Fog Detection for Autonomous Vehicles Using Convolutional Neural Network

Hongtao Hao
University of Wisconsin Madison
Madison, WI 53706
hhao9@wisc.edu

Abstract

As autonomous vehicles (AVs) gain popularity, traffic safety remains a top concern related to this emerging technology. Adverse weather conditions affect the functioning of sensors and cameras of AVs and pose a serious threat to traffic safety. Therefore, early detection of adverse weather is crucial for AV safety. This paper focuses on detecting fogs, which are a significant weather-related hazard. Existing fog detection technologies can be grouped into three categories: feature engineering, neural networks, and hybrid approaches. This study uses a convolutional neural network (CNN) to detect fogs in static images taken by an in-vehicle camera. Our approach achieves an accuracy of 73%, indicating that it is reliable to detect fogs for autonomous driving.

1 Introduction

As autonomous vehicles (AVs) are gaining popularity, traffic safety is a top concern related to this emerging technology. Ensuring safety is the primary concern for AV manufacturers. To bring AVs to the market faster, it is crucial to improve their safety on the road. Traffic accidents compromise the safety of road users and a significant cause of accidents is extreme weather conditions such as rain, haze, and snow. According to a report by the US Department of Transportation, 21% of crashes from 2007 to 2016 were caused by weather events [1]. This is especially true for AVs that rely on sensors and cameras as inclement weather hinders the functioning of sensing equipment [2], [3]. To improve AV safety, early detection of adverse weather conditions is crucial, followed by alerting either the driver or a central control system to react. This paper focuses on a specific adverse weather condition: fog.

2 Related Work

Existing fog detection technologies can be grouped into three categories: feature engineering approach, neural network approach, and hybrid approach.

Feature engineering approach refers to the method of extracting key features that are relevant to fogs and then do the classification based on these features. This method requires domain knowledge from experts.

Some of the features included are hough line and vanishing point [4], backscattered veil and halos surrounding light sources [5], power spectrum slop (PSS) [6], windowing [7], average saturation in a region of interest [8], etc. These methods are are shown to achieve high levels of accuracy (over 95%), for example, in [5]–[8].

An example of using neural networks is [9]. They compared the prediction results of deep neural network (DNN), recurrent neural network (RNN), long short-term memory (LSTM) and convolutional neural network (CNN). Data for this study is extracted from in-vehicle cameras. The results show that CNN with Adam optimizer yields the best prediction accuracy, i.e., 98%. The study demonstrates that a camera alone is able to allow accurate fog detection for autonomous driving.

[10] is an example where a hybrid approach is employed. Their method is hybrid because they used both feature selection and a neural network. Specifically, they chose Shannon entropy and discrete cosine transform (DCT) as predefined features. They also utilized a CNN for learnable features. They distinguish between three visibility levels and testing accuracy using their method reached 87%.

3 Methodology

3.1 Data

The fog data is from [11]. They generated synthetic fogs on the existing Cityscape dataset [12] using fog simulation. There are three levels of fogs based on visibility: low fog (600m visibility), medium fog (300 visibility) and high fog (150m visibility).

The whole data is very big (20G). It includes foggy images based on street scenes in Germany and Switzerland. This study only includes 522 images (174 scenes with three fog levels) in Aachen, Germany.

To reduce computational complexity, each image is resized to be 256×256 pixels using bilinear interpolation. These resized images in RGB format (with red, green, and blue channels) are used for training and testing. Samples of resized images are shown in Fig. 1.

3.2 Model

Tensorflow [13] is used to implement a CNN architecture with three input channels. The model is straightforward and simple: It consists of four convolutional layers, each of which is followed by a max pooling layer that reduces the dimensions by a factor of 2. Three dense layers followed the last max pooling layer. We trained our CNN model with a batch size of 32 and for 32 epochs. The model was trained using the Adam optimizer with a learning rate of 0.001. The loss function is cross-entropy loss. This model is trained on a CPU of a 2015 Mac and the training took 30 minutes. We randomly selected 80% (i.e., 417 pictures) of the data for training and the rest (105 pictures) to evaluate the model's performance.

The details of the architecture are as follows. The first layer is a convolutional layer with 32 filters and a kernel size of $(3, 3)$. ReLU activation is used. This layer takes in input images of size 256×256 pixels and the output is a tensor of shape $(32, 256, 56, 32)$. The second layer is a max pooling layer with a pool size of $(2, 2)$, reducing the dimensions of the output of the previous layer by half. The output of this layer is a tensor of shape $(32, 128, 128, 32)$.

The third layer is another 2D convolutional layer with 64 filters and a kernel size of $(3, 3)$. ReLU activation is used. This layer takes in the output of the second layer and produces a tensor of shape $(32, 128, 128, 64)$. The fourth layer is a max pooling layer with a pool size of $(2, 2)$. The output of this layer is a tensor of shape $(32, 64, 64, 32)$.

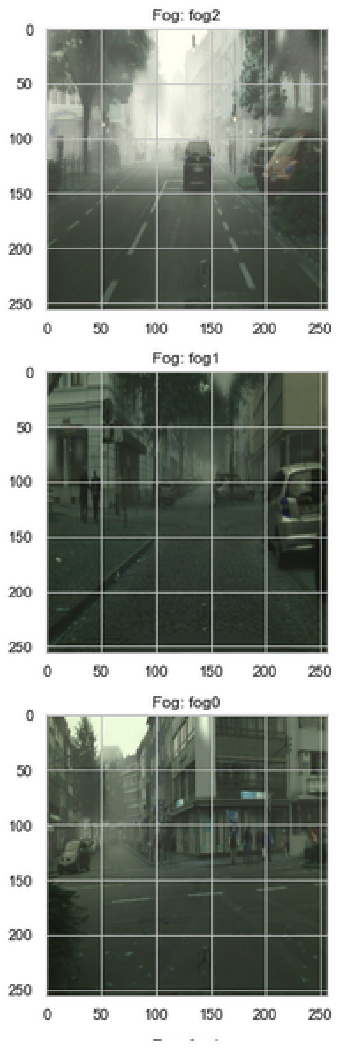
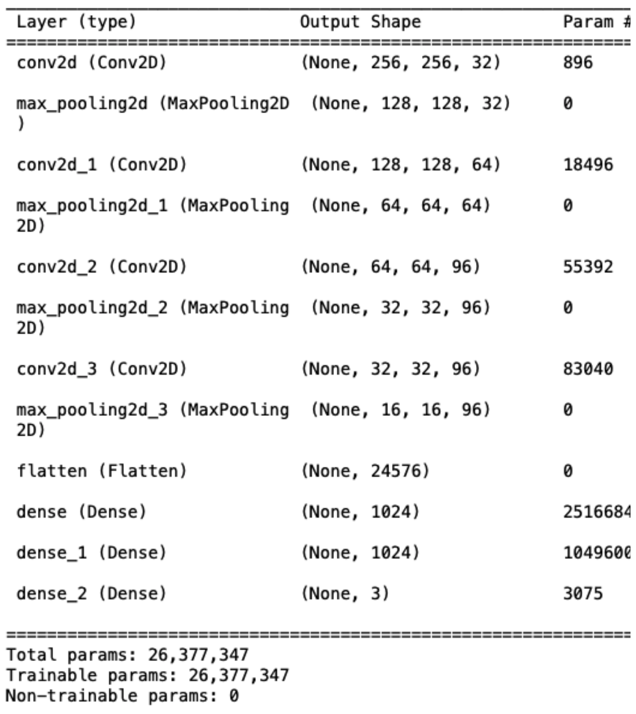


Figure 1: Sample resized images for training and testing

The fifth layer is another 2D convolutional layer with 96 filters and a kernel size of $(3, 3)$. ReLU activation is used. This layer takes in the output of the fourth layer and produces a tensor of shape $(32, 64, 64, 96)$. The sixth layer is a max pooling layer with a pool size of $(2, 2)$. The output of this layer is a tensor of shape $(32, 32, 32, 96)$.

The seventh layer is another 2D convolutional layer with 96 filters and a kernel size of $(3, 3)$. ReLU activation is used. This layer takes in the output of the sixth layer and produces a tensor of shape $(32, 32, 32, 96)$. The eighth layer is a max pooling layer with a pool size of $(2, 2)$. The output of this layer is a tensor of shape $(32, 16, 16, 96)$.

The ninth layer is a flatten layer that turns the output of the previous layer into a 1D tensor of



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 max_pooling2d (MaxPooling2D  (None, 128, 128, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 128, 128, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 64, 64, 96)        55392

 max_pooling2d_2 (MaxPooling  (None, 32, 32, 96)       0
 2D)

 conv2d_3 (Conv2D)           (None, 32, 32, 96)        83040

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 96)       0
 2D)

 flatten (Flatten)           (None, 24576)             0

 dense (Dense)               (None, 1024)              2516684

 dense_1 (Dense)             (None, 1024)              1049600

 dense_2 (Dense)             (None, 3)                 3075

=================================================================
Total params: 26,377,347
Trainable params: 26,377,347
Non-trainable params: 0
```

Table 1: CNN model specification

shape $(32, 24576)$.

The tenth layer is a dense layer with $1,024$ neurons and ReLU activation. It takes in the flattened tensor and produces the output tensor of shape $(32, 1024)$. The eleventh layer is another dense layer with $1,024$ neurons and ReLU activation. The output tensor is also of shape $(32, 1024)$.

The twelfth layer is the final dense layer with 3 neurons and softmax activation. This layer outputs the probability distribution over 3 classes the model is trained to predict.

This model has $26,377,347$ trainable parameters.

Table 1 shows the specification of the CNN model.

4 Results

As can be seen in Fig. 2, the model works relatively well. The decrease in loss is obvious. The accuracy starts at 30% and eventually reaches above 70%. Given the simple architecture of this CNN model, the performance is acceptable.

5 Discussion

This paper aims to detect fogs in images taken by an in-vehicle camera. A simple CNN model is able to reach 70% accuracy with three classes of fog: low, medium and high. The results indicate that cameras can be an inexpensive and reliable option for inclement weather detection for autonomous vehicles (AVs).

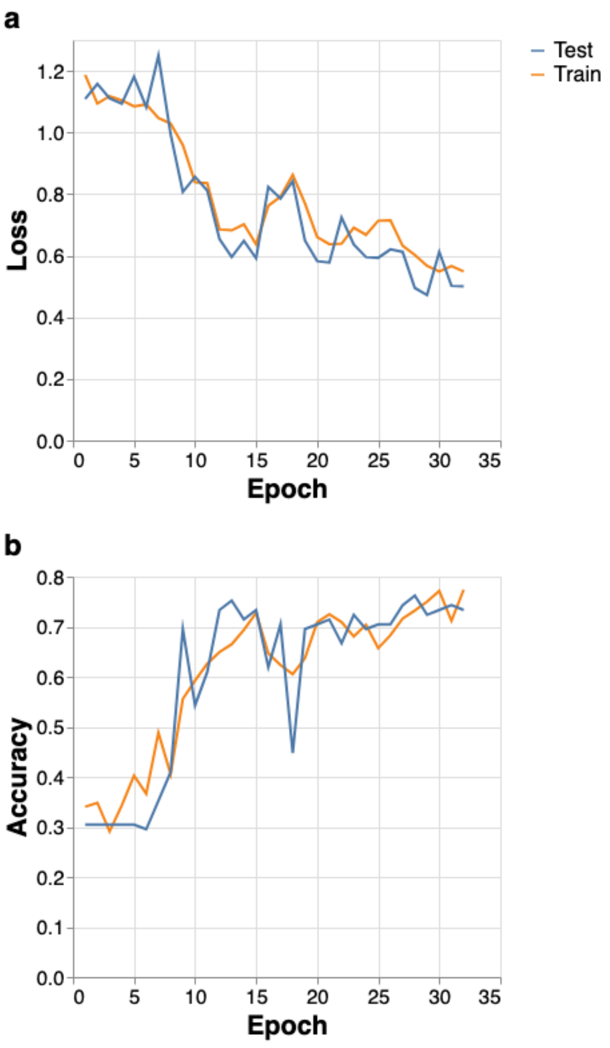


Figure 2: Loss and Accuracy

Accuracy of 70% may not be high enough in the setting of autonomous driving. This low accuracy may be due to the simplicity of the CNN architecture and the pre-processing of the images for training. To improve the accuracy, the following options can be considered:

- Increase image resolution when processing images for model training. For example, instead of using 256×256 pixels, use 512×512 pixels.
- Considering the resources we had, we chose to set the epoch to be only 32. Increasing this number may yield better results.
- We only chose a small fraction of the full

dataset in [11]. Utilizing the full data should increase the model performance.

- More complicated models than CNN can be used, for example, ResNet [14].

This study has several limitations. First, the dataset in [11] is synthetic. Model performance may not be as good for real foggy images. Second, for autonomous driving, it is necessary to detect fog in real-time. Our model may be very slow when classifying videos.

References

[1] U.S. Department of Transportation, *How do weather events impact roads?* `https://ops.fhwa.dot.gov/weather/q1_roadimpact.htm`, Last accessed on 2023-05-09.

[2] S. Zang, M. Ding, D. Smith, P. Tyler, T. Rakotoarivelo, and M. A. Kaafar, "The impact of adverse weather conditions on autonomous vehicles: How rain, snow, fog, and hail affect the performance of a self-driving car," *IEEE vehicular technology magazine*, vol. 14, no. 2, pp. 103–111, 2019.

[3] L. Hespel, N. Riviere, T. Huet, B. Tanguy, and R. Ceolato, "Performance evaluation of laser scanners through the atmosphere with adverse condition," in *Electro-Optical Remote Sensing, Photonic Technologies, and Applications V*, SPIE, vol. 8186, 2011, pp. 64–78.

[4] S. Bronte, L. M. Bergasa, and P. F. Alcantarilla, "Fog detection system based on computer vision techniques," in *2009 12th International IEEE conference on intelligent transportation systems*, IEEE, 2009, pp. 1–6.

[5] R. Gallen, A. Cord, N. Hautière, and D. Aubert, "Towards night fog detection through use of in-vehicle multipurpose cameras," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 399–404.

[6] R. Spinneker, C. Koch, S.-B. Park, and J. J. Yoon, "Fast fog detection for camera based advanced driver assistance systems," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 1369–1374.

[7] M. Pavlić, H. Belzner, G. Rigoll, and S. Ilić, "Image based fog detection in vehicles," in *2012 IEEE Intelligent Vehicles Symposium*, IEEE, 2012, pp. 1132–1137.

[8] K. Y. Choi, K. M. Jeong, and B. C. Song, "Fog detection for de-fogging of road driving images," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017, pp. 1–6.

[9] M. N. Khan and M. M. Ahmed, "Trajectory-level fog detection based on in-vehicle video camera with tensorflow deep learning utilizing shrp2 naturalistic driving data," *Accident Analysis & Prevention*, vol. 142, p. 105 521, 2020.

[10] V. Vaibhav, K. R. Konda, C. Kondapalli, K. Praveen, and B. Kondoju, "Real-time fog visibility range estimation for autonomous driving applications," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–6.

[11] C. Sakaridis, D. Dai, S. Hecker, and L. Van Gool, "Model adaptation with synthetic and real data for semantic dense foggy scene understanding," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 687–704.

[12] M. Cordts, M. Omran, S. Ramos, *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

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