Classification of LIDAR Sensor Contaminations with Deep Neural Networks

Jyothish K. James
University of Kaiserslautern
Kaiserslautern, Germany
jyothish@rhrk.uni-kl.de

Georg Puhlfürst
Valeo Schalter und Sensoren GmbH
Kronach, Germany
georg.puhlfuerst@valeo.com

Vladislav Golyanik
DFKI
Kaiserslautern, Germany
vladislav.golyanik@dfki.de

Didier Stricker
DFKI
Kaiserslautern, Germany
didier.stricker@dfki.de

ABSTRACT

Light detecting and ranging (LIDAR) sensors are extensively studied in autonomous driving research. Monitoring the performance of LIDAR sensors has become significantly important to ensure their reliability and hence guarantee the safety of the vehicle. Underestimation of sensor performance can give away reliable object data, overestimation may result in safety issues. Besides light and weather conditions, the performance is strongly affected by contaminations on the sensor front plate. In this paper, we focus on classifying different types of contaminations using a deep learning approach. We train a deep neural network (DNN) following a multi-view concept. For the generation of training and test data, experiments have been conducted, in which the front plate of a LIDAR sensor has been contaminated artificially with various substances. The recorded data is transformed to contain the essential information in a compact format. The results are compared to classical machine learning techniques to demonstrates the potential of DNN approaches for the problem under consideration.

KEYWORDS

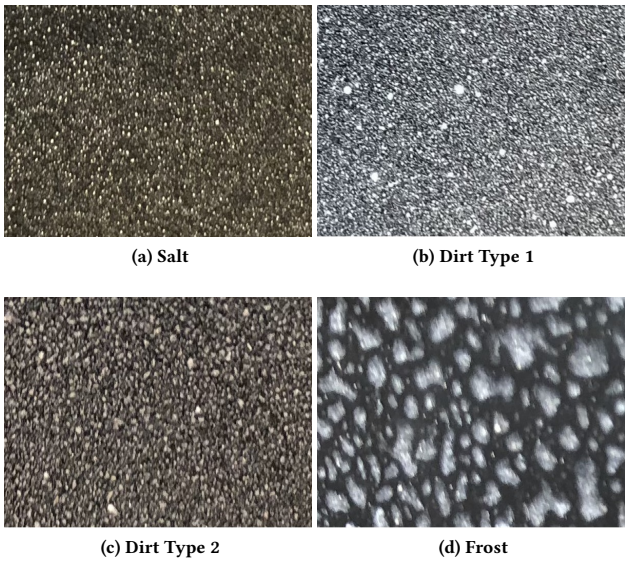
LIDAR, Deep-learning, Multi-view, Transfer learning, Scan, Scanpoints.

ACM Reference Format:
Jyothish K. James, Georg Puhlfürst, Vladislav Golyanik, and Didier Stricker. 2018. Classification of LIDAR Sensor Contaminations with Deep Neural Networks. In *2nd Computer Science in Cars Symposium - Future Challenges in Artificial Intelligence Security for Autonomous Vehicles (CSCS 2018), September 13–14, 2018, Munich, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3273946.3273952

1 INTRODUCTION

Light detecting and ranging (LIDAR) sensors [17] have become an important aspect for the autonomous driving industry. These components are used to detect lanes, cars, pedestrians and other objects [2, 11]. While a high performance of the corresponding detection functions is desirable, it is also important that this performance is monitored [27]. In case a relevant object is not detected by a LIDAR sensor, the system should take this performance limitation into consideration to prevent critical situations. Likewise, a good performance should not be underestimated to avoid giving away reliable data. In the context of sensor data fusion [7, 9], it is essential for each component to self-monitor its performance so that dependencies can be avoided. Hence, it is necessary to a) identify all influences on the sensor performance, b) detect and classify those influences and c) evaluate the impact of those influences on detection functions of the sensor. This allows to provide the system with reliable data, which is the foundation for an effective data fusion under varying traffic and environmental conditions.

In autonomous cars, LIDAR sensors are used along with cameras, radar and ultrasonic sensors. LIDAR sensors generate high resolution point clouds of the environment. A point cloud consists



(a) Salt **(b) Dirt Type 1**

(c) Dirt Type 2 **(d) Frost**

Figure 1: Different types of contaminations over the LIDAR front plate: (a) Salt, (b) Dirt Type 1, (c) Dirt Type 2, (d) Frost.

CSCS 2018, September 13–14, 2018, Munich, Germany

ACM ISBN 978-1-4503-6616-8/18/09…$15.00
https://doi.org/10.1145/3273946.3273952

of scan points, each of which is given a 3-dimensional position and, depending on the signal processing concept of the sensor, a quantity representing the intensity of the corresponding signal. The main advantages of LIDAR sensors are high range and angular resolutions in combination with large detection distances. The disadvantage of LIDAR sensors are the sensitivity to light, weather and sensor front conditions.

Contaminations or damages on the sensor front reduce the amount of transmitted light both in the sender and the receiver path. This is caused by scattering and absorption of the laser light. The point cloud is affected by this in various ways [21] a) The angular and range accuracy is decreased, b) the intensity of signals is reduced, which can lead up to c) missing scan points at certain distances and d) increased intensity and amount of near field signals. Those effects depend on the particular contamination or damage scenario, of which there are various combinations of types[1] (e.g. salt, dust, insects, snow, ice, scratches and stone chippings, few of which are shown in Figure 1) and distribution (e.g. full or partial blockages with different thicknesses of the blocking material). Hence, for the evaluation of the impact on the point cloud, not only the detection of sensor front influences, but also the classification is relevant. It also plays a significant role in improving the efficiency of heating and cleaning procedures. Contamination like dirt or salt need to be removed through cleaning procedures, snow or ice can only be removed through heating, while damages can not be removed at all.

In section 2, related works are discussed in the context of LIDAR performance as well as multi-modal fusion. Section 3 focusses on the experiments performed to record and generate the dataset, by applying different contaminations on the LIDAR sensor front plate. Furthermore, in this sections we propose a novel solution for classifying the contamination types with a deep neural network. The approach uses a multi-view deep neural network (LIDAR-Net), which fuses the front and the top view of the LIDAR sensors. In section 4, we evaluate the proposed approach and demonstrate that our results outperform those of classical algorithms. In our conclusion in section 5, we bring up possible aspects of future works.

2 RELATED WORK

In this section we review the algorithms and related works and position the proposed multi-view deep neural network among them.

2.1 Performance of laser scanners

The related work [21] tries to solve the problem of estimating the performance of LIDAR sensors by quantify the influence of contaminations on LIDAR sensors performance. The approach measures the transmission and reflecting properties and analyze LIDAR sensors position measurement accuracy, and identify potential forward scattering caused by contaminations. Real-world contaminations samples accumulated over the LIDAR front plate were collected and analyzed in the experiments, which is similar to our data collection pipeline. Our proposed approach classifies various types of contaminations. We measure the classification accuracy by performing various experiments, with different combinations of labels. In the related work [21], the influence of contaminations on the position measurement is calculated as a standard deviation relative to a clean sensor. The correlation between contaminations and the presence of rain and the performance degradation has also been studied. In our approach, we estimate the classification accuracy with deep neural networks. We performed various experiments to record the training and testing data from different traces. A part of the test dataset was used for validation, during the training phase. The final test results contributed to the classification accuracy.

2.2 Multi-modal Fusion

In the related work [2] a multi-view concept is introduced, which takes LIDAR point cloud and RGB images as input to predict 3D bounding boxes for object detection. The LIDAR point clouds are transformed to the front and top view respectively. Finally, the inputs to the network [2] are front view, top view and the RGB images. Similarly, in our approach we have introduced the multi-view concept by transforming the front and top view of the LIDAR sensor to usable image formats, which is given as inputs to a deep neural network (LIDAR-Net). The related work [2] introduces the concepts of early fusion, late fusion and deep fusion of deep neural networks. In our approach, the multi-model fusion was inspired by Fractal-Net [15] and Deeply-Fused Net [29], where we have adopted the late fusion over other fusion approaches. We implemented the late fusion of the views by applying the concatenation operation. Finally, our method gives the highest classification accuracy for classifying different types of LIDAR sensor front influences which are similar to the high accuracy results of object detection in the related work [2],

3 PROPOSED APPROACH

In this section we discuss a novel approach for classifying the contamination accumulated over the LIDAR sensor front plate. We use the method of deep-learning for solving this problem. Two different deep-learning approaches were used 1. Transfer learning [30], for carrying out the initial experiments and 2. Training a network from scratch, which overcomes the shortcoming of the first approach.

Initial evaluations, which were carried out using the deep-learning approach of transfer learning, are based on two image formats A and B as shown in Figure 2 and 3, which represents the front view and top view, respectively. Transfer Learning was performed separately with these image formats, to compare the performance of their classification. Later, we introduce a multi-view deep-learning network (LIDAR-Net), which takes image formats A and B as input to concatenate its features.

For the data acquisition, we performed various experiments to apply and record different contaminations over the LIDAR sensor front plate. The following contaminations were applied for the experiments: Salt, Dirt Type 1, Dirt Type 2 and Frost. All experiments were carried out manually. The recorded data transformed to image formats A and B were labeled with their respective contamination name type.

[1]Different contamination types used for the experiments include Salt, Dirt type 1 which is Arizona-Staub Quartz, Dirt type 2 which is KSL 11046 Prüfstaub and Frost.

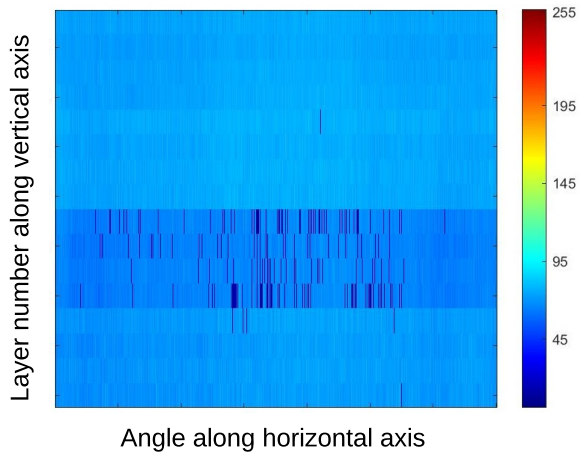


Figure 2: Image format A, where each pixel pair represents horizontal angle, which is in the range ±72.5 degree and layer number, ranging from 1 to 16.

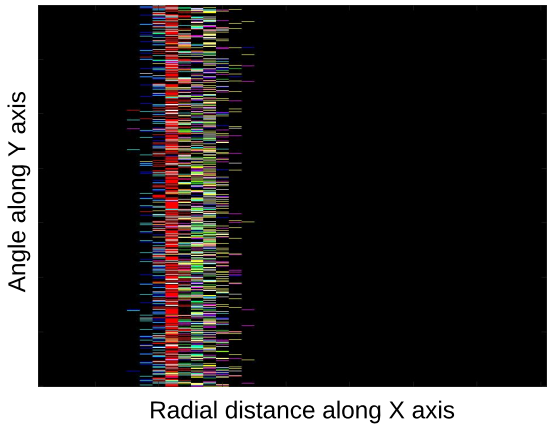


Figure 3: Image format B, where each pixel pair represents horizontal angle, which is in the range ±72.5 degree and radial distance(cm). The color indicates the layer number ranging from 1 to 16.

3.1 Dataset Acquisition

The dataset is acquired through a dataset creation pipeline as shown in Figure 7. We only considered the near field, which does not change under dynamic conditions. Hence no objects other than the contaminations appear at the near field of the LIDAR sensor. The recorded raw data is transformed into two image formats A and B, which represents the front and top view, respectively. Each image is then labeled with its contamination type. For all evaluations the dataset was separated into training and testing data in the ratio 4:1.

Datasets from same and different traces were made as part of the experiments. Same trace are made by reusing the contamination over the LIDAR sensor front plate (without re-applying the contamination). Different trace are made by re-applying the contamination over the LIDAR sensor front plate, after cleaning the sensor. During the dataset acquisition, the following recording parameters were considered: 1. Type of the contamination applied over the LIDAR sensor front plate, which was later used for labeling the data, 2. Number of layers for the contamination, 3. Weather conditions, 4. Light conditions and 5. Temperature.

The LIDAR sensor used for recording was developed by Valeo, as shown in Figure 4. It has a vertical field of view of ±5 degrees and horizontal field of view of ±72.5 degrees. Its resolution is 16 vertically and 701 horizontally. The LIDAR sensor used for this experiment has a frame rate of 25 frames per second. E.g., with a recording length of 10 seconds, 250 images could be generated for each image format.

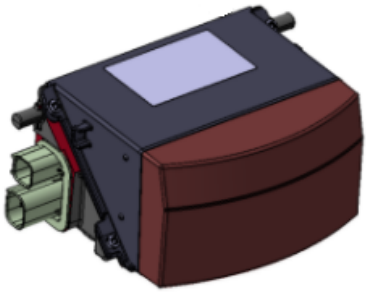


Figure 4: LIDAR sensor developed by Valeo.

The experimental settings were the following: 1. Experiments were conducted outside to have natural light conditions, 2. Some experiments were conducted during different times of the day and under different weather conditions, e.g. during sunny, cloudy or rainy weather. Therefore, the evaluated database includes recordings of different light conditions, 3. The variation of light conditions has not been studied systematically, neither have all conditions been analyzed (e.g. no experiments during night time) nor did the light conditions play any role in the separation of the database into training and testing data, 4. No ground, objects or any kind of obstacles were present in the near field.

For the experiments, the contaminations were mixed with water in the ratio of 1:1. Using a spray gun connected to a compressor, the front plate of the LIDAR sensor was contaminated with up to five layers of dirt or salt. Once the contamination over the LIDAR sensor front plate was dry, the recording was started.

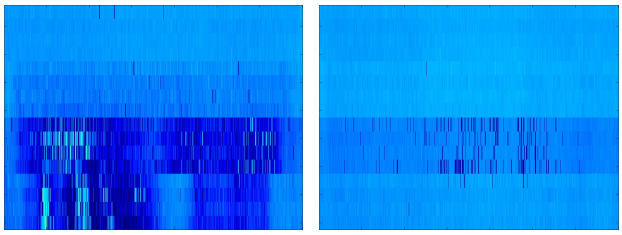
3.2 Raw Data Transformation

The raw data acquired is transformed and mapped into usable image formats with the Matlab [18] scripts. We transform the raw data into the image formats A and B using the data creation pipeline mentioned in the Figure 7. The image formats after transformation is visualized and compared in Figures 5,6.

3.2.1 Image Format A (Front View). The image format A is a 701x16 image. Each pixel pair represents horizontal angle and the layer number (Layers 1-16). The value of each pixel indicates the Echo

Pulse Width (EPW), which is an indicator of the reflectance of an object. The EPW values are normalized and mapped to a jet color map. Here only the first echo[2] of every scan point is considered, as the nearest scan points are relevant for detecting the contamination. The image has been enlarged in the Figures 2, 3, 5, 6 for visualization.

Figure 5 shows the comparison between a clean sensor and a sensor contaminated with dirt type 1, with image format A.

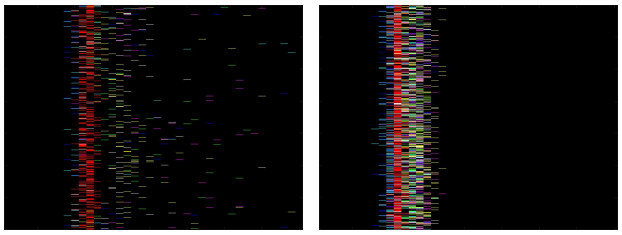


(a) Clean Sensor (b) Sensor with 5 layers of Dirt Type 1

Figure 5: Image format A compared for a clean sensor and a sensor contaminated with dirt type 1.

3.2.2 Image Format B (Top View). The image format B is a 710x40 image. Each pixel pair represents horizontal angle and radial distance. The values of each pixel indicate the layer number represented by RGB color code, exclusively assigned to each layer (Layers 1-16). Where ever there is no scan point, the pixel is represented by the color black. Here we have considered radial distance up to 160 centimeters from the LIDAR sensor front plate. The image has been enlarged in the Figures 2, 3, 5, 6 for visualization.

Figure 6 shows the comparison between a clean sensor and a sensor contaminated with Dirt type 1, with image format B.



(a) Clean Sensor (b) Sensor with 5 layers of Dirt Type 1

Figure 6: Image format B compared for a clean sensor and a sensor contaminated with dirt type 1.

[2]Every laser shot receives 3 echoes, each echo represents a scan point in the order of distance.

3.3 Deep-learning Architectures

The experiments for the problem of classification was carried out by two approaches: 1. Transfer Learning [30], 2. Training a Deep Neural Network from scratch.

3.3.1 Transfer Learning. Transfer Learning [30] was performed using the per-trained network InceptionV4 [28] . The experiments were carried out on a smaller dataset and the network was trained only over the last layer. Separate experiments were performed with the front view (image format A) and top view (image format B). While performing the experiments test data from same[3] and different trace[4] were used. The results for the classification declined over the number of labels and the contamination types.

3.3.2 Training a Deep Neural Network from scratch. We propose a novel architecture LIDAR-Net which takes two views of the LIDAR sensor, the front view (image format A) and top view (image format B) which are feed forwarded through the network simultaneously, with the same label name. The network was trained with different combinations of class labels with 2400 images per class per view.

The LIDAR-Net as shown in Figure 8, is a multi-view architecture [2] and has a structure with 3 convolution [14] layers, each having a filter size of (2x2). Each convolution layer is followed by a max pool [8] layer of filter size (2x2). This structure is duplicated to create the multi-view, i.e the feature maps from the 3rd Pooling layer are concatenated which is further connected to two fully connected layers. Each fully connected layer has a 1-Dimensional length of 2048. From the experiments performed by tweaking the architecture, we conclude that two fully connected layers with a dimension of 2048 was converging to a stable accuracy. The last fully connected layer is connected to a soft-max classifier [6].

Initial experiments were also made with the base architecture as VGG16 [23]. It was observed that a simple architecture with less number of layers gave similar results with better run time for training.

3.4 Training the Deep Neural Network

The labels used to train the network were 1. Clean, 2. Salt, 3. Dirt type 1, 4. Dirt type 2, and 5. Frost. A total of 2400 frames were collected for each label, which represents the contamination types. Each frame was used to generate two view, the front view (image format A) and top view (image format B). Hence a total of 4800 images per label were used to train the multi-view architecture. The test data was recorded separately with newly simulated contaminations, hence we have two different traces for training and testing. A portion of the test data was used as the validation data for fine tuning the hyper-parameters [5]. Finally, the weights of the network which gave the highest classification accuracy were saved for performing the test accuracy. The frameworks used for the experiments were Keras [3] and Tensorflow [1]. The Transfer learning experiments were carried out using Keras, where as the experiment for multi-view architecture was done with Tensorflow. The hardware equipments for training the Deep Neural Network

[3]Same trace are made by reusing the contamination over the LIDAR sensor front plate(without re-applying the contamination).

[4]Different trace are made by re-applying the contamination over the LIDAR sensor front plate, after cleaning the sensor.

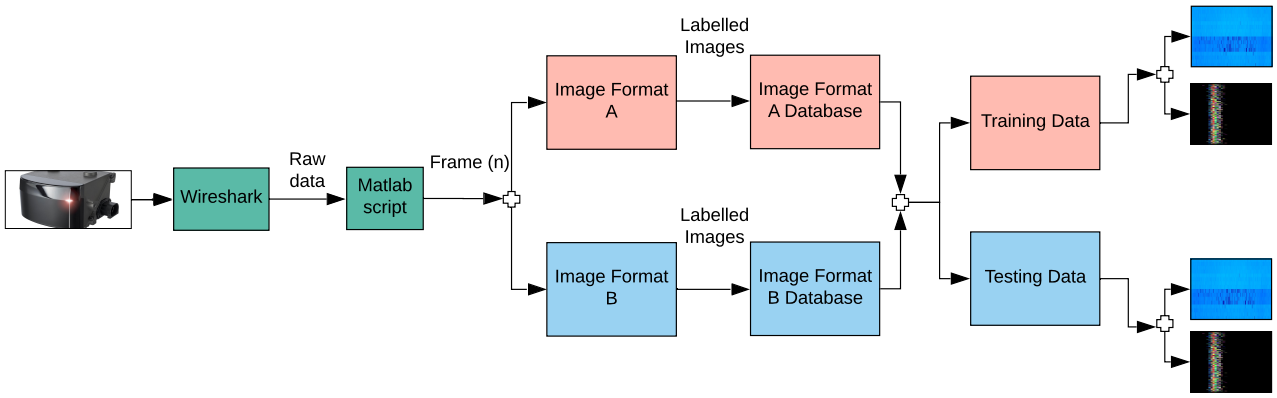


Figure 7: The dataset creation pipeline extracts the raw data from the LIDAR sensors via the open sourced package analyzer Wireshark [20]. Data transformations are done on the raw data to generate image formats A and B (with the Matlab scripts), to be stored in a database. Generated images are separated into training and testing data.

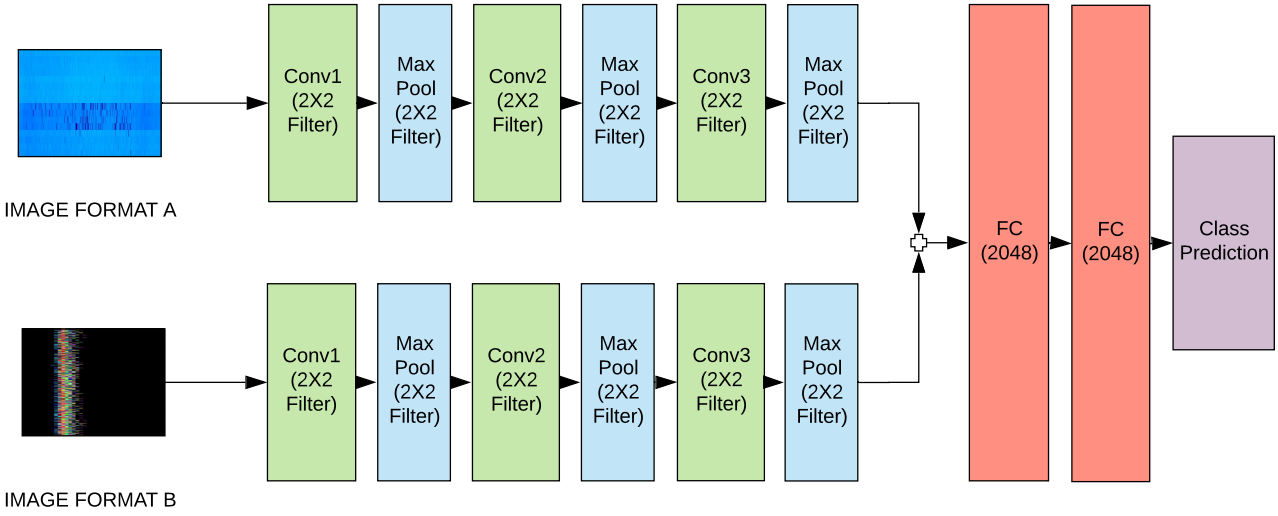


Figure 8: LIDAR-Net: A multi-view deep neural network for classifying different types of contaminations over the LIDAR front plate.

were the following: Three Nvidia Tesla X [16] GPU, and a training server.

3.5 Hyper Parameters

The following were the hyper parameters [5] for training the multi-view deep neural network:

3.5.1 Number of Hidden Layers and Units. We used 3 convolution layers along with 3 max pool layers for each view. The first convolution layer has an input of dimension 3, filter size of (2x2) and output feature map [24] size of 32. The second convolution layer has an input of dimension 32, filter size of (2x2) and output feature map size of 64. The third convolution layer has an input of dimension 64, filter size of (2x2) and output feature map size of 128. All max pool layers between the convolution layers has a filter size of (2X2). The first fully connected layer has an input dimension of (28x14x128) and an output dimension of 2048. The second fully connected layer has an input dimension of 2048 and an output dimension of 2048. The soft-max classifier has an input dimension of 2048. The final class size has been varied from 3 to 5 for various experiments.

3.5.2 Dropout. The dropout [26] is used to prevent overfitting of the data. We use SELU with the dropout probability of .05%.

3.5.3 Activation Function. The activation function used here is SELU or Scaled Exponential Linear Units [13]. The SELU activation

functions is preferred over the RELU because of its converging property.

$$selu(x) = \lambda \begin{cases} x, & if x > 0 \\ \alpha e^x - \alpha, & if x \leq 0 \end{cases}$$

Where the values α and λ are 1.673 and 1.050 respectively for mean 0 and standard deviation 1.

3.5.4 Loss Function. The loss function used here is Soft-max cross entropy [19]. The cross entropy is given by the equation:

$$H(y, p) = -\sum_i y_i log(p_i)$$

3.5.5 Optimizer. The Optimizer used here is the Adam optimizer [12]. The following are the configuration parameters for an ideal learning problem, alpha=0.001, beta1=0.9, beta2=0.999 and epsilon=10E-8.

3.5.6 Learning Rate. Here, we have used a decaying learning rate [25]. Table 1 shows the parameters for the decaying learning rate:

Table 1: Decaying Learning Rate

Parameters	Value
Initial learning rate	0.0001
Learning rate decay factor	0.7
Number of epochs before decay	2

3.5.7 Batch Size and Number of Epochs. The batch size is 128 and the network was trained for 20 epochs.

4 EVALUATION AND COMPARISON

In this section we evaluate and compare the proposed approach with the classical machine learning algorithms.

4.1 K-Means Clustering Algorithm

K-Means clustering algorithm [10] separates n data points into k clusters. The data points are assigned to the cluster with the closest mean value. The k-means algorithm is very similar to the k-nearest neighbor classifier [10].

Table 2: K-Means Clustering

EPW Centroids	Cluster size after 100 iterations
278.18	990
211.44	494
245.06	491
326.41	38
252.24	487

Here we cluster the mean EPW values of a complete scan, with 500 scan for each class. Mean EPW values from five different classes were used for the clustering. Over 100 iterations were made for a cluster size of 5. The results proved that the mean EPW values were not clustered equally around all the centroids, as shown in Table 2. The Figure 9 shows the clustering diagram of the EPW values from five different classes, where they are plotted along the x axis with value of y as 0. The Figure 9 has no axis along y and the plot along x axis has been shifted upwards for visibility. The experiment was also conducted with four clusters, four classes and provided similar clustering results.

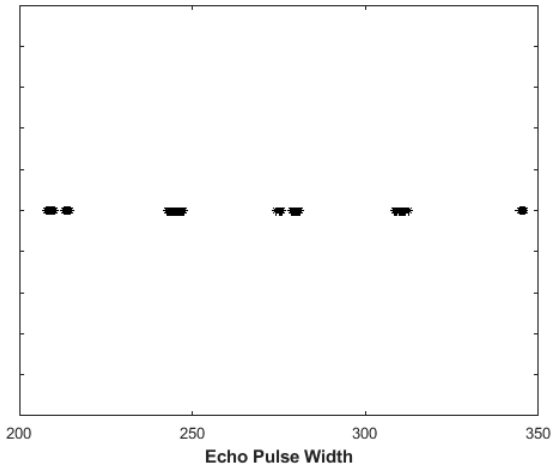


Figure 9: K means clustering of the echo pulse width values from five different classes.

4.2 K-Nearest Neighbors Algorithm

K-Nearest neighbors algorithm [4] is a classification algorithm. The input parameter k is the number of data points to be considered for deciding the class membership. The class membership is decided based on the score from the k nearest neighboring data points. If the value of k = 1, then the class membership is the same as that of the data point [4].

Table 3: K-NN Classifier

Class Label	Classification Accuracy
Clean, Dirt 1, Salt	66.66%
Clean, Salt, Frost	66.66%
Clean, Dirt 1, Dirt 2	47.61%
Clean, Dirt 1, Frost	66.66%
Clean, Dirt 1, Salt, Frost	50%
Clean, Dirt 1, Dirt 2, Salt, Frost	40%

We trained a K-NN classifier with k=4 and distance metric as euclidean distance. Here, the training and testing data are from different traces[5]. The features used for training and testing are the mean EPW[6] values of a complete scan. The labels used for the experiment were Clean, Dirt 1, Dirt 2, Salt and Frost. By plotting the

[5]Different trace are made by re-applying the contamination over the LIDAR sensor front plate, after cleaning the sensor.
[6]Mean EPW is the average of the EPW values of each scan point in a scan.

confusion matrix we infer that the K-NN Classifier gave a classification accuracy of 40%, also shown in Table 3. The confusion matrix from Figure 10 shows that the label Dirt 1 was misclassified with Clean, the label Salt was misclassified as Frost and the label Dirt 2 was misclassified as Dirt 1. The experiment was also conducted with labels Clean, Dirt 1, Salt and Frost, the classification accuracy was obtained as 50%.

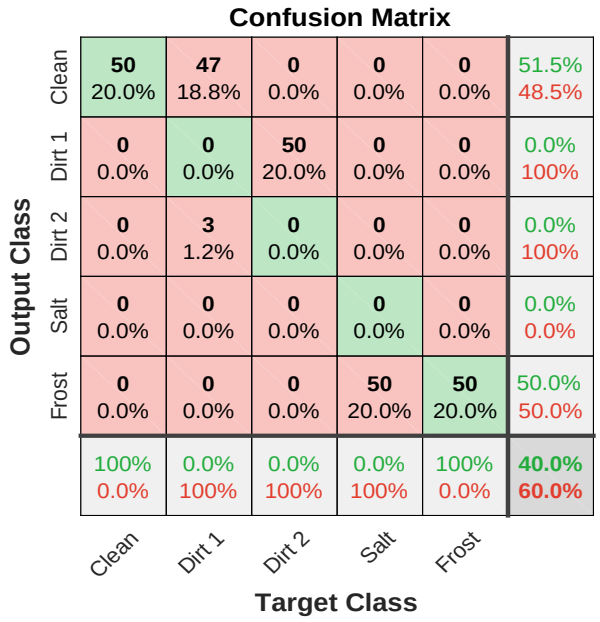


Figure 10: Confusion matrix for the labels a. Clean, b. Dirt type 1, c. Dirt type 2, d. Salt, e. Frost.

4.3 Deep Neural Network Approach

We started our experiment with the deep-learning approach of transfer learning. Here we used the Inception V4 architecture which was pre-trained on IMAGENET [22] dataset. We extract the features from a layer and add an additional fully connected layer before classification, so only the last layer is trained with the dataset from our domain of experiment. Experiments were performed separately for image formats A and B.

The Tables 4 and 5 summarize classification with transfer learning. The classification accuracy was high when we trained and tested the network with the same trace[7], but it declined drastically when a different trace[8] was used for testing. Hence, we combine the image formats A and B in the multi-view architecture.

[7]Same trace are made by reusing the contamination over the LIDAR sensor front plate (without re-applying the contamination).
[8]Different trace are made by re-applying the contamination over the LIDAR sensor front plate, after cleaning the sensor.

Table 4: Transfer Learning with Front View

Class Label	Classification Accuracy	Comments
Clean, Dirt 1	100%	Same trace
Clean, Dirt 1, Salt	90.44%	Same trace
Clean, Dirt 1	80%	Different trace

Table 5: Transfer Learning with Top View

Class Label	Classification Accuracy	Comments
Clean, Dirt 1	97.83%	Same trace
Clean, Dirt 1, Salt	94.29%	Same trace
Clean, Dirt 1	78.82%	Different trace

From the Table 6, we can say that the multi-view deep-leaning approach outperforms all other approaches in terms of classification accuracy. Here we have used a different trace 8^0 for testing all the cases, and we see that a higher accuracy is obtained in 4 out of 6 cases.

Compared to the evaluations made with the classical approaches, we obtained a 35% higher classification accuracy. Figure 10 shows the confusion matrix for the KNN-classifier which depicts the true and false positives for each label. We infer that only Clean and Frost are been classified correctly. Where as from the Table 6, we see that for the multi-view deep-learning approach, we get a classification accuracy of more that 95% while classifying Clean, Dirt 1, Salt and Frost. It was also observed that the classification accuracy declined while classifying two different types of dirt.

From the results, we also conclude that classifying the labels Dirt (Dirt 1 or Dirt 2), Salt, Frost and Clean resulted in a better accuracy compared to classifying different types of dirt.

Table 6: Multi View Architecture

Class Label	Classification Accuracy
Clean, Dirt 1, Salt	99.34%
Clean, Salt, Frost	95.41%
Clean, Dirt 1, Dirt 2	65.23%
Clean, Dirt 1, Frost	99.14%
Clean, Dirt 1, Salt, Frost	99.21%
Clean, Dirt 1, Dirt 2, Salt, Frost	77.98%

5 CONCLUSION

We have introduced a novel deep neural network approach for classifying various contaminations over the LIDAR sensor front plate. Classifying the contaminations are essential for the efficient heating and cleaning mechanism of the LIDAR sensor and hence its performance. We conclude that our approach with the deep neural network with a multi-view architecture proved to outperform classical approaches for the problem of contamination type classification in the performed experimental setting.

The data recorded with the recording setup originated from the same LIDAR sensor. We could bring more complexity to the dataset by making the data recording with multiple LIDAR sensors, which can be considered for the future works. Similarly, all the data recording experiments were conducted by applying full contamination over the LIDAR sensor, the application of partial contamination can be considered for future works. We can also consider the systematic variation of light or temperature conditions as a factor for data recording in the future works. In a real-world scenario where there are a mixture of contaminations (mixture of dust,salt or ice), the performance of the DNN approach has to be evaluated based on the classification accuracy, which also has a great potential for future work.

ACKNOWLEDGMENTS

The first author would like to thank Dr. Georg Puhlfürst and Vladislav Golyanik for their supervision. The author would also like to thank Prof. Dr. Didier Stricker for providing an opportunity to work with DFKI.

The first author would also like to thank anonymous referees for their valuable comments and helpful suggestions. The work is a part of the master's thesis supported by Valeo Schalter und Sensoren GmbH.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, Berkeley, CA, USA, 265–283. http://dl.acm.org/citation.cfm?id=3026877.3026899

[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2016. Multi-View 3D Object Detection Network for Autonomous Driving. *CoRR* abs/1611.07759 (2016). arXiv:1611.07759 http://arxiv.org/abs/1611.07759

[3] François Chollet et al. 2015. Keras. https://keras.io.

[4] T. Cover and P. Hart. 2006. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theor.* 13, 1 (Sept. 2006), 21–27. https://doi.org/10.1109/TIT.1967.1053964

[5] Gonzalo I. Diaz, Achille Fokoue, Giacomo Nannicini, and Horst Samulowitz. 2017. An effective algorithm for hyperparameter optimization of neural networks. *CoRR* abs/1705.08520 (2017). arXiv:1705.08520 http://arxiv.org/abs/1705.08520

[6] Kaibo Duan, S. Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. 2003. Multi-category Classification by Soft-max Combination of Binary Classifiers. In *Proceedings of the 4th International Conference on Multiple Classifier Systems (MCS'03)*. Springer-Verlag, Berlin, Heidelberg, 125–134. http://dl.acm.org/citation.cfm?id=1764295.1764312

[7] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh. 2011. Radar/Lidar sensor fusion for car-following on highways. In *The 5th International Conference on Automation, Robotics and Applications*. 407–412. https://doi.org/10.1109/ICARA.2011.6144918

[8] Benjamin Graham. 2014. Fractional Max-Pooling. *CoRR* abs/1412.6071 (2014). arXiv:1412.6071 http://arxiv.org/abs/1412.6071

[9] T. Herpel, C. Lauer, R. German, and J. Salzberger. 2008. Multi-sensor data fusion in automotive applications. In *2008 3rd International Conference on Sensing Technology*. 206–211. https://doi.org/10.1109/ICSENST.2008.4757100

[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. 2002. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 7 (Jul 2002), 881–892. https://doi.org/10.1109/TPAMI.2002.1017616

[11] D. U. Kim, S. H. Park, J. H. Ban, T. M. Lee, and Y. Do. 2016. Vision-based autonomous detection of lane and pedestrians. In *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*. 680–683. https://doi.org/10.1109/SIPROCESS.2016.7888349

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[13] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. *CoRR* abs/1706.02515 (2017). arXiv:1706.02515 http://arxiv.org/abs/1706.02515

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12)*. Curran Associates Inc., USA, 1097–1105. http://dl.acm.org/citation.cfm?id=2999134.2999257

[15] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. FractalNet: Ultra-Deep Neural Networks without Residuals. *CoRR* abs/1605.07648 (2016). arXiv:1605.07648 http://arxiv.org/abs/1605.07648

[16] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. 2008. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro* 28, 2 (March 2008), 39–55. https://doi.org/10.1109/MM.2008.31

[17] G. Lu and M. Tomizuka. 2006. LIDAR Sensing for Vehicle Lateral Guidance: Algorithm and Experimental Study. *IEEE/ASME Transactions on Mechatronics* 11, 6 (Dec 2006), 653–660. https://doi.org/10.1109/TMECH.2006.886192

[18] MATLAB. 2018. *version 9.4.0 (R2018a)*. The MathWorks Inc., Natick, Massachusetts.

[19] G. E. Nasr, E. A. Badr, and C. Joun. 2002. Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 381–384. http://dl.acm.org/citation.cfm?id=646815.708603

[20] Angela Orebaugh, Gilbert Ramirez, Jay Beale, and Joshua Wright. 2007. *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress Publishing.

[21] J. R. V. Rivero, I. Tahiraj, O. Schubert, C. Glassl, B. Buschardt, M. Berk, and J. Chen. 2017. Characterization and simulation of the effect of road dirt on the performance of a laser scanner. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. https://doi.org/10.1109/ITSC.2017.8317784

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. ImageNet Large Scale Visual Recognition Challenge. *CoRR* abs/1409.0575 (2014). arXiv:1409.0575 http://arxiv.org/abs/1409.0575

[23] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). arXiv:1409.1556 http://arxiv.org/abs/1409.1556

[24] A. Sluzek. 2003. Feature maps: a new approach in hierarchical interpretation of images. In *Proceedings. 2003 International Conference on Cyberworlds*. 245–250. https://doi.org/10.1109/CYBER.2003.1253461

[25] Leslie N. Smith. 2015. No More Pesky Learning Rate Guessing Games. *CoRR* abs/1506.01186 (2015). arXiv:1506.01186 http://arxiv.org/abs/1506.01186

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[27] M. Stampfle, D. Holz, and J. C. Becker. 2005. Performance evaluation of automotive sensor data fusion. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* 50–55. https://doi.org/10.1109/ITSC.2005.1520114

[28] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR* abs/1602.07261 (2016). arXiv:1602.07261 http://arxiv.org/abs/1602.07261

[29] Jingdong Wang, Zhen Wei, Ting Zhang, and Wenjun Zeng. 2016. Deeply-Fused Nets. *CoRR* abs/1605.07716 (2016). arXiv:1605.07716 http://arxiv.org/abs/1605.07716

[30] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *CoRR* abs/1411.1792 (2014). arXiv:1411.1792 http://arxiv.org/abs/1411.1792