# Comparing of Some Convolutional Neural Network (CNN) Architectures for Lane Detection

O. T. EKŞİ and G. GÖKMEN

***Abstract*— Advanced driver assistance functions help us prevent the human-based accidents and reduce the damage and costs. One of the most important functions is the lane keeping assist which keeps the car safely in its lane by preventing careless lane changes. Therefore, many researches focused on the lane detection using an onboard camera on the car as a cost-effective sensor solution and used conventional computer vision techniques. Even though these techniques provided successful outputs regarding lane detection, they were time-consuming and required hand-crafted stuff in scenario-based parameter tuning.**

**Deep learning-based techniques have been used in lane detection in the last decade. More successful results were obtained with fewer parameter tuning and hand-crafted things. The most popular deep learning method for lane detection is convolutional neural networks (CNN). In this study, some reputed CNN architectures were used as a basis for developing a deep neural network. This network outputs were the lane line coefficients to fit a second order polynomial. In the experiments, the developed network was investigated by comparing the performance of the CNN architectures. The results showed that the deeper architectures with bigger batch size are stronger than the shallow ones.**

*Index Terms*—**Deep learning, convolutional neural network, lane detection**

## I. INTRODUCTION

LANE DETECTION using computer vision techniques has been getting more attention for decades. Conventional computer vision techniques have addressed the lane detection system using three procedures as the following: 1) preprocessing, 2) feature extraction and model fitting, 3) lane tracking. These techniques generally are represented by two categories: feature-based, and model-based [1].

**O. T. EKŞİ,** Department of Mechatronics Engineering, Technology Faculty, University of Marmara, Istanbul, Turkey, (e-mail: osmantahireksi@gmail.com).

https://orcid.org/0000-0002-8994-0474

**G. GÖKMEN**, Department of Mechatronics Engineering, Technology Faculty, University of Marmara, Istanbul, Turkey, (e-mail: gokhang@marmara.edu.tr).

http://orcid.org/0000-0001-6054-5844

Manuscript received June 12, 2020; accepted Oct 29, 2020.
DOI: 10.17694/bajece.752177

Feature-based techniques depend on feature extraction regarding the lane such as edges and colors of the lane lines [1]. Kreucher and Lakshmanan [2] developed an algorithm called LANA (Lane-finding in Another domAin) which represented the features in the frequency domain and combined them with a deformable template to detect the lane markers. Collado et al. [3] created a bird-view of the camera images showing the road to eliminate the perspective view. They also utilized spatial lane features and Hough transform on this view to detect the lane adaptively. Borkar et al. [4] proposed a layered approach for lane detection in the night vision. They used a temporal blurred technique on the video frames to reduce the noise, and a local adaptive thresholding technique to take the binary images on which Hough transform implemented to find the lane markers. Lane detection process may fail, since the on-road vehicles can occlude the lane partially or completely. Satzoda and Trivedi [5] came up with ELVIS (Efficient Lane and Vehicle detection with Integrated Synergies) algorithm to handle this problem by detecting the on-road vehicles along with the lane. Mammeri et al. [6] used Maximally Stable Extremal Region technique with Hough transform for lane detection, and Kalman filter to track the detected lane. Jung et al. [7] introduced a spatiotemporal lane detection algorithm to take the advantage of some specific row pixels of the past images. In lane detection step, Hough transform was preferred.

Model-based techniques try to represent the lanes with some geometric models such as a linear model, a parabolic model, or most of the time spline models [1]. In [8], Catmull-Rom spline-based model was proposed to detect the lane boundary. This model, which has control points, enabled the detection system to represent arbitrary road shapes in a wider range compared to the previous models. Lim et al. [9] utilized Hough transform for lane detection within the near-field of the image. Since this model could not estimate the curvatures successfully at the far-field, a river flow method was proposed. To track the lane in the consecutive frames, a B-spline based Kalman filter system presented. Another two-field study was conducted by Tan et al. [10], and for the near-field, Hough transform was implemented. A hyperbolic model and RANSAC algorithm were used for creating the curved lines in the far-field. The coefficients for these lines were found by an improved river flow method. In [11], a method was proposed called Lane Detection with Two-stage Feature Extraction (LDTFE) against the challenges in lane detection due to lighting conditions and background clutters in the images. A modified Hough transform was applied

to extract small line segments of the lane, and then these segments are divided into two different classes by Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm in this method. Lastly, the lane identified by curve fitting.

Deep learning using convolutional neural network (CNN) for lane detection has been on the rise for the last decade as an alternative to the conventional computer vision techniques. Various CNN architectures have been developed, as the computer hardware improved.

In this paper, we proposed a CNN-based deep neural network model to detect the lane by representing the lines with second order polynomial's coefficients. We used some reputed CNN architectures in this model to compare their performance on lane detection. Fully connected neural network layers were built on the CNN part in a parallel way to find the coefficients.

This paper is organized as follows: In section two, a literature review about CNN-based lane detection works was presented. Section three describes CNN with its components and explain the designed architecture for the study. In section four, the experimental steps are explained using TuSimple dataset. Finally, the experimental results are compared and discussed at the conclusion section.

## II. Related Works

Gurghian et al. [12] put a down-facing camera on each side of a car for lane detection. This detection was an end-to-end process depending on a CNN model without any preprocessing or postprocessing. The model was trained using the collected real data and synthesized data. In the test, the model could estimate the lane with sub-centimeter accuracy. He et al. [13] proposed a Dual-View Convolutional Neutral Network (DVCNN) for lane detection. The images taken from the front-view camera of the car was transformed to bird-view using IPM. Then those images were sent to a CNN-based model with the original ones. The inputs of the model proceeded in a parallel way inside, and at the end, they merged with each other as an output. Finally, by a global optimization technique where the lane line probabilities, lengths, widths, orientations and the amount are considered, true lane lines could be detected. In [14], VPGNet that could detect the lane under harsh weather and lighting conditions was introduced. This architecture enabled the authors to identify and classify the lane and road signs and estimate the vanishing point. Pizzati et al. [15] modeled an end-to-end CNN architecture which was in a cascaded form. They trained it by 14336 lane boundaries from TuSimple dataset. In this way, the lane boundaries in the images were detected, clustered, and classified.

## III. Method

In this section, we described convolutional neural network (CNN). We also gave a deep neural network in which the most reputed CNN architectures were utilized for lane detection.

### A. Convolutional Neural Network

Convolutional neural network (CNN) is a subset of deep neural networks and specialized to process grid-like topological data. For instance, taken at regular intervals time series data can be thought of as a 1D grid, on the other hand, image data consisting of pixels can be thought of as a 2D grid. The name of this network comes from a linear mathematical operation called convolution which is implemented on the input data through the convolutional layers of the network [16].

Basically, CNN models include an input layer, an output layer and hidden layers as in deep neural networks. The layers can be in different forms as well except for the convolutional one to fulfill the objective of the application or to improve the performance of the network. Pooling layer, dropout layer, batch normalization layer, flatten layer, and fully connected layer are examples for these layers.

The pooling layers usually are placed after the convolutional layers to condense feature maps which are created during the convolution operations. Even though there are different kinds of pooling operations, the most common one is max-pooling. In max-pooling, a predefined pooling unit moves along the feature map and outputs the maximum activation in its region [17].

Dropout is a popular regularization technique to prevent the network from memorization of the data called overfitting. The dropout layer sets some feature outputs to zero during the training process with a given dropout rate [18, 19].

In CNN models, each layer input's distribution changes depending on the previous layer outputs. This situation causes a slow training step, since it requires a small learning rate for the optimizer and the parameter initialization carefully. At this point, the batch normalization layer enables the input of a layer to normalize the data before processing it. Thus, the requirements are dealt with substantially by provided generalization [20].

The flatten layer is used for transforming the feature maps, which are in a multidimensional matrix form, to one dimensional vector for the fully connected layers [21].

At the end of a CNN model, generally, the fully connected layers (dense layers) are used for classification of the input. Each neuron at each fully connected layer was linked to the neurons in the prior and following layers [21].

Another component of CNN is activation functions. These functions add nonlinearity to the model to represent the complex problems and catch the nontrivial relationships in the data. Many activation functions could be used in accordance with the target of the model like sigmoid, tanh, relu, elu, etc [19].

Evaluation of the difference between the target value and the estimated value from a CNN model is shown by a loss function, predefined by the designer. The model tries to minimize this function during the training process.

### B. Network Architecture

In the following, we describe our deep neural network architecture for lane detection as depicted in Figure 1. The basis of this architecture is composed of one of reputed CNN architectures which are Lenet5, AlexNet, VGG16, VGG19, ResNet50. After the CNN architecture, the fully connected

layers are available to estimate the line coefficients of the lane to fit second order polynomials.

In 1998, Yann Lecun et al. [22] developed LeNet5 to classify the handwritten digits. It took 32x32 input images and included two convolutional layers. These layers were individually followed by the average pooling layer, and finally there exist fully connected layers. tanh performed as the activation function.

AlexNet, which has five convolutional layers, max-pooling layers, and batch normalization layers, was trained on GPU in a parallel way using 224x224 input images. This architecture won first place by classifying different objects into 1000 categories with the highest accuracy in the 2012 ImageNet Large Scale Visual Recognition Competition (ILSVRC). relu activation function was used here for the first time [23].

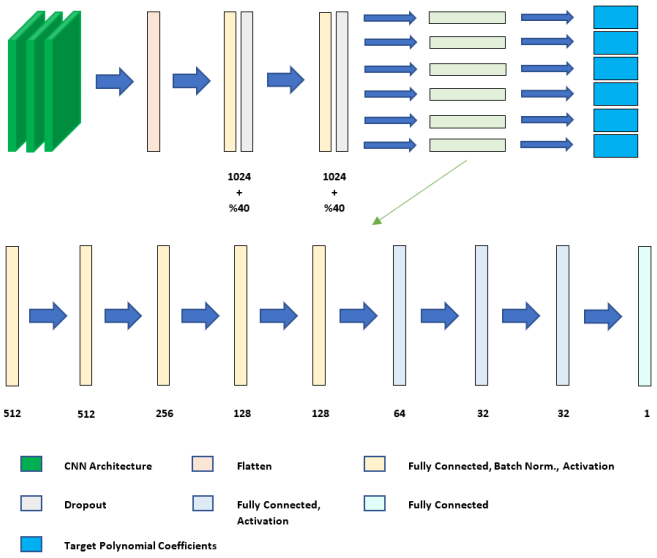


Figure 1. The developed architecture for lane detection

Vision Geometry Group from the University of Oxford invented VGG Net in 2014. Because this CNN architecture has small filters in the convolutional layers compared to previous ones, the network was able to be deeper to extract features from the complex input images without increasing the parameters extremely. The most common VGG Net architectures are VGG16 and VGG19 which are named regarding the total number of the convolutional and fully connected layers [24].

ResNet architecture came up with residual blocks to solve the vanishing gradient problem in the optimization and to get very deep CNN models with fewer parameters, in 2015 [25]. In this study, ResNet architecture with 50 layers is preferred to use.

The designed network architecture had fully connected layers after the CNN part. The first two fully connected layers were followed by the dropout layer with a 40% dropout rate. Then, six parallel fully connected branches process the data to estimate the lane coefficients for each line to fit a second order polynomial. As the activation function relu and elu were chosen for the CNN side and the fully connected side, respectively.

## IV. EXPERIMENTS

In this section, we examined the success of the CNN architectures with the developed fully connected architecture in terms of lane detection.

### A. *Dataset*

To train and test our approach, we did experiments on TuSimple dataset. This dataset was collected from highways in daylight. Figure 2 shows some images from the dataset. The weather was usually sunny and sometimes cloudy. It consisted of 3626 images for the training and 2782 images for the testing [26].

In the experiments, we divided the training set into two parts to make a validation after each training epoch. This validation set was 10% of the training set. The labels of the dataset showed the point coordinates of the lane lines with respect to the image origin.

Figure 2. Some images from TuSimple dataset

### B. *Preprocess*

The original images in the dataset had a resolution of 1280x720. We reduced the size of the images to 160x160 since the original resolution was too high to process in the network regarding the computational cost. The lane line points were scaled with the same ratio of the resolution change due to the change in the size of the images. The points were used for getting the target coefficients for a second order polynomial.

The dataset had images from highway. Thus, multilane lines ranging from two to five existed. In the scope of this study, only the ego lane was examined and so filtered out from the other lanes. Figure 3 demonstrated the ego lane lines with red and blue polylines.

Figure 3. Acquired ego lane lines for TuSimple dataset

Most of the images were taken in clear conditions such as in daylight, without shadows or extreme illumination changes. This situation was not good for the generalization of the training process, since it would be biased to a clear view of the camera. Hence, we implemented data augmentation for the input images by changing their brightness in some limits before they go into the network as illustrated in Figure 4.

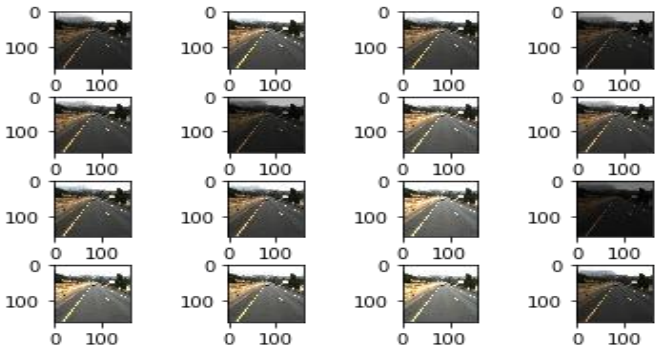

Figure 4. The augmented input data for the developed architecture

### C. *Training the Network*

The designed deep neural network was composed of two parts which were the CNN architecture and the parallel fully connected architectures. The abovementioned CNN architectures were put into the first part for each training process. They were compared in terms of performance and accuracy. Estimation of the designed network was compared with the target coefficients with mean squared error loss function using RMSprop optimizer. The loss function represented the average of all the coefficients error. The learning rate of the optimizer was initialized with 0.001 and decreased after each epoch by 5%. The total number for the epoch was set at 100. If the no changes were observed by a callback function that follows the changes in the loss function during the predefined period, training could be discontinued without waiting for the end of the epochs.

Besides, the effect of batch size was also investigated in this study choosing them of 8, 16 and 32. The loss function changing were depicted in Figure 5 regarding the mentioned batch sizes, respectively.

### D. *Evaluation Metric*

The result of the data testing after the training phase was evaluated by intersection over union (IoU) to measure the similarity between the targets and the test outputs [27]. In this metric, the lane area between the lines, which were represented by second order polynomial's coefficients by the designed architecture, was segmented with white pixels. The region outside the lane was segmented with black pixels as illustrated in Figure 6.

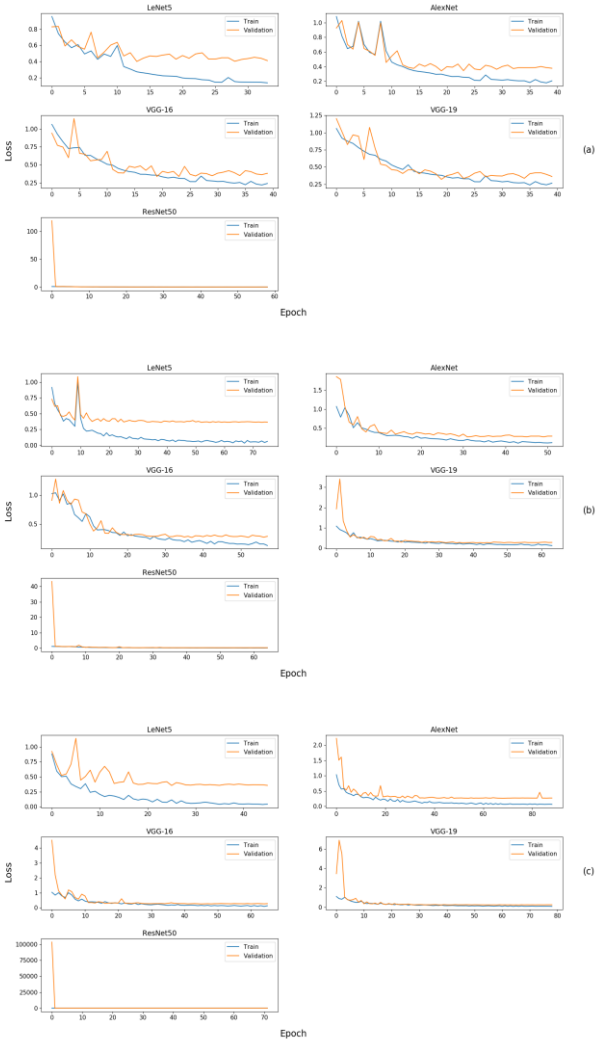


Figure 5. Loss function change depending on batch sizes of (a) 8, (b) 16, (c) 32, respectively

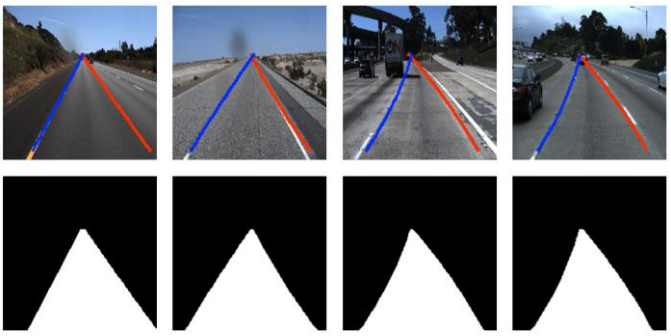

Figure 6. Segmented lanes which colored with white pixels

## V. EXPERIMENTAL RESULTS

The designed model based on reputed CNN architectures was tested on 2782 test images of TuSimple dataset. The performance of this model was calculated by taking the average of the evaluation metric for all the test images. In addition to this, the effect of batch size was considered for the CNN architectures individually. The results were compared with each

other in Table 1. The most successful experiment results, which belong to ResNet50 architecture base by a batch size of 32, were depicted in Figure 7 as the detected lane area using green color.

TABLE I
COMPARATIVE RESULTS of CNN ARCHITECTURES

| CNN BASE | Batch Size | Epoch Number | IoU Score |
|---|---|---|---|
| Lenet5 | 8 | 34 | **0.86518** |
| AlexNet | 8 | 40 | 0.84630 |
| VGG16 | 8 | 40 | 0.85010 |
| VGG19 | 8 | 40 | 0.85531 |
| ResNet50 | 8 | 59 | 0.85607 |
| Lenet5 | 16 | 76 | 0.85135 |
| AlexNet | 16 | 52 | 0.84399 |
| VGG16 | 16 | 58 | 0.87786 |
| VGG19 | 16 | 64 | **0.89115** |
| ResNet50 | 16 | 65 | 0.87930 |
| Lenet5 | 32 | 46 | 0.87737 |
| AlexNet | 32 | 89 | 0.82226 |
| VGG16 | 32 | 66 | 0.87183 |
| VGG19 | 32 | 79 | 0.87273 |
| ResNet50 | 32 | 72 | **0.89570** |

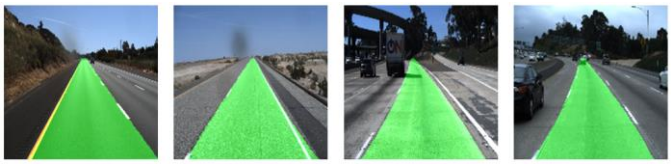

Figure 7. ResNet50 outputs for lane detection by the batch size of 32

## VI. CONCLUSIONS

In this paper, a CNN-based lane detection deep neural network model was proposed. In comparison with most deep learning-based lane detection methods, which output the lane area by segmentation, the developed model gives more crucial lane line coefficients for advanced driver assistance keeping the care safely in the lane. Our experiments showed that CNN architectures played an active role in lane detection regarding their complexity to extract more sophisticated image features. Even though LeNet5 is an outdated architecture, it gave good results for some cases. Deeper architectures such as VGGNet and ResNet could detect the lane more accurately. It was observed that the batch size influenced the detection accuracy, and about 90% IoU score was taken with ResNet50 by the batch size of 32. The batch size also affected the number of epochs, which was limited to 100, utilizing an early stop callback function in a direct proportion.

Future studies extending to multi-lane detection and adding on other CNN architectures are required to develop a real-time application on a car with a computer like NVIDIA TX2.

## REFERENCES


[1] Y. Xing, C. Lv, L. Chen, H. Wang, H. Wang, D. Cao, E. Velenis, F. Wang, "Advances in vision-based lane detection: Algorithms, integration, assessment, and perspectives on ACP-based parallel vision," IEEE/CAA Journal of Automatica Sinica, Vol. 5, No. 3, 2018, pp. 645-661.

[2] C. Kreucher, S. Lakshmanan, "LANA: A lane extraction algorithm that uses frequency domain features," IEEE Transactions on Robotics and Automation, Vol. 15, No. 2, 1999, pp. 343-350.

[3] J. M. Collado, C. Hilario, A. de la Escalera, J. M. Armingol, "Adaptative road lanes detection and classification," Advanced Concepts for Intelligent Vision Systems: 8th International Conference, Antwerp, Belgium, 2006.

[4] A. Borkar, M. Hayes, M. T. Smith, S. Pankanti, "A layered approach to robust lane detection at night," 2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, Nashville, US, 2009, pp. 51-57.

[5] R. K. Satzoda, M. M. Trivedi, "Efficient lane and vehicle detection with integrated synergies (ELVIS)," 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, US, 2014, pp. 708-713.

[6] A. Mammeri, A. Boukerche, Z. Tang, "A real-time lane marking localization, tracking and communication system," Computer Communications, Vol. 73, 2016, pp. 132-143.

[7] S. Jung, J. Youn, S. Sull., "Efficient Lane Detection Based on Spatiotemporal Images," in IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 1, 2016, pp. 289-295.

[8] Y. Wang, D. Shen, E. K. Teoh, "Lane detection using spline model," Pattern Recognition Letters, Vol. 21, No. 8, 2000, pp. 677-689.

[9] K. H. Lim, K. P. Seng, L. Ang, "River Flow Lane Detection and Kalman Filtering-Based B-Spline Lane Tracking," International Journal of Vehicular Technology, Vol. 2012, 2012, pp. 1-10.

[10] H. Tan, Y. Zhou, Y. Zhu, D. Yao, J. Wang, "Improved river flow and random sample consensus for curve lane detection," Advances in Mechanical Engineering, Vol. 7, No. 7, 2015.

[11] J. Niu, J. Lu, M. Xu, P. Lv, X. Zhao, "Robust lane detection using two-stage feature extraction with curve fitting," Pattern Recognition, Vol. 59, 2016, pp. 225-233.

[12] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, V. N. Murali, "DeepLanes: End-to-end lane position estimation using deep neural networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, Las Vegas, US, 2016, pp. 38-45.

[13] B. He, R. Ai, Y. Yan, X. Lang, "Accurate and robust lane detection based on Dual-View Convolutional Neutral Network," 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 2016, pp. 1041-1046.

[14] S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T. Lee, H. S. Hong, S. Han, I. S. Kweon, "VPGNet: Vanishing Point Guided Network for lane and road marking detection and recognition," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 1965-1973.

[15] F. Pizzati, M. Allodi, A. Barrera, F. Garcia, "Lane detection and classification using cascaded CNNs," Computer Aided Systems Theory – EUROCAST 2019, 2020, pp. 95-103.

[16] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, The MIT Press, 2017, p. 330.

[17] M. Nielson, Neural Networks and Deep Learning, Determination press, 2015, p. 174.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "{Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, Vol. 15, No. 1, 2014, pp. 1929-1958.

[19] F. Chollet, Deep Learning with Python, Manning, 2017, p.109.

[20] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," Proceedings of the 32nd International Conference on International Conference on Machine Learning – Volume 37, Lille, France, 2015, pp. 448-456.

[21] D. Foster, Generative Deep Learning, O'Reilly Media, 2019, p. 38.

[22] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, Vol. 86, No. 11, 1998, pp. 2278-2324.

[23] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Proceedings of the 25th

International Conference on Neural Information Processing Systems, Lake Tahoe, US, 2012, pp. 1097-1105.
[24] K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” International Conference on Learning Representations, 2015.
[25] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, US, 2016, pp. 770-778.
[26] "TuSimple/tusimple-benchmark", GitHub, 2020. [Online]. Available: https://github.com/TuSimple/tusimple-benchmark. Accessed: 17- May- 2020
[27] F. V. Beers, A. Lindström, E. Okafor, M. A. Wiering, “Deep neural networks with intersection over union loss for binary image segmentation,” Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods, 2019

## BIOGRAPHIES

**Osman Tahir EKŞİ** was born in 1991, in Düzce, Turkey. He received the B.S. degree in Mechatronic Engineering from the University of Kocaeli, Kocaeli, Turkey in 2013. From 2015 and 2017, he worked as a research and development engineer at Semiz Elektronik. In 2018 July, he joined to AVL Turkey and still works here as an autonomous driving development engineer.

**G. Gökmen,** was born in 1974. He received B.S, M.S and PhD degrees from Marmara University, Istanbul, Turkey. He has been working as a full professor at Marmara University. His current interests are measurement method, signal processing and artificial intelligence techniques.