factors influence the final performance of deep learning models. When fine-tuning is performed, one key consideration is the dataset used for pre-training. If the pre-training task differs too much from the fine-tuned task, then the patterns learned during pre-training might not be relevant to the fine-tuned task.

The VGG16 was pre-trained on the places365 dataset in accordance with the method presented by K. Ozcan et al. [5]. This dataset contains images of scenes with the intention of teaching the model to recognize traits that all images of a certain scene have in common. It is logical to use this dataset for pre-training when fine-tuning the model for road condition classification since the goal is to classify a scene as either 'a snowy road' or 'a clear road'.

The vision transformer was pre-trained on the ImageNet-21K dataset and then fine-tuned on the ImageNet 2012 dataset. Although not presented as a result in this thesis, an experiment where the vision transformer was only pre-trained on the ImageNet-21K dataset was run. Fine-tuning this model resulted in a significantly worse performance than the one first fine-tuned on the ImageNet 2012 dataset. This further emphasizes the importance of choosing a good dataset for pre-training.

It would be interesting to see what performance could be achieved with a vision transformer pre-trained on the Places365 dataset. However, to the author's knowledge, there are no available pre-trained models.

Another important factor for the final performance of a model is the hyperparameters. An important part of finding the best model is to tune the hyperparameters. Tuning the hyperparameters is a computationally costly procedure involving training the model several times with different hyperparameters. Due to this, no extensive tuning was possible. Instead, only a few different values for the learning rate were tested. The impressive performance of the vision transformer without a proper hyperparameter search shows a potential for an even better performance.

Both the VGG16 and vision transformer find it more difficult to predict 'snow' images correctly. This can be seen in the confusion matrices found in Figure 5.8. This might be for the same reason mentioned in Section 6.1.2: there are many images with just a small amount of snow on the road.

The distributions of the predicted probabilities can be seen in Figure 5.9. The vision transformer shows an almost ideal distribution. When it predicts correctly, it generally predicts a high probability, and when it predicts incorrectly, the probabilities generally lower. It would be preferable to see a distribution for the incorrect

images with a peak around the lower probabilities. In a practical implementation, this would result in incorrect predictions having a too low probability to be useful, and they would be disregarded. Incorrect predictions are, however, difficult to avoid because of the many cases of snowy roads where the snow level is so low that even a person might misclassify the image.

The VGG16 seems to struggle more with these close-call instances. It shows a higher uncertainty when classifying 'snow' rather than 'no snow'. This is a highly unwanted pattern since it is more important to correctly classify 'snow' images with a high probability than 'no snow'.

Figure 5.10 shows images were the model is wrong but predicts with a high probability for obvious cases. Most images are cases where the road condition is 'snow', but the amount of snow is relatively low. There are also some mistakes made where the road condition is clear, but the model still predicts wrong. Some examples can be found in Figure 5.10. These errors often come down to factors such as bright asphalt, snow covering the sides of the road, and the snow being positioned between the lanes. These factors make it difficult for the model to distinguish snow from asphalt and to determine what area is part of the road. This results in the model thinking that snow-covered parts of the road are part of the terrain.

6.2.2 Snow/Wet/Dry

This section discusses the results of classifying snow/wet/dry using the deep learning method. Given the overlap between this section and Section 6.2.1, this section will be less detailed. The general discussion of hyperparameter tuning and choice of the pre-training dataset is covered in Section 6.2.1.

Although no direct comparisons between classifying snow/no snow and snow/wet/dry should be made since the datasets are not the same. It is no surprise that the performance for classifying snow/wet/dry is significantly lower than for classifying snow/no snow. The classification problem is more difficult because of the similarities between wet and dry roads. The difference often comes down to just a darker shade of grey on the asphalt. The number of images in the dataset is also smaller than the one used for classifying snow/no snow. Ideally, the dataset should be larger for a more complex classification problem. This is necessary for the model to learn the more complex patterns of the data.

An interesting result is the distribution of the predicted probabilities for correctly predicted 'snow' images. Figure 5.13 shows that VGG16 does not predict higher probabilities for 'snow' images than the other classes. This is surprising since 'snow' images would be expected to be easier to distinguish. The vision transformer, however, predicts higher probabilities for 'snow' images, suggesting that it can more easily detect snow.

Most errors made by the vision transformer were made for images with 'close-call' conditions where the road condition is not obvious. Some predictions were wrong where the road condition was clear. Why these mistakes were made is difficult, but a key factor seems to be subtle changes in the grey level in the asphalt. These changes are common in distinguishing between dry and wet but can occur for multiple reasons, leading to confusion. Some examples of clear mistakes can be found in Figure

5.14.

6.3 Concluding Discussion

This section aims to discuss general issues and comparisons between the conventional and deep learning method. Direct comparisons should not be made between the performance of the two methods since the datasets used are different. Still, some remarks about the advantages and disadvantages of the methods can be made. This section will first discuss the practicality and generalization of each method. This will be followed by cases where each method might be preferred.

6.3.1 Practicality and Generalization

As mentioned previously, the conventional method is heavily reliant on a good segmentation of the road. The large variety of road layouts makes it difficult to train a road segmentation model that can guarantee good segmentations. This makes the conventional method difficult to scale. An advantage of the conventional method is the small amount of data required to train the classifier to achieve good performance. The computational cost of the conventional method comes from extracting features. Once the features have been extracted, the inference is cheap.

The deep learning method, on the other hand. requires more data to train efficiently, but once it is trained, it can process any image and generalize well to new stations. Deep learning models require a higher computational cost for training and inference than conventional models. They are also hard to interpret, making them unusable in cases where it is important to be able to explain the decision. This can make it difficult to find reasons why the model predicts incorrectly and remove potential biases.

6.3.2 Use Cases

The conventional method is a viable method if the number of CCTV stations is small. In this case, the roads can be segmented by hand, and the classifier can be trained on these CCTV stations specifically. This can result in the classifiers being 'overfitted' to these stations and increasing performance. This does, however, worsen the ability to generalize to new stations. This overfitting might also work with simpler features that are relatively cheap to extract.

This form of overfitting might also allow for the model to accurately predict more complex road conditions, such as wet, ice, and snowy with tracks. This is not explored in this thesis, but the author's understanding of the papers published by P. Jonsson [3] and Z. Sun and K. Jia [4] is that they employed this type of overfitting and were able to produce almost perfect classification performance.

The deep learning method is a better alternative than the conventional method if there is enough data to train it and the number of CCTV stations is large. The minimal amount of data needed to successfully fine-tune a vision transformer is not explored in this thesis, but approximately 9000 images from ca. 80 stations have been shown to be enough for the training of a reliable snow/no snow classifier.

For the practical implementation of a vision transformer, it is necessary to set a limit at which the predicted probability of a road condition is to be considered reliable. In the case of classifying snow/no snow, Figure 5.9 would suggest that a good limit might be around 80%. At this level, the model is relatively sure of its prediction, and most incorrect predictions would not be considered reliable enough. When classifying snow/wet/dry, a good limit might be 70%. This would allow for most 'snow' images to be correctly classified, and few incorrect predictions would be considered reliable. The problem would be that many correct predictions of 'dry' and 'wet' would also be lost. The limit could be lowered, but this would come at the cost of more incorrect predictions being considered reliable.

7
Conclusion and Future Work

This thesis has explored the practicality and performance of previously presented methods for classifying road conditions using images from CCTV stations. The aim was to find the best method and model for classifying snow/no snow and snow/wet/dry road conditions. The method and model need to be accurate, predict high probabilities when correct and low probabilities when incorrect, and generalize to new CCTV stations reliably. The best set of methods has been shown to be deep learning methods, and the best-performing deep learning model has been shown to be a fine-tuned vision transformer. The performance was evaluated on a test set comprising images from CCTV stations not present in the training data. The fine-tuned vision transformer achieved a mean accuracy of 87.9% and 75.3% on the classification tasks snow/no snow and snow/wet/dry, respectively.

During fine-tuning, computational and time restrictions hindered a proper hyperparameter search. This could potentially increase the performance of the model even further. This step is crucial for finding the optimal vision transformer. This step would allow for a benchmark performance useful for future research. It could also lead to an increase in performance.

In further research, many interesting concepts might be worth investigating as alternative models. For instance, there is a temporal relationship between subsequent images taken by a CCTV station. The strength of this relationship depends on how often images are taken, but including temporal information might be able to aid the model in predictions. The complexity of this problem is not investigated in this thesis, and the author is unsure about what model architecture best suits the task. There are several available models for video classification, and maybe these can be useful.

Another interesting relationship that might help the classifier is the spatial relationship between the stations. The idea is that road conditions at CCTV stations near each other are correlated. This was investigated by A. Abdelraouf et al. [6]. They presented a vision transformer that uses spatial awareness to classify different levels of wet roads with good performance. This was done by using CCTV stations positioned along a highway in Florida and using spatial information from their positions to aid the vision transformer. This showed promising results but is unfortunately not directly applicable to the problem explored in this thesis. This thesis explores images from CCTV stations scattered throughout several countries. Since the coordinates of each station are available, it might be possible to create a similar model to the one presented by A. Abdelraouf et al. [6].

This thesis covered the classification of snow/no snow, and snow/wet/dry, and considered these perfectly separable. This meant that an image with only a small

amount of snow would be considered as snowy as an image covered in snow. Many previous studies divided the classes into levels of snow and wet. For instance, Z. Sun and K. Jia [4] classified dry/mild snow/heavy snow, and A. Abdelraouf et al. [6] classified combinations of road conditions and downpour. Dividing the classes into subclasses might create a more difficult classification task for the more subtle subclasses but help the model identify the extreme subclasses better. For instance, if the model is taught to classify dry/mild snow/heavy snow instead of snow/no snow, then the model might struggle to separate dry and mild snow but easily separate heavy snow from dry. This might be beneficial as the most important class to correctly classify is heavy snow.

Bibliography

[1] Federal Highway Administration U.S. Department of Transportation. *How Do Weather Events Impact Roads?* (2023). URL: https://ops.fhwa.dot.gov/weather/q1_roadimpact.htm.

[2] Andreas Kuehnle and Wilco Burghout. "Winter road condition recognition using video image classification". In: *Transportation research record* Vol. 1627.1 (1998), pp. 29–33.

[3] Patrik Jonsson. "Classification of road conditions: From camera images and weather data". In: *2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings.* (2011), pp. 1–6.

[4] Zhonghua Sun and Ke-bin Jia. "Road Surface Condition Classification Based on Color and Texture Information". In: *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2013), pp. 137–140.

[5] Koray Ozcan et al. "Road Weather Condition Estimation Using Fixed and Mobile Based Cameras". In: *Advances in Intelligent Systems and Computing* Vol. 943 (2019), pp. 192–204.

[6] Amr Abdelraouf, Mohamed Abdel-Aty, and Yina Wu. "Using Vision Transformers for Spatial-Context-Aware Rain and Road Surface Condition Detection on Freeways". In: *IEEE Transactions on Intelligent Transportation Systems* Vol. 23.10 (2022), pp. 18546–18556.

[7] Chi Seng Pun, Kelin Xia, and Si Xian Lee. *Persistent-Homology-based Machine Learning and its Applications – A Survey.* (2018). arXiv: 1811.00252.

[8] Batta Mahesh. "Machine learning algorithms-a review". In: *International Journal of Science and Research (IJSR)* Vol. 9 (2020), pp. 381–386.

[9] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. "Supervised machine learning: A review of classification techniques". In: *Emerging artificial intelligence applications in computer engineering* Vol. 160.1 (2007), pp. 3–24.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd. Springer, (2009).

[11] Stuart J. Russell and Peter Norvig. *Artifical Intelligence: A Modern Approach.* 4th. Pearson Education, (2022), pp. 801–839.

[12] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR).* (2015). arXiv: 1412.6980.

[13] George Hinton, Nitish Srivastava, and Kevin Swersky. *rmsprop: Divide the gradient by a running average of its recent magnitude*. (2012). URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[14] Anh H. Reynolds. *Convolutional Neural Networks (CNNs)*. (2019). URL: https://anhreynolds.com/blogs/cnn.html.

[15] Jason Brownlee. *How to Visualize Filters and Feature Maps in Convolutional Neural Networks*. (2019). URL: https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, (2016), pp. 335–339.

[17] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. (2015), pp. 1–14. arXiv: 1409.1556.

[18] Khuyen Le. *An overview of VGG16 and NiN models*. (2021). URL: https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models-96e4bf398484.

[19] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. (2017).

[20] Aston Zhang et al. *Dive into Deep Learning*. Cambridge University Press, (2023). URL: https://D2L.ai.

[21] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. (2021). arXiv: 2010.11929.

[22] Canlin Zhang et al. "Biomedical word sense disambiguation with bidirectional long short-term memory and attention-based neural networks". In: *BMC Bioinformatics* Vol. 20 (2019), p. 502.

[23] Shervin Minaee et al. "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 44.7 (2022), pp. 3523–3542. arXiv: 2001.05566.

[24] Swarnendu Ghosh et al. "Understanding Deep Learning Techniques for Image Segmentation". In: *ACM Comput. Surv.* Vol. 52.4 (2019). ISSN: 0360-0300.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. (2015). arXiv: 1505.04597.

[26] Divyanshu Mishra. *Transposed Convolution Demystified*. (2020). URL: https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba.

[27] Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. 2nd. CRC Press LLC, (2016), pp. 31–34.

[28] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers Electrical Engineering* Vol. 40.1 (2014), pp. 16–28. ISSN: 0045-7906.

[29] Lisa Sullivan. *Hypothesis Testing - Analysis of Variance (ANOVA)*. URL: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_hypothesistesting-anova/bs704_hypothesistesting-anova_print.html.

[30] Brian C. Ross. "Mutual Information between Discrete and Continuous Data Sets". In: *PLOS ONE* Vol. 9.2 (2014), pp. 1–5.

[31] Anne Humeau-Heurtier. "Texture Feature Extraction Methods: A Survey". In: *IEEE Access* Vol. 7 (2019), pp. 8975–9000.

[32] Mary M. Galloway. "Texture analysis using gray level run lengths". In: *Computer Graphics and Image Processing* Vol. 4.2 (1975), pp. 172–179. ISSN: 0146-664X.

[33] A. Chu, C.M. Sehgal, and J.F. Greenleaf. "Use of gray value distribution of run lengths for texture analysis". In: *Pattern Recognition Letters* Vol. 11.6 (1990), pp. 415–419. ISSN: 0167-8655.

[34] What-When-How. *Run Lengths (Biomedical Image Analysis)*. URL: http://what-when-how.com/biomedical-image-analysis/run-lengths-biomedical-image-analysis/.

[35] Larry Wasserman. "Topological Data Analysis". In: *Annual Review of Statistics and Its Application* Vol. 5.1 (2018), pp. 501–532.

[36] Applied Algebraic Topology Network. *Introduction to Cubical Complexes and Persistence - Damiano - 2020*. (2020). URL: https://www.youtube.com/watch?v=r6mu8YnDAHk&ab_channel=AppliedAlgebraicTopologyNetwork.

[37] Henry Adams et al. "Persistence Images: A Stable Vector Representation of Persistent Homology". In: *Journal of Machine Learning Research* Vol. 18.8 (2017), pp. 1–35.

[38] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the International Conference on Machine Learning (ICML)*. (2020). arXiv: 1905.11946.

[39] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[40] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition* (2009), pp. 248–255.

[41] Hugging Face. *Vision Transformer (base-sized model)*. URL: https://huggingface.co/google/vit-base-patch16-224.

[42] Scikit-learn. *Sklearn Documentation*. URL: https://scikit-learn.org/stable/.

[43] Editors of Encyclopaedia Britannica. *homomorphism*. Encyclopedia Britannica, (2008). URL: https://www.britannica.com/science/homomorphism.

[44] Herbert Edelsbrunner and John L. Harer. *Computational topology : an introduction*. American Mathematical Society, (2010), pp. 177–185.

A

Tuning Parameters for Classifiers in the Conventional Method

This section presents the parameters tuned for each classifier used in the conventional method. The parameters are presented in tables along with what values were explored and what parameters were optimal for classifying snow/no snow and snow/wet/dry. Not all parameters for every classifier was tuned, to find information about values, see the default values found in the sci-kit learn documentation [42].

Fully connected neural network					
Parameter	Parameter search space		Selected for each dataset (snow/all)		
	Range	Nr. of samples	GLRLM	PD basic	PI
Learning rate	$[10^{-4}, 10^{-1}]$	30	$10^{-4}/10^{-4}$	$7.0 \times 10^{-3}/1.0 \times 10^{-2}$	$5.9 \times 10^{-2}/1.7 \times 10^{-2}$
Hidden layers (size and number)	$[(100,1),(100,2),(200,1),(200,2)]$	-	$(200,2)/(100,2)$	$(200,1)/(200,1)$	$(100,1)/(200,1)$

Table A.1: Table of the search space for each tuned parameter in the FCN classifier. The tuned parameters are the learning rate and network size. Here the 'Hidden layers' parameter refers to the number of hidden layers and the size of each layer, denoted (size, number of layers). The 'Range' refers to what range of values or what values were explored. The 'Nr. of samples' column refers to the number of values explored in the range. The optimal parameter for each dataset is shown in the 'Selected for each dataset column'. The chosen parameter is shown for both snow/no snow classification and snow/wet/dry classification, these are denoted 'snow' and 'all'. Here 'PI' refers to the 'Persistent images' dataset.

Support Vector Machine					
Parameter	Parameter search space		Selected for each dataset (snow/all)		
	Range	Nr. of samples	GLRLM	PD basic	PI
C	$[0.1, 100]$	30	51.8/10.4	100.0/41.4	86.2/17.3
kernel	[Polynomial, Radial basis function]	-	Poly/Poly	RBF/RBF	RBF/RBF

Table A.2: Table of the search space for each tuned parameter in the SVM classifier. The parameters tuned are the slack variable and the kernel. The kernel was either a polynomial of degree 3 or the radial basis function. The 'Range' refers to what range of values or what values were explored. The 'Nr. of samples' column refers to the number of values explored in the range. The optimal parameter for each dataset is shown in the 'Selected for each dataset column'. The chosen parameter is shown for both snow/no snow classification and snow/wet/dry classification, these are denoted 'snow' and 'all'. Here 'PI' refers to the 'Persistent images' dataset.

Random Forest					
Parameter	Parameter search space		Selected for each dataset (snow/all)		
	Range	Nr. of samples	GLRLM	PD basic	PI
Number of trees	$[1, 300]$	30	101/231	251/211	181/41
Max depth of tree	$[1, 20]$	4	16/16	16/16	11/16
Impurity metric	[Gini,Entropy,Log loss]	-	Entropy/Gini	Gini/entropy	Log loss/Log loss

Table A.3: Table of the search space for each tuned parameter in the random forest classifier. The parameters tuned are the number of trees, the maximum depth of the trees, and the impurity metric. The 'Range' refers to what range of values or what values were explored. The 'Nr. of samples' column refers to the number of values explored in the range. The optimal parameter for each dataset is shown in the 'Selected for each dataset column'. The chosen parameter is shown for both snow/no snow classification and snow/wet/dry classification, these are denoted 'snow' and 'all'. Here 'PI' refers to the 'Persistent images' dataset.

B

Vision Transformers, Day and Night

This section presents the probabilities predicted of the vision transfomer. The distributions shows the probabilities based on whether the image was taken during the night or day.

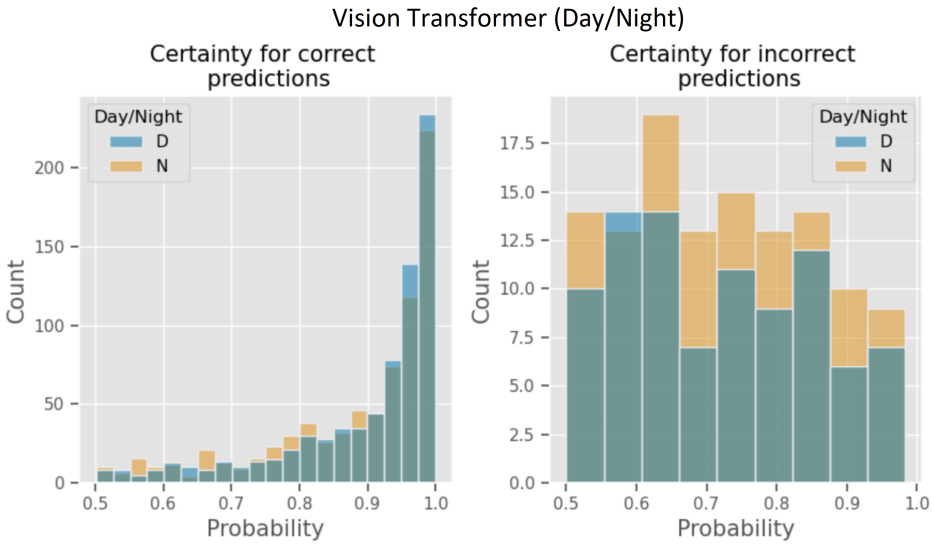

Figure B.1: The probabilities predicted by the vision transformer based on the time of day when predicting snow/no snow. The labels 'D' and 'N' refer to day and night, respectively. To the left is shown the distribution when the model predicts correctly, and to the right, when the model's prediction is wrong.

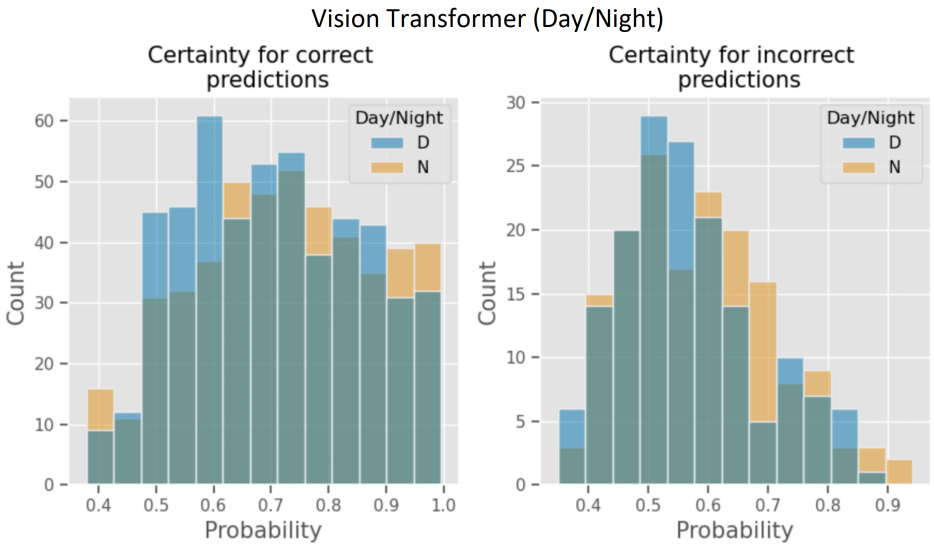

Figure B.2: The probabilities predicted by the vision transformer based on the time of day when predicting snow/wet/dry. The labels 'D' and 'N' refer to day and night, respectively. To the left is shown the distribution when the model predicts correctly, and to the right, when the model's prediction is wrong.

C

Persistent Homology

This section presents the mathematics behind persistent homology. The aim is to present a mathematical foundation along with intuitive examples. It begins with defining simplicial complexes and homology. Persistent homology is then presented as a tool for extracting meaningful features from high-dimensional data.

C.1 Simplicial Complexes

The foundation of persistent homology (PH) is the simplicial complex [7]. A simplex can be seen as a generalization of triangles in higher dimensions and a simplicial complex is simply the combination of simplexes under certain rules.

Definition 1: *A geometric k-simplex* $\sigma^k = \{v_0, v_1, v_2, ..., v_k\}$ *is the convex hull formed by k+1 affinely independent points* $v_0, v_1, v_2, ..., v_k$ *in Euclidean space* $\mathbb{R}^d$ as follows,

$$\sigma^k = \Big\{\lambda_0 v_0 + \lambda_1 v_1 + ... + \lambda_k v_k \Big| \sum_{i=0}^{k} \lambda_i = 1; 0 \leq \lambda_i \leq 1, i = 0, 1, ..., k\Big\}.$$

A face τ *of k-simplex* σ^k *is the convex hull of a non-empty subset of* σ^k*. We denote it as* $\tau \leq \sigma^k$.

A 0-simplex can be thought of as a vertex, a 1-simplex as an edge, a 2-simplex as a triangle, etc.

Definition 2: *A geometrical simplicial complex K is a finite set of geometric simplices that satisfy the following conditions:*

1. *Any face of a simplex from K is also in K.*
2. *The intersection of any two simplices in K is either empty or shares faces*

For example, a simplicial complex, K, containing the 1-simplex $\{0, 1\}$ must also include the 0-simplices $\{0\}$ and $\{1\}$, and K cannot contain a 0-simplex 3 which lies on the edge defined by $\{0, 1\}$. The dimension of K dictates the highest possible dimension of a simplex contained in K.

Definition 3: *An abstract simplicial complex K is a finite set of elements* $\{v_0, v_1, v_2, ..., v_n\}$ *called abstract vertices, together with a collection of subsets*

$(v_{i_0}, v_{i_1}, ..., v_{i_m})$ *called abstract simplices, with the property that any subset of a simplex is still a simplex.*

For an abstract simplicial complex, K, there exists a geometrical simplicial complex, $K^{'}$, whose vertices correspond one-to-one with the vertices of K. Any simplex in $K^{'}$ must correspond to a simplex in K. $K^{'}$ is called the geometric realisation of K. For example, the geometric realization of the abstract simplicial complex $\{\{0\},\{1\},\{2\},\{0,1\},\{0,2\},\{1,2\}\}$ is an empty triangle where the corner vertices are $\{0,1,2\}$.

There are many ways to create simplicial complexes with varying suitability for different data types. The grid structure of images makes the use of cubical complexes suitable where we can consider the center of each pixel as a 0-simplex.

Definition 4: *An elementary interval is a closed interval $I \subset R$ of the form $I = [l, l+1]$ or $I = [l, l]$, for some $l \in Z$. The interval $[l, l]$ contains only one point and is degenerate.*

Definition 5: *An elementary cube Q is a finite product of elementary intervals, that is $Q = I_1 \times I_2 \times ... \times I_d \subset R^d$, where I_i is an elementary interval.*

Definition 6: *A set $K \subset R^d$ is a cubical complex if K can be written as a finite union of elementary cubes.*

The cubical complex of an image can be seen as a discrete surface where the discrete points correspond to the centers of all pixels and the height of the surface is determined by the greyscale level of each pixel.

C.2 Homology

Homology aims to quantify an equivalence class of cycles that enclose a k-dimensional hole, i.e. cycles that are not also boundaries of k-simplices [37].

Definition 7: *An orientation of a k-simplex $\sigma^k = \{v_0, v_1, v_2, ..., v_{k+1}\}$ is an equivalence class of orderings of the vertices of σ^k, where $(v_0, v_1, v_2, ..., v_{k+1}) \sim (v_{\tau_0}, v_{\tau_1}, v_{\tau_2}, ..., v_{\tau_{k+1}})$ are equivalent orderings if the parity of the permutation τ is even. An oriented simplex is denoted by $[\sigma^k] = [v_0, v_1, v_2, ..., v_{k+1}]$*

For example, the oriented 2-simplex $[\sigma^2]_1 = [1,2,3]$ has the same orientation as $[\sigma^2]_2 = [2,3,1]$ but not $[\sigma^2]_3 = [1,3,2]$ [7]. A linear combination of k-simplices, $c = \sum_i \alpha_i \sigma_i^k,\ \alpha_i \in \mathbb{Z}_2$, is called a k-chain.

Definition 8: *A k-th chain group $C_k(K)$ of a simplicial complex K is an Abelian group made from all k-chains from the simplicial complex K together with addition operation.*

Consider a simplicial complex K containing the following 2-simplices $\{\{1,2,3\},\{2,3,4\}\}$,

the 2nd chain group is $C_2(K) = \{\{1,2,3\},\{2,3,4\}\},\{1,2,3\} + \{2,3,4\}\}$. All k-chains from a simplicial complex K together with modulo-2 addition form an Abelian group $C_k(K, \mathbb{Z}_2)$. Homomorphism can be defined between these groups.

Homomorphism is a correspondence between the elements of two algebraic structures [43]. Two homomorphic systems have the same basic structures and the results on one system often apply to the other. Let G and H be groups with elements $g_0, g_1, ...$ and $h_0, h_1, ...$, and they are subject to an operation $\circ$. A homomorphism from G to H is a correspondence between all elements in G and some elements in H that has the following property:

$$\text{if } g \to h \text{ and } g' \to h', \text{ then } g \circ g' \to h \circ h'$$

In other words, homomorphism is a transformation of one group to another, which preserves the associated operation.

Definition 9: *The boundary operator $\partial_k : C_k \to C_{k-1}$ of an oriented k-simplex $[\sigma^k] = [v_0, v_1, v_2, ..., v_{k+1}]$ can be denoted as $\partial_k[\sigma^k] = \sum_{i=0}^{k}[v_0, v_1, v_2, ..., \hat{v}_i, ..., v_{k+1}]$, where $[v_0, v_1, v_2, ..., \hat{v}_i, ..., v_{k+1}]$ is an oriented (k-1)-simplex generated by all vertices contained in $[\sigma^k]$ except $\hat{v}_i$.*

The boundary operator ∂_2 applied to the oriented 2-simplex $[\sigma^2] = [1,2,3]$ would result in $\partial_2[\sigma^2] = [2,3] - [1,3] + [1,2]$, i.e. the boundary of the oriented 'triangle' $[\sigma^2]$ consists of 3 oriented 1-simplices or edges. The boundary operator satisfies $\partial_0 = 0$ and $\partial_{k-1}\partial_k = 0$ [7]. With the boundary operator, we can define the k-th cycle group and the k-th boundary group.

Definition 10: *The k-th cycle group is defined as:*

$$Z_k = \text{Ker } \partial_k = \{c \in C_k | \partial_k c = 0\}$$

Definition 11: *The k-th boundary group is defined as:*

$$B_k = \text{Im } \partial_{k+1} = \{c \in C_k | \exists d \in C_k : c = \partial_{k+1} d\}$$

The cycle and boundary groups satisfy $B_k \subseteq Z_k$ since $\partial_{k-1}\partial_k = 0$. Wit this in mind we define homology groups as:

Definition 12: *The k-th homology group is the qoutient group $K_k = Z_k/B_k$.* The rank of the k-th homology group is called the k-th Betti number and satisfies $\beta_k = \text{rank } H_k = \text{rank } Z_k - \text{rank } B_k$.

The Betti numbers can be considered as a count of the number of k-dimensional 'holes'. The first three Betti numbers can be seen as the number of connected components, the number of loops, and the number of trapped volumes.

C.3 Persistent Homology

Persistent homology is a method that aims to track the evolution and scales of topological features during a filtration [44]. The method uses geometry to define a function in a topological space, and algebra to turn the function into a measurement.

Definition 13: *A filtration of a simplicial complex K is a nested sequence of subcomplexes, $\emptyset \subseteq K^0 \subseteq K^1 \subseteq ... \subseteq K^n = K$. A simplicial complex with a filtration is called a filtered complex.*

The method of filtration differs depending on the data type and the type of simplicial complex. For a point cloud, one method is to assign a sphere to each data point and let these expand with time. At fixed points in time, a simplicial complex can be built based on the overlap of the spheres. In the case of image data and cubical complexes, it is suitable to view the image as a discrete surface where the center of each pixel represents a point on the surface, and the height of the point is determined by the greyscale level of the pixel [36]. A cross-section is then applied at different greyscale levels, and subcomplexes are built based on the surface below the cross-section.

Throughout this filtration, the topological evolution is tracked through the homology groups of each subcomplex [44]. For every $i \leq j$ we have an induced homomorphism from K_i to K_j through $f_k^{i,j} : \mathrm{H}_k(K^i) \to \mathrm{H}_k(K^i)$. This corresponds to a sequence of homology groups.

$$0 = \mathrm{H}_k(K^0) \to \mathrm{H}_k(K^1) \to ... \to \mathrm{H}_k(K^n) = \mathrm{H}_k(K)$$

As the filtration progresses, new homology classes will appear, and some homology classes will disappear, this is referred to as the birth and death of classes. By grouping classes born at or before a certain time and which die after another time, the persistent homology groups can be defined.

Definition 14: *The k-th persistent homology groups are the images of the homomorphisms induced by inclusion, $H_k^{i,j} = im\ f_k^{i,j}$, for $0 \leq i \leq j \leq n$. The corresponding k-th Betti numbers are the ranks of these groups, $\beta_k^{i,j} = rank\ H_k^{i,j}$*

In simpler terms, let γ be a class in $\mathrm{H}_p(K^i)$ then γ is born at K^i if $\gamma \notin \mathrm{H}_k^{i-1,i}$. γ dies at K^j if it merges with an older class as the filtration step goes from $j-1$ to j.

The result of the filtration can be represented as pairs of birth and death times of groups [7].

$$l_j^k = \{a_j^k, b_j^k\} \text{ with } k \in \mathbf{N}$$

The persistence of a class is $(b_j^k - a_j^k)$. These pairs can be visualized in persistence diagrams (PD). In persistent diagrams, each pair is represented as a point with coordinates (a_j^k, b_j^k).

There are many ways to extract features usable for a machine learning model from persistent diagrams [7]. The simplest features are the basic statistical features

such as the mean persistence, mean death time, variance of death time, etc. More complex features can be extracted using various distance measures, kernels, and persistent images.

Persistent images were first introduced by Adams et al. [37]. It is a transformation of persistence diagrams into images that are robust to small noise. Persistence images are created by first rotating the persistent diagram using the transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$, $T(x, y) = (x, y - x)$.

A weighted probability distribution is then assigned to each point in the rotated persistence diagram in order to create a persistence surface. Let $\phi_u : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable probability distribution with mean $u = (u_x, u_y) \in \mathbb{R}^2$. The most common choice of ϕ_u is the normalized symmetric Gaussian:

$$\phi_u = \frac{1}{2\pi\sigma^2} e^{-[(x-u_x)^2+(y-u_y)^2]/2\sigma^2}.$$

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a non-negative weighting function that is zero along the horizontal axis, continuous and piecewise differentiable.

Definition 15: *Let B be a persistence diagram and T(B) be the rotated persistence diagram , the corresponding persistence surface* $\rho_B : \mathbb{R}^2 \to \mathbb{R}$ *is the function*

$$\rho_B(z) = \sum_{u \in T(B)} f(u)\phi_u(z).$$

The persistent image is then created by fixing a grid on the persistent surface and integrating it over each grid cell. Each grid cell then corresponds to a pixel in the image.

Definition 16: *Let B be a persistence diagram and* ρ_B *be its persistent image. The persistent image is then defined as the collection of pixels.*

$$I(\rho_B)_p = \int\int_p \rho_B dy dx$$

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY