

MSc. Thesis Report

Weather Condition Estimation in Automated Vehicles

J.C.H. Wymenga

Supervisors

ir. J.F.M Domhof

prof. dr. D.M. Gavrilu

April 23, 2018



MSc. Thesis Report

Weather Condition Estimation in Automated Vehicles

by

J.C.H. Wymenga

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 15th of May, 2018

Student number: 4090284
Project duration: December, 2016 – March, 2018
Thesis committee: Prof. dr. D.M. Gavrilă (chair)
dr. J.F.P. Kooij
dr.ing. J. Kober
ir. J.F.M. Domhof

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Contents

1	Background	5
1.1	Vision-based weather classification	5
1.2	Weather sensor-based weather estimation	10
1.3	Weather conditions	14
1.4	Classification evaluation metrics	15
1.5	Prediction Fusion	16
2	Dataset	21
2.1	Available datasets	21
2.2	Data collection	23
2.3	Data labelling	28
3	Classifiers	31
3.1	Classifier algorithm	31
3.2	Hyperparameter optimisation	31
3.3	Classifier results	35
4	Data fusion	41
4.1	Model matrices	41
4.2	Fusion results	41
5	Discussion	47
5.1	Dataset	47
5.2	Individual classifiers	49
5.3	Fusion	49
6	Conclusion and Recommendations	53
6.1	Conclusion	53
6.2	Recommendations	54
	Bibliography	59

Introduction

The field of autonomous vehicles is advancing rapidly. Not only are companies actively pursuing the production and sale of autonomous vehicles (AVs) in the near future [57], Advanced Driver Assistance Systems (ADAS) are already used more and more in modern vehicles. In vehicles equipped with ADAS, control is taken over in part by autonomous systems (e.g. lane-keeping systems or adaptive cruise control) but other aspects of control are exerted by the driver¹. Autonomous vehicles or those equipped with an ADAS make use of a number of sensors, often including distance sensors, camera(s), and radio detection systems (RADAR), in order to model their environment.

Adverse weather conditions may affect many aspects of the driving environment. For example, wet roads may become slippery posing a hazard to drivers [1]. Additionally, weather conditions may affect drivers' visibility, sensor measurements and, not unimportant, the behaviour of other road users around them [26, 45]. Sensors affected by weather conditions like rain or fog may not be able to assess this influence by themselves. For example, object detection models using camera may not be able to discriminate between a situation in which they do not detect some object because there is there, and one in which nothing is detected due to fog affecting the camera's visibility.

A robust and accurate weather context may help mitigate these problems by means of providing a weather *context* to an ADAS or AV system. Modelling and control subsystems of the ADAS/AV can use such context information to account for weather conditions in their processes. For example, if rain or fog are known to impair the reliability of some sensor, the system may adjust its modelling parameters or driving behaviour accordingly.

This research aims to construct such a context using information that is available in the vehicle or can be easily measured using on-board sensors. These constraints are put in place to allow the vehicle system to operate independently of information from outside sources (e.g. weather forecasts from the internet). The context should be supplied fast, reliable, and not be affected by changes of scene type (e.g. urban or highway). To improve reliability of the model, it should not rely fully on a single sensor in the vehicle, but instead should provide a framework in which multiple different (or redundant) sensors could be used.

This report aims to provide a framework and proof-of-concept for estimating weather conditions in-vehicle, based on sensor measurements and without the dependence on external sensor information. The latter requirement is based on the notion that, because of safety concerns, future automated vehicles cannot rely on external data for their operation. The resulting framework should be flexible and modular in order to allow changes in sensor configuration as the availability of sensors may change in future vehicles.

Modelling approach

Many approaches may be taken to the kind of weather condition estimation system presented in the previous section. Research questions are formulated to limit the scope of the research to make the problem tractable, and to ensure that the resulting model is as useful as possible in real-world applications. The main research question for this work is:

How to construct an accurate and robust weather estimation model using multiple on-board sensors in a vehicle?

A preliminary study in related works shows that two approaches for weather condition estimation in general are the most common:

- For driving situations, the use of a computer vision model and on-board camera is an often-used state-of-the-art approach. For example, the related works [32, 43] and [56] employ classical machine learning techniques and image features in order to estimate the weather condition from image data. Other works like [28] and [19] use image data as well, but derive custom situation-specific visual models to do so.

¹In the remainder of this section the term 'automated' driving is used for both the partial autonomy of ADAS systems and the full autonomy of self-driving cars.

- For local weather prediction in a more general sense, the common approaches [16, 37, 53] make use of easily-observed weather variables like temperature, pressure and humidity to estimate or predict weather conditions.

Both of these models can satisfy the requirement that the modelling should be carried out in-vehicle without the need for external information. As front-facing cameras are becoming more and more common on modern vehicles and many weather variables can be measured using sensors that are inexpensive and easy to obtain, both these approaches may be used at the same time in a vehicle. The two approaches could potentially be combined in order to construct a model that takes the strengths of both method and is more robust than either of the two models by itself. This approach is considered worth exploring, and as such three sub-questions are formulated to focus on the idea of combining two approaches for improved results:

1. *How can Machine Learning techniques be employed to construct an accurate visual weather estimation model using camera images?*
2. *How can locally measured weather variables be used to construct a weather sensor-based weather estimation model?*
3. *How to combine information from visual and weather sensor-based weather condition estimation models in order to improve both estimation accuracy and reliability, and yield a practical model?*

High-level model architecture

A general high-level architecture of the intended model is convenient in order to define model components and the resulting modelling approach. Three possible architectures are shown in Figure 1, with each a different level at which data of the different sources is combined. Though each of these architecture has advantages and disadvantages, Architecture **B** from Figure 1 is considered the most promising for two practical reasons:

1. addition, removal, or change of a sensor affects each of the intermediate steps (preprocessing, classification and fusion). In architectures **A** and **C** this means that preprocessing and classification are affected for all sensors, whereas in **B** only the steps of that specific sensor and the Fusion steps may require changes.
2. The nature of the two information sources proposed differ in measurement frequency, dimensionality and rate of change. As such, combination in raw (**C**) or pre-processed form (**A**) is expected to cause problems in later intermediate steps. These effects differences on classification have to be carefully assessed and may not be clear from the surface. For example, combining data of large and smaller dimensionality may have implications when classification is done using automated feature selection, as one sensor provides many more features than the other.

A more specific implementation of architecture **B**, using the two models (Vision model and Weather sensor model) presented in this section is shown in Figure 2. The architecture in figure 2 is be used further in this research. The intermediate steps Fusion architecture will be explored and implemented in order to create a weather condition estimation model and assess its strengths and weaknesses with regards to the research questions.

Report structure

The first chapter gives an overview of the theoretical background and existing methods used to construct the model components shown in Figure 2. Data is collected in order to evaluate the model and the collection process is described in Chapter 2. The third chapter shows the implementation and configuration of independent classifiers for the vision and weather sensor model used. The fourth chapter describes the Fusion model that is used to combine estimations made by these individual classifiers. The fifth and final chapter shows conclusions and discussion with regard to the research questions stated earlier in this introduction.

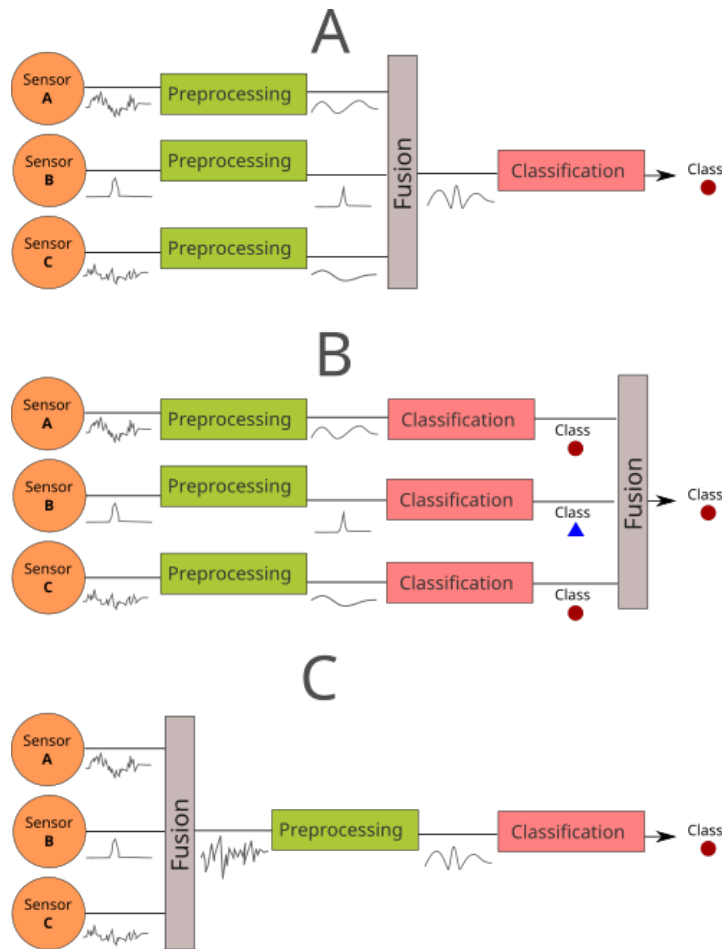


Figure 1: Example of three possible architectures for data fusion of information from different sensors. In architecture **A** pre-processed data is combined before classification takes place, in **B** classification takes place for each sensor and the results are fused. In **C** raw data is fused, and preprocessing and classification take place on the combined raw data.

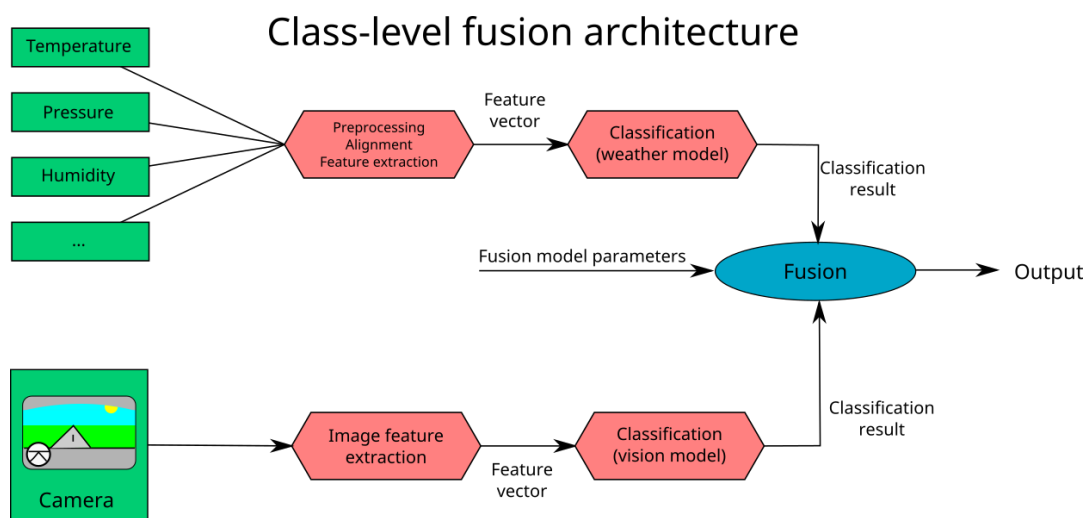


Figure 2: High-level model architecture of the sensor-based weather condition estimation model.

Background

In the introduction, an overview and high-level architecture is given of a possible approach to construct a weather condition estimation model. This chapter describes a number of existing methods that are used to implement the individual model components shown in Figure 2, as well as the data Fusion approach and model evaluation metrics. To construct the model described in the previous chapter, a number of existing techniques is used. Since the research involves constructing a vision classifier, weather sensor classifier and fusion model, the modelling techniques used to construct each of those components are discussed in this section. To get an unequivocal definition of the estimation targets, the considered weather conditions are defined in Section 1.3. Furthermore, an overview of metrics used to evaluate the classifiers is presented in Section 1.4. Finally, this chapter presents some definitions and notations used in further chapters of this reports.

1.1. Vision-based weather classification

Weather classification on images using Machine Learning techniques, as applied in e.g. [32, 43, 56, 59] is done in two steps. First, informative data is extracted from images in the form of *features*. Features are calculated on raw image data, and convey information believed relevant to the classification process. Then, a *Supervised Learning Algorithm* [35] is used to train one or more classifiers that learn to distinguish images of taken in different weather conditions based on those features.

This section shows the image features selected for the first step, and introduces an intermediate step intermediate step (ROI selection) proposed in [43]. The choice and implementation of the Supervised Learning Algorithm used here is discussed in Chapter 3.

Representation of image data

For convenience, the same notation and definitions for image data are used throughout this section. Images, here, are interpreted as $w \times h \times 3$ matrices \mathbf{I} , where w is the image width, h the image height and the elements of \mathbf{I}_{wh} are the Red, Green and Blue intensities. These values can be directly obtained from raw image data and are integers ranging from 0 through 255. In other words, the image I of dimensions $w \times h$ is constrained to:

$$\begin{cases} I(j, k, c) & \in \mathbb{Z}_{255} \\ j & \in \mathbb{Z}_w \\ k & \in \mathbb{Z}_h \\ c & \in [R, G, B] \end{cases} \quad (1.1)$$

Where \mathbb{Z}_j denotes the set of integers $[0...j]$.

1.1.1. Image feature extraction

Feature extraction tends to reduce the dimensionality of the image data greatly, and aims to extract only aspects of the image that are relevant with regard to the classification task at hand and invariant to other conditions. The relevance of specific image features depends greatly on the task at hand, and the authors

of [43] and [59], among other, both propose features that are found to be useful in classifying weather conditions. This section describes a number of image features used for the Vision classifier used in this work. These features are sourced from a number of works ([32, 43, 56, 59]) and selected by three criteria imposed by the research questions at hand:

- **Fast in computation:** Since the research considers driving applications, features should be calculated fast enough to keep up with the rapidly changing data the on-board camera provides. Single-frame computation times under $1Hz$ would be ideal, but it is difficult to predict computation time on arbitrary hardware. As a rule of thumb, features will be limited to those that can be computed using linear computations on pixel values (addition, subtraction, division, etc). More complex features that require, for example, probabilistic search algorithms (like the automated road region detection presented in [19]) or pattern matching algorithms (as in [28]) are therefore excluded.
- **Independent of road scenery:** The behaviour of features should not be dependent on road scenery. Features that only work in situations with a visible horizon (like those in [19, 39]) are excluded, as such conditions are impractical in real-world situations. Instead, features should be generic and not require any scene-related assumptions in their algorithms. This is to ensure that the Vision classifier works in every scenery type, and that only one classifier is required for all scenes.
- **Absence of temporal relations:** it must be possible to compute the features on single frames, as not all test datasets contain sequential video frames. Also, in real-world situations the intervals between camera frames may not be constant.

Apart from these practical criteria the features should of course be useful for the prediction of weather conditions, which is shown in the works where they were originally proposed. The following seven features were collected from different state-of-the-art works ([43, 49, 59]) and considered the most suitable according to the above criteria. All features calculations are normalised to return values between 0 and 1 to avoid scaling problems.

Patch-based feature computation

All features are computed over 10×10 patches; the mean or median is used to combine features over the patches into a single number between 0 and 1 as suggested in [43]. This increases speed, reduces complexity and allows for the computation of features which cannot be calculated on single pixels. With the definitions from Section 1.1, we can denote a patch P as we would an image I , where $P(x, y, c)$ denotes the value of channel $c \in [R, G, B]$ of the pixel at $(x, y) \in ([1, 100] \times [1, 100])$. The following subsections show the specifications used for the calculation of each selected feature, Table 1.1 gives a summary of features and their symbols.

Table 1.1: Image features and their symbols

Feature	Symbol	Reference
Noise feature	N	[49]
Edge Energy	E_E	[49]
Local Contrast	C_{local}	[43]
Local Saturation	S_{local}	[43]
Minimum brightness	B_{min}	[43]
Dark Channel Intensity	I_{DC}	[59]
Hue	H	[43]

Noise Feature

In rainy and snowy conditions, atmospheric particles create noise in the image according to [49]. The authors estimate image noise using the standard deviation of the Laplace-filtered image. The Laplace filter used here is the 3×3 kernel matrix L :

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

To compute the noise feature N , the entire image is convoluted with the Laplace kernel L and then, for each ROI, the standard deviation of the patch intensity is used as the ROI feature value. Since, for the 10×10 patch the maximum standard deviation is 0.5, the resulting standard deviation is divided by 0.5 so that the range of the feature value is 0..1.

Edge Energy

Metrics of Edge Energy are used in in [43, 49] and fog-estimation works by many other authors. Edge Energy gives an indication of how many edges can be observed in the image, and their relative strength, using the notion that in blurry (or foggy) images these strengths are reduced due to scattering light. To detect or define edges, many image filters exist. The filter used here is the Sobel operator [46], as it is fast in computation and isotropic by nature. The Edge Energy is calculated using the following process, quite similar to the computation used in [43]:

1. Compute the *Sobel* filter response on the image in the horizontal (S_h) and vertical (S_v) direction
2. Define S_n as the Euclidean norm for each pixel value in S_h and S_v
3. Define the edge threshold τ_S as the mean value of S_n plus two times its standard deviation (over the entire image)

The edge energy E_E over a patch is calculated as the mean S_n for each edge pixel (each pixel for which $S_n > \tau$) in the patch. Since the *Sobel* responses are already on the [0..1] range, the maximum value of the resulting feature value is $\sqrt{2}$. The feature value is divided by this value to normalise the it to [0..1].

Local Contrast

Local contrast is shown to be a good contrast and fog indicator in [49] and [43]. The implementation here is similar to the one used in [49].

For each pixel $P(x, y, c)$ on the patch, the relative Luminance $L(x, y)$ is computed using the *Luma* formula (equation 1.5), then the local contrast value C_{local} is computed over the patch as:

$$C_{local}(P) = \sqrt{\frac{\sum_P [L^2(x, y)] - \frac{(\sum_P [L(x, y)])^2}{N}}{N}} \quad (1.2)$$

The value N denotes the total number of pixels on the patch (100 in our case).

Local Saturation

Saturation, as shown in [59] is a good illumination-independent indicator of weather condition. The patch is converted to HSV color space, and the values of the saturation channel S on patch P are denoted $S_P(x, y)$. The value is normalised using the maximum and minimum over the entire image, and then the mean over the patch is computed as the feature value:

$$S_{local}(P) = \frac{\sum_P \frac{S_P(x, y) - \min(S)}{\max(S) - \min(S)}}{N} \quad (1.3)$$

Here, the values $\min(S)$ and $\max(S)$ are the natural minimum and maximum of the saturation channel, 0 and 255 respectively.

Minimum brightness

The local minimum brightness is the normalised minimum value for the Luminance over a patch. Luminance, a derived color channel is computed from RGB values using the *Luma* formula [9] shown in Equation 1.5. Since the coefficients of the Luma formula sum up to one, the features is normalised using the natural minimum- and maximum values 0 and 255.

$$B_{min}(P) = \min_P (L_P(x, y)) \quad (1.4)$$

$$L_P(x, y) = 0.2125 \cdot P(x, y, R) + 0.7154 \cdot P(x, y, G) + 0.0721 \cdot P(x, y, B) \quad (1.5)$$

Dark Channel Intensity

The *Dark Channel* feature, used in [59] and other works, is related to the observation that in haze-free images, the minimum intensity of pixels in any of the three color channels is very low[59]. Haze-affected images generally contain less-dark pixels due to scattered light. The *Dark channel* value of a pixel is the value with the lowest intensity among that pixel's R, G and B values. In image terms, pixels in Hazy images are often displayed 'whiter' than in Dry images, as the fog adds a white layer over the original. An example of the Dark Channel values for a sunny and hazy image is shown in Figure 1.1. For a patch, the dark channel value is the minimum value for each channel, for each pixel in the patch:


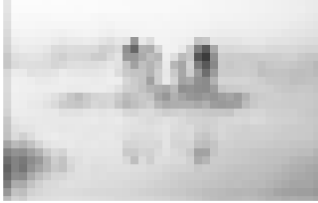

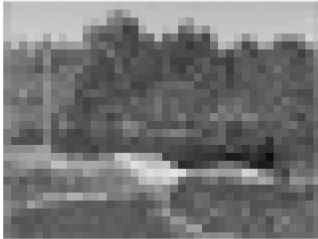
Class	Original image	Dark Channel value (on patch)
Hazy		
Sunny		

Figure 1.1: Example of Dark-Channel result on a hazy image (top) and on a sunny image (bottom). The white 'veil' resulting from the fog reduces the occurrence of pixels that are very dark in every channel. Images taken from the MWI dataset[59].

$$I_{DC}(P) = \min_{x,y,c}(P(x, y, c)) \quad (1.6)$$

Since I_{DC} is a color channel value, it is normalised by dividing over 255.

Hue

The mean value of the Hue value, extracted from the HSV transform is used as the feature value for the Hue feature. The hue values are normalised using their natural minimum- and maximum values in images, 0° and 180° .¹

$$H(P) = \frac{\sum P}{N} \quad (1.7)$$

1.1.2. ROI selection

Roser & Moosmann [43] demonstrate an efficient yet simple method to encode interesting region and distance information (relative to vehicle) in their features. The authors select Regions of Interest (ROIs) based on rectangular subspaces of the image which usually correspond to different scene regions and distances. For example, in almost every situation, the top center part of an image taken using a windshield camera shows mostly sky, whereas the center bottom part shows road, and the absolute center usually contains other vehicles and the horizon. The left and right areas may contain the same, but also show buildings and other objects just off the road. The authors note that this assumption may be less valid in urban situation as it is in expressway situations.

¹In [9], the HSV color space has a 360-degree *hue* channel. In the popular OpenCV image processing software however, Hue values are halved so that the result fits in a single 16-bit integer

An alternative to using rectangular ROIs to extract interesting image regions is using algorithms to detect and extract sky or road regions [19] but these add complexity and the result may not be meaningful for all scenery situations. As this research does not focus explicitly on creating the most optimal visual classifier, flexibility is preferred, and the ROIs described in [43] are used.

Regions of Interest

Image data is split in 13 ROIs numbered R_0 through R_{12} . As in [43] first ROI, R_0 is the full image and the other 12 ROIs are created by slicing the image in three equal parts horizontally, and in four equal parts vertically. The numbering is from left to right and from top to bottom: the first row is numbered R_1 through R_4 , the second row R_5 through R_8 , and so on. An image showing the ROIs on an example image frame is shown in Figure 1.2. The feature values described Section 1.1.1 are calculated for each of these ROIs. The dimensionality of the feature vector obtained this way is:

$$10 \text{ bins} \times 13 \text{ ROIs} \times 7 \text{ features} = 910D$$



Original image



Regions of Interest (ROIs)

Figure 1.2: Schematic of the ROI specification. The original image (top) is divided into 12 Regions of Interest (ROIs). The ROIs are divided in For each feature and ROI combination, mean feature value is computed and added to the feature vector. The ROI specification was originally proposed in [43].

Dimensionality reduction

The $910D$ feature vector seems to introduce a huge computational overhead for the visual classifier. Therefore, after computing feature values for each patch, the mean over all patch values on the ROI is used as the feature value, giving an $1D$ feature value per ROI, or an $91D$ total feature vector. An image describing this process is shown in Figure 1.3. It is expected that this may affect the classifier performance a bit, but that reducing the dimensionality by an order of magnitude justifies this choice. The rationale behind this assumption is that in both cases, the feature values tell us something about the values in the patches. When using histograms, information about the distribution of feature values over the patch is retained, while using the mean discards this information, except for the mean itself. However, if the mean is useful to the classifier, it will need to consider all of the histogram bins to evaluate it. This is a trade-off between locality of information and complexity: if the variance among the patch values is small, the mean gives a lot of information about their values. The larger this variance, the less informative this mean becomes.

In a more practical sense, many classifiers, including the one used here (Section 3) operate on single feature values at a time. In this case, determining whether a value should be considered ‘high’ or ‘low’ would take at least 5 decisions using 10-bin histograms (as the majority of values may be in any of the first 5 bins), whereas it takes only one decision when considering the mean value.

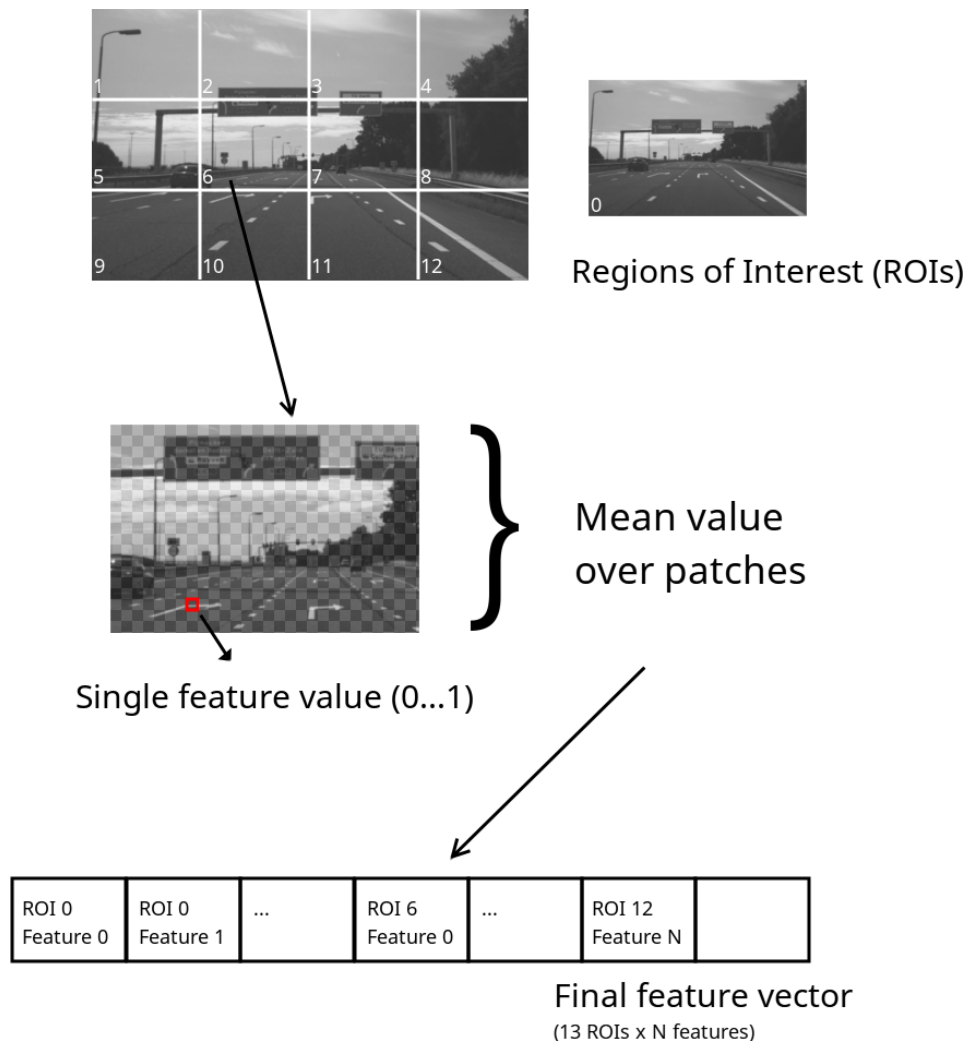


Figure 1.3: Feature vector construction. A single feature value generates 13 values in the feature vector, one for each ROI.

1.2. Weather sensor-based weather estimation

Professional weather forecasting, used for e.g. the news weather message or aviation is generally done using weather stations that measure, among others atmospheric quantities like cloud cover (height) and high-atmospheric pressure fields. These quantities are predicted using lattice methods [52] (historically) and, more recently, by satellite and RADAR[4]. In a driving situation, it is infeasible to make direct use of a weather radar or satellite imagery, and also the highly complex numerical models used in modern day weather prediction are not considered a viable prediction option. Instead *local* methods that depend on data that can be measured from the ground have to be used.

Such local weather predictions are created by (amateur) meteorologists using much more portable weather sensors, often thermometers, humidity sensors (hygrometers) and air pressure sensors (barometers). Works like [25, 37] gives some examples of how quantities like humidity and air pressure may estimate (or predict) the weather. Other works like [3, 58] use an approach similar to the one used here by predicting Fog and using data mining techniques and (in part) locally measured variables.

1.2.1. Locally observable weather variables

This section presents weather-related quantities that are considered predictors for weather classification. Candidates are sourced from a number of related works [3, 25, 58] and selected by their fitness regarding a number of constraints relevant here in order to make the resulting model applicable and useful in real-world applications:

1. They must be measurable using only sensors that are either present on modern-day cars, or sensors that can be added to the car in an unobtrusive, inexpensive way
2. They must be measurable accurately with regard to their estimated effect size and at a sufficiently high rate (at $< 10s$ intervals, preferably $< 1s$).
3. There must be a visible effect size or association of the single variable and the weather conditions defined in Section 1.3.

Temperature

Temperature may be the most used weather variable in weather prediction through the ages[33], and its value used as both a predictor and predictand². It is very easy to measure as both digital and analog temperature sensors nowadays are cheap and easy to obtain. Both thermodynamics and meteorology show a well-studied relations between weather effects and temperature. When, for example, rain causes cold water from the colder, high atmosphere to fall down, local temperature tends to fall. The other way round, cooling air can contain less moisture: cooling, saturated air may need to ‘lose’ whether by condensation which may result in rain and/or fog.

An example quantification of the cooling effect is shown in Figure 1.4. It shows that, in the Netherlands, rain is much more likely to occur with dropping temperatures (changes from $-1^{\circ}C/hr$ to $-10^{\circ}C/hr$), and occurs rarely together with rising temperatures. If temperature change can be measured with at least $1^{\circ}C$ precision, this effect should show itself in the data its effects may be used for weather estimation.

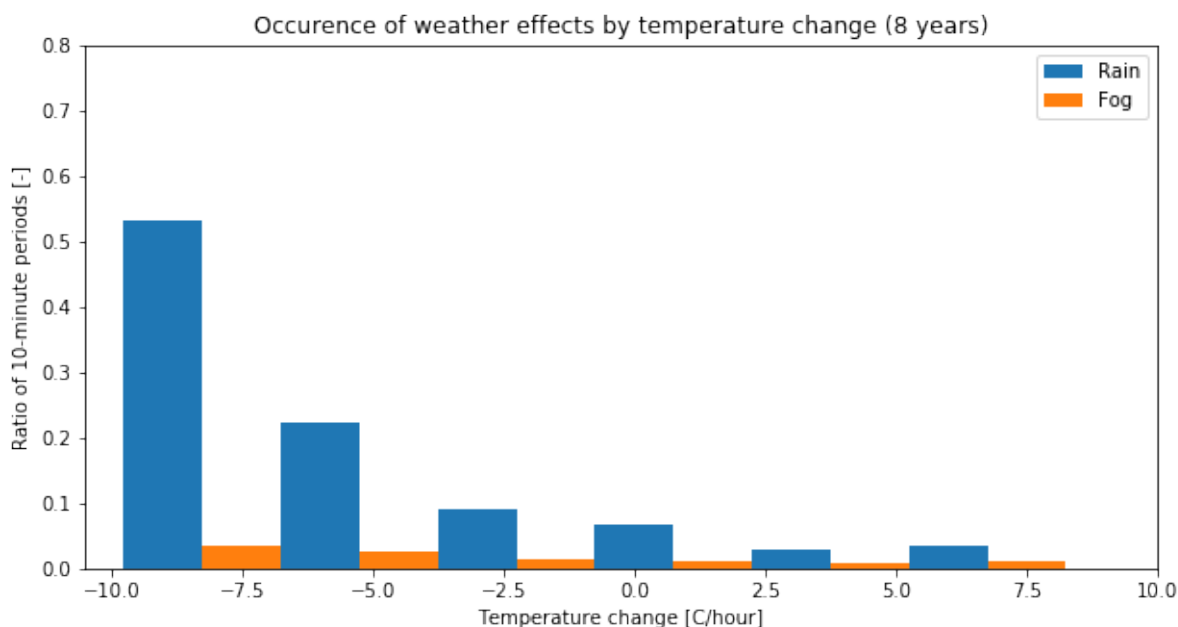


Figure 1.4: Mean rain/fog frequencies versus temperature change. This shows ratio of all data points with one associated temperature change rate bin ($^{\circ}C/hr$) that contain rain (blue) or fog (orange). Created from hourly rain data over a 16-year period in the Netherlands. Data from CESAR [12].

²A variable that is to be predicted

Humidity

Relative air humidity (RH) is defined as the amount of water in air relative to the point of saturation. Since air cannot contain more water than its saturation point it will need to lose water (by condensation) when it is over-saturated. This effect manifests itself as rain or fog, and is shown in Figure 1.5. The plot shows a distribution of rain rates that peaks at saturated (> 100%) and near-saturated (> 90%) humidity levels. Also, it shows that fog occurs almost exclusively, and very frequently, at any over-saturated humidity level.

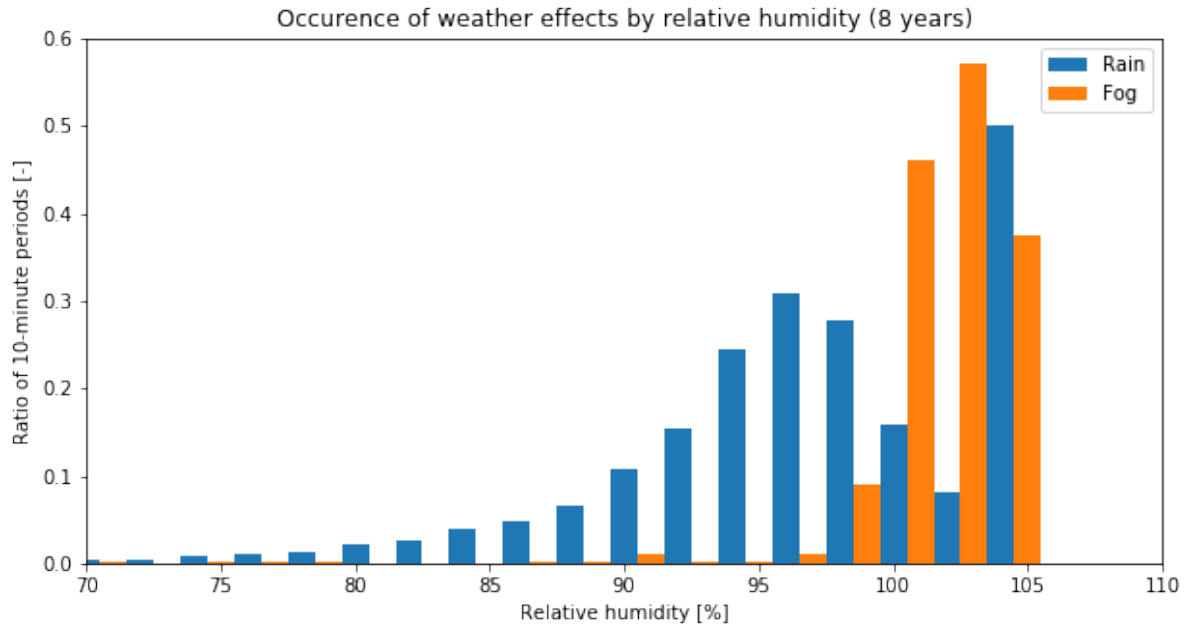


Figure 1.5: Mean rain/fog frequency versus relative humidity. This shows the relative frequency of rain and fog among all data points with one associated relative humidity bin ($\%RH$). Created from hourly rain data over a 16-year period in the Netherlands. Data from CESAR [12].

Air pressure

Most modern-day weather forecasts rely heavily on the estimation of high-and low pressure areas high up in the sky [4, 37, 53]. In fact, air pressure measurements in weather forecasting have been in use for over a century [16, 33]. Already in 1897, barometric pressure effects on the ground were correlated with weather conditions [16].

To predict weather conditions, the author in [16] uses three variables: barometric pressure, its rate of change, and wind direction to predict weather conditions for the eastern United States. The author uses four pressure thresholds, each 680 Pa apart³ and descriptions like ‘rising’ or ‘falling rapidly’ for the derivatives.

Another source [54] defines fast and slow changes in barometric as shown in Table 1.2. Weather prediction using such tables, or from personal experience, is used often among amateur meteorologists. Many sources on the internet explain how to read and use a barometer, and give an indication of the tables to use. Though the exact effects depend on climate, season and other factors it can, from these sources be expected that if changes in barometric pressure can be consistently measured these can be used for weather estimation in a Machine Learning setting. In this setting, the Machine Learning algorithm acts as an operator which learns the effects of pressure on weather condition in its training scenario.

The effect size of pressure and its rate of change can be estimated from Table 1.2 and [16]. The table in [16] uses a step size 680Pa to discretise absolute pressure, and Table 1.2 uses intervals of 100Pa/hr to discretise rates of change.

Instead of relying on relations between air pressure and weather conditions made a long time ago for other continents, recent data from the Netherlands can be used to get an idea of the effects and their sizes

³Represented as 0.20 inches of Mercury pressure in the original work.

Table 1.2: Rate of change definitions in barometric pressure from [54], converted to SI units.

Class	Change [Pa/hr]
Slow	0 - 100
Moderate	100 - 200
Rapid	200 - 400

as well. Figure 1.6 shows that rainfall is much more associated with low absolute pressure levels ($< 980hPa$) than it is with higher levels. The same way, it indicates an association of fog with high pressure, occurring almost exclusively at pressures over $> 1020hPa$.

Figure 1.7 shows the association of rain with the rate of change in air pressure well, and agrees with the finding in [16] that rapid changes in air pressure (both rising and falling, $> \pm 200Pa/hr$) often occur together with rain. An association with fog here is not very visible though, for temperature changes in the $-10^{\circ}C/hr - 10^{\circ}C/hr$ the distribution seems somewhat skewed toward rising temperatures. As such, a measurement resolution of about $100Pa$ is considered reasonable, and both air pressure (in kPa) and its time derivative (in kPa/hr , calculated over 5 minutes) are added to the weather metrics.

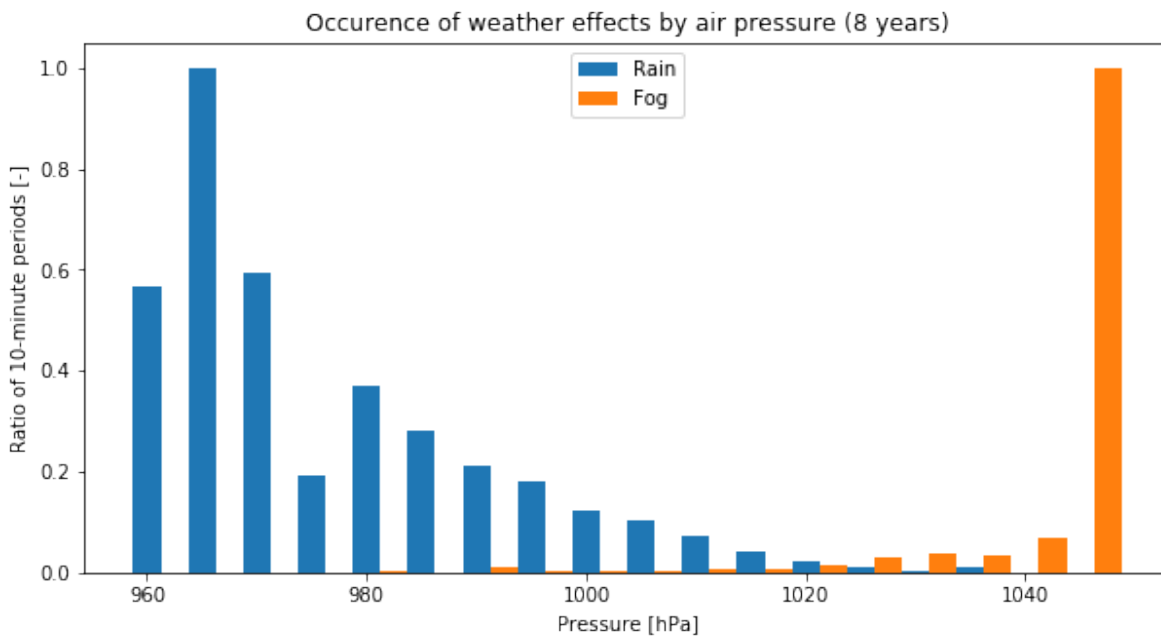


Figure 1.6: Relative rain/fog frequency versus air pressure. This shows the ratio all data points grouped by air pressure value (hPa), that contain rain (blue) or fog (orange). Created from hourly rain data over a 16-year period in the Netherlands. Data from CESAR [12].

1.2.2. Calculated weather metrics

This section gives background and a few metrics calculated from weather sensor data that are commonly used for local weather prediction. These metrics are used as independent variables for classification, along with the measured variables.

Dew point

The *Dew point* is the temperature at which air, at the current Relative Humidity, is *saturated* with water [53]. In other words, if, at some relative humidity level, the temperature drops to the dew point temperature, it is saturated with water. If the temperature drops even further, air will condensate forming drops which may present themselves as fog, a phenomenon known since at least 1917 [51]. The relation of the dew point with temperature and relative humidity is usually calculated using empirical approximations, of which the *Magnus formula* [29] is a well-known variant. The accuracy of the approximation is around $\pm 1^{\circ}C$ at normal outside temperatures ($0 - 30^{\circ}C$) and humidity levels ($50\% - 80\%$) [29]. Alternatives exist that are accurate at more

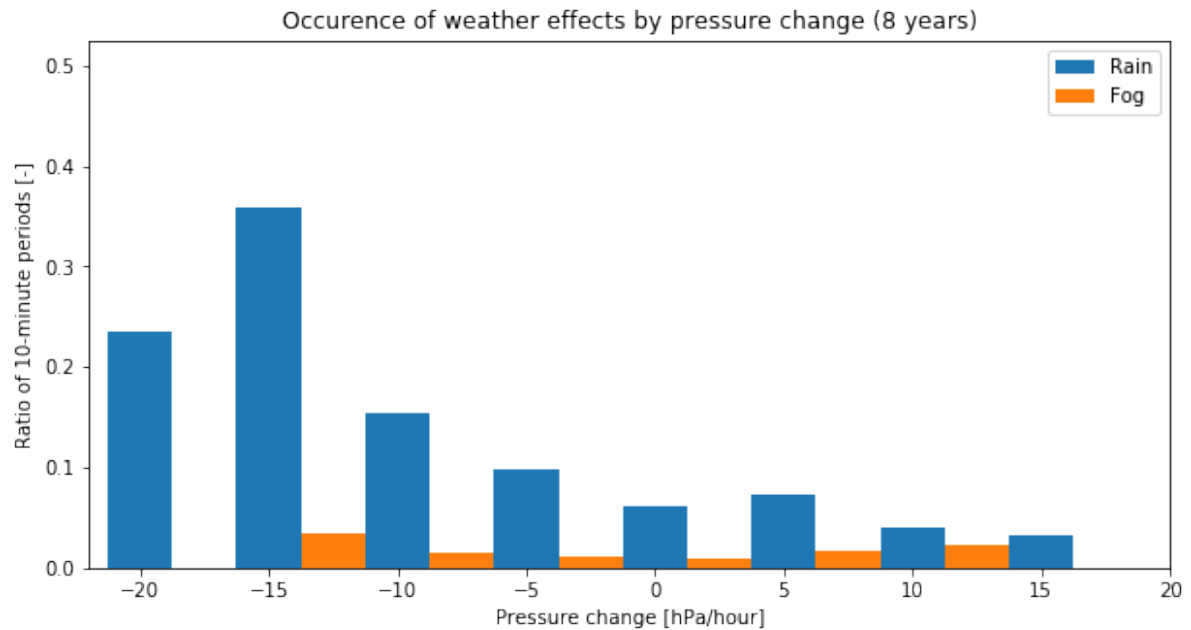


Figure 1.7: Relative rain/fog frequency versus air pressure change. This shows the ratio all data points grouped by air pressure change ($[hPa/hr]$), that contain rain (blue) or fog (orange). Created from hourly rain data over a 16-year period in the Netherlands. Data from CESAR [12].

tropical data ranges (e.g. the Arden Buck [10] equation). For normal, outside measurements, the accuracy and operating range of the Magnus formula are considered sufficient here.

The Magnus formula, for temperature T [$^{\circ}C$] and relative humidity H [%] is defined as [29]:

$$\begin{cases} T_d(T, H) &= (c \cdot \gamma(T, H)) / (b - \gamma(T, H)) & [^{\circ}C] \\ \gamma(T, H) &= \ln\left(\frac{H}{100}\right) + \frac{b \cdot T}{c + T} & [mbar] \\ b &= 17.63 & [mbar] \\ c &= 243.0 & [^{\circ}C] \end{cases} \quad (1.8)$$

The formula uses two coefficients b and c , and, as for the approximation, know many variants with varying accuracy. The coefficient values used here are taken from [29].

Dew point distance

As shown above, temperatures at or below the current dew point temperature often indicate fog. To aid the classifier, this notion is directly encoded into the derived weather sensor metrics, by adding the dew point distance metric ΔD :

$$\Delta D = T - T_d \quad (1.9)$$

Note that this is not an absolute distance because the sign is important here: temperatures slightly below the dew point temperature have different implications than temperatures slightly over it. This shows in Figure 1.5: a ΔD value of 0 is the point where the relative humidity crosses the 100% line.

1.3. Weather conditions

The number of conceivable weather conditions is to consider infinite. To find out which weather conditions are relevant and practical for consideration in this research, a broad categorisation, based on human perception of weather conditions is used here:

- **Precipitation:** Water particles, in any of its different states, fall down from higher regions of the atmosphere where they are formed due to thermodynamic process. Examples are *rain* (liquid), *hail* (solid), *snow* (crystalline) or mixtures thereof.

- **Suspension:** Water particles are formed but suspended in the air at ground level. This is generally referred to as fog and may have different names based on its cause (ground fog, ice fog, etc).
- **Temperature:** Ground temperatures below the freezing point of water are generally denoted as *frost*, high temperatures (often combined with high humidity) are described as *heat waves*.
- **Wind:** Though always present in some way, wind may be defined as a weather condition if it exceeds some intensity, like a *breeze*, *gale* or *storm*.
- **Lightning:** Electronic discharge between clouds or between a cloud and the earth surface. Lightning combined with strong winds and rain it is often called a *thunderstorm*.

In non-extreme cases, low temperatures, wind, and lighting generally do not have much influence on driving conditions. Precipitation and suspension however are known to reduce visibility. Additionally, precipitation causes roads to become wet (or icy if combined with or followed by frost) which is known to reduce road friction. As extreme cases occur infrequently by definition, only more prevalent effects (precipitation and suspension) are considered interesting and practical enough to consider in this research. Since the distinction between sub-classifications of both precipitation and suspension are difficult to make and the difference between their effects on driving are considered too small to include sub-classes like *hail* or *snow*.

In summary, the two weather conditions considered here the general classes of precipitation-based (all denoted *Rain*) and suspension-based effects (all denoted *Fog*). In situations in which neither of these conditions occurs are denoted *Dry*, as both rain and fog are, in fact forms of moisture.

1.4. Classification evaluation metrics

Machine-learned classification is the process of assigning a *class* to some observation based on its *attributes*. A *classifier* is a model that is created, often using a set of observation with known classes, to automate this process. The class variable used in this research (weather condition) is categorical and mutually exclusive. Classifiers thus map some set of observations (cardinal variables) to one element of a set of classes.

Most classifier can either output the predicted class for some observation (*hard* classification), or a probability distribution over the known classes (*soft* or *probabilistic* classification). Given the label is known for some observation, the performance of the classifier can be assessed by observing the difference between the classes predicted by the classifier and the known classes. This section presents four evaluation metrics and their implications for classifier performance.

Definitions and notation

A single observation is denoted \mathbf{x} and belongs to some class y . The classifier may output either:

- A hard class prediction \hat{y}
- A class probability distribution $\hat{\mathbf{z}}_c$ with elements $p(\hat{y} = c|\mathbf{x})$ for each class c known to the classifier.

The set of N observations used for evaluation is denoted $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1 \dots \mathbf{x}_N]$ and its associated set of known class labels $\mathbf{y} = [y_0, y_1 \dots y_n]$. The classifier produces the set (hard) of class predictions $\hat{\mathbf{y}} = [\hat{y}_0 \dots \hat{y}_N]$ or (soft) class probability distributions $\hat{\mathbf{z}} = [\hat{\mathbf{z}}_{0,c}, \hat{\mathbf{z}}_{1,c} \dots \hat{\mathbf{z}}_{N,c}]$.

Each accuracy metric is normalised so that its value is 1 if classification is perfect and 0 if classification bears no information at all.

1.4.1. Accuracy

The most straightforward performance metric is accuracy, which is the ratio of correctly predicted classes. It operates on hard classifications and does not take class imbalance in account. The accuracy *acc* for a set of predictions $\hat{\mathbf{y}} = [\hat{y}_0 \dots \hat{y}_N]$ with known classes $\mathbf{y} = [y_0 \dots y_N]$ is defined as:

$$acc = \frac{1}{N} \sum_{i=0}^N \mathbb{1}(y_n = \hat{y}_n) \quad (1.10)$$

Here $\mathbb{1}(x)$ is the *indicator function* which has value 1 if x is true and 0 otherwise.

1.4.2. Class-weighted F1 score

If the distribution of the labels y among the used dataset is imbalanced, the accuracy metric may be skewed and misrepresent classifier performance. For example, if a classifier is evaluated on a dataset with 90 observations of class \mathcal{A} and 10 observations of class \mathcal{B} , a classifier that assigns any observation to 'A' would receive an accuracy score of 0.9 while misclassifying 100% of class- \mathcal{B} points. The F1-score, originally defined in [50] operates on hard classifications and binary classes. It computes the true positive (tp), false positive (fp) and false negative (fn) ratios and defines, using these ratios, measures of *precision* and *selectivity*.

Since the metric considers binary classification, it is evaluated in a multi-class setting using the one-versus-all method [35]. The metric is evaluated for each class and the resulting score is the mean of each class weighted by its relative frequency in the dataset. As such, it accounts for class imbalances as well. The F1-score for some class j is defined as[50]:

$$\left\{ \begin{array}{l} tp \\ fp \\ fn \\ precision \\ recall \\ F1_j \end{array} \right. = \left\{ \begin{array}{l} \sum_{i=0}^N \mathbb{1}(y_i = j) \cdot \mathbb{1}(\hat{y}_i = j) \\ \sum_{i=0}^N \mathbb{1}(y_i \neq j) \cdot \mathbb{1}(\hat{y}_i = j) \\ \sum_{i=0}^N \mathbb{1}(y_i = j) \cdot \mathbb{1}(\hat{y}_i \neq j) \\ \frac{tp}{tp+fp} \\ \frac{tp}{tp+fn} \\ 2 \cdot \frac{precision \cdot recall}{precision+recall} \end{array} \right. \quad (1.11)$$

The class-weighted F1-score for J classes $[0\dots j]$ is here denoted $F1_{cw}$ and defined as:

$$F1_{cw} = \frac{1}{J} \sum_j \left[F1_j \cdot \frac{\sum_{i=0}^N \mathbb{1}(\hat{y}_i = j)}{N} \right] \quad (1.12)$$

1.4.3. Mean Average Error

The Mean Average Error (MAE) considers the total difference between the predicted class probability distribution and real class probability distribution. It is a linear metric that is influenced equally by small and large deviations from the true value and is defined as:

$$MAE = \frac{1}{2N} \sum_{i=0}^N \sum_{j=0}^J (|z_{i,j} - \hat{z}_{i,j}|) \quad (1.13)$$

As the MAE is an error metric that ranges from 0 to 1, the reported performance with regard to MAE is $1 - MAE$

1.4.4. Normalised Root Mean Squared Error

The Normalised Root Mean Squared Error (NRMSE) is a quadratic metric and considers soft classification. It penalises small deviations from the true distribution less than large deviations, making it less sensitive to noise in the result (which is usually small and around the true value) and more sensitive to bias (structural deviation). It is the sum of the squared differences in true and predicted probability distribution, for each class and observation:

$$NRMSE = \frac{1}{2N} \sqrt{\sum_{i=0}^N \sum_{j=0}^J (z_{i,j} - \hat{z}_{i,j})^2} \quad (1.14)$$

Like MAE, the NRMSE is an error metric, and with this definition also ranges from 0 to 1, the performance regarding this metric is shown as $1 - NRMSE$

1.5. Prediction Fusion

This section presents two simple, probabilistic methods that can be used to combine several classifier predictions: The Naive Bayes Classifier (NBC) [35] and the Hidden Markov Model (HMM) [35, 40]. When viewed

broadly, the difference between both methods is subtle. To produce an estimate at some point in time, the NBC uses only the current observations (classifier predictions) and the á priori estimate (or *prior*), whereas the HMM uses the previous model estimate and a learned temporal relation between consecutive estimates.

1.5.1. Naive Bayes Classifier

The Naive Bayes Classifier[35] (NBC) models a class probability distribution of multiple variables that are conditionally independent that class. The NBC is a specialisation of the more general Naive Bayesian Model, where the outputs are restricted to the discrete, finite classes. In the NBC, the probability distribution of some conditionally independent set of N variables $\mathbf{x} = [x_1 \dots x_N]$, given a set of modelling parameters θ , data set \mathcal{D} and class label y is modeled by equation 3.54 in [35]:

$$p(\mathbf{x}|y = c, \theta) = \prod_{i=1}^N p(x_i|y = c, \theta) \quad (1.15)$$

Since this is a generative model, Bayes' rule is used to invert the conditioning. To avoid overfitting, the term $p(y = c|\mathcal{D})$ can be introduced representing the prior probability of $[y = c]$ based on some dataset \mathcal{D} . Prediction is given by equation 3.63 in [35]:

$$p(y = c|\mathbf{x}) \propto \underbrace{p(y = c|\mathcal{D})}_{\text{Prior}} \cdot \underbrace{\prod_{i=1}^N p(x_i|y = c, \mathcal{D})}_{\text{Observation}} \quad (1.16)$$

This equation gives the probability that observation \mathbf{x} belongs to class c as a function of two terms: the prior probability $p(y = c|\mathcal{D})$ (the probability of *any* observation belonging to class c), and the probability of obtaining observation \mathbf{x} given that the observation originates from class $[y = c]$.

This model can be used to combine outputs from different classifiers if their predictions are conditionally independent [2]. That is, the estimation results should be independent *given the weather condition*. This is an intuitive, almost obvious assumption in the problem presented here: there should be no factors that influence both the weather sensor model and visual model in the same way apart from the weather condition.

Given some set of classifiers $C_0 \dots C_N$ that produce conditionally independent estimations of the same variable, we can denote their outputs $x_0 \dots x_N$. If the classifiers are evaluated on unseen data, a reasonable estimation of the conditional probability $p(x_j|y = c)$ can be obtained by counting the cases [2] in which classifier j produced prediction x_j given the known class was c . In fact, in the case of a finite, discrete set of classes, such a probability distribution is an often-used tool to assess the performance of a classifier called the *confusion matrix*. The structure of such a matrix is for a three-class situation is shown in Table 1.3 and contains all the probability estimates required to compute the second term on the RHS of Equation 1.16. The other term, the prior $p(y = c|\mathcal{D})$ is easily computed by counting each of the class labels in \mathcal{D} . Alternatively, a different prior may be inserted probability if the real-world representation requires so. A schematic of the Naive Bayesian classifier is shown in Figure 1.8.

Table 1.3: Definition of the confusion matrix for a three-class classifier containing probability values in each cell. The classes are labeled 1, 2 and 3

True class	Predicted class		
	1	2	3
1	$p(x_j = 1 y = 1)$	$p(x_j = 2 y = 1)$	$p(x_j = 3 y = 1)$
2	$p(x_j = 1 y = 2)$	$p(x_j = 2 y = 2)$	$p(x_j = 3 y = 2)$
3	$p(x_j = 1 y = 3)$	$p(x_j = 2 y = 3)$	$p(x_j = 3 y = 3)$

Combining predictions of the same variable (the predicted variable or *predictand*) from different classifiers using this NBC classifier can as such be done using the following process:

1. Train a set of classifiers $C_1 \dots C_N$ on training data
2. For each classifier C_i , evaluate the classifier on the validation set with known labels \mathbf{y} , yielding predictions $\hat{\mathbf{y}}$. The relative frequencies of all the combinations $(\hat{\mathbf{y}}, \mathbf{y})$ are counted and used to construct observation matrix \mathbf{F}_i

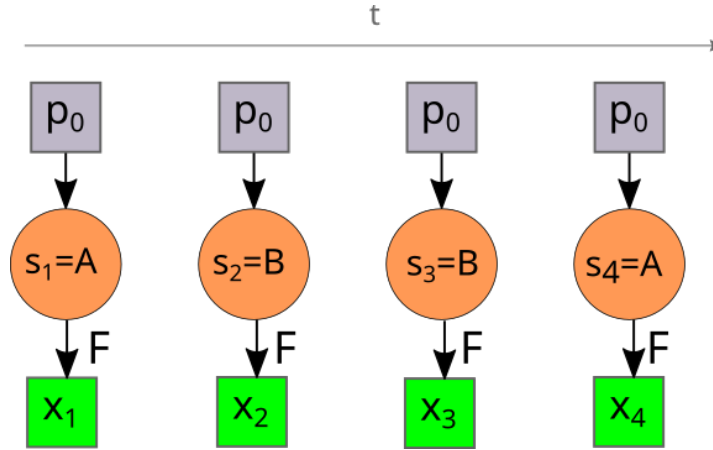


Figure 1.8: Schematic of the prediction process using a Naive Bayesian Model. All considered observations at some time step t are denoted \mathbf{x}_t , the predictand is the hidden variable \mathbf{s}_t . At each step, the prior probability distribution p_0 of the hidden variable is known, and the observation matrix \mathbf{F} are use to estimate \mathbf{s}_t . In the case where the space of \mathbf{s} consists of finite, discrete classes, the model is called a Naive Bayesian Classifier or NBC. The constant matrix or function \mathbf{F} is called the *observation matrix* contains the conditional probabilities $p(\mathbf{x}|\mathbf{s})$. Note that the prediction of \mathbf{s} is independent of the time-step here, a major difference with the Hidden Markov Model.

3. Substitute the values in Equation 1.16 and normalise

In the case that the classifier output is a probability distribution over the classes (i.e. soft prediction result $\hat{\mathbf{z}}_{i,j}$ one can pre-substitute Equation 1.16 the equation for combined prediction result \hat{y}_i :

$$p(\hat{y}_i = c) \propto \underbrace{p(y = c)}_{\text{Prior}} \cdot \underbrace{\prod_{j=1}^N \hat{\mathbf{z}}_{i,j} \mathbf{F}_j}_{\text{Observation}} \quad (1.17)$$

1.5.2. Hidden Markov Model

The Hidden Markov Model [40] (HMM) is a generalisation of the *Markov Chain* [35] that allows for the use of observation data. It makes use of Observation Matrices like the NBC, but models a temporal relation between the current state and one or more previous states. A first-order Hidden Markov Model describes the two relations between some non-directly observable (ie. *hidden*) state at the current time instance t and [35]:

1. The **observations** at this current time instance, through an **observation model**
2. The **previous** hidden state, through a **transition model**

A schematic of the Hidden Markov Model is shown in Figure 1.9, the differences with Figure 1.8 are clear: the insertion of the prior at every time step is replaced with a temporal relation wit the previous time step.

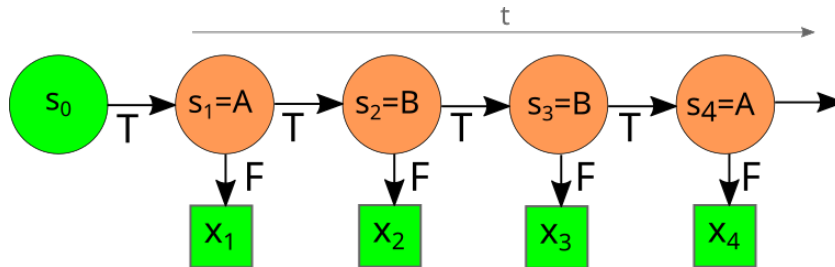


Figure 1.9: Schematic of the prediction process using a Hidden Markov Model. The difference with the NBC is that the estimation $\hat{\mathbf{s}}_{t-1}$ are used for estimation and the prior probability p_0 is omitted. The transition matrix \mathbf{T} contains conditional probabilities $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ Refer to Figure 1.8 for symbol definitions.

The hidden state in the example of the previous section is the actual weather condition at time t denoted y_t : if it were directly observable we would not need to estimate it. The previous hidden state value, then, is

the weather condition at the previous time-step y_{t-1} . Since it is, by definition, unobservable, the estimation result at the previous time-step \hat{y}_{t-1} is used. The observations are the classifier outputs \mathbf{x} . For prediction, the relations are written as:

$$p(\hat{y}_{t,i} = c) \propto \underbrace{p(\hat{y}_{t,i} = c | \hat{y}_{t-1,i} = c)}_{\text{Transition}} \cdot \underbrace{p(\hat{y}_i = c | \mathbf{x})}_{\text{Observation}} \quad (1.18)$$

The observation model is the same as in the previous section. Since Markov Models are time-invariant, the conditional probability $p(\hat{y}_i = c | \hat{y}_{i-1} = c)$ is modelled constant. The approach shown in the previous section can be used with a few changes: instead of Equation 1.16, the RHS of Equation 1.18 is substituted. The prior is omitted, but the new term $p(\hat{y}_{t,i} = c | \hat{y}_{t-1,i} = c)$ must be inserted instead.

This term is expressed as a transition matrix $T_{i,j}$, in which element i, j represents the conditional probability $p(y_t = i | y_{t-1} = j)$. The transition matrix \mathbf{T} has the same dimensions as the observation matrices, and is found by determining the relative frequency of each transition [$y_{t-1} = j \rightarrow y_t = i$] in the training set. With transition matrix \mathbf{T} , the confusion matrices \mathbf{F}_j and the notation from the previous section we can write the prediction equation as:

$$p(\hat{y}_{t,i} = c) \propto \underbrace{\hat{\mathbf{y}}_{t-1} \mathbf{T}}_{\text{Transition}} \cdot \underbrace{\prod_{j=1}^N \hat{\mathbf{z}}_{i,j} \mathbf{F}_j}_{\text{Observation}} \quad (1.19)$$

To start the 'chain' depicted in Figure 1.9, a prediction for the first time step $\hat{\mathbf{y}}_0$ is required. Before prediction starts, there is no available observation. Therefore, the probability for each class given *no* information is substituted, which is the prior:

$$p(\hat{y}_{0,i} = c) = p(y_i = c) \quad (1.20)$$

2

Dataset

In order to build and verify a weather estimation system as described in the introduction, there is a need for accurate, labeled weather data. As this model should consider both visual data (camera images) and weather sensor data, it would be ideal for generalisation to have a dataset comprising both these data components, preferably recorded in a driving situation. In fact, an ideal dataset would satisfy all of the following requirements:

- It contains both visual (images) and weather sensor (measurements) data so that is directly applicable
- The visual and weather sensor data are recorded in exactly the same situation, that is, at the same moment and location, so that a joint prediction can be made for a single moment
- Data is recorded in a real-world driving scenario as this makes generalisation much easier and there is no *transfer of learning* problem.
- All of the data points are labeled using verifiable and relevant weather conditions. In the case that the data needs to be recorded, Rain and Fog seem the most promising classes for this research as, from the conditions that may affect driving, these are the only somewhat prevalent ones in the Netherlands, as discussed in Section 1.3.

2.1. Available datasets

The process of data collection requires a lot of effort and resources. First of all, training Machine Learning models often requires a lot of data. Depending on variance and dimensionality of the data, numbers of data points in the order of 100 to 100.000 may be required to train a basic model. Additionally, resources like time, a measurement vehicle, and operators would be required to collect a dataset satisfying the criteria. Therefore, making use of existing datasets may save a lot of time and resources. This section discusses a few available datasets that seem to satisfy at least a some of the requirements, and their applicability/usability to this research.

2.1.1. MWI Dataset

The *MWI Dataset* [59] is made available by the authors for research purposes. The dataset contains a large number of images (± 20.000 images) and considers four weather conditions (images are labeled *Sunny*, *Hazy*, *Snowy* and *Rainy*). Figure 2.1 shows a few sample images from the MWI dataset. It is a great source of visual data, but since the images are taken from internet sources rather than taken in driving conditions, the variance is large and the set does not contain many driving-related situations. The data is labeled independently by the authors and volunteers without knowledge of the original weather situation in which the photo was taken, and as such does not contain weather sensor data.

2.1.2. Oxford robotcar Dataset

The *Oxford RobotCar Dataset*[34] is the result of ongoing robotic vehicle research at Oxford University. The public data repository provides vast amounts of driving footage (over 100 recordings along one route), along

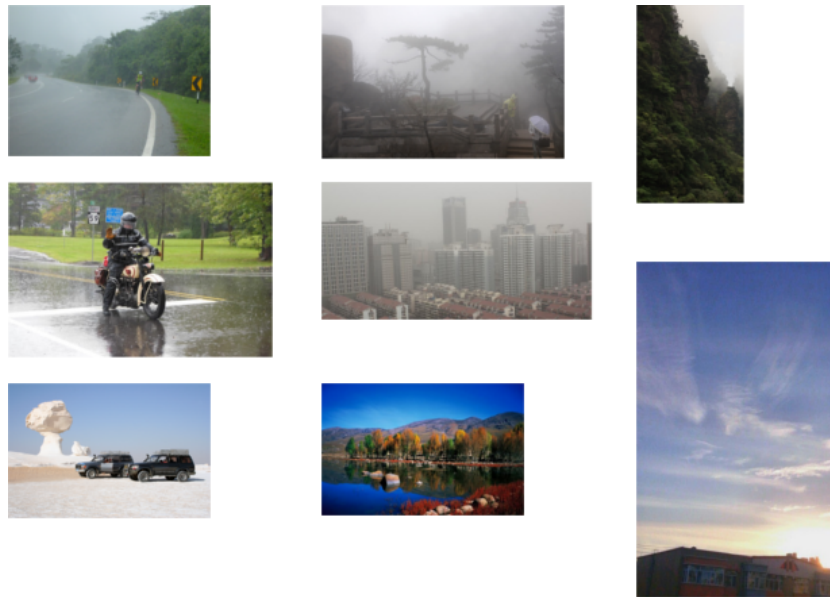


Figure 2.1: Sample images from the MWI dataset[59]. The dataset contains a total of 20,000 images with four weather classes (Sunny, Rainy, Hazy and Snowy).

with sensor data from vehicle sensors. Each recording in the dataset is labeled with a single weather condition. An example showing two datasets is shown in Figure 2.2. As with the MWI dataset, this dataset also lacks weather sensor data, and the origin of the tags describing weather conditions is unclear.



Figure 2.2: Index page showing three dataset samples from the Oxford Robotcar Database[34].

2.1.3. CESAR Dataset

The *CESAR Database* [12] is collected at the Cabauw Experimental Site for Atmospheric Research (CESAR), a weather observatory located in Cabauw, the Netherlands. The weather station provides historic weather data including rainfall amounts and visibility distance and images from a camera located at the site. The weather data is available for recent years (2008-present) at 10-minute interval and at 1-hour intervals for years prior to 2001. The data is recorded at a static weather station and also includes visual data (photos of a field taken from the observatory), but is not recorded in a driving situation. As such, it is not useful for training a Vision classifier, but may be very useful for inference based on weather sensor data.

2.1.4. KNMI dataset

The Royal Dutch Meteorology Institute (KNMI) provides historical weather data at a one-hour temporal resolution for 50 different observatories in the Netherlands. The data includes temperature, rainfall, visibility

distance and other data, but no visual data.

2.2. Data collection

The search for applicable datasets discussed in Section 2.1, did seem to yield any dataset that was public and satisfies all the applicability requirements. Therefore, over the course of eight months, a dataset was collected for the purpose of creating the model. This section discusses process and results of this dataset collection.

2.2.1. Equipment

A camera-equipped Toyota Prius, provided to the TU Delft by the Dutch Autonomous Vehicle Initiative¹ was used to collect data. Visual data was collected in the form of video frames using a camera mounted under the central rear-view mirror in the vehicle. To collect weather sensor data, the vehicle was equipped with a Barometer and a digital Hygrometer/Thermometer. Details of these sensors are shown in Table 2.1.



Figure 2.3: Sensor placement position, the rectangular opening around the latch lets air through for measurements by the barometer. The red circle shows the latch behind which the weather sensors are placed.

Table 2.1: Details of the weather sensors used for data collection

Quantity	Sensor	Sensor model	Resolution	Accuracy	Datasheet
Air pressure	Barometer	NXP MPL115A2	0.15kPa	±1kPa	[44]
Outside temperature	Thermometer	Aosong DHT22	0.1°C	±0.5°C	[31]
Relative air humidity	Hygrometer	Aosong DHT22	0.1%	±5%	[31]

2.2.2. Weather sensor placement

The weather sensors are placed behind a small opening in the bodywork near the left rear wheel of the vehicle as shown in Figure 2.3. This placement was chosen in order to:

1. Keep the sensors as close as possible to the outside air in order to measure the actual outside temperature and humidity, and avoid influence of the vehicle's heating/air conditioning system.
2. Keep the sensors out of the air stream around the vehicle when it is moving. This is in order to avoid pressure readings being affected by the air flow velocity (by Bernoulli's Principle).

¹DAVI, <http://davi.connekt.nl/>

It was ensured that measurements inside the vehicle were not influenced by turning the vehicle heating or air conditioning on/off and comparing readings. No pressure effects were found to occur resulting from vehicle velocity.

2.2.3. Camera

The camera used is an UI3060CP Rev.2 by IDS Imaging[21]. It is a global-shutter camera with a 1936×1216 resolution and equipped with a LM8HC 8mm lens by Kowa American Corp. It is mounted close to the windshield, about 10cm below the rear-view mirror. As it is the right camera of a stereo camera setup, its vertical position is 20cm right of the vertical center of the windshield. A photo showing the camera placement is shown in Figure 2.4.



Figure 2.4: Interior photo showing camera placement. The red circle indicates the camera location. The camera orientation is horizontally and vertically level. Image taken from passenger side.

2.2.4. Driving route

In order to create a model that is valid in different scenery types, it is important to collect data from different scenery situations. To collect such data, a driving route is carefully chosen that contains city, industrial and highway scenery. To be able to assess the variance in the data due to the weather and thus minimise the variance originating from the route itself, every recording was done along the same route where possible. This choice also has the advantage that all of the collected datasets are roughly the same length. The route is shown in Figure 2.5. The route starts and ends at the Delft University and takes around 20 minutes (depending on traffic) from start to finish.

2.2.5. Driving frequency and data sampling

During a period of around eight months, the route shown in Figure 2.5 was driven on one or two occasions every week (see Table 2.2 for the full schedule). During these recordings data was collected using both the camera and weather sensors. Full-color video frames from the camera were recorded at one-second intervals (1Hz), to produce around 1200 frames per recording. Weather data from the sensors was recorded at 2-second intervals (1/2Hz) and linearly interpolated to match timestamps of the camera images. The weather sensor recording rate was chosen as the DHT22 sensor is known to be susceptible to self-heating effects [31] at faster rates.

In total, 21 recordings were made, totalling 390 minutes of driving time, or around 23.000 data points. Of these recordings, two were excluded: In one case the Hygrometer got wet during the recording and stopped working, in the other case camera recording stopped working after some time during the recording due to

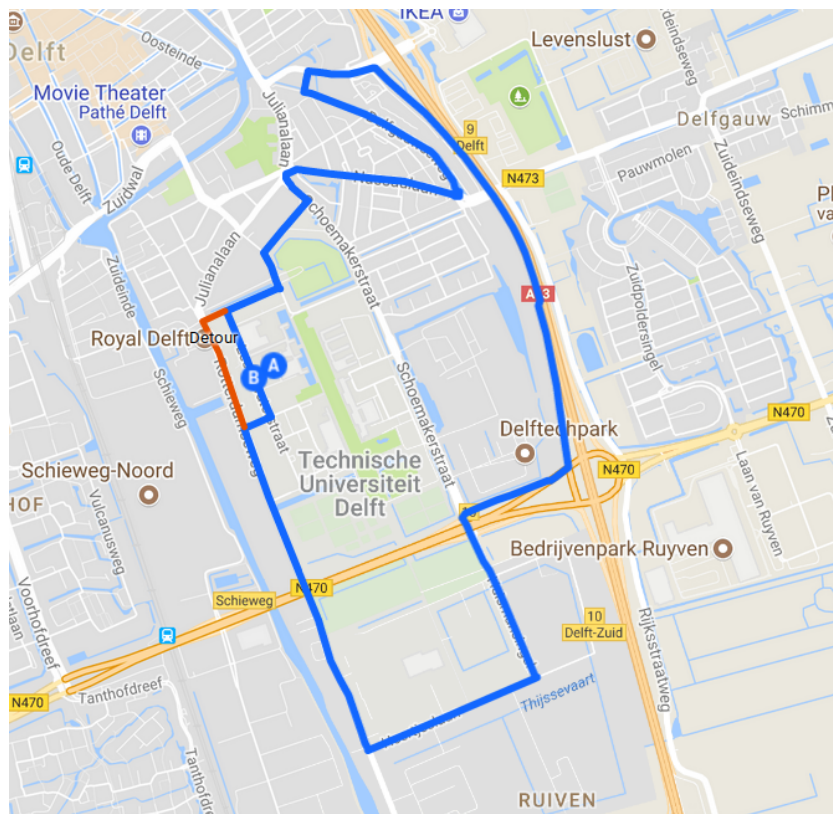


Figure 2.5: Map showing the standard driving route. The blue line shows the standard route, the red route shows a detour taken during roadworks. The route was always driven clockwise.

software malfunction. Table 2.2 gives an overview of the recorded datasets. To prevent effects of camera underexposure, all recordings were done during daylight, at least one hour after dawn and one hour before sunset.

2.2.6. Data collection results

The available raw weather sensor data at each time step is shown in Table 2.3. The visual data at each time step consists of a video frame, which is a 1936×1216 array of pixels that with values between 0 and 255 for each color channel (red, green and blue). The total dimensionality of visual data is thus $w \times h \times 3$ and its integer range 0..255. After interpolation and exclusion of bad datasets 20938 image/weather sensor combinations are available. An example of raw data at a single recording is shown in Figure 2.6, three example frames are shown in Figure 2.7.

2.2.7. Visual data processing

Visual data is pre-processed using feature extraction. The feature extraction process is described in Section 1.1. The result of the feature extraction is a 91D vector containing floating-point values, rounded to 3 decimals, between 0.000 and 1.0000 for each feature/ROI combination.

2.2.8. Sensor data processing

After calculating the derived metrics and time derivatives for the weather sensor data, the used data for each time instance is an 8-Dimensional feature vector. An overview of the calculated quantities and derivatives is shown in Table 2.4.

Table 2.2: Overview of collected datasets (rides), the date/times and general weather observations. These observations are made by the operator and not used for labelling. For the time of day, *Morning* rides took place between 08:00 and 12:00, *Early afternoon* rides between 12:00 and 14:00 and *Afternoon* rides between 14:00 and 18:00.

Date	Time of day	Duration [s]	Weather	Notes
2017-05-19	Early afternoon	1054		Excluded
2017-06-02	Early afternoon	1056	Dry	
2017-06-09	Early afternoon	1028	Dry	
2017-06-16	Early afternoon	1168	Dry	
2017-06-22	Afternoon	990	Rainy	
2017-06-22	Afternoon	950	Rainy	
2017-06-30	Early afternoon	974		Excluded
2017-07-07	Early afternoon	1084	Dry	
2017-07-18	Early afternoon	950	Dry	
2017-09-01	Early afternoon	1052	Rainy	
2017-09-08	Morning	1006	Rainy	
2017-09-15	Morning	872	Dry	
2017-09-28	Morning	984	Dry	
2017-10-06	Morning	910	Rainy	
2017-10-11	Morning	964	Dry	
2017-11-20	Afternoon	1090	Rainy	
2017-11-23	Early afternoon	966	Rainy	
2017-11-27	Early afternoon	1052	Rainy	
2017-11-30	Early afternoon	1092	Rainy	
2017-12-01	Morning	1172	Foggy	
2017-12-07	Afternoon	974	Foggy	

Table 2.3: Overview of available weather sensor data at a single time instance

Quantity	Unit
Air pressure	<i>kPa</i>
Relative air humidity	%
Temperature	$^{\circ}\text{C}$

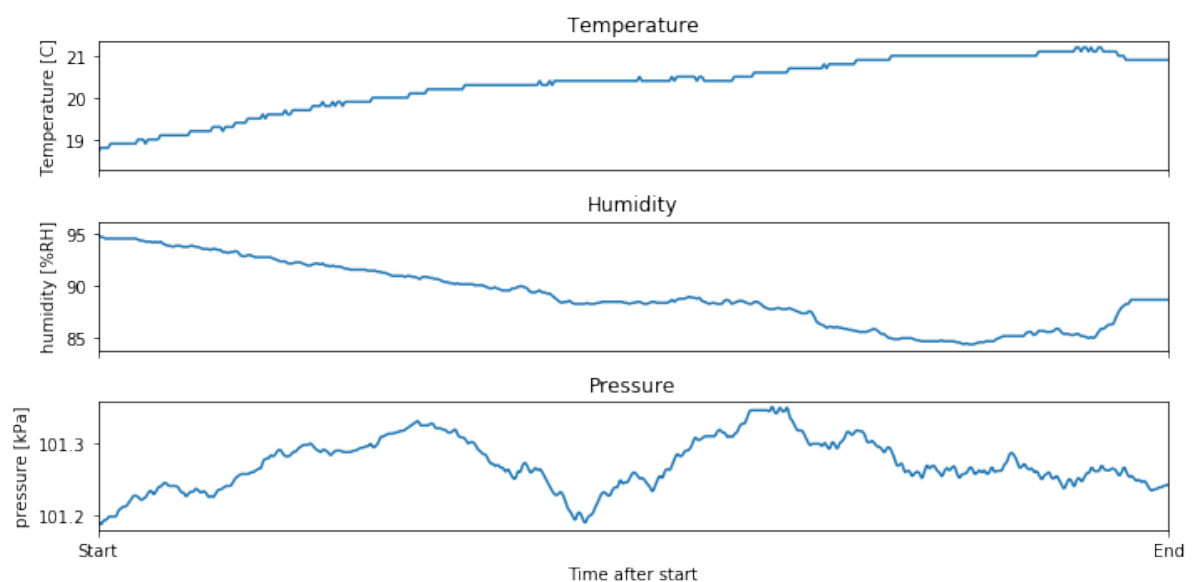


Figure 2.6: Example of raw data available for a certain recording (15th of September 2017). These show the measured weather sensor data: temperature, air pressure, and relative humidity. The total recording time from start to end for this ride is 15 minutes and 36 seconds.



Figure 2.7: Six examples of captured frames from the recording at September 1st 2017

Table 2.4: Sensor data and calculated features

Symbol	Quantity	Source
T	Outside temperature	Measured
P	Air pressure	Measured
H	Humidity	Measured
T_d	Dew point temperature	Section 1.2
ΔD	Dew point distance	$T - T_d$
dT_{60s}	10-minute derivative of T	Differentiation
dP_{60s}	10-minute derivative of P	Differentiation
dH_{60s}	10-minute derivative of H	Differentiation

2.3. Data labelling

As stated in the introduction, the goal of the research is to estimate weather conditions at the current moment, given image and weather sensor data. However, quantification of these weather conditions is not straightforward. This quantification has been discussed in Section 1.3 for this research, and resulted in three weather classes: *Rain*, *Fog*, and the ‘default’ *Dry* class as defined in Section 1.3. This section provides a more specific definition of these three weather classes, so that exact class labels can be assigned to data points.

2.3.1. Rain labels

Rain labels are calculated using rain rates (in mm of precipitation per hour) at 5-minute intervals and at a 1km-by-1km spatial resolution. The data is measured by the KNMI and distributed by Buienradar², a service that formats and provides this data for consumer use. The data is processed by taking rain data for the 1km-by-1km square where the recordings take place, for the nearest two 5-minute intervals around each time stamp. To match the (much shorter) measurement intervals, the rain rate is interpolated linearly between these data points. Figure 2.8 shows an example of a plot retrieved from the website, with the orange flag indicating the rain rate at a certain time (15:50) on some example day.

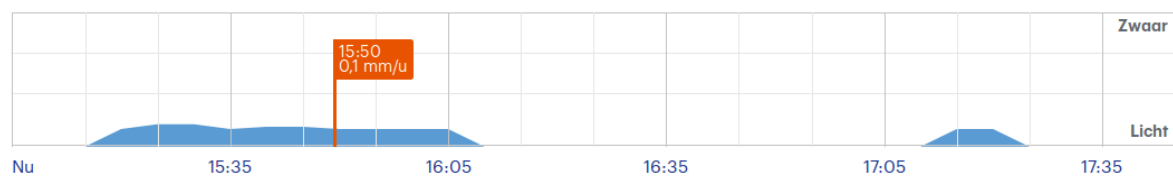


Figure 2.8: Example rain rate chart. This is a sample graph for some day in January. The horizontal axis shows the time of day, the vertical axis shows the rain rate (on some arbitrary scale from *light* to *heavy*). The tooltip shows the rain rate in *mm/hr* for 15:50. This graph considers a specific $1\text{ km} \times 1\text{ km}$ area. Chart from <https://www.buienradar.nl>

Label definition The American Meteorological Society (AMETSOC, [48]) defines rain rates below 0.01 in (0.254 mm) per hour as *trace*, or too light to measure. To eliminate questionable situations, a similar threshold is applied to the data here. Data points for which the rain rate exceeds 0.25 mm/hr are labeled *Rainy*. In related works like [43] two rain classes (*Light* and *Heavy*) are formulated, although without an exact definition. As situations defined by AMETSOC[48] as *Heavy rain* (rain rates exceeding 7.6 mm/hr) are not present in our dataset this class is omitted.

Although it would be interesting to collect heavy rain data, it is, in our situation, impractical to obtain as these situations occur only around five times per year in the Netherlands[23]³.

2.3.2. Fog labels

Fog data is provided as a visibility distance in meters, interpolated spatially from 24 observatories around the country. The data is also provided by Buienradar, and cross-checked with data from the two nearest observatories provided by the KNMI (see Section 2.1). The temporal interpolation of this visibility distance is carried out in the same way as for rain labels. Figure 2.9 shows an example of a visibility distance graph provided by Buienradar for a single weather station.

Label definition A threshold visibility distance is set at 1000 m . Data points with a visibility distance under 1000 m are labeled *Fog*, all other data points are labeled *Clear*. The 1000 m threshold corresponds to the international definition of Fog[47], and stems from practical considerations: Visibility of objects more than 1000 m away is, in driving situations, very often limited by other objects (e.g. vehicles, trees and guardrails). Finally, visibility limitations at distances greater than 1000 m are assumed to affect drivers very little, as situations more than 1000 m away usually gives gives them well over 20 seconds to anticipate⁴. There were no questionable situations, as for each datapoint in the collected data the two weather stations either both showed sub- 1000 m values or both showed values larger than 2000 m .

²<https://www.buienradar.nl>

³Table on page 5 (in Dutch)

⁴At driving speeds of $< 130\text{ km/h}$

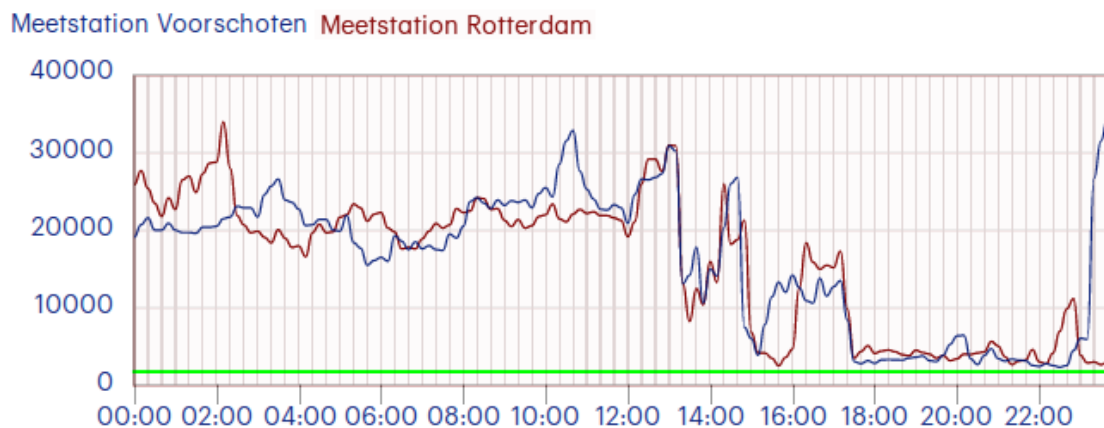


Figure 2.9: Example visibility distance chart for a given day and observatory (23rd of January 2018) and two different observatories (Rotterdam and Voorschoten). The red and blue lines represent data from the different observatories, the green line indicates the 1000m threshold used for fog labels. Image from <https://www.buienradar.nl>, merged by author.

2.3.3. Weather condition label

Data points labeled *Fog* are labeled as such no matter the corresponding *Rain* class. This choice is made as 1) rain and fog very rarely occur together and 2) visibility distances of less than 1000m caused by rain are very rare in the Netherlands [23], making it safe to assume that such situations are caused by fog. In the present dataset, rain and fog never occur together, so this choice does not change labels used.

2.3.4. Pre-processed dataset

The pre-processed dataset contains 19 recordings and 20.938 data points. The class distribution over the data points is shown in Table 2.5. A graph showing the data points and their class for each recording is shown in Figure 2.11. It should be noted that within this dataset, there is almost no variance in labels within the recordings: all recordings were either *Dry*, *Rainy* or *Foggy* from start to end with the exception of a 40-second period⁵.

An interesting observation from the Table 2.5 is that the fractions of Dry and Rainy data points (0.42 resp. 0.47) are much larger than the fraction of Foggy points (0.11). The main reason for this imbalance is the relative rarity of fog in the Netherlands: Fog occurs in less than 1% of daytime periods as shown in Figure 2.10. As such, it proved difficult to plan and execute a recording at the few and brief periods of fog that occurred during the recording period. It is known [30] that data imbalance may affect performance of learned classifiers if some class is underrepresented. The actual effect of the imbalance on the classifier training is discussed in Chapter 3.

Table 2.5: Overview of pre-processed dataset

Label	Measurements	Frames	Ratio
Dry	8.807	8.807	0.419
Rainy	9.779	9.779	0.469
Foggy	2.352	2.352	0.112
Total	20.938	20.938	1.000

⁵2017-11-23, between minutes 9 and 10. This is shown as the grey patch in the 5th bar from bottom in Figure 2.11

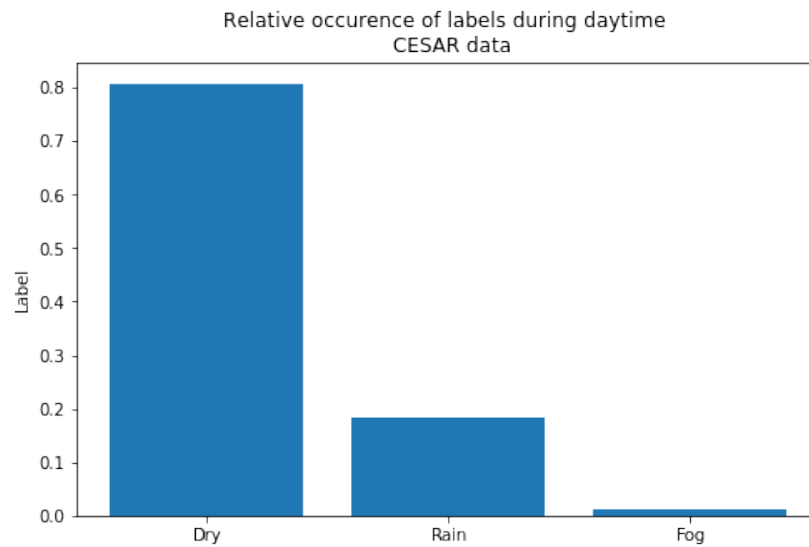


Figure 2.10: Frequency of the weather condition labels defined in Section 2.3. The histogram represents the frequency of 10-minute periods labeled Dry, Rainy or Foggy in 8 years of data (2008-2016) for Cabauw, The Netherlands. Data from the CESAR database [12].

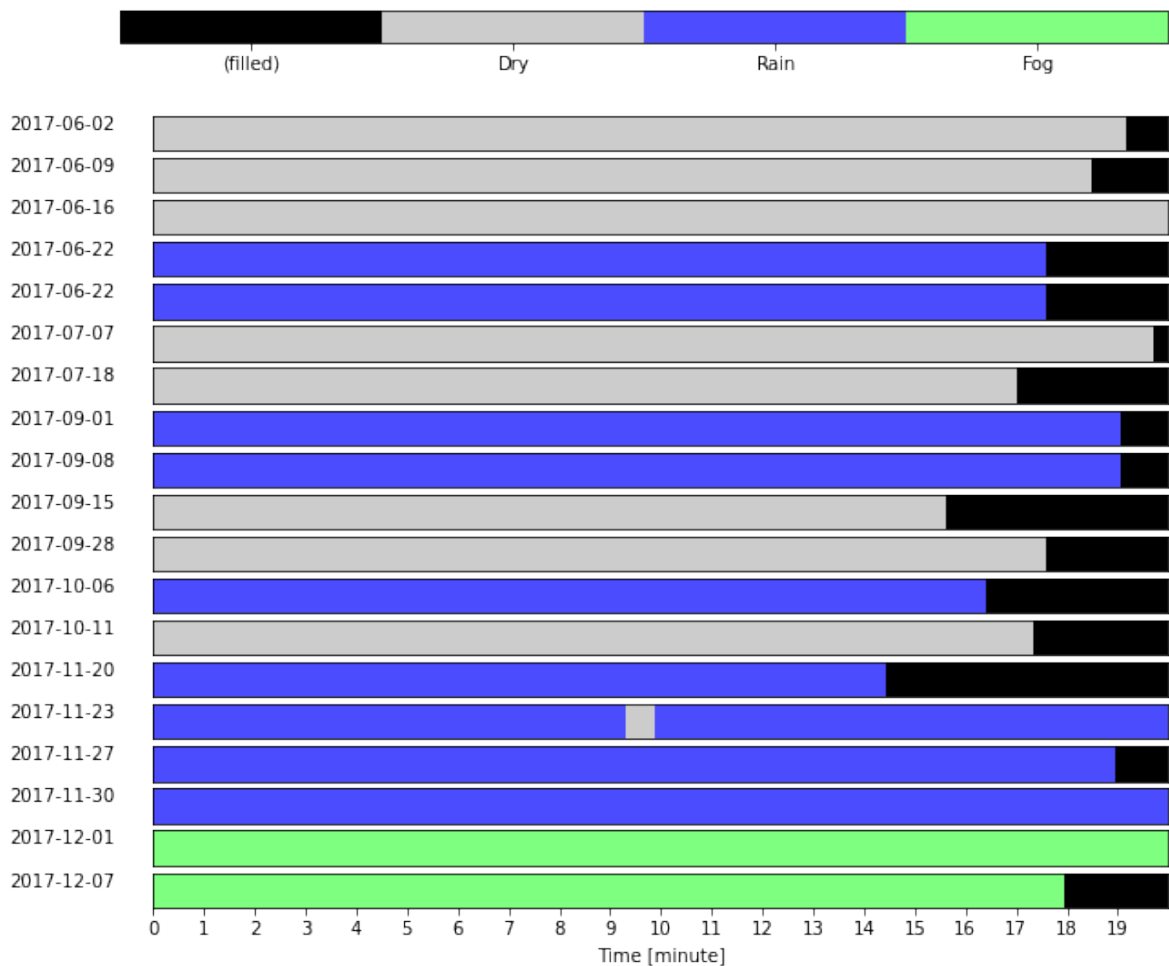


Figure 2.11: Overview of labels for each recording in the dataset. Each row corresponds to a single recording. The horizontal axis shows time, each color corresponds to a weather condition label. Black parts are only for display purposes. The titles left of each recording show the recording date.

3

Classifiers

This section describes the process of classifier construction for the vision and weather sensor data used here. The first subsection discusses type of classifier algorithm used and motivation for using this particular algorithm. The next subsection discusses classifier hyperparameters for each of the classifiers (vision and weather sensor data). Finally, the third subsection discusses the classification results for each component (vision and sensor data) in predicting the weather condition labels defined in Section 2.3.

3.1. Classifier algorithm

The used classifiers are called *Random Forests* [7, 20, 35]. A Random Forest is an ensemble of Decision Trees [35] classifiers that that are combined using e.g. Bagging or Boosting [7] techniques. Though the exact type of classifier algorithm does not seem to have a large influence on the result: with constant complexity, preliminary research showed similar results for k-NN (as used in [49]) or SVM (as in [43]). Works like [56] also show little difference (around 1%) in accuracy among different algorithms, for a similar problem. However, there are a few compelling reasons here to choose Random Forests over similar algorithms.

Decision Trees, the weak learners in Random Forests[8] are easy to interpret[35]¹ as they describe intuitive chains of single-variable decision boundaries (like a flowchart), and have no scaling issues. Though single Trees are known to have smaller accuracy on some problems than compared to other algorithms, the strong ensemble techniques like Bagging or Boosting used to construct the forest may (partly) solve this problem. An additional advantage is that Random Forests are simple to control: there are few hyperparameters that influence important effects (overfitting and the time-accuracy trade-off) quite directly. Since the goal is to obtain classifiers that can eventually be used to explore the possibilities of a combined model, and not to achieve the best single-classifier accuracy, this is considered an advantage.

The Random Forest implementation is provided by the `scikit-learn` Machine Learning toolkit[38]. Each Decision Tree in the Random Forest is trained using the modified CART[8] algorithm as implemented by `scikit-learn`. For decision scoring, the Gini index criterion is used[17]. The minimum number of datapoints for a split (2) and for a leaf node (1) are not limited. The other hyperparameters used in training Decision Forests to control overfitting and space complexity are optimised in the following section.

3.2. Hyperparameter optimisation

Overfitting control is done by limiting the maximum tree depth for each Decision Tree in the forest. The space-accuracy tradeoff is controlled by limiting the number of estimators The maximum tree depth (d_{max}) and number of Trees in for each forest are empirically optimised. An additional hyperparameter is introduced by subdividing the dataset, which is a necessity stemming from the format of the dataset (a set of independent recording sequences). Optimisation of these parameters is done independently for both classifiers, since the dimensionality and variance of the datasets are quite different, and they measure completely different quantities.

¹Page 550

In total, there are three hyperparameters to be determined:

- The number of estimators (Decision Trees) in the Random Forest n_{trees} , used to control the accuracy-complexity trade-off
- The maximum depth for the Decision Trees in the Random Forest, used to control overfitting
- The number of subdivisions n_{slices} for each recording discussed in Section 3.2.2

3.2.1. Number of Trees

The number of classifiers in an ensemble, or, in our case, the number of Decision Trees in each Decision Forest controls the trade-off of accuracy versus complexity. Time complexity scales linearly with the number of trees, and classifier accuracy increases with this number with diminishing returns [6]. As such, there should be some optimal value, for which the added complexity of training more trees does is no longer justified by ever shrinking increase in classifier accuracy.

Figure 3.1 shows the classifier accuracy versus tree depth for both classifiers and different numbers of trees. In general, larger numbers of trees show higher test set accuracies as expected, but the result varies with tree depth. The lines in figure 3.2 show the relative increase in classifier test accuracy per tree versus the number of trees. The number of trees where the orange line in the figure (the 0.01% threshold) meets the blue line (accuracy increase per tree) is the point after which adding a single extra tree to the forest adds a relative accuracy increase of less than 0.1%. The lines meet around $n_{trees} = 40$ for the vision classifier and around $n_{trees} = 20$ for the weather sensor classifier, which are selected as the optimal values for the complexity-performance trade-off.

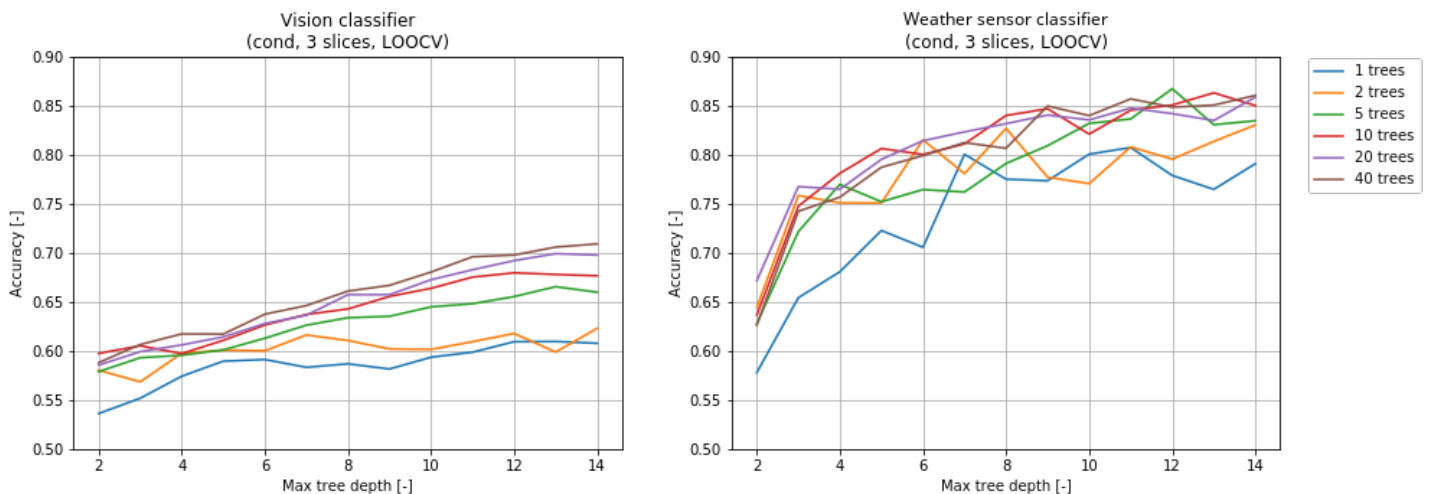


Figure 3.1: Classifier accuracy versus maximum tree depth for a number of Forest sizes. The left plot shows results for the vision classifier, the right graph for the weather sensor classifier. These values are calculated for 3 subdivisions per recording and using LOOCV.

3.2.2. Data slicing

As described in Chapter 2, the dataset consists of individual sequences (*recordings*) recorded on different days or at different times. The label and feature variation of data within these recordings is much smaller than the variation between recordings. If we would create training datasets by collecting the data points for all recordings, and then randomly sample a train and test set from those, this may cause the problems with generalisation of the results.

The classifiers should be able to accurately estimate the weather on unseen recordings, that is, work on recordings that it was not trained on. In a real life situation, it is impractical and beats the purpose if the classifier must be trained every time one chooses to start driving. As such, for proper classifier evaluation, the training subset should not contain data points from recordings that also have points in the test subset, and vice versa. Practical considerations make this difficult: to be able to accurately classify on a large number of situations, a large number of these in situations is required for classifier training. In our case, it

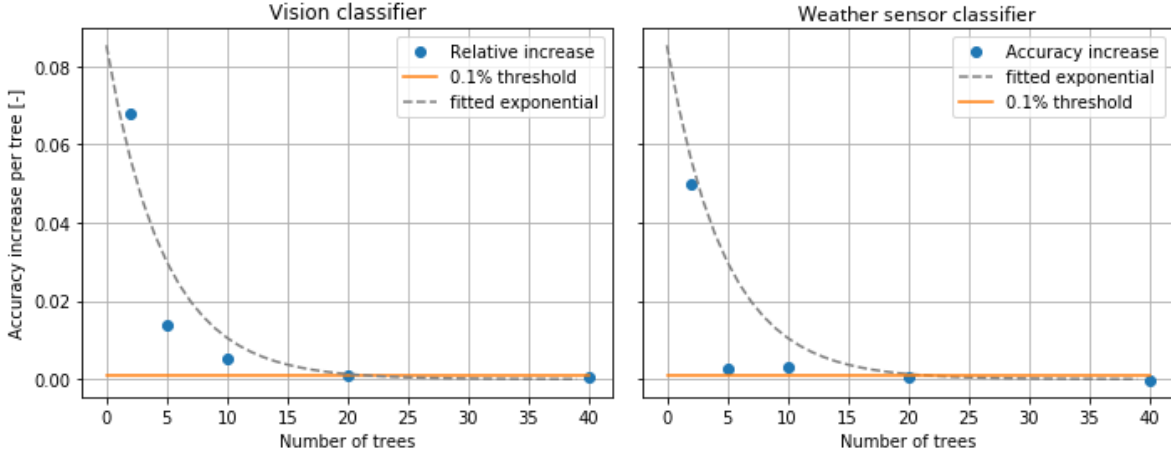


Figure 3.2: Differential accuracy increase (per tree), averaged over all tested tree depths (2 through 14) versus number of trees per forest. The orange line indicates the limit where adding a single extra tree to the forest gives a mean accuracy increase of less than 0.1%. The lines meet around $n_{trees} = 20$ for both classifiers.

is not feasible to completely split training and test subsets by full recordings, as on the present dataset, large between-recording variation and small within-recording variation would make classifier training to yield very bad results.

As a compromise, the assumption is done that two measurements -even when originating from the same recording- are dissimilar enough if there is sufficient time between them. This minimum time difference should be kept as large as possible (using data points as far apart as possible for different sets), while small enough to produce acceptable classifier results. This value is found as follows:

1. Each recording is subdivided in n_{slices} equal, sequential parts, yielding $N = n_{rides} \times n_{slices}$ slices. Points in different slices are, as such, at least $14.5/n_{slices}$ minutes apart².
2. Classifier performance at different values of n_{slices} is evaluated using LOOCV.

The results are visualised in Figure 3.3. For $n_{slices} = 1$, the case in which each individual recording is assigned to either the test or training set, test accuracy for both the vision classifier as the vision classifier (both < 0.5) is too low for further use. With two slices, accuracies increase (≈ 0.65 for vision, $\approx .70$ for weather sensors) which is better, at three slices, the classifiers seem to reach their full potential with test accuracies of around 0.70 (vision) and 0.85 (weather sensors). Making the slices even smaller does not seem to really increase the result. Also, a time difference of ≈ 5 minutes between data points is assumed to introduce enough variance to make the results consistent for unseen 5-minute periods.

3.2.3. Data augmentation

Similar works regarding weather classification on single images, like [43, 59] show classifier accuracies of 0.84 and 0.73, respectively. Since preliminary results to the classifier constructed here shows a classification accuracy of around 0.70, it performs slightly worse than expected. As there are suspicions that this may due to too little available data, a Data Augmentation technique is evaluated. This technique is proposed for CNN-based image recognition in [27] where it is successfully used to reduce classifier error rates.

To evaluate this technique here, the data is split into a training and test set³, and the training set is augmented with the horizontally mirrored version of each image. Since the features used (Section 1.1) are isotropic and are computed over symmetric ROIs (Figure 1.2), this can easily be implemented in a computationally inexpensive way by swapping out elements of the feature vector. Figure 3.4 shows the accuracy of a vision classifier for a number of forest sizes and tree depths and compares the augmented and non-augmented data sets. The similar lines in the left and right plot indicate that the data augmentation seems

²The shortest recording has a duration of 14.5 minutes. The average duration is 18 minutes.

³Using 5-Fold cross validation

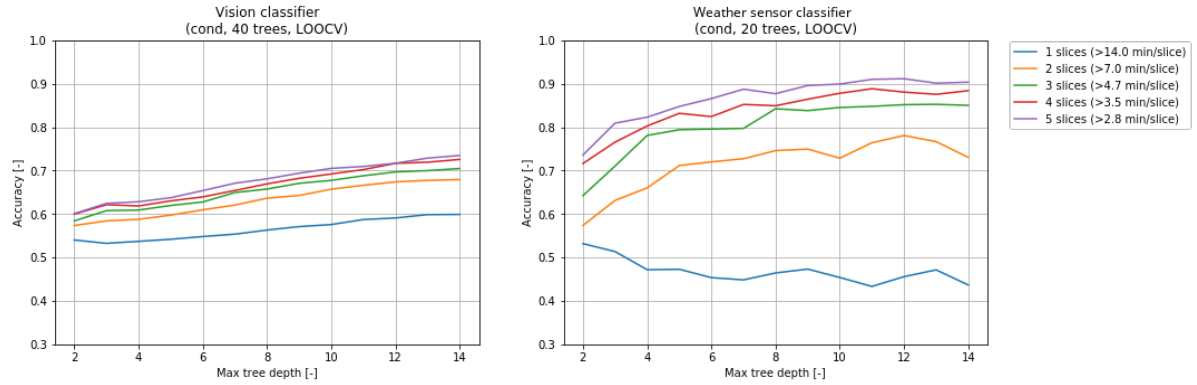


Figure 3.3: Testing accuracy versus tree depth for a number of values for n_{slices} . Taking a lot of slices increases reported accuracy, but makes generalisation harder. Using only one slice (the full recording) removes the generalisation problem, but shows poor accuracy (< 0.6 , sometimes even < 0.5). Testing scores are calculated using LOOCV.

not to improve classifier performance by a substantial amount. In fact, the result improvement using data augmentation shows to be less than 2%. Since using augmented data is computationally expensive⁴ and seems to have very little effect on the result, it is not pursued any further.

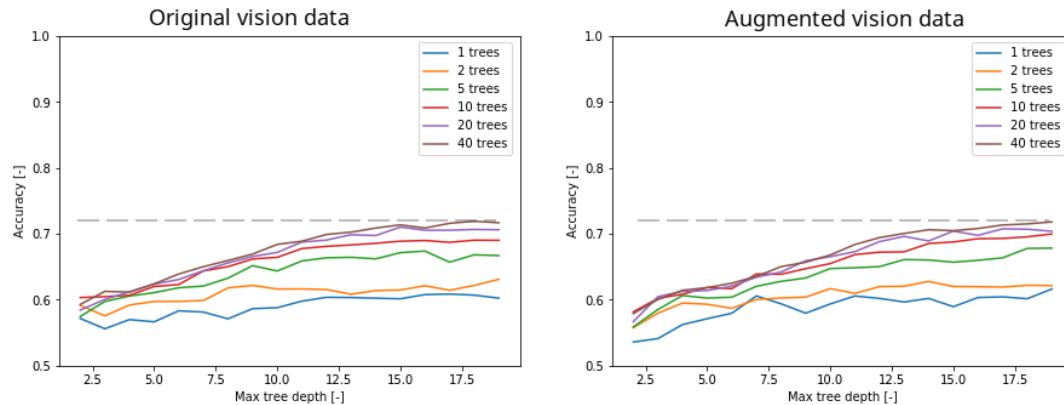


Figure 3.4: Comparison of vision classifier results on original (left) and augmented (right) data. The dashed line is for visual reference and shows that the highest-scoring result does not show substantial improvement when using data augmentation.

3.2.4. Tree depth

With all the other hyperparameters determined, a final validation graph can be used to assess and control classifier *overfitting* [35]. Overfitting, a well known problem in Machine Learning techniques is the situation where a classifier models relations that are too complex (i.e. over-analyses) with regard to the dataset. In essence, the classifier models effects that are specific to (and often occur by chance in) the training set do not exist in a ‘real-world’ situation. This harms generalisation and tends to result in worse performance on new data points.

Overfitting often manifests itself as training accuracy that keep increasing with classifier complexity, while testing accuracy stagnates or even decrease [35]. By plotting complexity (here controlled by maximum tree depth) against the training and testing score of the classifiers, we can find this point⁵. Figure 3.5 shows this plot for the hyperparameters determined earlier in this section.

The expected effect of gradually diminishing improvement in testing accuracy is clearly visible in both graphs. For the vision classifier, for depth values larger than 13, there seems to be almost no increase in testing accuracy. For the weather sensor classifier this is the case for depth values larger than 12. An exception seems to exist at depth 17, but from the plot this looks like a statistical anomaly. These values are recognised as the

⁴Computational complexity of the used CART algorithm scales by a factor $\mathcal{O}(n \log n)$ with n the number of samples[41]

⁵[35], paragraph 1.4.7. Note that the validation chart in the book has its x-axis inverted compared to the charts presented below

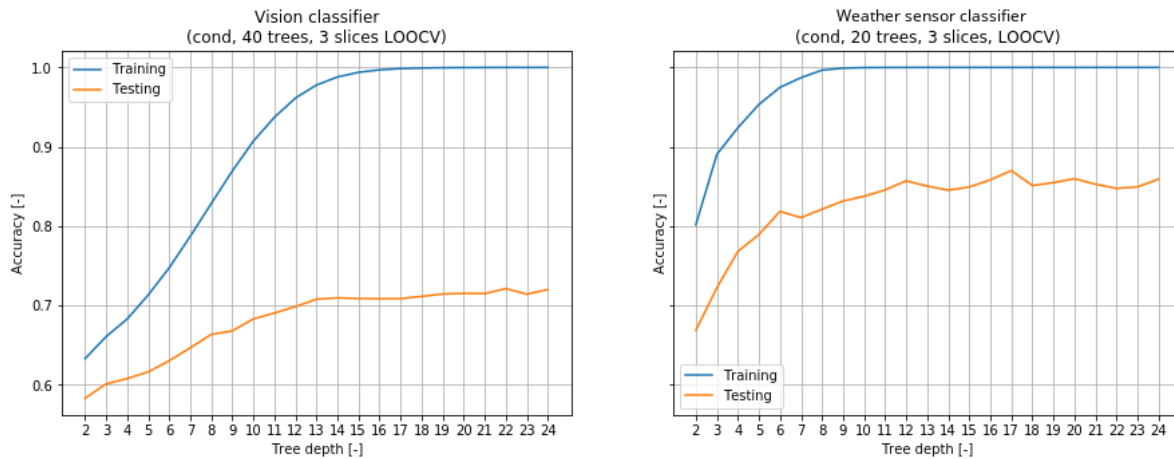


Figure 3.5: Classifier training and testing accuracy for both classifiers, plotted against maximum tree depth.

‘overfitting border’, and will be used as the maximum tree depth hereafter.

3.2.5. Overview

Table 3.1 shows the optimised and default hyperparameters used for classifier training.

Table 3.1: Tuned classifier hyperparameter values for both classifiers

Parameter	Symbol	Vision classifier	Weather sensor classifier
Number of trees	n_{trees}	40	20
Max tree depth	d_{max}	13	10
Recording slices	n_{slices}	3	3

3.3. Classifier results

The values for a number of evaluation metrics, (see Section 1.4 for an explanation of these), are shown in Figure 3.6 and tabulated in Table 3.2. It is observed easily that the weather sensor classifier outperforms the vision classifier for every metric, with a margin between 12% (weighted F1) and 35% (MAE). The differences are larger for accuracy and MAE than they are for F1 and NRMSE. Since the former two penalise inaccuracy more than bias, this raises the suspicion that the vision classifier is more noisy, whereas the weather sensor classifier may be biased towards some classes.

The confusion matrices in Figure 3.7 seem to confirm this suspicion. Bias in confusion matrices is indicated by imbalanced error terms on the sides of the diagonals. The weather sensor classifier shows an error rate 0.15 for *Dry* versus *Rain*, but only a 0.02 error for *Rain* versus *Dry*: an almost 8-fold difference. Similarly, it shows a 0.07 error rate for *Rain* versus *Fog* but a 0.24 for *Fog* versus *Rain*, a 3.5-fold difference. The vision classifier shows less imbalance (0.15 versus 0.30) for *Rain/Dry*, but 0.55 versus 0.05 (an 11-fold difference) for *Rain/Fog*. The small portion of Fog data in the dataset (Table 2.5) makes that this bias has less influence on the macro-averaged metrics.

Finally, the prediction results for all data (using LOOCV) and both classifiers are shown in Figure 3.8. The accuracy/precision difference between the classifiers is clearly visible: the vision classifier (middle bar for each recording) shows the correct class predictions on average (with the exception of a few recordings), but shows a lot of points where it predicts the wrong class for a small amount of time, shown by the thin vertical lines in a different color. The weather sensor classifier shows fewer of these thin lines, but does show (for example near the end of the 2017-09-01 recording) parts where it predicts the wrong class label for longer continuous periods.

Table 3.2: Accuracy metrics (Section 1.4) for the vision and weather sensor classifiers.

Classifier	Accuracy	MAE	NRMSE	F1 (weighted)
Vision	0.70	0.58	0.69	0.79
Sensors	0.85	0.79	0.82	0.89

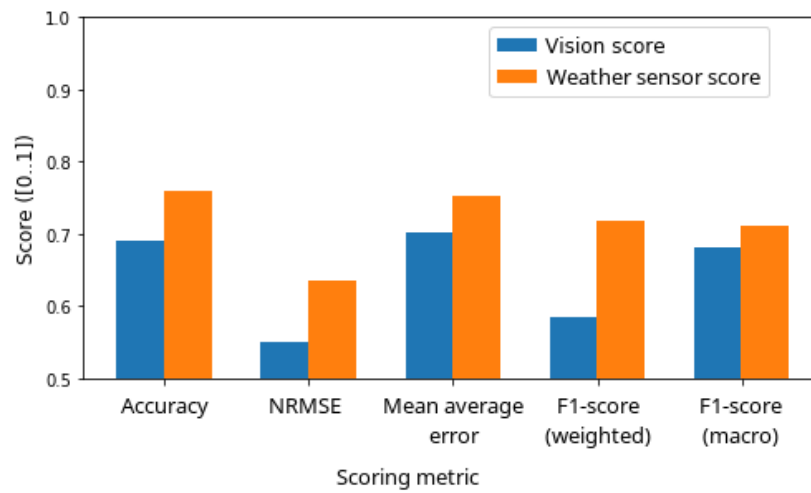


Figure 3.6: Vision and weather sensors classifier results for a number of performance metrics discussed in Section 1.4. The sensors classifier performs better (higher score) for each of the metric. Results were evaluated using LOOCV and with hyperparameters shown in Table 3.1.

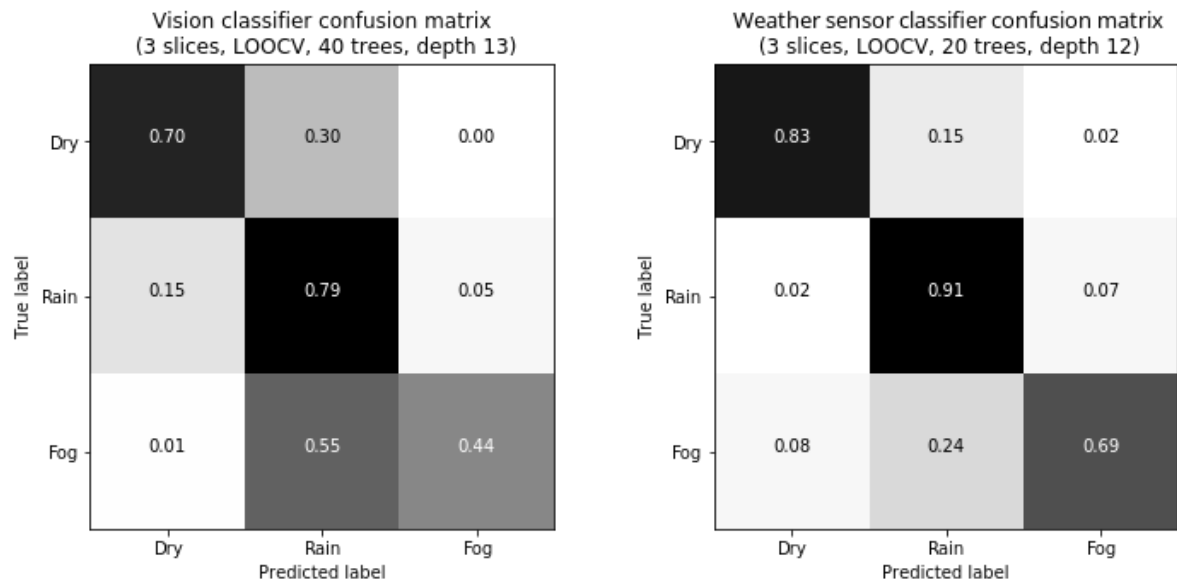


Figure 3.7: Confusion matrix for both classifiers, vision (left) and weather sensors (right). This was evaluate for the final hyperparameters (from Table 3.1) using LOOCV.

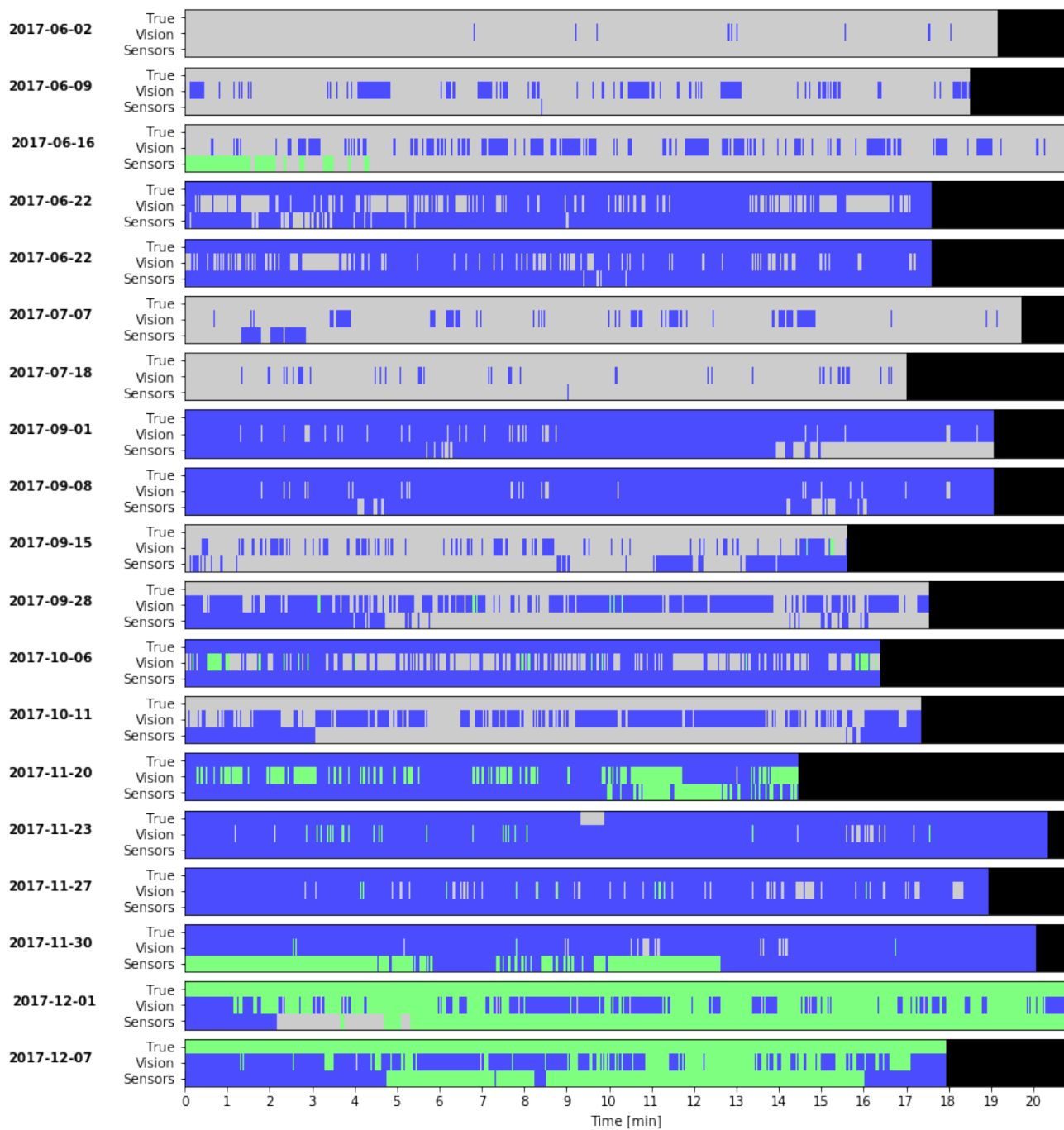


Figure 3.8: Classification results for both classifiers. Each image represent a recording slice. The top row shows true classes, the middle row predictions from the vision classifier and the bottom column predictions from the weather sensor classifier. The images are read from top to bottom (seconds) and from left to right (minutes). The colors indicated the classes. Black colors indicate filler data used to make the results fit in the plots. The weather sensor classifier results (bottom row for each recording) are labeled *Sensor* for brevity.

3.3.1. Discussion

No other works could be found that employs weather sensor-based weather classification in driving situations. This makes comparing the results of the vision classifier difficult. For the vision classifier however, works like [43, 59] have similar prediction goals and datasets and can be used for comparison.

Vision classifier results

The accuracy of the vision classifier presented here is around .70. In [43], the authors present a feature set and learning method which achieves an accuracy of around .85 for the ‘Expressway/Rural/Urban’ scenario presented in their work. A few non-classifier related key differences between this work and [43] may explain the difference in these results.

First of all, in [43] the authors use 1250 images taken randomly from a database 150 sequences (totalling 500.000 frames), but the article does not mention the exact size of the training set. Also, there is no mention of whether there were any measures taken to prevent different images from the same *sequence* ending up in both the training or test set (which may result in both better accuracy and generalisation problems, as discussed Section 3.2.2). In the work here, separating the sequences between the test and training set significantly influences the classifier result as shown in Section 3.2.2 and Figure 3.3 for the dataset used here.

Furthermore, as the authors mention that their classifier has a particularly high error rate between the *Dry* and *Light rain* classes, and between the *Light rain* and *Heavy rain* classes while distinguishing between *Dry* and *Heavy rain* very well. The data used in this research has very little situations representing *Heavy Rain*: if the definition of heavy rain by AMETSOC[48] is used ($> 7.6mm/hr$), there are none (the heaviest rain observed here is $2.5mm/hr$).

If only *Dry* and *Light Rain* are considered, we can compute, from Table I in [43], an accuracy of 0.76 for the mixed-scene (‘Subset 3’) situation. Considering the differences in dataset, the small variance in weather condition and the use of 10 times fewer features, the accuracy score of 0.70 compared to the 0.76 in [43] seems both acceptable and expected.

In the other compared work, [59], a visual classifier distinguishes between four classes (Dry, Rain, Haze and Snow) in a scene-free situation with an accuracy of 0.73. Since the MWI dataset (Section 2.1.1) was provided, the method presented here could be evaluated on that same dataset and showed an accuracy of 0.86 when Snow-labeled images are omitted, versus an accuracy of 0.74 when omitting Snow in Figure 6 of [59]. The confusion matrix for the MWI dataset is shown in Figure 3.9. This difference is expected as the variance between the hand-labeled photos used in the work are expected make classification more difficult.

In conclusion, the performance of the visual classification method is considered acceptable, and no unexpected errors or effects are present. Still, there is room for improvement. A good start would be to evaluate all of the features in [43, 59] and [56] to determine which features are the most informative in this situation. Additionally, the mentioned data issues may be solved by performing a lot more recordings so that there are plenty of recordings for each weather condition, and that the dataset will also contain *Heavy Rain* and, if possible, Snowy situations.

Sensors classifier results

The weather sensor classifier shows a relatively high scores for the evaluation metrics (Table 3.2), but shows substantial bias towards *Dry* in rain situations. Also, it seems to be influenced rather heavily by the data slicing process (Section 3.2.2), which may indicate that it ‘recognises’ some of the recordings and assigns classes as such. Figure 3.8 shows that the weather sensor classifier has much less high-frequency noise compared to the vision classifier which is expected due to the stable short-term nature of weather effects: weather variables change much slower than images taken from a driving vehicle do. The little noise also indicates a sense of robustness: small, random changes that are naturally present in the measured data (due to measurement inaccuracy) do not seem to affect the classifier output much.

Combined model

The classifiers presented in this section give, though far from perfect, informative results, with accuracy values of 0.71 and 0.86 for the vision and weather classifier respectively. Using the confusion matrices shown in Figure 3.7, a combined classification model based on the Naive Bayesian Classifier presented in Section 1.5.1 may be constructed.

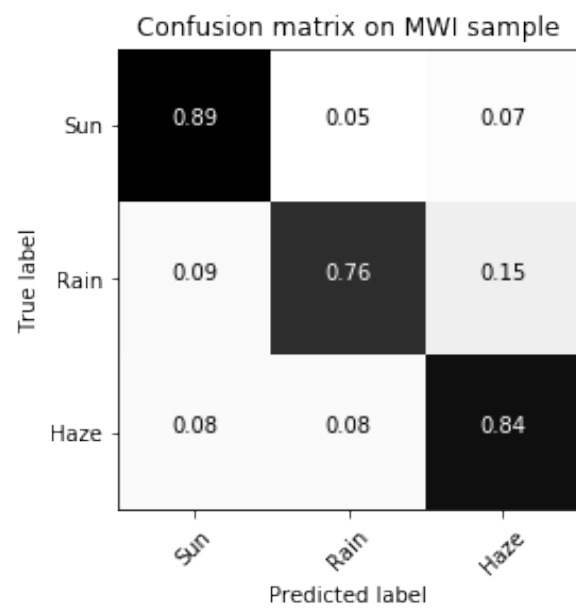


Figure 3.9: Confusion matrix of the vision classifier on MWI dataset. The *Snow* classes is omitted since no snow-specific features are used.

4

Data fusion

The process of combining classifier predictions made using the Vision and Sensor classifier presented in Chapter 3 to obtain improved results is referred to as *fusion*. The fusion process is explored using two approaches: the Naive Bayesian Classifier (NBC) approach, also known as ‘Simple Bayes’ outlined in Section 1.5.1, and the Hidden Markov Model shown in Section 1.5.2. The two fusion models are constructed using the processes for the NBC and HMM described in Section 1.5. The observation matrices for the classifier are obtained from the classifier results. The prior probability (for the NBC and HMM) and transition model (for the HMM) are obtained from the data set. These matrices, along with the classifier results, allow the two processes described in Section 1.5 to be carried out and obtain combined results using the NBC and HMM approaches.

4.1. Model matrices

The prior probability, counted from Table 2.5 is shown in Table 4.1. The transition matrix \mathbf{T} defined in Section 1.5.2 is calculated from the class labels and shown in Table 4.2. The observation matrices are determined before, and are shown in Figure 3.7. For convenience, the values from these observation matrices are tabulated in Tables 4.3 (for the Vision classifier) and 4.4 (for the Weather sensor classifier).

Table 4.1: A priori class probability, counted from class ratios

	Class		
	Dry	Rain	Fog
$p(y_i = c)$	0.42	0.46	0.11

Table 4.2: Transition matrix \mathbf{T} calculated from the class labels.

Current class y_t	Previous class y_{t-1}		
	Dry	Rain	Fog
Dry	.9994	.0005	.0000
Rain	.0004	.9996	.0000
Fog	.0000	.0000	1.0000

4.2. Fusion results

The fusion process was applied to each recording slice using LOOCV. The error metrics (Section 1.4) are evaluated for each result and their means are tabulated in Table 4.5, and additionally plotted in Figure 4.1. From the plot and table, we see that, although the differences are small, the fusion model achieves the highest score for all metrics. The addition of the fusion models improves the metric scores with margins from 2.5% to 9%, and the improvement will be discussed further in the next section. The classifier predictions for each recording are shown in Figures 4.2 and 4.3. These figures show, for all rides, the true class (top bar), the

Table 4.3: Observation matrix F_V for the Vision classifier, calculated from the classifier results. Classifiers trained as described in Chapter 3 and evaluated using LOOCV. See Table 1.3 for a formal definition.

True class	Predicted class		
	Dry	Rain	Fog
Dry	0.70	0.30	0.00
Rain	0.15	0.79	0.05
Fog	0.01	0.55	0.44

Table 4.4: Observation matrix F_S for the weather sensor classifier, calculated from the classifier results. Classifiers trained as described in Chapter 3 and evaluated using LOOCV. See Table 1.3 for a formal definition.

True class	Predicted class		
	Dry	Rain	Fog
Dry	0.83	0.15	0.02
Rain	0.02	0.91	0.07
Fog	0.08	0.24	0.69

classifier predictions (as in Figure 3.8) and the predictions after fusion using the two fusion methods (bottom two bars). The bars shown in the Figures are stacked probability plots that sum up to one. For example, an instance (vertical line) that is blue from top to bottom means a 100% *Rain* probability, whereas an instance that is 30% grey and 70% blue means a 30% *Dry* and 70% *Rain* probability.

A few expected differences between the NBC and HMM show clearly: the NBC shows longer periods for which there is one predicted class with a large probability and another one with small probability, but the HMM does not. For example, for the recording at 2017/09/15 (Figure 4.2, last group) shows a small *Rain* probability throughout in the NBC result, but a 100% *Dry* probability for the HMM result (bottom bar). This is expected as in the HMM, the previous prediction is propagated, which means that after extended periods in which some class has a > 50% probability, the HMM will predict 100% probability for this class. As soon as another class shows > 50% probability (from the observation) for a longer period, this will gradually change to a 100% probability for that class. This slow change is visible in the recording of 2017-09-28 (Figure 4.3, top group). In the third slice, the HMM prediction starts out at 100% dry probability, then slowly moves to 100% rain probability as the observations are just over 50% for *Rain* for an extended period. Just before the end of the slice, the HMM prediction slowly shifts back to 100% *Dry* probability. A similar effect is shown for *Rain* and *Fog* in the third slice of the recording at 2017-11-20 (Figure 4.3, fourth group from top). This makes it clear that the HMM acts as the NBC with an added low-pass filter over the class probabilities.

The rate of change, however, is proportional to the ratio of the observation probabilities. For example, in the first slice recording of 2017-12-01 (Figure 4.3, second group from bottom), around the center of the slice, the weather sensor classifier (and the NBC) estimates an almost 100% *Dry* probability. In this case, the HMM ‘follows’ only seconds after.

Table 4.5: Evaluation scores for individual classifiers and the fusion model for a number of evaluation metrics. Best scores for each metric are shown in **bold**.

Method	Accuracy	NRMSE	MAE	F1 (weighted)
Feature classifier	0.71	0.70	0.59	0.61
Sensor classifier	0.87	0.82	0.79	0.83
NBC Fusion	0.88	0.84	0.81	0.84
HMM Fusion	0.89	0.89	0.89	0.86

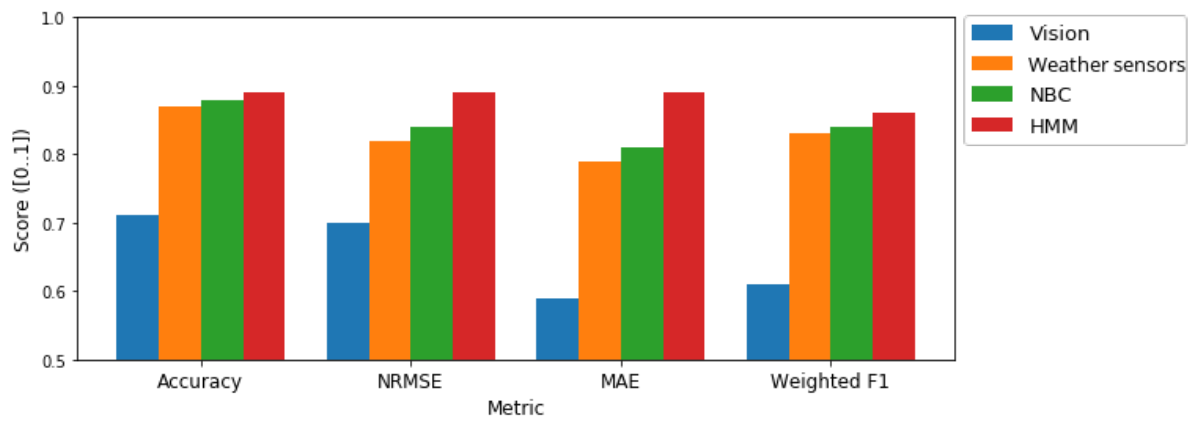


Figure 4.1: Error metrics for the classifiers and for the fusion model. The results are tabulated in Table 4.5

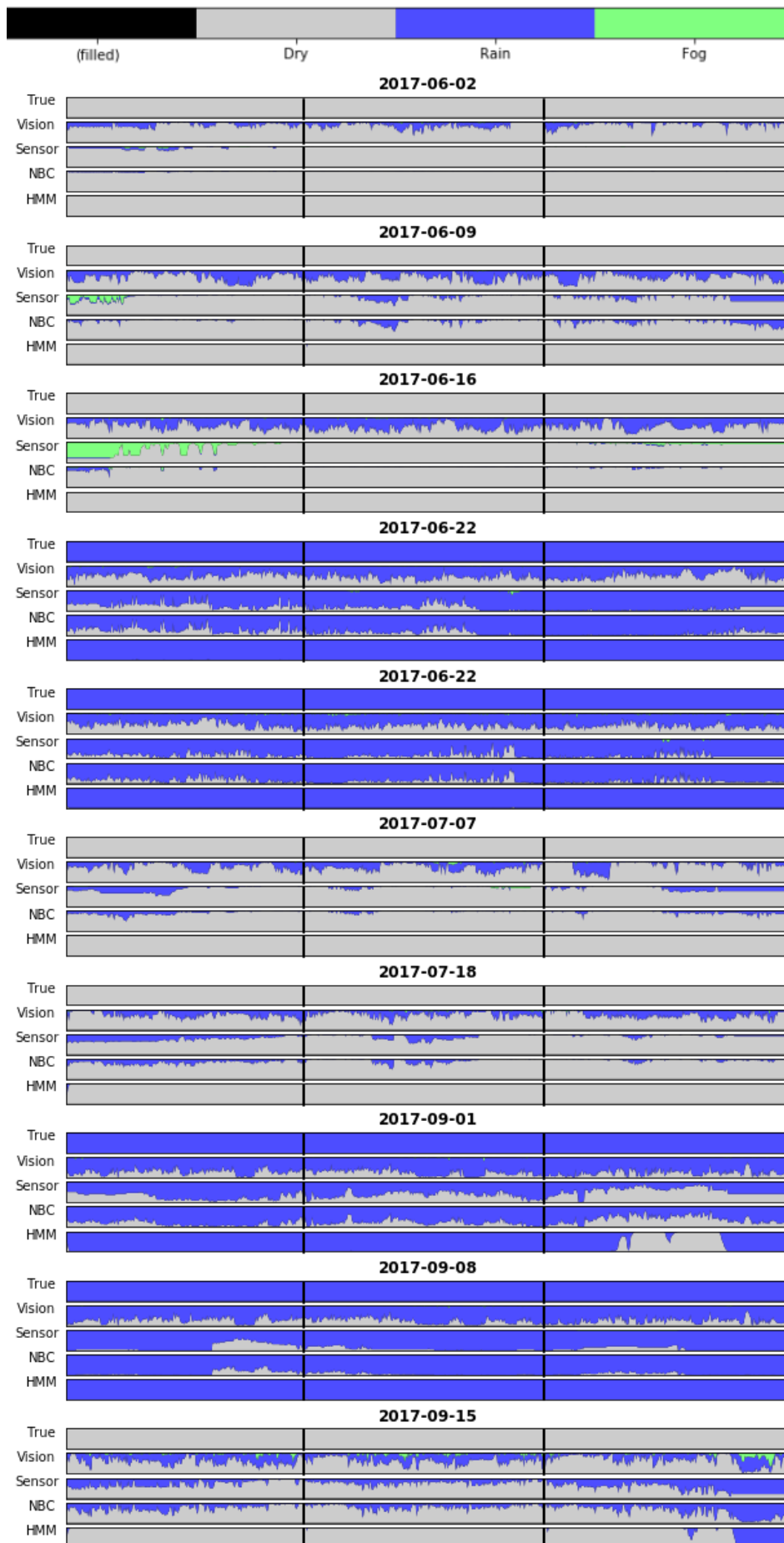


Figure 4.2: Predictions made by all the classifiers including ground truth for the first 10 recordings. Each bar represents a prediction made for one entire recording by one classifier. The bars are stacked plots (summing up to one) with predicted probabilities for each weather condition. Black vertical lines show the borders between recording slices. Note that the time axis is scaled for each ride to have equal width. Results of the weather sensor classifier labeled *Sensors* for brevity.

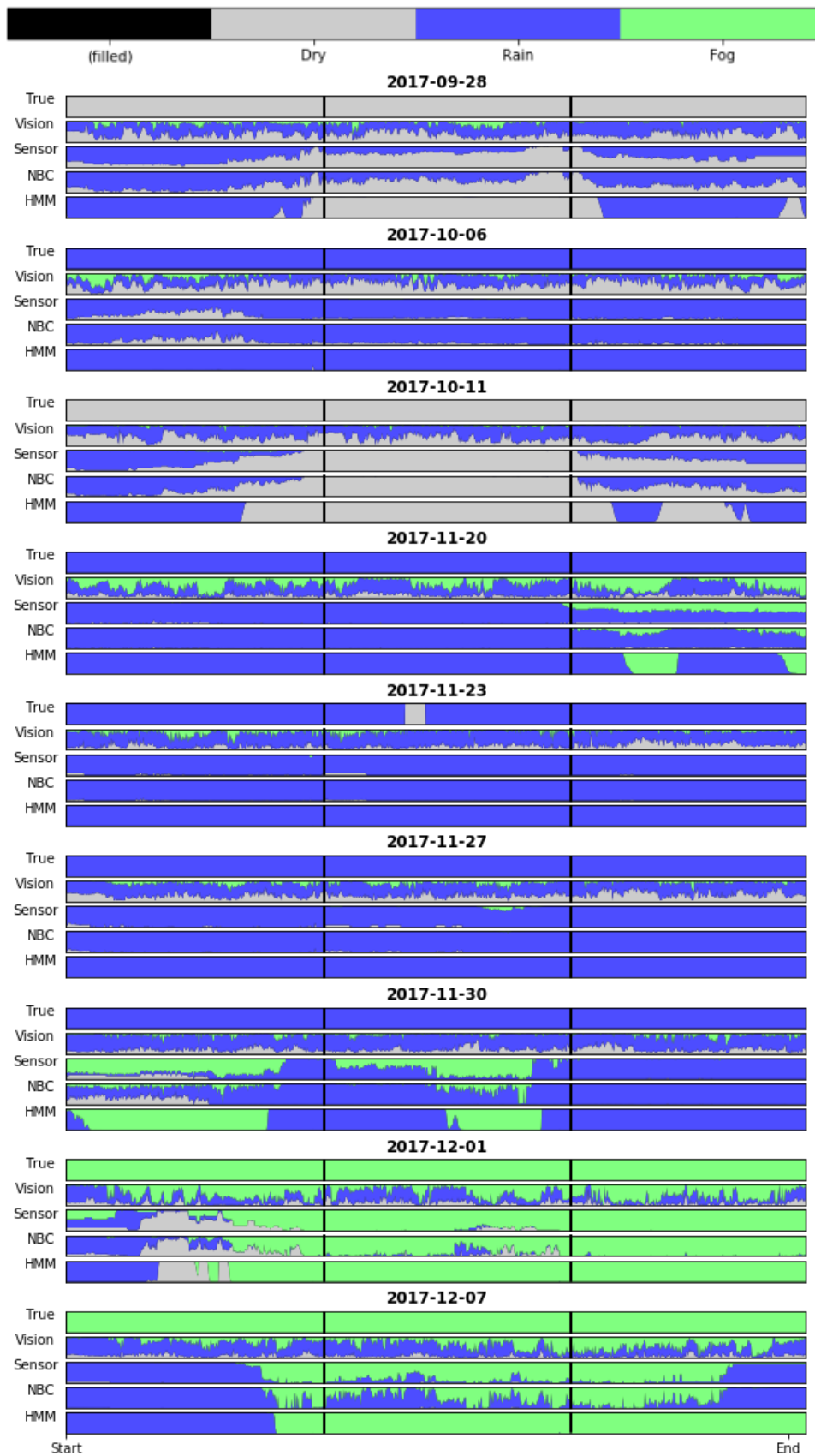


Figure 4.3: Predictions made by all the classifiers including ground truth for the last 9 recordings. Each bar represents a prediction made for one entire recording by one classifier. The bars are stacked plots (summing up to one) with predicted probabilities for each weather condition. Black vertical lines show the borders between recording slices. Note that the time axis is scaled for each ride to have equal width. Results of the weather sensor classifier labeled *Sensors* for brevity.

5

Discussion

The results of the data collection process (Section 2.2.6), classification using individual classifiers (Section 3.3), and of course that of the fusion techniques (Section 4) constitute a basis for discussion, as not all results are as expected. Implications stemming from the results -expected and unexpected- are discussed in this chapter. It provides a detailed discussion regarding all the obtained results, comparing them to what would be expected for each component of the model used to draw conclusions and formulate recommendations in the following chapter.

For each of the model components shown in Figure 2, the results are briefly reiterated and discussed in this section. The partial discussion regarding the classifier results (Section 3.3.1) is revised, as the fusion results presented in Section 4 show new implications with regard to the data set and classifier results.

5.1. Dataset

The dataset discussed in Section 2 contains a large number (over 20k) of observations, from driving situations, suitable for a number of uses. The recordings span a total of 390 minutes of driving time along a single route, at different days, times of day and in different weather conditions. Image data comprises high-resolution (1936×1216 px) sequential images recorded with a high dynamic contrast camera, allowing for a great amount of visual information to be extracted. The addition of weather-specific sensor data measured in-vehicle makes the dataset especially suitable for analysis and prediction of weather-related effects, but the video frames alone could be used for other uses like (non-stereographic) detection and tracking of objects as well.

Data labels

Over the course of the research, the notion emerge that it is nontrivial to obtain unambiguous and verifiable weather labels for the situation in which the data was recorded.

One reason for this is that vehicle is not stationary and weather effects tend to vary strongly with location. Since the travelled distance is small here (the vehicle does not leave a $\approx 3km \times 3km$ area), means over the this area are used, but if the vehicle would travel further, location-specific weather data would be required.

Another reason is that measuring e.g. rain intensity for a specific location and over very short time intervals may not be accurate which means that methods like interpolation or (statistical downscaling [18]) are required. More reliable weather labels could be obtained by recording weather conditions using local sensor systems, e.g. consumer weather stations at a high spatial resolution in and around the area where the recordings take place.

Another labelling problem is the selecting applicable weather conditions, discussed in Sections 1.3 and quantification of those conditions (Section 2.3). Although sources like [47, 48] give good technical, meteorological definitions of rain and fog, the applicability of these to driving situations is debatable, and relevance varies with climate of the considered area and other factors. For example, classes like *Light Rain* considered here may have an arguably small effect on driver behaviour[26] and visibility[23] relative to heavier rain conditions or snow. The very low incidence of snow, on the other hand, is so low (under 20 days per year[24]) in the southern Netherlands that it may be considered irrelevant and tedious to study here. This makes the

dataset less applicable for use in countries where snow and the effects of snow on driving behaviour are prevalent (like Finland [26]).

In the relatively new field of Automated Vehicles, there is little public research to the effects of weather conditions on the AV systems and their object detection subsystems. Such research would be a valuable aid in determining on which exact weather conditions further research should focus.

Data distribution

Comparing the distribution of classes and transitions in the dataset, found in Tables 4.1 and 4.2 (the prior and transition probabilities) with real-world estimations from CESAR data [12] shows these may not conform to real-world values. The prior probability, for example, can be compared to that calculated for the 16-year CESAR dataset shown in Table 5.1. A dataset with this class distribution better resembles the real-world situation (making it easier to generalise) but comes with its own difficulty: the class imbalance in real-world weather situation (*Dry* : *Rain* : *Fog* ratio of 81 : 18 : 1) may make it difficult to train classifiers [30]. Table 4.2 shows another issue: the probability of a *Fog* situation changing to *Dry* or *Rain* is zero. This means that the *Fog* state is *absorbing*, and the HMM model will eventually get ‘stuck’ in the *Fog* state until it is re-set. This is what happens in the HMM prediction for the recording of 2017-12-07 (Figure 4.3, bottom group, third slice): near the end of the slice, the classifiers change their predictions from *Fog* to *Rain*, but the HMM (bottom bar) remains predicting *Fog*, as it cannot make this change.

Table 5.1: Prior probability of weather classes defined Section 1.3 calculated on 16 years of hourly weather data from the CESAR dataset

	Class		
	Dry	Rain	Fog
$p(y_i = c)$	0.81	0.18	0.01

Data variability

Finally, as presented in Section 3.2.2, the recordings used here have to be subdivided in order to ameliorate generalisation issues while at the same time obtaining meaningful classifier results. The time difference between recordings (> 7 days) and their low number (19 included recordings) has the effect that the inter-recording variance of data is quite large compared to the within-recording variance of data points. Analysis shows that this effect is most pronounced in weather sensor data, which is expected: weather sensor data tends to change slowly compared to video frames recorded while driving. In the current datasets, two recordings contain *Fog* labels, nine contain *Dry*, and nine contain *Rain* and the labels of single recordings are almost uniform, with one short exception (Figure 2.11 shows a brief period of *Dry* in-between *Rain* periods in the 2017-11-23 recording).

Because of the large between-ride variance, and the uniformity of labels within rides, great care must be taken in assessing the validity of a model trained directly on this data. The fact the weather sensor classifier performs poorly if the dataset is not sliced (Figure 3.3 shows <0.5 accuracy scores for the weather sensor classifier if slicing is omitted), the use of use data points recorded on the same day in both the train and test set is required to ensure some classifier performance. In other words, the data set is not *Independent and Identically distributed* (iid, [35], 2.6.3). The assumption in Section 3.2.2 that the gap between train and test data is large enough to ensures validity may not be true in general, and the only ways to omit such an assumption are to either use more data, or limit the scope of situations.

Since it may not be the case that the results for the weather sensor classifier as obtained for the sliced data can be generalised, it remains to be proven that the sensor classifier result works as well on a real, regularly obtained dataset of sufficient size. These results, evaluated on the dataset obtained here (the ≈ 0.5 accuracy shown for single slices in Figure 3.3) do not seem very convincing because of the variability issues described above. However, to give an indication of the results that can be obtained on a large, well-behaved iid dataset, the classifier can be trained and evaluated on the CESAR [12] dataset, split into a test and training set. The CESAR repository provides weather sensor data over a period of 16 years (2000-2016), but only records visibility distance since 2008. For an indication of the weather sensor classifier performance on a proper dataset, the following approach is evaluated:

1. Extract 8 years of weather sensor data (at 10-minute intervals) from the CESAR [12] dataset, taking only the quantities required for labelling (as in Section 2.3) and the measured quantities shown in Table 2.3, so that its form is exactly like that of the dataset collected here
2. Pre-process the sensor data, adding the calculated metrics and derivatives shown in Table 2.4
3. Perform the hyperparameter tuning as shown in Section 3 for the weather sensor classifier
4. Evaluate the Random Forest using K -fold cross validation (setting $K = 10$), as LOOCV with this amount of data large is untractable

The result, using the metrics shown in Section 1.4 is shown in Table 5.2. The table shows that classifier performance on the CESAR dataset, according to all metrics, is less than that on the sliced dataset collected here, but is much larger than the performance on the unsliced dataset. It also shows that even though the results are worse, the sensor classifier still provides very meaningful results. From this observation, it may be concluded that even though there is a strong suspicion that the weather sensor classifier performance presented here may be artificially high, the weather sensor classifier can provide a meaningful addition to the multi-sensor system proposed here.

Table 5.2: Evaluation metrics for sensor model

Dataset	Accuracy	MAE	NRMSE	F1
Current data	0.87	0.79	0.82	0.83
CESAR 8-year data [12]	0.77	0.64	0.71	0.79

5.2. Individual classifiers

The individual classifiers described in Chapter 3 and their performance (Table 3.2) are discussed in part in Section 3.3.1. Individual classifier performance does not seem very high (accuracy ≈ 0.70) for the vision classifier, and is quite dependent on the slice size as shown in Section 3.2.2. The classifier performance for the vision classifier might be improved by selecting individual frames, rather than full recordings, for the training and testing set which may be, as discussed in the previous section, valid for visual data but less so for weather sensor data. As such, the vision classifier performance may be improved by selecting individual frames, rather than recording slices, for the train and test set. However, training the vision and weather sensor classifier differently makes fusion as presented in Section 1.5.1 require many more intermediate steps.

5.3. Fusion

Figure 4.1 and Table 4.1 show that the fusion models outperform the individual classifiers for all of the considered performance metrics. A normalised version of Table 4.5 is shown in Table 5.3. This table shows that the performance increase by the NBC classifier is between 1.1% (for accuracy) and 2.5% (for NRMSE and MAE). The HMM model shows higher performance increases, between 2.3% (Accuracy) and 12% (MAE).

To be able to assess these results, it is important to note that Table 4.2 shows a transition matrix with very low off-diagonal terms, which means that the transition model essentially acts as a low-pass filter. Considering that noise in the classifier results, especially for the vision classifier (Chapter 3, Figure 3.8) is substantial, the differences between the two classifiers are expected. In the NBC this noise is propagated to the fusion result: it has no temporal relations so it does not filter noise. In the HMM, the transition model judges sudden condition changes are unlikely so only prediction changes consistent over an extended period are propagated, an effect visible in Figures 4.2 and 4.3. This result is intuitive: the weather changes slowly, so the probability of the weather condition changing during any single-second interval should not be very high. An example where this noise filtering does not work is in the HMM fusion results for the recording *2017-11-30*¹. The fusion results show, near the end of the second slice, two situations in which the predicted result changes from *Rain* to *Fog* and back in respectively ≈ 4 and ≈ 40 seconds.

In the Accuracy and F1 metrics, noise in the probability distribution has a smaller influence because the

¹Figure 4.3, 8th group from top

$argmax$ for hard classification. This explains why the Accuracy and F1 metrics increases (1 resp. 3 percentage points) are smaller than the NRMSE and MAE increases (6 resp. 10 percentage points). The other metrics (NRMSE and MAE) that operate on the soft prediction do account for differences in predicted probability, which explains the greater relative increase of NRMSE and MAE (8% resp. 13%) than the increases in Accuracy and F1 (2% resp. 4%). Additionally, as mentioned in Section 1.4, MAE (a linear metric) penalises noise more than NRMSE (a quadratic metric). This may explain why the relative NRMSE increase from NBC to HMM (2% to 8%) is smaller than MAE increase (3% to 13%): the effect of removing noise is more pronounced in metrics that penalise it more.

Table 5.3: Normalised version of Table 4.5. The score of the weather sensor classifier (the strongest of the two individual classifiers described in Chapter 3) is set to 1.00, and the other results are scaled proportionally.

Method	Accuracy	NRMSE	MAE	F1 (weighted)
Vision classifier	0.816	0.854	0.747	0.735
Weather sensor classifier	1.000	1.000	1.000	1.000
NBC Fusion	1.011	1.024	1.025	1.012
HMM Fusion	1.022	1.085	1.126	1.036

Contribution of the Fusion model

The greatest contribution to the overall accuracy of the model is the addition of the transition model, the difference between the NBC and HMM. In this light, it is considered a lot of effort (adding weather sensors, synchronising data, construction the fusion model) for the mere 1% – 3% increases in performance achieved by the NBC. The direct application of a fusion model to the vision classifier should be explored as it could produce equally promising results while being much easier to implement. Using the NBC-based fusion model however does add a few advantages that a single-sensor-transition model does not.

First of all, the addition of any additional sensor/classifier system in the modelling framework presented here is trivial: using a classifier and some confusion matrix, the predictions can be easily added into Equation 1.17. This works both ways: if a component (camera or weather sensor) is considered broken, it can be easily disabled without hurting the result much. If any of the two classifiers used here remain active, the results should not become worse than the worst results shown in Table 3.2. An Autonomous Vehicle system may self-assess the status of sensors, and use this assessment to decide which sensors are, at some instance, suitable to be used in the weather condition estimation system.

Second, the influence of a sensor on the estimation result is proportional to its pre-estimated accuracy with regard to the problem. As such, less accurate sensors do not ‘harm’ the prediction much, neither do periods in which some sensor is less accurate. A reliable sensor (one which has a high accuracy for most of the time) has a larger contribution to the result than one less reliable (showing periods of low accuracy). If the reliability of sensors changes over time, it should not be difficult to re-evaluate the confusion matrices.

5.3.1. Comparison to state-of-the-art related frameworks

Table 5.4: Comparison for the accuracy metric with related works. All works include a ‘base’ class (*Sunny, Clear or Dry*). Note that each method is evaluated on its own dataset. Refer to the text for discussion.

Method/source	Data	Conditions	Method	Accuracy
This work, NBC	Driving	Rain, Fog	Vision + Weather sensors	0.88
This work, HMM	Driving	Rain, Fog	Vision + Weather sensors	0.89
Roser & Moosmann [43]	Driving	Rain (light + heavy)	Vision	0.85
Roser & Moosmann [43]	Driving (expressway)	Rain (light + heavy)	Vision	0.98
Lu et al. [32]	Photography	Cloudy	Vision	0.77
Song et al. [49]	Traffic camera	Rain, Snow, Fog	Vision	0.93
Zhang et al. [59]	Photography	Rain, Snow, Fog	Vision	0.72

Weather detection, is explored for practical use in modern vehicles by Dannheim et al. [13]. The system proposed by the authors uses visual detection as presented in this work, but employs mostly frequency-space features rather than those taken directly from the image color space. Additionally, the authors propose the use of LIDAR data and Kalman Filtering for detection of weather conditions.

A visibility distance estimation framework for in-vehicle use is presented by Gabb et al. [15]. This work uses sensor fusion to create joint probabilistic estimations from camera and RADAR data using Gaussian Mixture Models. The authors show that their estimations converges and correlates well with visibility distances from human perception. However, the fog case on which the authors evaluate the model is one of very dense fog (45-60m estimated visibility distance).

Both these works provide practical, applied frameworks for weather estimation. As those works employ more modern and sophisticated statistical models, their accuracy may be expected to be high, but comparable figures are not given by their respective authors. Since the availability of both RADAR and LIDAR on vehicles is expected to increase in the future, the use of models leveraging these technologies is considered a good addition.

The method presented here, in comparison, uses simple established statistical models and a more general approach to classification (Random Forests). The optimal balance between more specialised methods (those in [13, 15]) and the more general one presented here remains difficult to find: the former tend to offer greater accuracy, whereas the latter remains flexible to future demands.

Results of the methods proposed in in [32, 43, 59] are shown in Table 5.4. This table shows the accuracy results for each methods as evaluated on the dataset used by the author of the work in which the method is proposed. The methods could not all be evaluated on a single dataset since for some works neither the implementation or dataset was available to the authors. Since method by Roser & Moosmann[43] shows a better accuracy score and raises the question if the addition of the weather sensors and fusion model is justified. However, apart from the accuracy scores shown here, the fusion model offers an increased sense of robustness (it offers redundancy by the use of multiple different sensors) and modularity. The addition of other sensors (e.g. acoustic or capacitive rain sensors) is easy in the model architecture shown in Figure 2.

Conclusion and Recommendations

This chapter draws conclusions from the results and the discussion in the previous chapters, with regard to the research questions formulated in the introduction. The recommendations that follow aim to provide a good basis for further research into the subject, with the goal of creating a weather condition estimation model that can be applied to real-world situations.

6.1. Conclusion

This work presents two architectures: a Naive Bayesian Classifier (NBC) and a Hidden Markov Model (HMM), that can be used to combine weather condition estimations from different sources providing robustness against single-sensor failure. The implementation used here shows that there is a (though very small) overall improvement of estimation results can be achieved by this framework. To implement the framework in the way presented here, improvements to the dataset described in Section 5.1 may be required to obtain better results and allow more extensive analysis. Figure 6.1 shows an overview of the model architecture (Figure 2) with the names of model components replaced by the implementation used here, for reference.

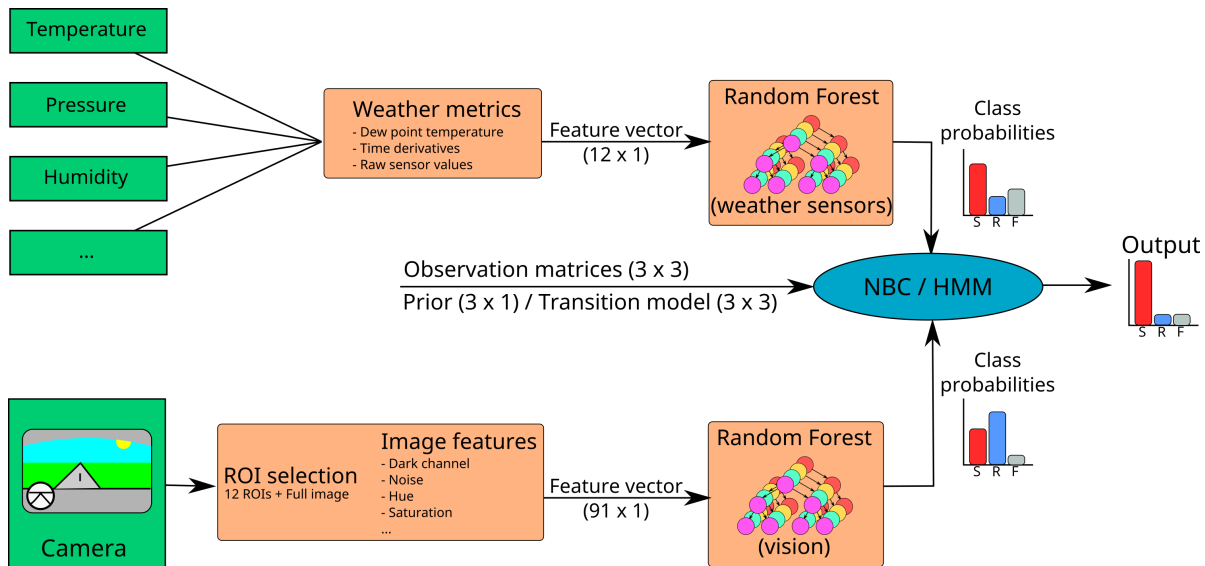


Figure 6.1: Model architecture (Figure 2), with the components as implemented in the model used here.

1. *How can Machine Learning techniques be employed to construct an accurate visual weather estimation model using camera images?*

The combination of image features on Regions of Interest described in Section 1.1 with Random Forests (Section 3.1 onward) shows to produce a visual classifiers with reasonable prediction accuracy (0.70) and F1 score (0.79) on a weather condition estimation dataset consisting of in-vehicle images.

2. *How can locally measured weather variables be used to construct a weather sensor-based weather estimation model?*

Like visual information, the use of features, in this case calculated metrics (Section 1.2) and the same Random Forests shows to produce classifiers with accurate results. Because of the limited, non-iid data set collected here, generalisation of the results may be difficult. The problem of limited data is ameliorated by slicing (Section 3.2.2), but should in fact be solved by collecting more data. The effect sizes shown in Section 1.2 suggest that even more accurate results may be possible, but this is difficult to assess using only the dataset used in this research. The discussion in Subsection 5.1 however, shows that the classifier as constructed here may very well give meaningful results on a large and well-behaved dataset like the CESAR [12] dataset. The CESAR dataset provides a large amount of well-distributed weather sensor data, but its application to a driving situation constitutes a *transfer learning* problem, which requires further research.

3. *How to combine information from a visual and weather sensor-based weather estimation model in order to improve both estimation accuracy and robustness?*

A combination of classifier predictions without a temporal model, in the form of a Naive Bayes Classifier yields an >1.0% increase in hard-classification accuracy and F1 score, and a >2.4% increase in soft-classification NRMSE and MAE score. The addition of a single-step transition model in the form of a HMM yields an increase of >2.0% for the hard-classification metrics and >8.0% for the soft-classification metrics. Both models provide a sense of robustness: they allow classifiers to be disabled if the AV system considers them broken or failing. As such the system is not dependent on a single classifier. Additionally, it provides flexibility, as the addition of a new (type of) sensor-classifier system is easy if its confusion matrix is known. Larger increases in accuracy may be obtained by modelling the individual classifier performance in to specific situations (e.g. scenery).

For both explored methods (NBC and HMM), it has been shown that these contribute to a more accurate and robust result in the research setting here, but the accuracy improvement cannot be guaranteed in general because data set issues raised in Section 5.1. Section 1.5.1 and Chapter 4 do show, however, that they can be applied to fuse multi-classifier predictions in a straightforward way.

This is the main contribution of this work: the collection of data, construction of individual classifiers and implementation of the fusion models together show a proof-of-concept that the proposed modelling framework can be implemented and yield a robust and reliable model. Furthermore, the loose coupling of the model components (sensors, classifiers, fusion models) in the modelling architecture (Figure 2) allows the use of more specialised variants of any modelling component. The model architecture shown in Figure 6.1 shows this: each of the model components (orange blocks) has inputs and outputs independent of the other components, and can as such be replaced with a different implementation, as long as its inputs and outputs are the same. For example, the *Random Forest*-based Vision classifier could be replaced by the MKL-based classifier proposed in [59] or the SVM classifier proposed in [43] without requiring any change to the other components. Likewise, the *Image features* step could be replaced by the feature selection techniques presented in those works, or even frequency space methods as proposed in e.g. [36].

6.2. Recommendations

From the discussion and conclusions shown above, recommendations can be formulated for further research with the goal of construction a valid weather condition estimation system applicable to driving situations.

6.2.1. Use of scenery information

As mentioned briefly in Section 3.3.1, the authors of [43] label and subdivide their dataset depending on the scenery (Highway, Rural or Urban) and the results seem to vary substantially among the different combinations of these. Only the addition of scene information as a feature may aid the training of classifiers like Decision Trees. If detection of the scenery type (on new observations) through a side channel is accurate and inexpensive, scenery information may be exploited to improve visual classification results: individual classifiers could be trained for each scenery type, and the appropriate classifiers used for new data of the corresponding scenery type.

Another possibility to explore is the use of scene information in the Fusion method. If the Vision and weather sensor classifier show very different confusion matrices for different scenery types, then scene-specific observation matrices may be an easy to implement improvement of the model.

6.2.2. Use of seasonal information

Weather and weather effects are known to change with season[42], and this information may be used to improve the weather sensor classifier. Seasonal differences are used often in Markov Chain modelling of rainfall [11, 42]. The weather transition matrix used in the HMM fusion could thus be replaced by a season-specific one, as the current season is generally known. In fact, the use of season information can be applied at multiple levels of the model, with increasing degrees of work required:

- Priors and transition matrices specific to the current season can be used. In this case, seasonal effects on classifier prediction are not accounted for, but the season-specific information embedded in the prior and transition matrix may improve performance of the fused estimation. Since these matrices can be obtained from various sources (like in this case, CESAR [12]) and are easy to compute, this is straightforward to implement.
- Season-specific observation matrices can be used in the fusion model. This accounts for classifier errors specific to season without the need for individual season-specific classifiers. It requires data labeled by season to obtain season-specific confusion matrices for each classifier.
- Individual classifiers could be trained on season-specific data, creating an individual classifier for each season. A system chooses the proper classifier to use for each season. This should yield classifiers with less variance, but requires training each classifier, on smaller amounts of data per classifier. As the classifiers may be very different, observation matrices should also be re-evaluated for each seasonal classifier.

6.2.3. Reducing data variability

Aspects of the data set used here, discussed in Section 3.2.2 (use of sequential recordings, non-iid data), Section 5.1 (data imbalance and imperfect representation) are the key issues with regard to validation of the model presented. These issues do however, provide a good idea about how a data set *should* look like in order to create a valid model. To construct a model that is verifiable and applicable in a general sense, the data set used for training and evaluation should:

1. Contain a large number of recordings containing situations of each class. This may be required for classifiers to perform well on completely unseen situations.
2. Contain different recordings for each weather condition, so that the variability between recordings of different weather classes is 'natural' (iid). This allows individual observations to be selected for the test or training set without causing generalisation concerns. To achieve this, single conditions should be recorded on different occasions (e.g. not all fog recordings should be made in winter, or in the morning)
3. Show distributions of weather conditions and conditions that reflect their real-world counterparts well. That is, the data set must be a very good representation of real-world weather conditions. For example, a generalisable model dataset should definitely contain situations where *Fog* changes to another another condition.

Satisfying these criteria may require a great amount of resources. To account for all large sources of variance, recordings should take place at different times of day (morning, afternoon, evening), every season and weather condition. This is expected to require recordings over the period of at least one full year.

6.2.4. Visual classifier improvements

The current performance of the visual classifier is far from optimal, as some performance improvements were omitted in favour of simplicity, runtime and ease of tuning. Both the set of image features (Section 1.1) and classifier algorithm (Section 3.1) were chosen this way.

With regard to image features, Roser & Moosmann [43], Zhang et al. [59] and Lu et al. [32] show feature extraction techniques that may yield a higher performance. Additionally, Pavlic et al. [36] and Hautiere et al. [19]

show features with very good classification accuracy on their datasets, and the use of these may be worth exploring on the dataset presented here.

Machine Learning methods that seem perform well from other works are e.g. the SVM method from [43], the multi-kernel dictionary learning scheme from [59]. Another state-of-the art method used for weather classification on images in uses Convolutional Neural Networks [14], and its very high performance (accuracy > 0.91) may justify use in this model as well.

The latter approach, involves the use of modern general-purpose image classification techniques, employing Artificial Neural Networks, like Convolutional Neural Networks (CNN, applied to weather detection in [14]) or Long Short-term memory Networks (LSTM, applied to visual driving models in e.g. [55]). Such networks generally require more computing power than the classical Random Forest method used here, but show very promising results in terms of accuracy.

Appendix A: Unscaled versions of prediction plots

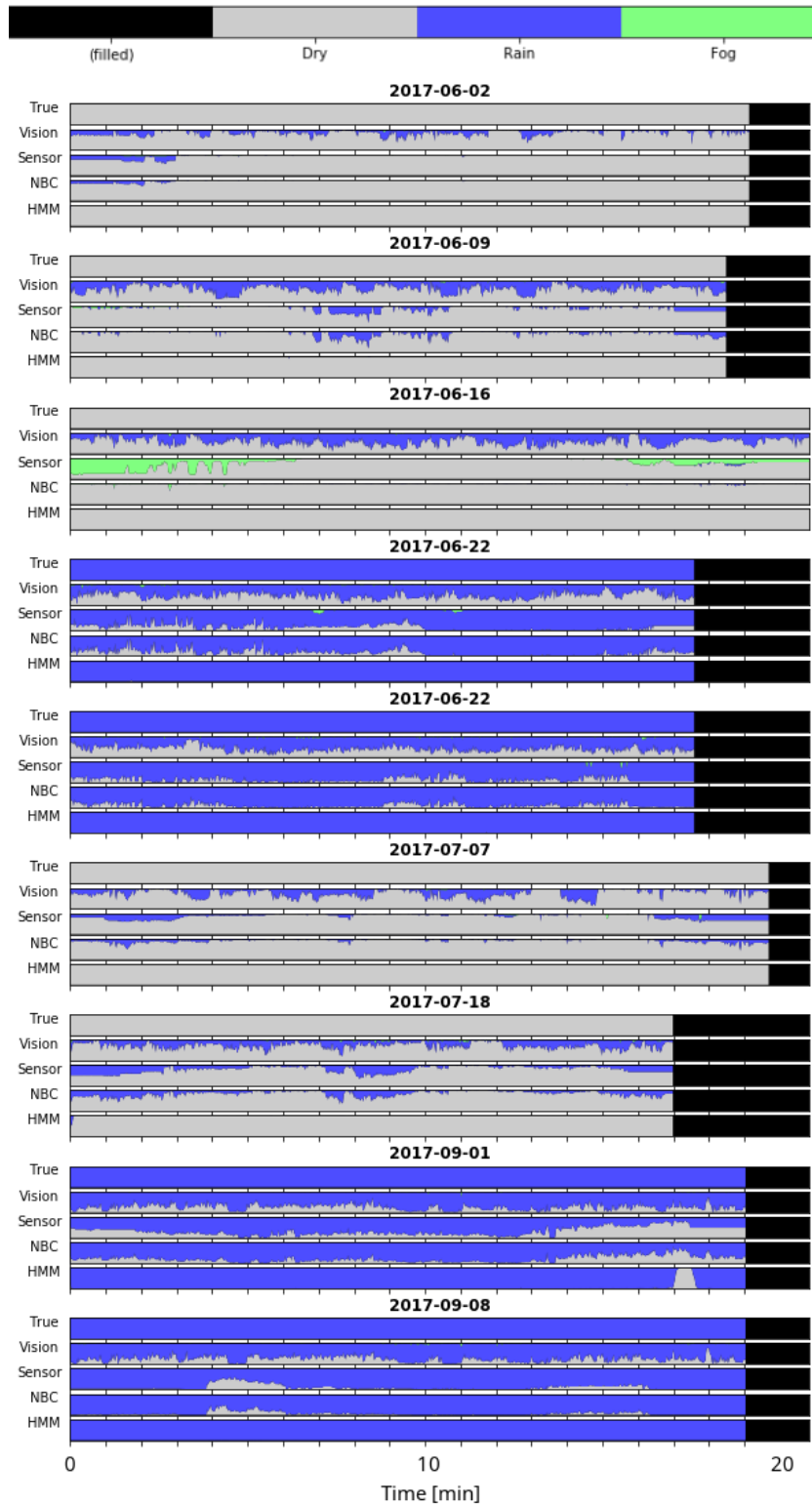


Figure 2: Fusion prediction results for the first 9 recordings, un-scaled version. The label *Vision* denotes the vision classifier results, *Sensor* the weather sensor classifier, *NBC* the NBC fusion model results and *HMM* the HMM fusion results.

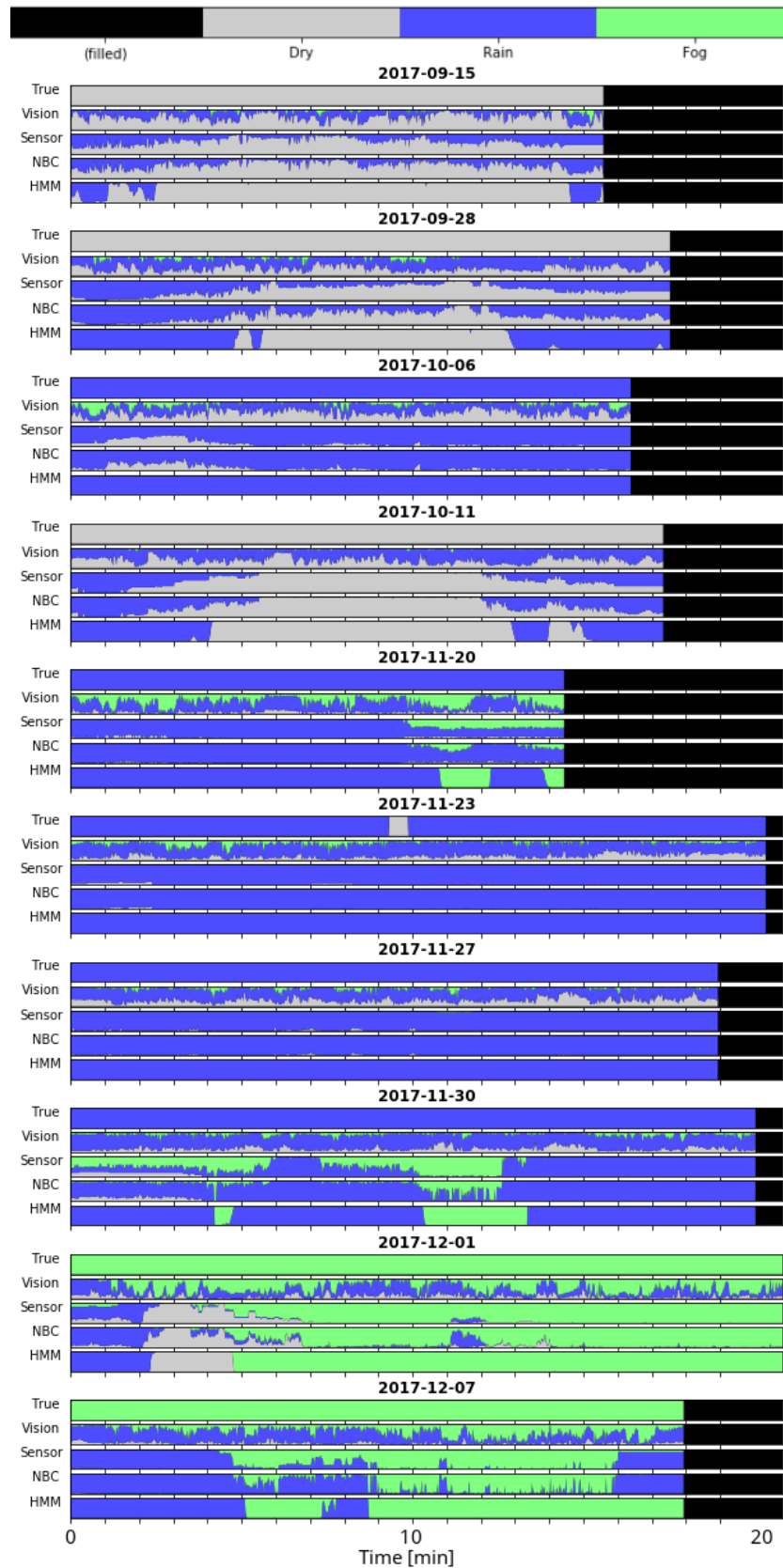


Figure 3: Fusion prediction results for the last 10 recordings, un-scaled version, each tick indicates one minute of recording time. The label *Vision* denotes the vision classifier results, *Sensor* the weather sensor classifier, *NBC* the NBC fusion model results and *HMM* the HMM fusion results.

Bibliography

- [1] Jean Andrey and Sam Yagar. A temporal analysis of rain-related crash risk. *Accident Analysis & Prevention*, 25(4):465–472, 1993.
- [2] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
- [3] Juraj Bartok, František Babic, Peter Bednár, Ján Paralic, Jozef Kovac, Ivana Bartokova, Ladislav Hluchy, and Martin Gera. Data mining for fog prediction and low clouds detection. *Comput Inf*, 31(6):1441–1464, 2013.
- [4] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.
- [5] David Bolton. The computation of equivalent potential temperature. *Monthly weather review*, 108(7):1046–1053, 1980.
- [6] Hamed R Bonab and Fazli Can. A theoretical framework on the ideal number of classifiers for online ensembles in data streams. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2053–2056. ACM, 2016.
- [7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [8] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
- [9] RECOMMENDATION ITU-R BT. Basic parameter values for the hdtv standard for the studio and for international programme exchange. 1990.
- [10] Arden L Buck. New equations for computing vapor pressure and enhancement factor. *Journal of applied meteorology*, 20(12):1527–1532, 1981.
- [11] James E Caskey. A markov chain model for the probability of precipitation occurrence in intervals of various length. *Monthly Weather Review*, 91(6):298–301, 1963.
- [12] The Cesar Consortium. Cesar database, 2013. URL <http://www.cesar-database.nl/>.
- [13] Clemens Dannheim, Christian Icking, Markus Mäder, and Philip Sallis. Weather detection in vehicles by means of camera and lidar systems. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on*, pages 186–191. IEEE, 2014.
- [14] Mohamed Elhoseiny, Sheng Huang, and Ahmed Elgammal. Weather classification with deep convolutional neural networks. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 3349–3353. IEEE, 2015.
- [15] Michael Gabb, Sebastian Krebs, Otto Lohlein, and Martin Fritzsche. Probabilistic inference of visibility conditions by means of sensor fusion. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 1211–1216. IEEE, 2014.
- [16] E.B Garriott. Wind-barometer table. *Monthly Weather review*, pages 204–205, 1897.
- [17] Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*, 1912.
- [18] José Manuel Gutiérrez, Antonio S Cofiño, R Cano, and Miguel A Rodríguez. Clustering methods for statistical downscaling in short-range weather forecasts. *Monthly Weather Review*, 132(9):2169–2183, 2004.

- [19] Nicolas Hauti re, Jean-Philippe Tarel, Jean Lavenant, and Didier Aubert. Automatic fog detection and estimation of visibility distance through use of an onboard camera. *Machine Vision and Applications*, 17(1):8–20, 2006.
- [20] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.
- [21] IDS Imaging. Ui3060cp datasheet. https://en.ids-imaging.com/IDS/datasheet_pdf.php?sku=AB00605.
- [22] Koninklijk Nederlands Meteorologisch Instituut. Risicosignalering zware regen. http://bibliotheek.knmi.nl/klimaatbrochures/Risicosignalering_Zware_regen.pdf, 2007.
- [23] Koninklijk Nederlands Meteorologisch Instituut. Regenintensiteit. <https://www.knmi.nl/kennis-en-datacentrum/uitleg/regenintensiteit>, 2007.
- [24] Koninklijk Nederlands Meteorologisch Instituut. Knmi waarschuwingen gladheid en winterse neerslag, 2017. URL http://bibliotheek.knmi.nl/weerbrochures/FS_Gladheid_winterse_neerslag.pdf.
- [25] Wilfried Jacobs, V Nietosvaara, A Bott, J Bendix, J Cermak, M Silas, and I Gultepe. Short range forecasting methods of fog, visibility and low clouds. *Earth System Science and Environmental Management Final Rep. on COST-722 Action*, 2007.
- [26] Markku Kilpel inen and Heikki Summala. Effects of weather and weather forecasts on driver behaviour. *Transportation research part F: traffic psychology and behaviour*, 10(4):288–299, 2007.
- [27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [28] Hiroyuki Kurihata, Tomokazu Takahashi, Yoshito Mekada, Ichiro Ide, Hiroshi Murase, Yukimasa Tamatsu, and Takayuki Miyahara. Raindrop detection from in-vehicle video camera images for rain-fall judgment. In *ICICIC (2)*, pages 544–547, 2006.
- [29] Mark G Lawrence. The relationship between relative humidity and the dewpoint temperature in moist air: A simple conversion and applications. *Bulletin of the American Meteorological Society*, 86(2):225–233, 2005.
- [30] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.
- [31] Aosong Electronics Co. Ltd. Dht22 datasheet. <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>.
- [32] Cewu Lu, Di Lin, Jiaya Jia, and Chi-Keung Tang. Two-class weather classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [33] Peter Lynch. The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.
- [34] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford Robot-Car Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. doi: 10.1177/0278364916679498. URL <http://dx.doi.org/10.1177/0278364916679498>.
- [35] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [36] Mario Pavlic, Gerhard Rigoll, and Slobodan Ilic. Classification of images in fog and fog-free scenes for use in vehicles. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 481–486. IEEE, 2013.
- [37] Robert Penrose Pearce. *Meteorology at the Millennium*, volume 83. Academic Press, 2002.

- [38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [39] Dean Pomerleau. Visibility estimation from a moving vehicle using the ralph vision system. In *IEEE Conference on Intelligent Transportation Systems*, pages 906–911, 1997.
- [40] Daniel Ramage. Hidden markov models fundamentals - css229 section notes. <http://cs229.stanford.edu/section/cs229-hmm.pdf>, 2007.
- [41] rapaio (<https://stats.stackexchange.com/users/16709/rapaio>). Literature on the algorithm for optimal splitting in the growing of classification trees. Cross Validated. URL <https://stats.stackexchange.com/q/85698>. URL:<https://stats.stackexchange.com/q/85698> (version: 2014-02-06).
- [42] Clarence W Richardson. Stochastic simulation of daily precipitation, temperature, and solar radiation. *Water resources research*, 17(1):182–190, 1981.
- [43] Martin Roser and Frank Moosmann. Classification of weather situations on single color images. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 798–803. IEEE, 2008.
- [44] NXP Semiconductors. Mpl115a2 datasheet. http://cache.freescale.com/files/sensors/doc/data_sheet/MPL115A2.pdf.
- [45] Venkataraman Shankar, Fred Mannering, and Woodrow Barfield. Effect of roadway geometrics and environmental factors on rural freeway accident frequencies. *Accident Analysis & Prevention*, 27(3):371–389, 1995.
- [46] Irwin Sobel. History and definition of the sobel operator. *Retrieved from the World Wide Web*, 2014.
- [47] American Meteorological Society. Ametsoc glossary - fog. <http://glossary.ametsoc.org/wiki/Fog>, 2012.
- [48] American Meteorological Society. Ametsoc glossary - rain. <http://glossary.ametsoc.org/wiki/Rain>, 2012.
- [49] Hongjun Song, Yangzhou Chen, and Yuanyuan Gao. Weather condition recognition based on feature extraction and k-nn. In *Foundations and Practical Applications of Cognitive Systems and Information Processing*, pages 199–210. Springer, 2014.
- [50] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5: 1–34, 1948.
- [51] GI Taylor. The formation of fog and mist. *Quarterly Journal of the Royal Meteorological Society*, 43(183): 241–268, 1917.
- [52] Joseph J Tribbia and Richard A Anthes. Scientific basis of modern weather prediction. *Science*, 237: 493–500, 1987.
- [53] John M Wallace and Peter V Hobbs. *Atmospheric science: an introductory survey*, volume 92. Academic press, 2006.
- [54] [weatherfaqs.com](http://weatherfaqs.org). Wind direction and barometer tendency, 2017. URL <http://weatherfaqs.org.uk/node/166>.
- [55] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint*, 2016.
- [56] Xunshi Yan, Yupin Luo, and Xiaoming Zheng. Weather recognition based on images captured by vision system in vehicle. In *International Symposium on Neural Networks*, pages 390–398. Springer, 2009.
- [57] Autonomous Vehicles: Are you Ready for the New Ride? Mit technology review insights. <https://www.technologyreview.com/s/609450/autonomous-vehicles-are-you-ready-for-the-new-ride/>, 2017.

-
- [58] Gaetano Zazzaro, Francesca Maria Pisano, and Paola Mercogliano. Data mining to classify fog events by applying cost-sensitive classifier. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, pages 1093–1098. IEEE, 2010.
 - [59] Zheng Zhang, Huadong Ma, Huiyuan Fu, and Cheng Zhang. Scene-free multi-class weather classification on single images. *Neurocomputing*, 2016.