Koninklijk Meteorologisch Instituut van België

Institut Royal Météorologique de Belgique

Royal Meteorological Institute of Belgium

Automated fog detection on the RMI webcam images using machine learning techniques

Lars Heylen, Maarten Reyniers
Royal Meteorological Institute of Belgium

August-September 2021
published March 2023

Wetenschappelijk rapport van het
KONINKLIJK METEOROLOGISCH INSTITUUT VAN BELGIE
Ringlaan 3, B-1180 Brussel

Rapport scientifique de
L'INSTITUT ROYAL METEOROLOGIQUE DE BELGIQUE
Avenue Circulaire 3, B-1180 Bruxelles

Automated fog detection on the RMI webcam images using machine learning techniques

Lars Heylen, Maarten Reyniers

DOI: 10.5281/zenodo.7685079

Koninklijk Meteorologisch Instituut van België
Institut Royal Météorologique de Belgique
Königliches Meteorologisches Institut von Belgien
Royal Meteorological Institute of Belgium

Contents

1 Short introduction to CNNs

A convolutional neural network (CNN) takes an image as input and gives probabilities for different classes in the dataset as output. A CNN consists of several subsequent layers, which are in general split into two major parts. The first part are a few convolutional layers which are alternated with maximum pooling layers. The second part consists of a few dense layers which allow to obtain probabilities for the different classes in the dataset.

The convolutional layers use a kernel to scan the image and calculate the input of the next layers. A kernel is a rectangular shape that takes a small part of the image into account, calculates a value from it, and then moves to the next part of the image. Doing so, it creates a new 'image' of those calculated values that can be used as input for the next convolutional layer. Before using it as input for the next layer, the values are first transformed by an activation function, which allows to introduce some non-linearities into the model. The kernels used in this project were squares of size 3, 4 or 5, but they can be any rectangular shape. This step of scanning the image with a kernel is typically done several times using different kernels to obtain multiple output 'images' of the convolutional layer. Each convolutional layer is followed by a maximum pooling layer. This layer divides the image into several rectangles and returns the largest value from each rectangle, it is a dimensionality reduction technique.

The final part of the CNN consists of several dense layers. These layers contain multiple neurons that allow to calculate all sorts of relations between the input values. The input values of these dense layers are the flattened outputs of the convolutional/maximum pooling layers from the first part of the CNN. The final output from these dense layers are probabilities for all classes in the dataset.

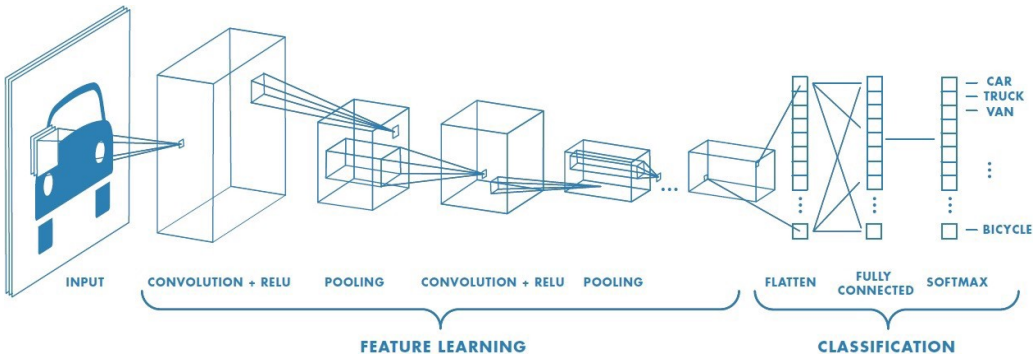


Figure 1: General example of the structure of a CNN (source: `towardsdatascience.com`).

For a slightly more elaborate yet easy to understand explanation, see e.g. the introduction on `https://towardsdatascience.com`. For this project, the TensorFlow library in Python was used to construct the network. TensorFlow (`https://tensorflow.org`) is an open source library developed by Google that allows to build neural networks of various complexities and for various causes.

2 Metrics

Two metrics were used for this project:

- Accuracy
- Confusion matrix and related scores

Accuracy is a very basic metric: it is simply the fraction of the images that were predicted correctly by the CNN. However, this metric is not ideal for datasets with irregularly distributed data. Take for example a dataset with 2 classes, ‘cats’ and ‘dogs’. If 90% of the images were of cats and 10% were of dogs, then the CNN would be able to get 90% accuracy by always predicting ‘cat’. However, such model clearly has not much skill.

The second metric is a confusion matrix. This metric compares the true labels with the predicted labels in a matrix as shown in figure 2. This matrix will be extended to respectively five and three classes in the next sections of this report. From the confusion matrix, a few scores can be calculated. The scores are calculated for a particular class, so they do not apply to the full dataset. The formulas shown below are calculated for the ‘Positive’ class in figure 2.

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- F1 Score = $\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$

Precision can be interpreted as the fraction of predictions of a certain class that actually belong to that class, so it is related to the probability to get false alarms. Recall is the fraction of images of a class that are also predicted in that class, so conversely it says something about the misses. The F1 Score is an arithmetic mean of precision and recall. These scores can be used to select CNNs based on certain criteria and allow to focus on some classes more than on others.

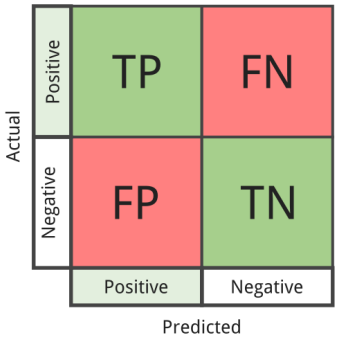


Figure 2: General example of a confusion matrix (with 2 classes).

3 Experiment 1: single webcam and five visibility classes

This section describes the steps taken and the final results for a CNN based on images of the Mont Rigi webcam in 2020. For this section, five classes were defined: 'clear view', 'dawn/dusk', 'fog', 'night' and 'reduced visibility'. The distribution of the images over the different classes is shown in table 1 Examples of these images can be found in appendix B. A distinction is made between the images of only October 2020 and all images of 2020. This is because the October images were the first that were labelled and thus also the ones that were used for the first analyses. Most images correspond to the 'clear view' or the 'night' classes, while the other classes roughly contain between 5% and 10% of the images. In this first exploratory experiment, the accuracy will be the main metric.

	October 2020	2020
clear view	2077 (23.5%)	38295 (37.4%)
dawn/dusk	323 (3.7%)	6436 (6.3%)
fog	876 (9.9%)	8685 (8.5%)
night	4234 (47.8%)	40978 (40.0%)
reduced visibility	1339 (15.1%)	8058 (7.9%)
total	8849	102452

Table 1: Distribution of the Mont Rigi images over the five different classes.

3.1 Images of October 2020

3.1.1 Labelling of the images

The labelling process consists of 2 parts:

1. Labelling the images a first time.
2. Training a basic model on those labelled images and using this model to detect potential wrongly labelled images. The images for which the model assigned a probability below 70% to the true label were relabelled.

The first step is very time consuming since all images have to be looked at individually and have to be given a label. However, this can be done more or less efficiently since the images are ordered chronologically and multiple subsequent images thus correspond to the same class.

The second step is used to find and correct labelling mistakes. The idea is that a model trained on the labelled images will find the general pattern in the images. That means that any images that do not match the pattern, either because they are edge cases between two classes

or because their label is wrong, will be predicted wrongly (or with low confidence) by the model. Those images are then relabelled. For this step however, the advantage of chronologically ordered images disappears. But since the number images that have to be relabelled is a lot smaller than the full dataset, this step can still be done in an acceptable amount of time. Of course, not all images in the relabelling process receive a new label, but the obvious labelling mistakes will be corrected.

After the labelling process, the labelled images are used to create a directory structure that can be used as input for the CNNs. These models require the images to be in directories with their class labels as folder names. For this project, the images were not directly copied to their respective folders, but instead symbolic links ('symlinks') to the images were added in the folders. This allows to save a significant amount of disk space by not having to copy thousands of images.

3.1.2 Finding the optimal hyperparameter configuration for the CNN

Multiple hyperparameter configurations are tested. The hyperparameters that are varied are:

- The **number of convolutional layers**: 2 or 3. The first part of the CNN consists of convolutional layers alternating with maximum pooling layers. In case of 3 convolutional layers, each maximum pooling layer has a 2x2 view. In case of 2 convolutional layers, the first maximum pooling layer has a 2x2 view and the second one a 4x4 view. This is done to keep the number of input values for the dense layers more or less the same between the cases with 2 and 3 convolutional layers.
- The **kernel size**: 3x3, 4x4 or 5x5.
- The **stride length**: 1 or 2. This is the step size by which the kernel scans the image. After a first test it was clear that stride length 1 outperformed stride length 2, so this hyperparameter was kept at 1 afterwards.
- The **activation function**: 'relu', 'tanh', 'sigmoid', 'softplus', 'elu'. These functions are used to transform the output of the convolutional layers and the dense layers. They introduce some non-linearities in the model. The are plotted in figure 3.
- The **number of kernels in the first convolutional layer**: 8 or 16. This defines the number of output 'images' of the first convolutional layer. This value is multiplied by 2 for each subsequent layer.
- The **size of the first dense layer**: 64 or 128. This defines the number of neurons in the first dense layer.

A final value that was also varied but that is not really a hyperparameter is the **resolution of the input images**. There are two reasons to reduce the input resolution: memory use and training time. However, reducing the input resolution also reduces the performance of the model so a strong resolution reduction is not advised. This trade-off led to a final resolution reduction by a factor 8 in both height and width. In practice, the native resolution of the archived webcam images was scaled down from 1280x960 pixels to 160x120 pixels, while keeping the original aspect ratio of 4:3.

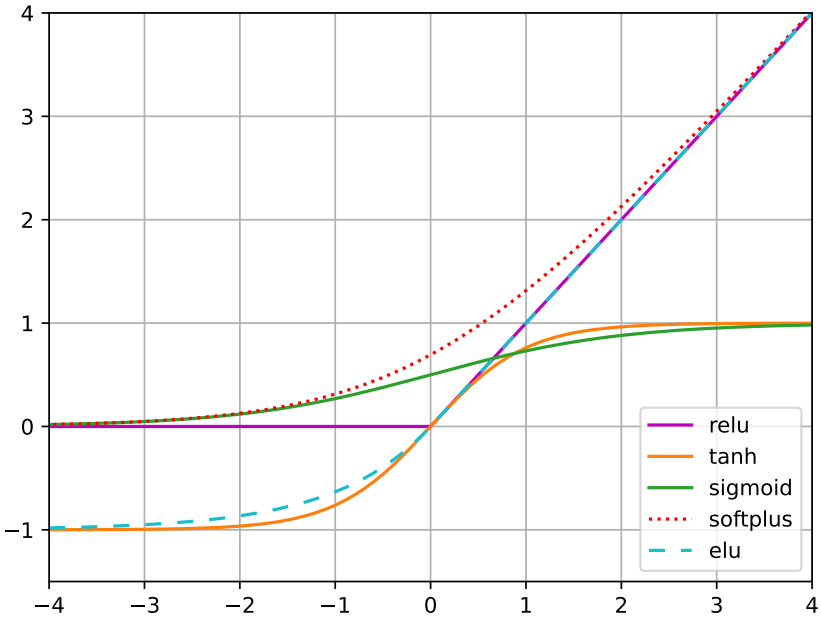


Figure 3: The different activation functions used.

The optimal hyperparameter configuration was found using KFold cross-validation (for this project, k=5). In this case the dataset was split into 5 folds of equal size. The model was then trained on 4 of those sets and validated on the fifth set. This is done 5 times, each time with a different validation set. Finally, the results are averaged and one final accuracy was obtained for the model. Doing this for all combinations of the hyperparameters leads to a result as in figure 4. The bottom line here is that the different configurations span a range of accuracies. Most of them are high but there are also model configurations with rather bad scores. Also clear in the figure is that the training accuracy is typically higher than the validation accuracy. However, since the validation accuracy does not decrease after a certain number of epochs, there seems to be no problem with overfitting. For further steps, the model configuration with the highest validation accuracy was chosen.

The loss function that is also shown on the figure is the *SparseCategoricalCrossentropy* function of TensorFlow.

In this case, the chosen model had the following hyperparameter configuration:

- 2 convolutional layers
- 'tanh' activation function
- 3x3 kernel
- stride length 1
- 16 kernels in the first convolutional layer
- 128 neurons in the first dense layer

Figure 5 shows the equivalent figure of figure 4, but now with only the chosen model. The highest validation accuracy was reached after 17 training epochs so this is also the number of epochs used to train the final model.

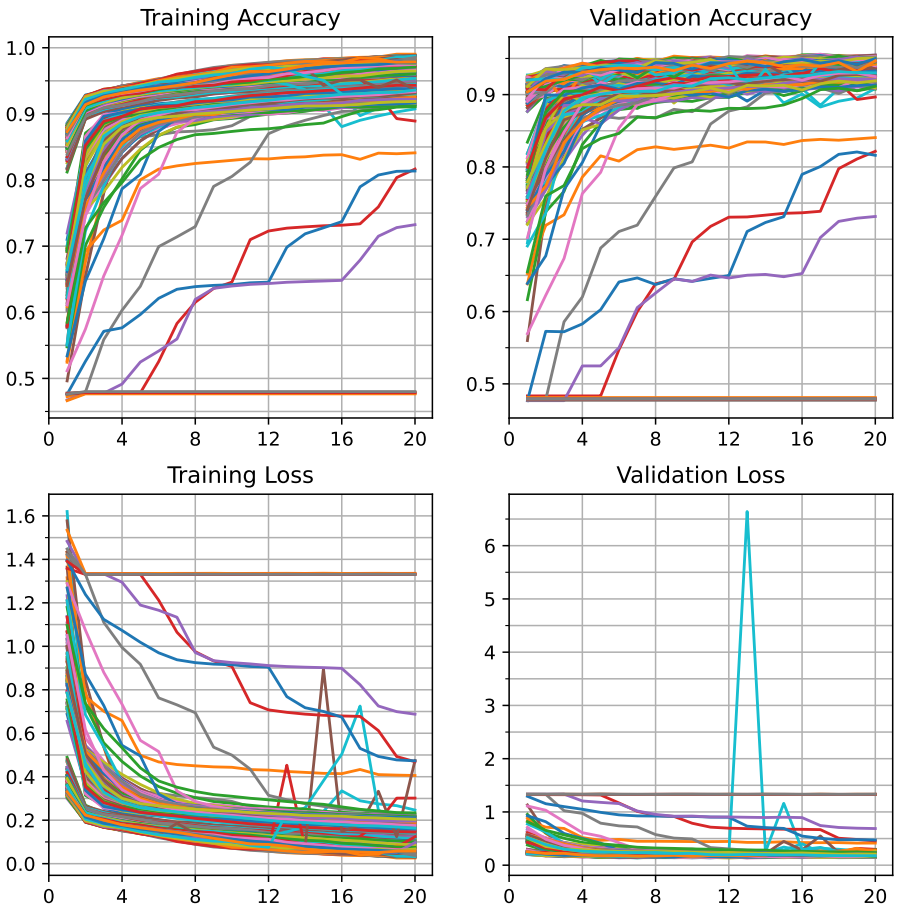


Figure 4: Results of the hyperparameter tuning using KFold cross-validation with k=5.

3.2 All 2020 images

For this part, many results are taken from the previous part. Some things are different, however. During the labelling process, the relabelling step was not done because this is very time consuming with a large set of images (around 16000 in this case). Also, the KFold cross-validation step was not done due to computer time constraints with the very large dataset. As a result, the chosen hyperparameter configuration based on the analysis with only images of October 2020 was used to train a model on all images of 2020. This is probably not the same configuration that would result from a full cross-validation analysis, but a significant difference in accuracy is not expected.

First, the model was trained on 80% of the images and evaluated on the other 20% to get a performance measure of the model. The results are summarised in a confusion matrix which is shown in figure 6. The overall accuracy is 93.1%. The different scores calculated from this confusion matrix are given in table 2.

From the calculated scores it is clear that the two main classes (‘clear view’ and ‘night’) are predicted correctly over 98% of the time. The recall score of the other classes is a little bit lower, however still 92% of the fog cases are detected. When taking a closer look at the confusion matrix in figure 6, we can also see where the wrong cases are located. Here it is clear that they mainly correspond to edge cases. For example 29% of the ‘dawn/dusk’ images are predicted as

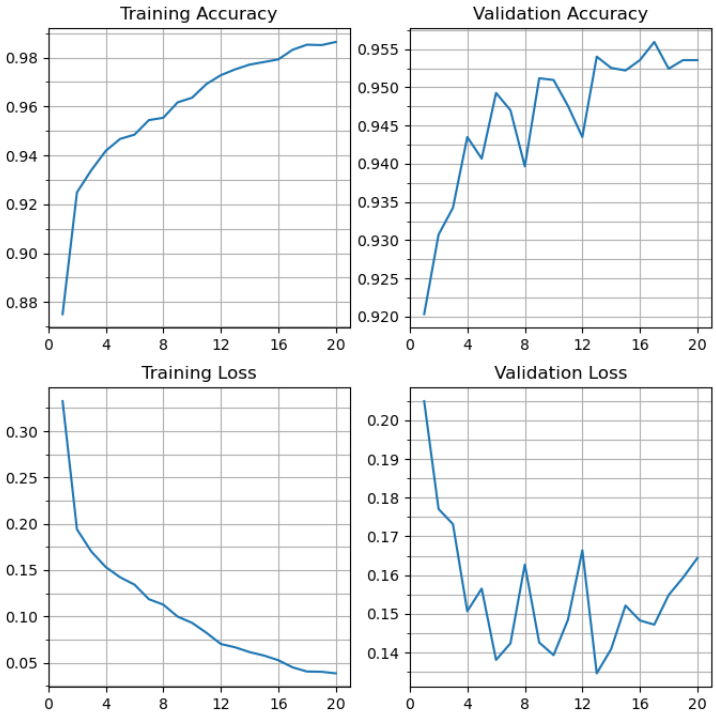


Figure 5: Results of the model with the highest validation accuracy.

	Number of images	Precision	Recall	F1 Score
clear view	7766	94.4%	98.0%	96.1%
dawn/dusk	1260	86.2%	54.9%	67.1%
fog	1755	90.1%	92.3%	91.2%
night	8079	95.7%	99.2%	97.4%
reduced visibility	1630	79.4%	70.6%	74.7%

Table 2: Performance of the CNN based on all Mont Rigi images of 2020.

'night', but this makes sense because there is no hard boundary between those classes. The same holds for 'fog' images that are predicted as 'reduced visibility' or for 'reduced visibility' images that are predicted as 'clear view' or as 'fog'. Other wrongly predicted images are a bit more strange, for example the 'fog' image that is predicted as 'clear view'. This case is probably due to a labelling error. Since the relabelling step was not done for the full 2020 dataset, it is very likely that there are still such wrongly labelled images present in the dataset.

In general, the model performs well. The loss in accuracy or lower other scores is mainly due to the fact that the different classes have no clear boundaries, which means that the model is probably not really wrong either for those cases. Some other wrongly predicted images are most likely due to labelling errors.

As a final step, the model is trained again but now on the full dataset. This model is then saved and can, for example, be used to predict new images in real time.

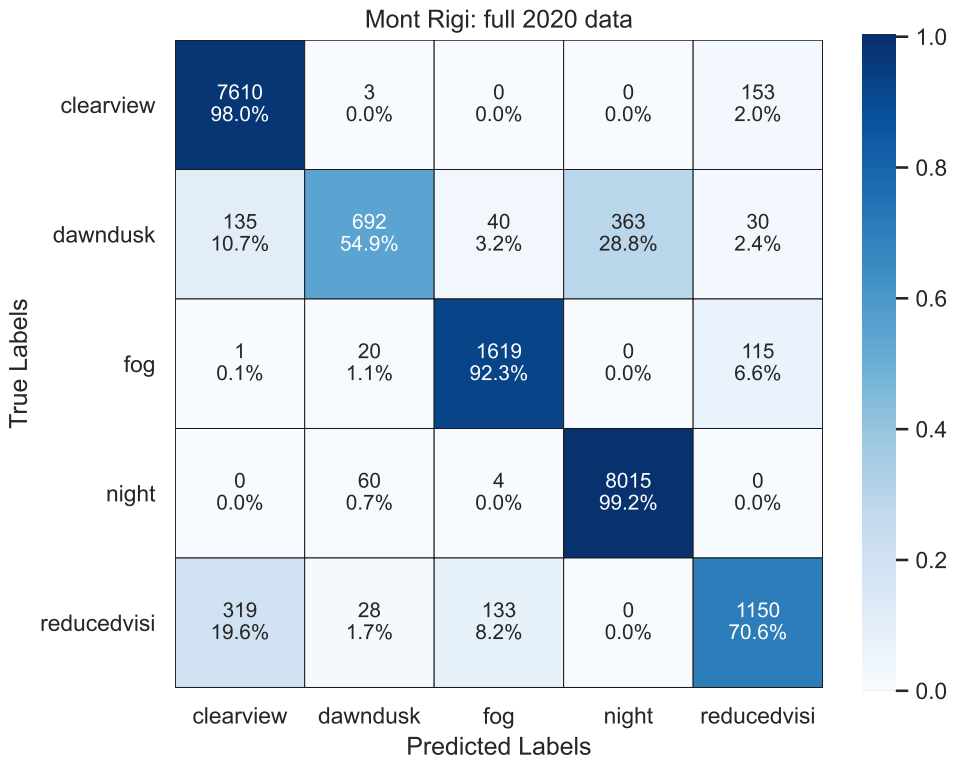


Figure 6: Confusion matrix for the Mont Rigi model calculated from a test set containing 20% of the images.

4 Experiment 2: multiple webcams and three visibility classes

This section describes the steps taken and the final results for a few CNNs based on images of six RMI webcams in 2020. The six webcams are located in Diepenbeek, Melle, Mont Rigi, Stabroek, Wideumont and Zeebrugge; their location is shown on the map in figure 7. In the current experiment, only images of 20 days were chosen for each webcam. From those 20 days, 10 days were selected from the days for which people indicated that there was fog in the RMI app. This was done to make sure that there were an acceptable number of 'fog' images in each dataset. These 10 days were spread more or less evenly throughout the year. The other 10 days were chosen randomly from all other days of 2020. Contrary to the previous section, now only three classes were defined: 'clear view' (no fog), 'fog' and 'too hard to see'. Most images could be assigned to the 'clear view' or 'fog' classes. The distribution of the images over the different classes is shown in table 3. Examples of these images can be found in appendix B.

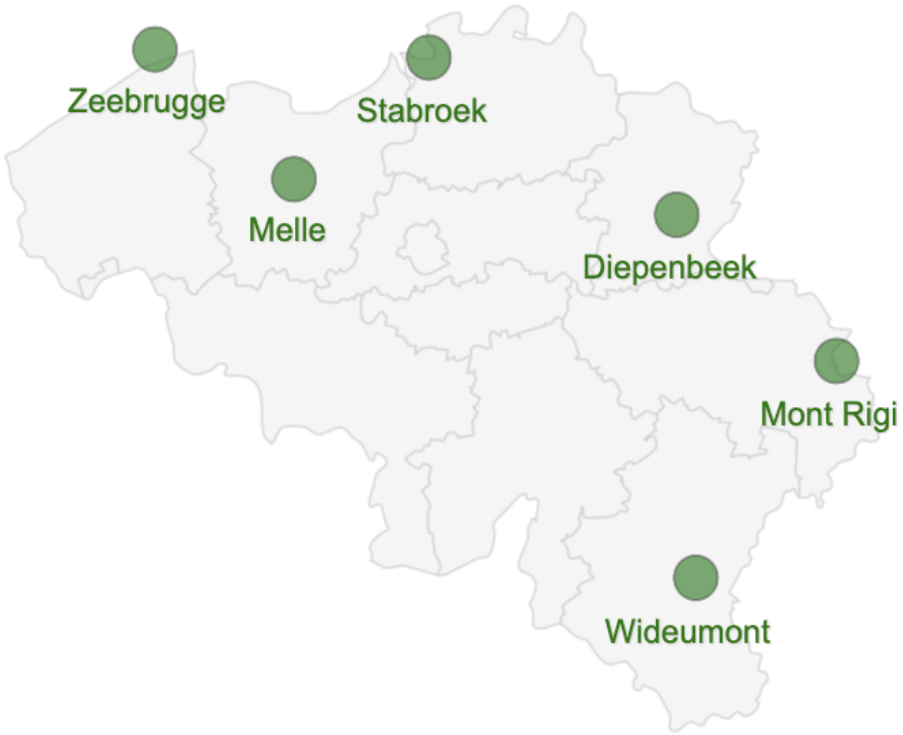


Figure 7: The location of the six RMI webcams included in Experiment 2.

For most webcams, the percentage 'fog' images is between 15% and 20%. It is lower in Stabroek and higher in Mont Rigi. Also clear in the table is that not all webcams have images in the 'too hard to see' class. These images typically correspond to very dark nights, which occur mostly in the Ardennes (Mont Rigi and Wideumont). The few 'too hard to see' images in Stabroek are due to the fact that the webcam in Stabroek is a black-and-white camera. This made it sometimes difficult to really distinguish fog from no fog. The final line in table 3 shows the sum over all webcams. These values are also shown because a separate CNN will also be trained on the combined set of images instead of only on images of one webcam.

	clear view	fog	too hard to see	total
Diepenbeek	4607 (82.3%)	988 (17.7%)	0 (0%)	5595
Melle	4646 (80.7%)	1114 (19.3%)	0 (0%)	5760
Mont Rigi	2873 (50.4%)	1300 (22.8%)	1527 (26.8%)	5700
Stabroek	5087 (89.7%)	525 (9.3%)	60 (1.1%)	5672
Wideumont	3491 (60.6%)	1073 (18.6%)	1194 (20.7%)	5758
Zeebrugge	4680 (81.3%)	1075 (18.7%)	0 (0%)	5755
Combined	25384 (74.1%)	6075 (17.7%)	2781 (8.1%)	34240

Table 3: Distribution of the images for the different webcams over the different classes.

4.1 Labelling of the images

The steps taken to label the images were exactly the same as for the images of the Mont Rigi webcam in the previous experiment. Now, since there were only 20 days of images used for each webcam, the relabelling step was also done.

4.2 Finding the optimal hyperparameter configuration for the CNNs

This part was done in a very similar way to the one described in section 3.1.2. However, a few insights were used to reduce the number of configurations that had to be tested. The top results all had a stride length of 1, and the models with a 'sigmoid' or 'softplus' activation function performed not as good as the models with a different activation function. All other configurations (72 in total) were again tested using KFold cross-validation with 5 folds.

Contrary to the hyperparametertuning in section 3.1.2, now the validation accuracy was not used as the measure to decide which configuration to choose for the final model. In this case, the average validation recall of the 'clear view' and 'fog' categories was used and the configuration that reached the highest value was chosen. The recall was chosen since it treats the labels individually: what is the probability that the model will correctly predict the label, given that the situation corresponds to a certain class. The precision is not very useful in this situation since it also depends on the distribution of the classes over the dataset: what is the probability that the model will predict a certain label, given that the situation does not correspond to that class. However, since the number of 'fog' images in the dataset is higher than in reality due to the selection process of the days used for labelling, this score is not used. Since the F1 score is the arithmetic mean of the recall and precision, this metric is also not used.

The scores shown in figure 8 are the results of the chosen model based on the combined set of images of the six webcams. For this model, however, it has to be noted that not all configurations were tested due to lack of time and some hardware problems in the final week

of the project. 44 of the 72 configurations were tested, 31 of those had 2 convolutional layers and the other 13 had 3 convolutional layers. However, since the top models all get very similar scores, even if a non-tested model would perform better, this wouldn't be by a large margin. For the models based on images of individual webcams, all 72 configurations were tested. In the following, we will refer to the model based on the combined set of images as the *combined model*, while the models based on images of only one webcam will be referred to as the *single models*.

The chosen combined model, for which the results are shown in figure 8, has the following hyperparameter configuration:

- 3 convolutional layers
- 'relu' activation function
- 3x3 kernel
- stride length 1
- 16 kernels in the first convolutional layer
- 64 neurons in the first dense layer

The corresponding figures and model configurations for the single models are shown in appendix C. The final model configurations are also summarised in table 4.

	Number of convolutional layers	Activation function	Kernel size	Stride length	Number of kernels in the first convolutional layer	Size of the first dense layer
Diepenbeek	2	tanh	3x3	1	16	64
Melle	3	tanh	4x4	1	8	64
Mont Rigi	2	relu	4x4	1	8	64
Stabroek	2	tanh	4x4	1	8	64
Wideumont	2	relu	3x3	1	8	128
Zeebrugge	2	elu	3x3	1	16	64
Combined	3	relu	3x3	1	16	64

Table 4: Configurations of the final models used for predicting images of six webcams with three classes.

4.3 Performance of the models

To test the performance of the models, the images were again split into a training set containing 80% of the images and a test set containing the other 20% of the images. To be able to compare

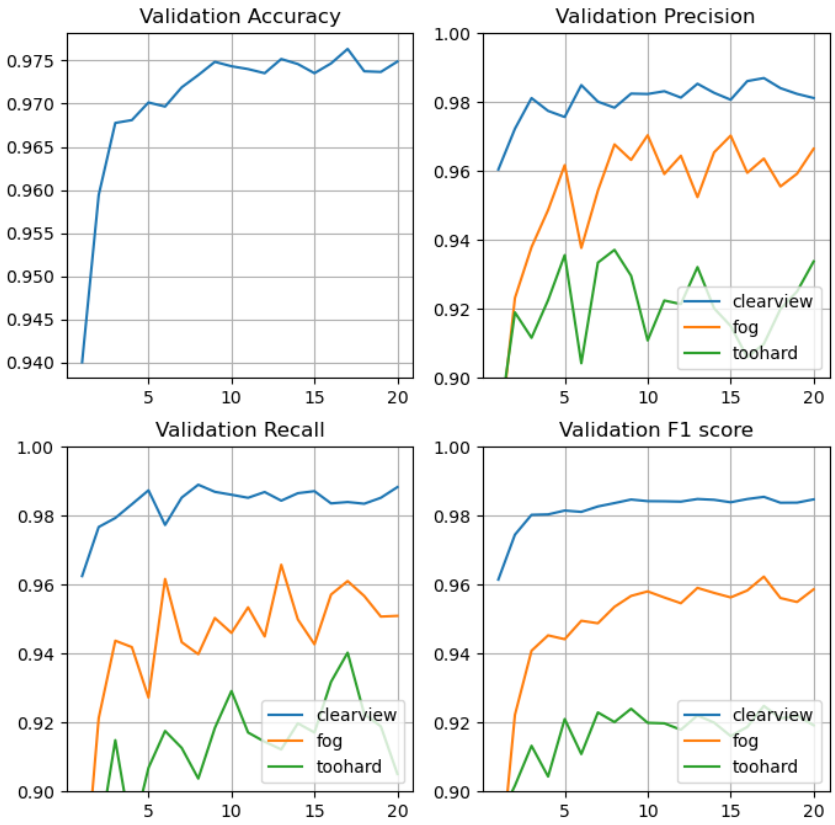


Figure 8: Results of the hyperparametertuning for the model based on the combined set of images of the six webcams.

the results between the combined model and the single models, the 80/20 split was made for all webcams individually and the resulting training and testing sets were combined into two final sets for the combined model. Important to note here is that this 80/20 split was made randomly without paying particular attention to keeping the class distribution constant in the training and testing set.

Figure 9 shows the confusion matrix constructed for the combined model based on the testing set. The overall accuracy is 97.6%, but all individual classes have a recall of almost 95%, with the 'clear view' and 'fog' classes even more. We can conclude that the model performs very well.

The confusion matrices shown in figure 10 compare the performance of the combined model with the performance of the single models. Each confusion matrix in the figure is calculated from the respective test set. The left column shows the confusion matrices of the single models, while the right column shows the confusion matrices of the combined model. A few general observations can be made:

- The overall performance is very similar between the single models and the combined model.
- The single model always performs better than the combined model in one of the 'clear view' or 'fog' classes, and worse in the other one. The class where the single model performs better is almost always the 'clear view' class, except for the webcam in Zeebrugge.

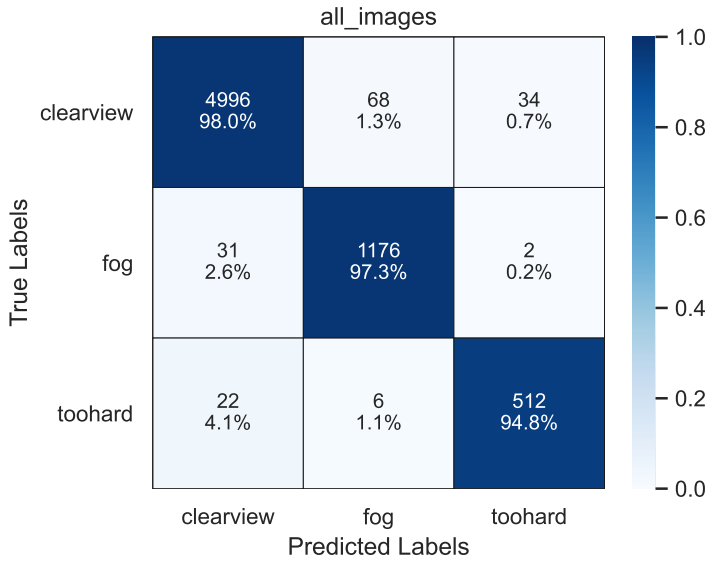


Figure 9: Confusion matrix of the combined model. The confusion matrix is based on a test set containing 20% of the images.

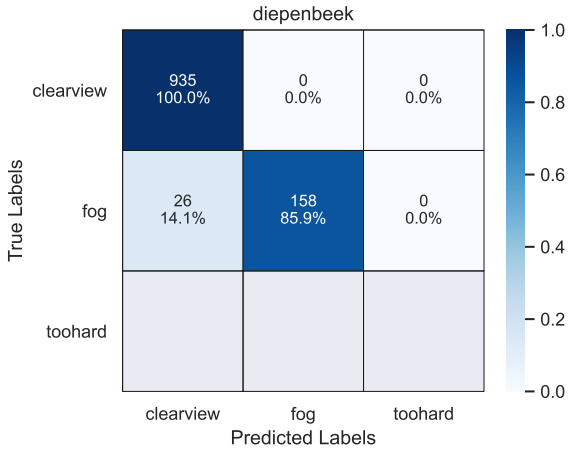
- The combined model always performs better than the single models for images corresponding to the 'too hard to see' class.
- The combined model never predicts a 'too hard to see' label for webcams for which such images do not exist.

The fact that the single models and the combined model each perform better in one of the 'clear view' and 'fog' classes, can be explained by how often each model predicts a certain class. Take the webcam in Diepenbeek as an example: the single model predicts 'clear view' more often, leading to a better recall for the 'clear view' class and a lower recall for the 'fog' class. Since most single models have a higher recall for the 'clear view' class, more false fog alarms can be expected, however the number of misses for the 'fog' class is lower. Depending on what one wants from the model, the single models or the combined model can be used.

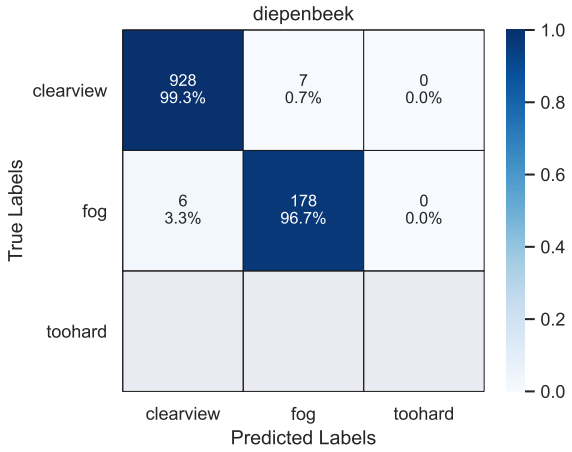
4.4 Performance on images of an unseen webcam

A final experiment that is performed, is a test to check whether the combined model can be used to also classify images of other webcams. To achieve this, the combined model was trained on all images of five of the six webcams and the images of the other webcam where then used as test set. This was done for each webcam.

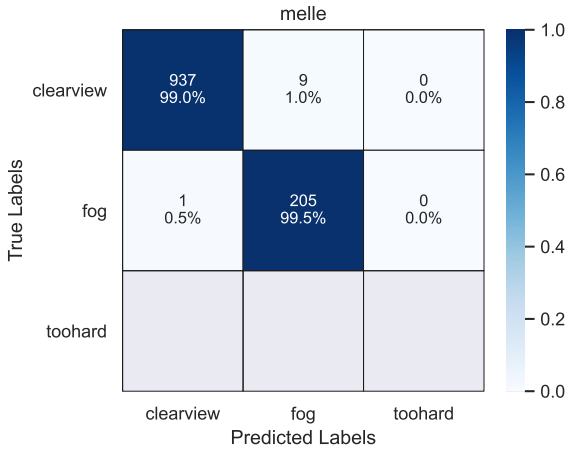
The results are shown in the confusion matrices in figure 11. From those figures, it is clear that the combined model cannot be used to classify images from webcams not included in the training set. For the webcam in Diepenbeek, the model classifies the images too often in the 'clear view' class. For the webcams in Melle, Wideumont and Zeebrugge, the model classifies the images too often in the 'fog' class. For the webcams in Mont Rigi and Stabroek, there is a mix between those two cases.



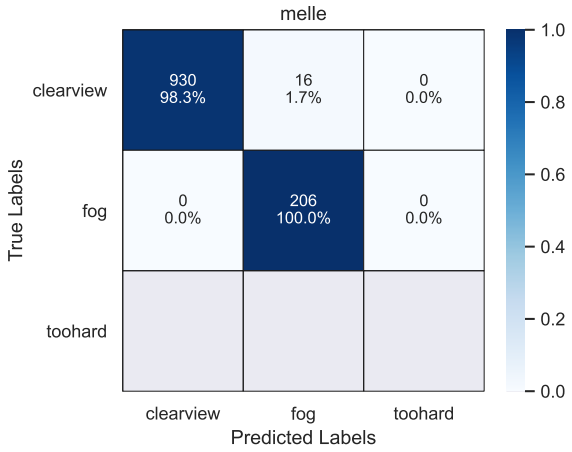
(a) Diepenbeek, single model



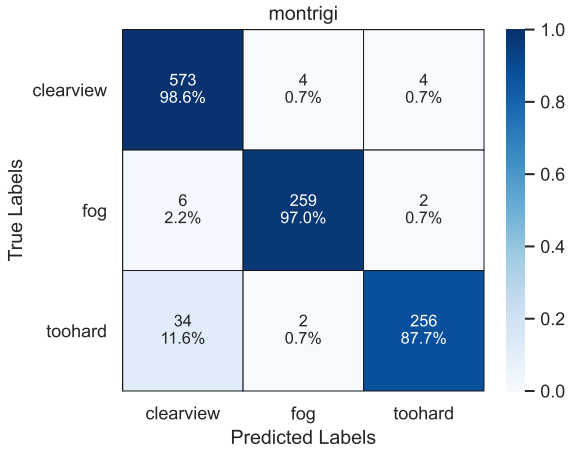
(b) Diepenbeek, combined model



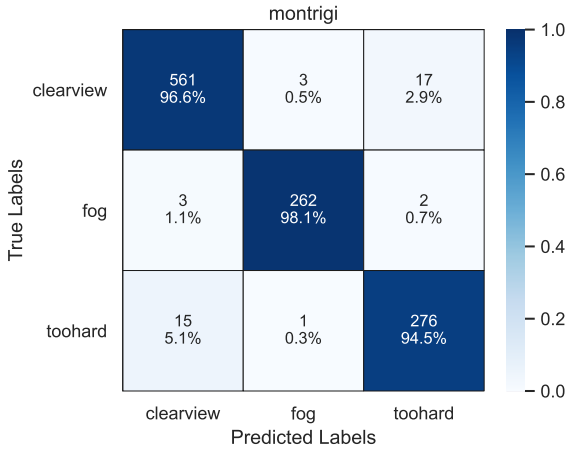
(c) Melle, single model



(d) Melle, combined model

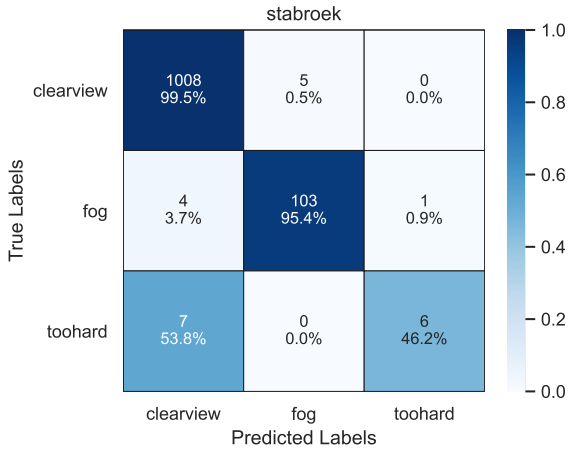


(e) Mont Rigi, single model

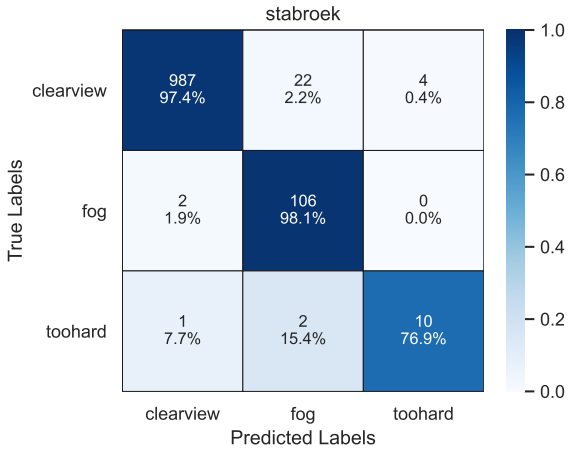


(f) Mont Rigi, combined model

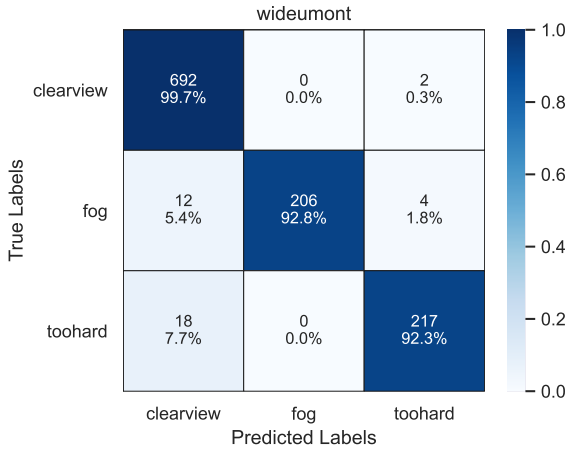
Figure 10: Confusion matrices for the six webcams. The left images show the results of the models based on only images of a single webcam, the right images show the results of the model based on the combined set of images of all webcams.



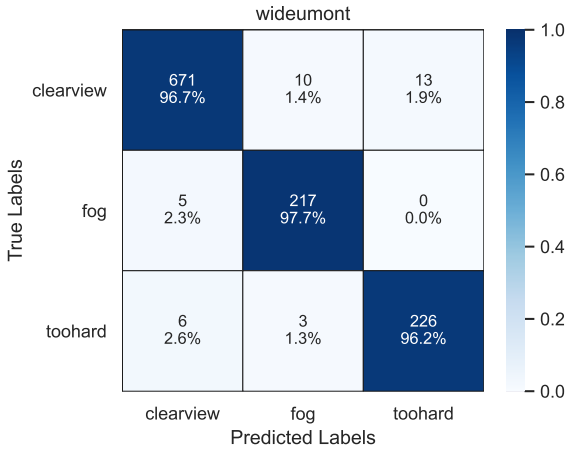
(g) Stabroek, single model



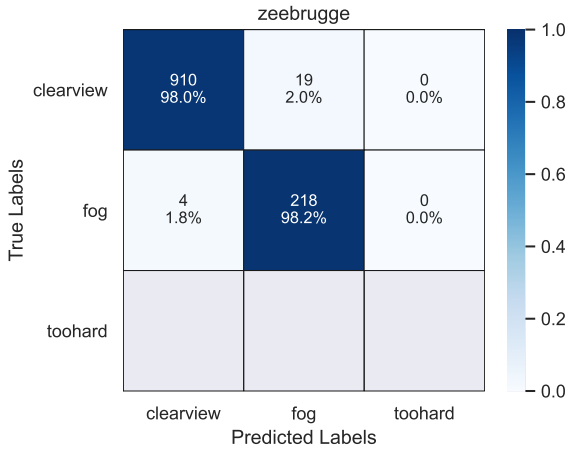
(h) Stabroek, combined model



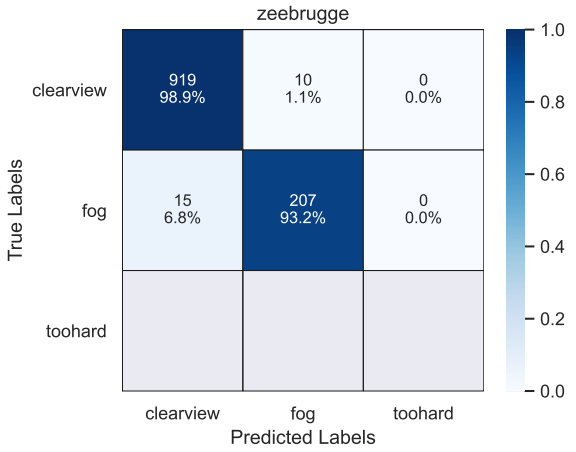
(i) Wideumont, single model



(j) Wideumont, combined model

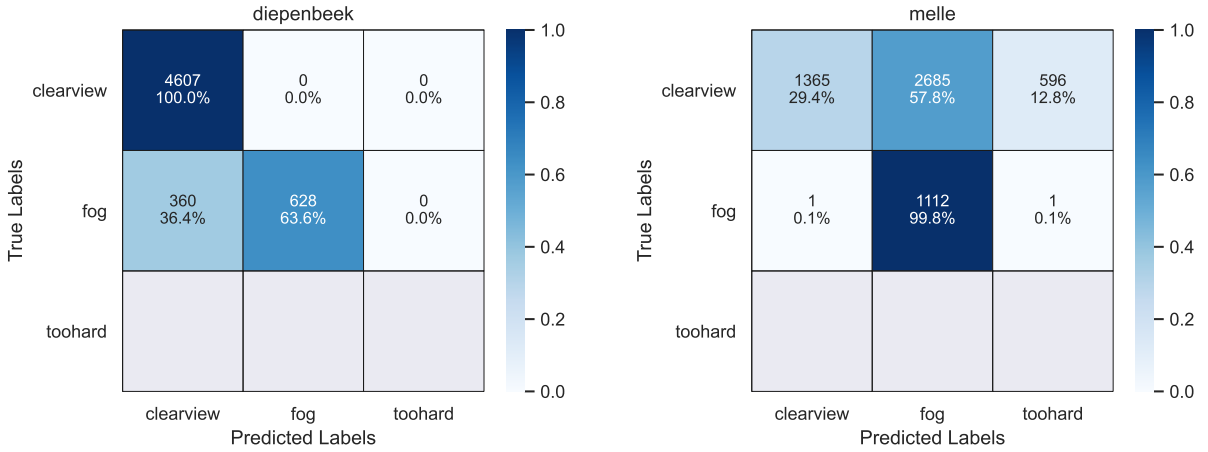


(k) Zeebrugge, single model



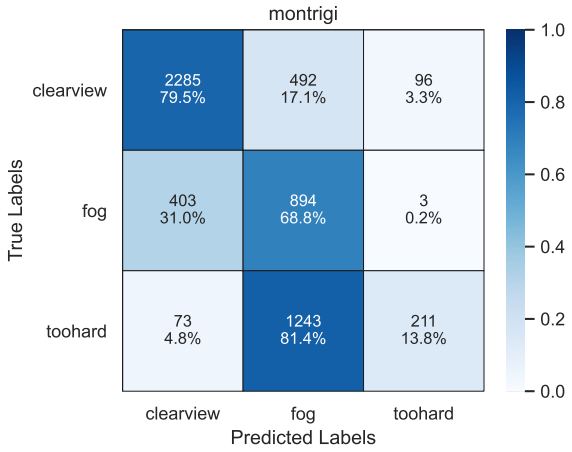
(l) Zeebrugge, combined model

Figure 10: Confusion matrices for the six webcams. The left images show the results of the models based on only images of a single webcam, the right images show the results of the model based on the combined set of images of all webcams.

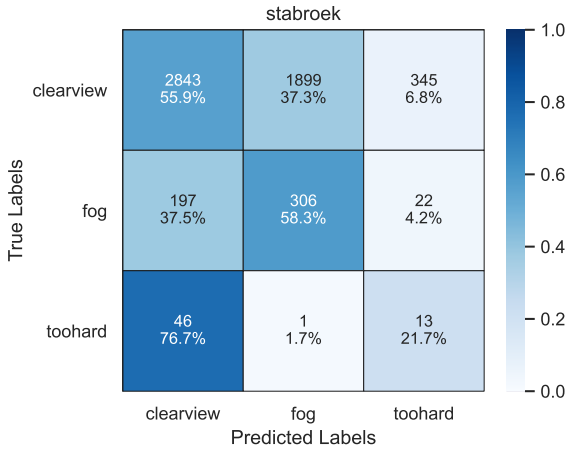


(a) Diepenbeek

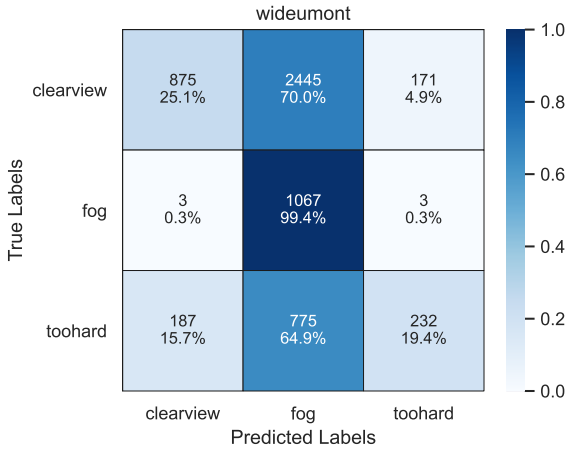
(b) Melle



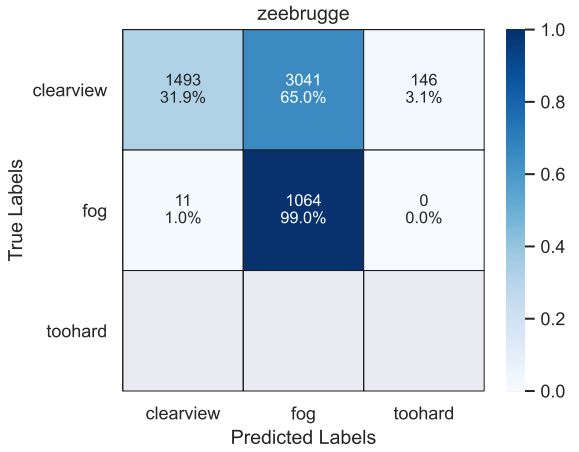
(c) Mont Rigi



(d) Stabroek



(e) Wideumont



(f) Zeebrugge

Figure 11: Confusion matrices for six models. The models are each trained on images of five webcams and tested on the images of the sixth webcam. For each confusion matrix, this sixth webcam is also mentioned below the figure.

4.5 Conclusion

Several models were constructed based on images of six webcams divided into three classes. The single models typically have a comparable performance to the combined model. Depending on the user's preference (fewer misses or fewer false alarm), one can either choose to use the single models to classify new images for the respective webcam, or one can use the combined model to classify new images of all webcams.

However, the models cannot be used to classify images of webcams not included in the training set. Hence, a minimal labelling effort is required for each webcam (the whole process of selecting the 20 days and labelling and relabelling those images takes 3-4 hours). After labelling the images of the new webcam, a model has to be trained. This can take a few days depending on whether the hyperparametertuning step is done or if a generic model configuration is chosen. However, contrary to the labelling process, the training process of the model is fully automated and can run without any active supervision.

A Distribution of the images over the different classes

clear view	dawn/dusk	fog	night	reduced visibility	total
38295 (37.4%)	6436 (6.3%)	8685 (8.5%)	40978 (40.0%)	8058 (7.9%)	102452

Table 5: Distribution of the images for the Mont Rigi webcam over the different classes (corresponds to table 1 in the text).

	clear view	fog	too hard to see	total
Diepenbeek	4607 (82.3%)	988 (17.7%)	0 (0%)	5595
Melle	4646 (80.7%)	1114 (19.3%)	0 (0%)	5760
Mont Rigi	2873 (50.4%)	1300 (22.8%)	1527 (26.8%)	5700
Stabroek	5087 (89.7%)	525 (9.3%)	60 (1.1%)	5672
Wideumont	3491 (60.6%)	1073 (18.6%)	1194 (20.7%)	5758
Zeebrugge	4680 (81.3%)	1075 (18.7%)	0 (0%)	5755
Combined	25384 (74.1%)	6075 (17.7%)	2781 (8.1%)	34240

Table 6: Distribution of the images for the different webcams over the different classes (corresponds to table 3 in the text).

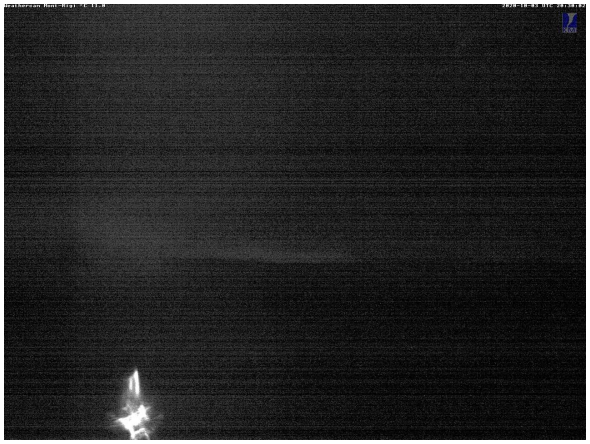
B Examples of webcam images and their classification

B.1 Examples of the images of the Mont Rigi webcam (five classes)

(a) clear view

(b) dawn/dusk

(c) fog



(d) night

(e) reduced visibility

Figure 12: Examples of the images of the Mont Rigi webcam (five classes).

B.2 Examples of the images of the multiple webcams (three classes)

B.2.1 Diepenbeek

(a) clear view, example 1

(b) clear view, example 2

(c) fog, example 1

(d) fog, example 2

Figure 13: Examples of the images of the Diepenbeek webcam (three classes).

B.2.2 Melle

(a) clear view, example 1

(b) clear view, example 2

(c) fog, example 1

(d) fog, example 2

Figure 14: Examples of the images of the Melle webcam (three classes).

B.2.3 Mont Rigi

(a) clear view, example 1

(b) clear view, example 2

(c) fog, example 1

(d) fog, example 2

(e) too hard to see

Figure 15: Examples of the images of the Mont Rigi webcam (three classes).

B.2.4 Stabroek

(a) clear view, example 1

(b) clear view, example 2

(c) fog, example 1

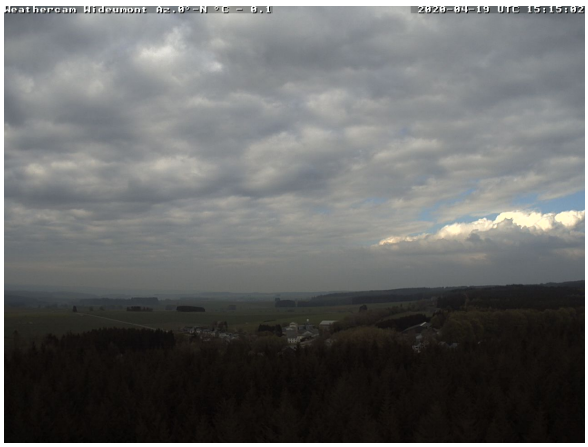
(d) fog, example 2

(e) too hard to see, example 1

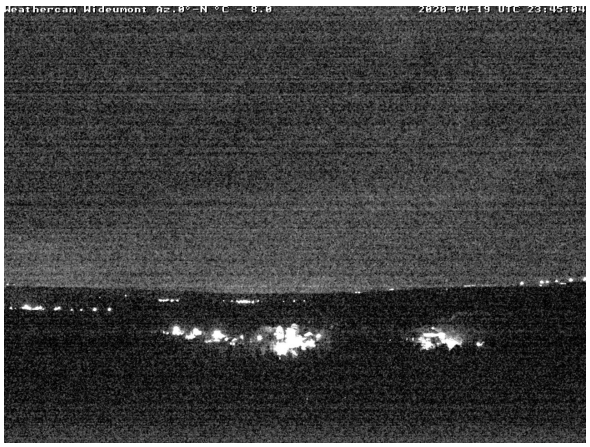
(f) too hard to see, example 2

Figure 16: Examples of the images of the Stabroek webcam (three classes).

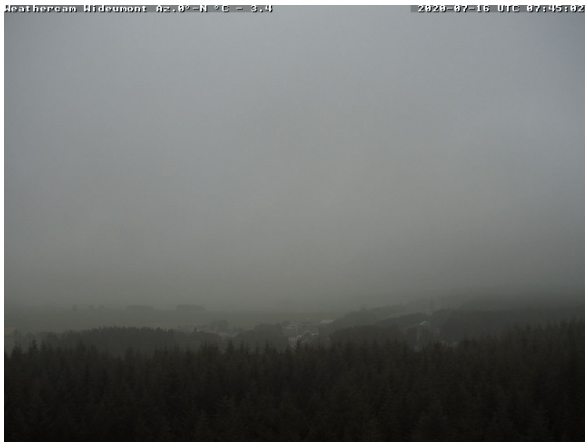
B.2.5 Wideumont



(a) clear view, example 1



(b) clear view, example 2



(c) fog, example 1

(d) fog, example 2

(e) too hard to see, example 1

(f) too hard to see, example 2

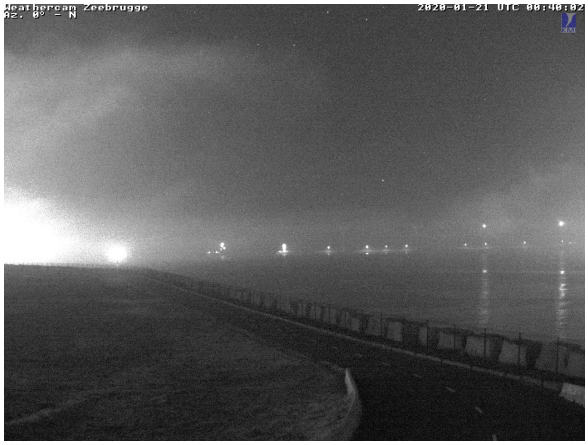
Figure 17: Examples of the images of the Wideumont webcam (three classes).

B.2.6 Zeebrugge

(a) clear view, example 1

(b) clear view, example 2

(c) fog, example 1



(d) fog, example 2

Figure 18: Examples of the images of the Zeebrugge webcam (three classes).

C Hyperparametertuning results (three classes)

C.1 Diepenbeek

Configuration of the final model:

- 2 convolutional layers
- ‘tanh’ activation function
- 3x3 kernel
- stride length 1
- 16 kernels in the first convolutional layer
- 64 neurons in the first dense layer

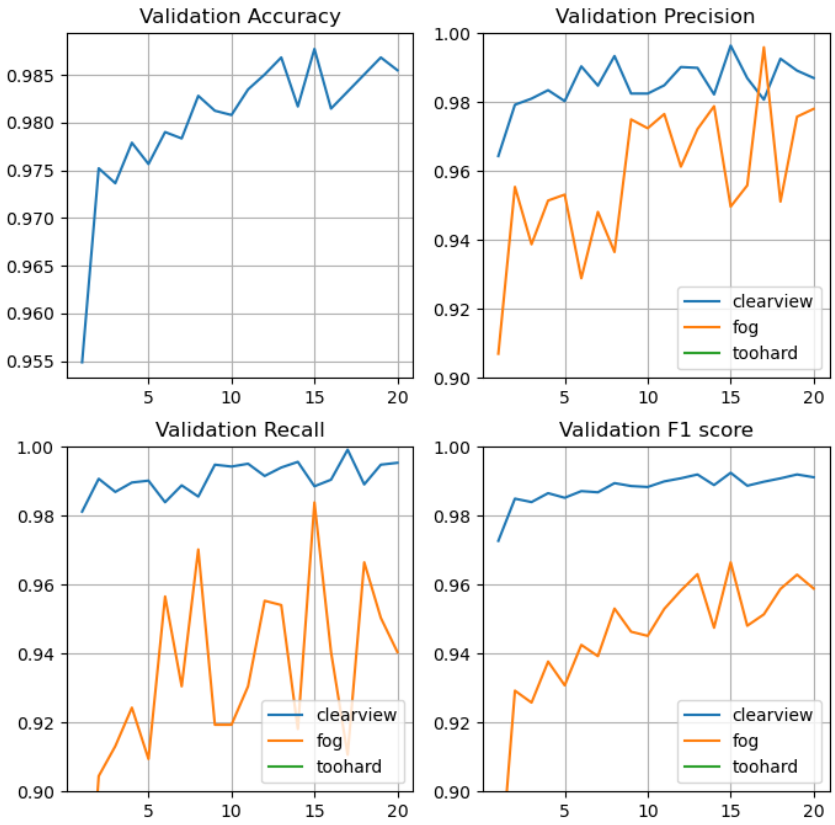


Figure 19: Results of the hyperparametertuning for the model based on the images of the Diepenbeek webcam.

C.2 Melle

Configuration of the final model:

- 3 convolutional layers
- ‘tanh’ activation function
- 4x4 kernel
- stride length 1
- 8 kernels in the first convolutional layer
- 64 neurons in the first dense layer

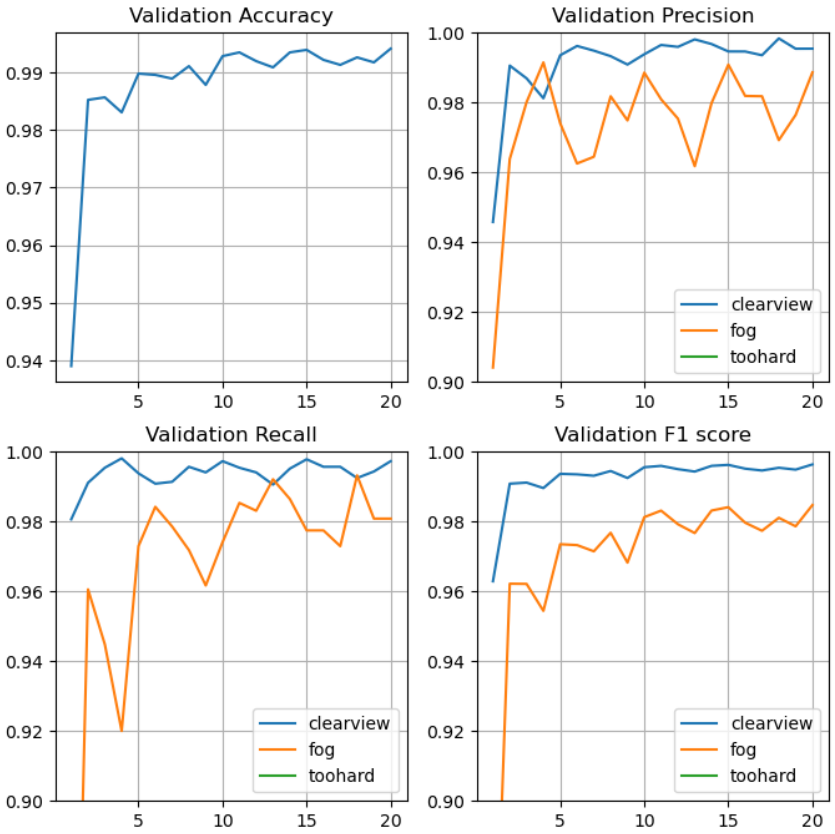


Figure 20: Results of the hyperparametertuning for the model based on the images of the Melle webcam.

C.3 Mont Rigi

Configuration of the final model:

- 2 convolutional layers
- ‘relu’ activation function
- 4x4 kernel
- stride length 1
- 8 kernels in the first convolutional layer
- 64 neurons in the first dense layer

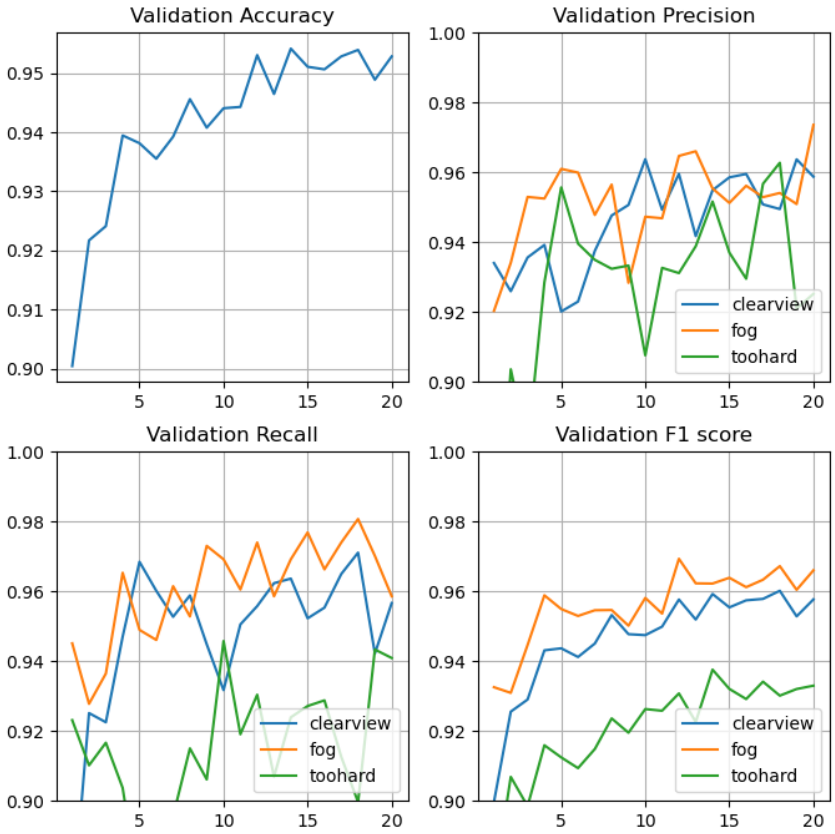


Figure 21: Results of the hyperparametertuning for the model based on the images of the Mont Rigi webcam.

C.4 Stabroek

Configuration of the final model:

- 2 convolutional layers
- ‘tanh’ activation function
- 4x4 kernel
- stride length 1
- 8 kernels in the first convolutional layer
- 64 neurons in the first dense layer

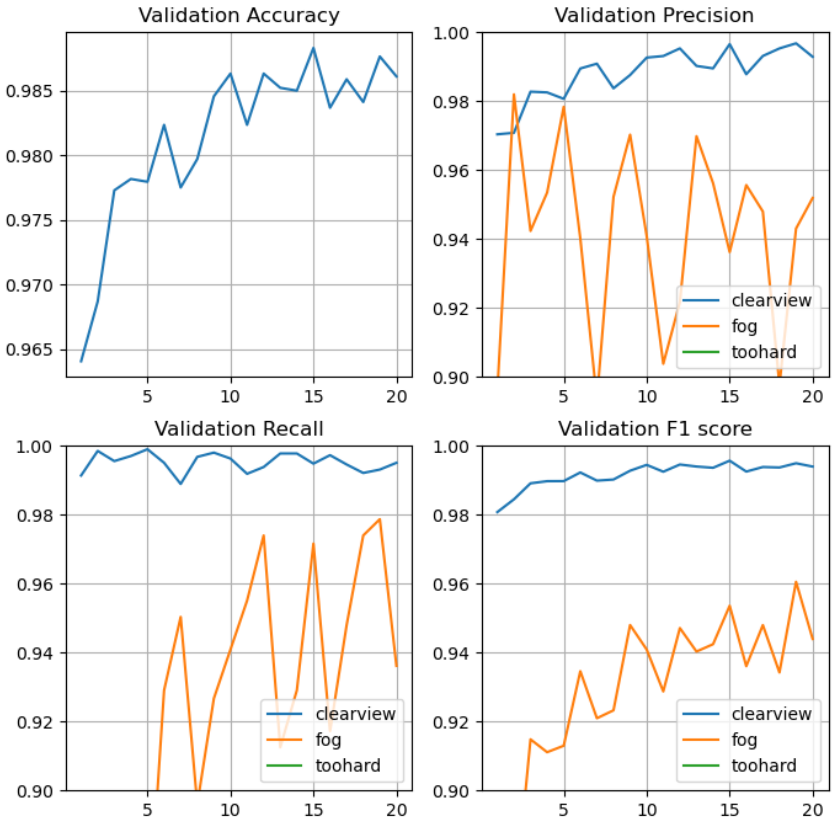


Figure 22: Results of the hyperparametertuning for the model based on the images of the Stabroek webcam.

C.5 Wideumont

Configuration of the final model:

- 2 convolutional layers
- ‘relu’ activation function
- 3x3 kernel
- stride length 1
- 8 kernels in the first convolutional layer
- 128 neurons in the first dense layer

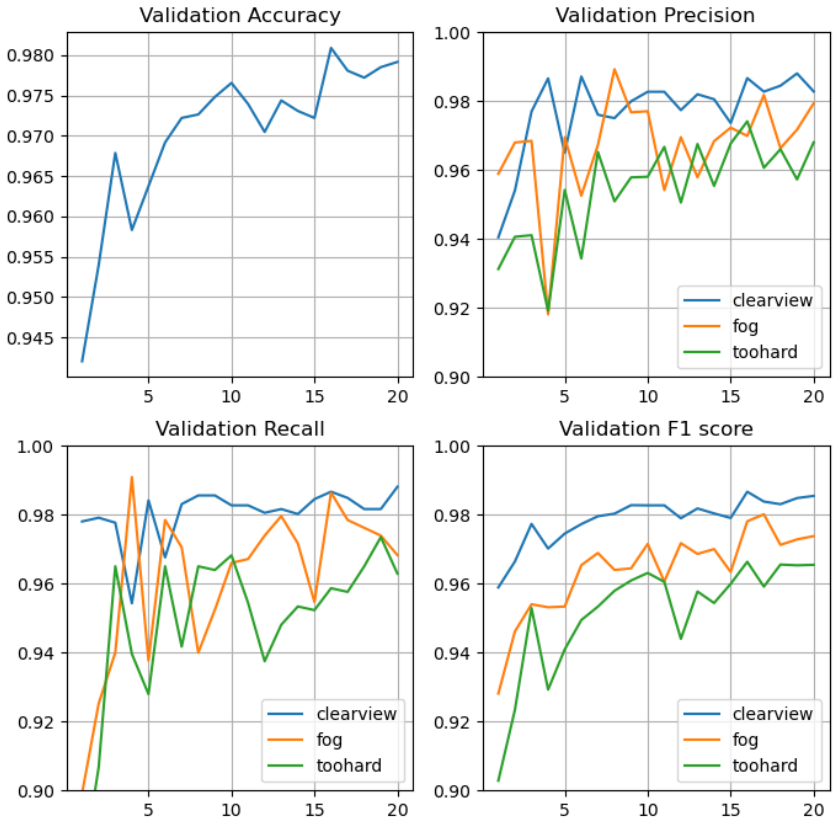


Figure 23: Results of the hyperparametertuning for the model based on the images of the Wideumont webcam.

C.6 Zeebrugge

Configuration of the final model:

- 2 convolutional layers
- 'elu' activation function
- 3x3 kernel
- stride length 1
- 16 kernels in the first convolutional layer
- 64 neurons in the first dense layer

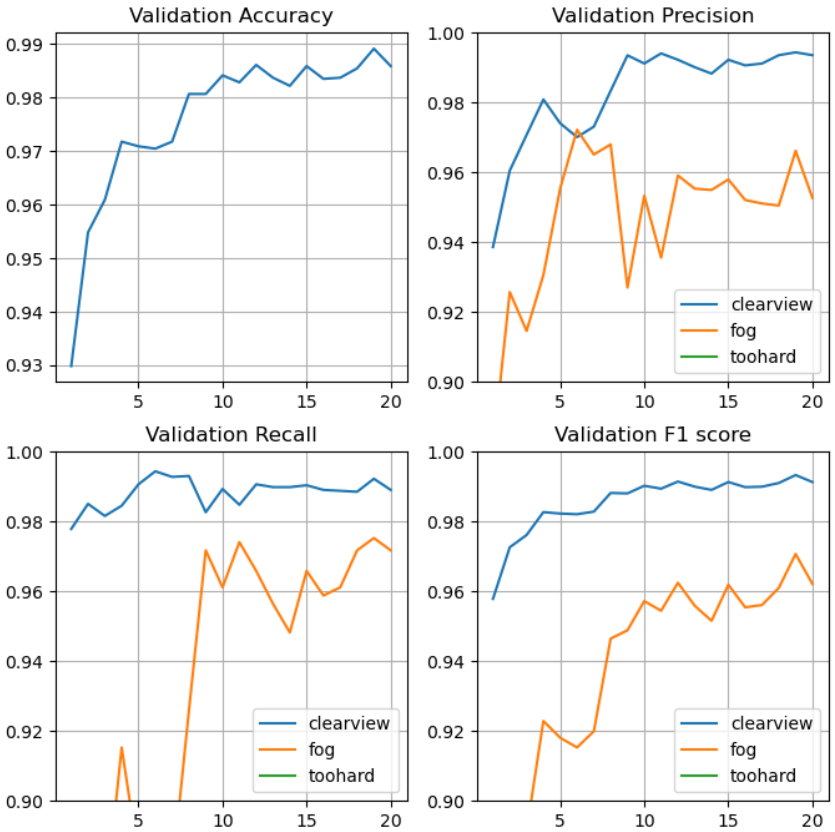


Figure 24: Results of the hyperparametertuning for the model based on the images of the Zeebrugge webcam.

Automated fog detection on the RMI webcam images using machine learning techniques

Lars Heylen, Maarten Reyniers

DOI: 10.5281/zenodo.7685079

Koninklijk Meteorologisch Instituut van België
Institut Royal Météorologique de Belgique
Königliches Meteorologisches Institut von Belgien
Royal Meteorological Institute of Belgium