

Road Lane Segmentation Using Deconvolutional Neural Network

Dwi Prasetyo Adi Nugroho and Mardhani Riasetiawan^(✉)

Universitas Gadjah Mada, Yogyakarta, Indonesia
dwi.prasetyo.a@mail.ugm.ac.id, mardhani@ugm.ac.id

Abstract. Lane departure warning (LDW) system attached to modern vehicles is responsible for lowering car accident caused by inappropriate lane changing behaviour. However the success of LDW system depends on how well it define and segment the drivable ego lane. As the development of deep learning methods, the expensive light detection and ranging (LIDAR) guided system is now replaced by analysis of digital images captured by low-cost camera. Numerous method has been applied to address this problem. However, most approach only focusing on achieving segmentation accuracy, while in the real implementation of LDW, computational time is also an importance metric. This research focuses on utilizing deconvolutional neural network to generate accurate road lane segmentation in a realtime fashion. Feature maps from the input image is learned to form a representation. The use of convolution and pooling layer to build the feature map resulting in spatially smaller feature map. Deconvolution and unpooling layer then applied to the feature map to reconstruct it back to its input size. The method used in this research resulting a 98.38% pixel level accuracy and able to predict a single input frame in 28 ms, enabling realtime prediction which is essential for a LDW system.

Keywords: Convolutional Neural Network · Semantic segmentation
Ego lane segmentation · Lane departure warning

1 Introduction

With the development of urban transportation system, safety become one of the most important aspect when developing advanced driving system. According to WHO (2009), road incident is the highest cause of death for people in age 15–29, higher than suicide or HIV/AIDS [1]. This road accident according to IFRCRCS is up to 80–90% caused by bad driving behavior, such as improper lane changing [2].

However, road accident caused by the improper lane changing behavior can be reduced by 11–23% with the implementation of Lane Departure Warning (LDW) system [3]. LDW works by segmenting the drivable lane of the road and guide the driver to follow this lane. Most of the segmentation approach are using LIDAR sensor. However, this sensor is relatively expensive and unable to give precise information of color and texture of the road. On the other side, segmentation can also be done by analyzing image captured by digital camera that is relatively cheaper compared to light detection and ranging (LIDAR) sensor and able to capture road detail in high resolution images.

While there are many approach to segment drivable lane using digital image, most of them only focus on getting high accuracy. In the real implementation, computational time is also an important metric. An LDW system should be able to warns the driver immediately when the driver leave their drivable lane improperly. The use of fully convolutional network (FCN) that is first introduced in [4] opens possibility for real time image segmentation. However, the simple upsampling method employed in FCN produce inaccurate reconstruction of segmented image.

2 Related Works

Road and lane segmentation is an active research for past few years. Methods, both using traditional handcrafted features as well as deep learning based are employed to solve the segmentation problem.

A work by Aly [15] make use of Inverse Perspective Mapping (IPM) transformation followed by Random Sample Consensus (RANSAC) to extract road lane candidates from input images. RANSAC spline fitting method then applied to the candidates to retrieve appropriate road lane. Similar work to extract road lane features has been done by Niu et al. [14]. Features extracted using curve fitting that produce small lane segment followed by clustering it. The final segmentation obtained by using particular features such as distance between dashed line and its vanishing point.

He et al. [12] used the dual-view convolutional network to solve the segmentation problem. The first view shows road from the driver's perspective. The second view is the result of IPM transformation, resulting a bird eye view of the road. Each view feeded into two different Convolutional Neural Network (CNN) that are then joined to a single fully connected layer at the end.

An attempt to reduce the dependency on human labeling on solving road and lane segmentation was done in [13]. OpenStreetMap data was used to reconstruct the surrounding environment, resulting additional training data. Fully Convolutional Network (FCN) is then trained on the combined data.

Contextual Blocks was introduced in [11] as a way to provide contextual information to the classifier. These blocks are classification, contextual blocks and road blocks. In order to determine the class of a classification blocks, it uses the feature extracted from its own block as well as the corresponding contextual and road blocks. The feature is then concatenated to a single vector an inputted to a standard MLP.

Mohan in [5] proposed the use of deconvolutional neural network to address road segmentation problem. The deconvolutional layer (also known as transposed convolution) is used as learnable upsampling part of a deep neural network. This research also introduce multipatch training method, in which the input image is separated to several patch and the same number of network is trained on each patch independently. Although this method produce outputs with high F-score, it takes 2 s per image on its inference time. Other methods with similar F-score are able to segment an input in less than 0.2 s.

A convolutional patch network for road and lane segmentation is proposed in [6]. On this method, a standard CNN is trained to predict each pixel on a input image by looking at the value of its surrounding pixel. The output of the network is binary

decision stating whether the current pixel is part of the road/lane or not. The inference process should then be done to each pixel on input image. This means that the inference process must be iterated up to the total number of pixel in the image. Thus, resulting a 30 s inference time of a single input image.

Fully Convolutional Network (FCN) has been employed to address slow inference time problem [7]. This network remove the fully convolutional layers on a standard Convolutional Neural Network. By removing the fully connected layers, the number of weight to be multiplied on a forward phase reduced significantly. This research also employ Network in Network architecture that is claimed to reduce inference time. However, the FCN architecture is only used when inferencing a new image. The network is trained in patch based and the network transformation to become FCN requires a complex prerequisite.

3 Method

3.1 Network Architecture

The network architecture is consist of two part. The first part is the encoder, which is used to generate segmentation map from input image. The second part is the decoder, which is used to reconstruct the segmentation map back to its input size.

The final network is structured by stacking several encoders followed by a same the number of decoders as shown in Fig. 1.

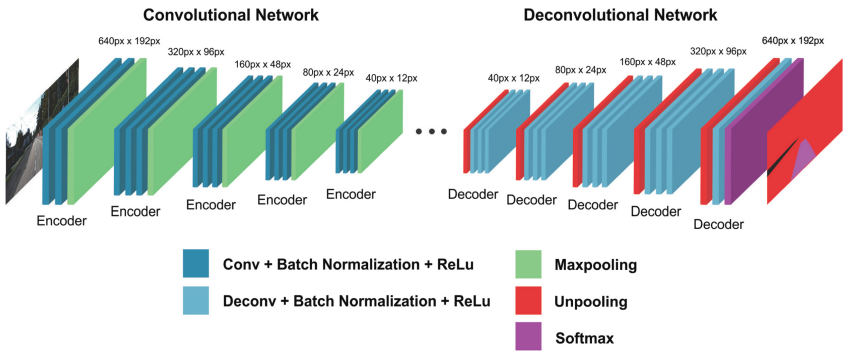


Fig. 1. Encoder-decoder architecture

3.2 Encoder Architecture

The encoder is used to encode input image to a segmentation map. It consist of a series of convolution and pooling operation. The encoder segment the image by classifying each pixel on its input image to either part of ego lane or not. By classifying all pixel on the input image, all pixel will be labeled. Thus, giving different color code to each pixel label will output a segmentation map of the input image.

Convolution. The first operation in the encoder part is the convolution operation. In our final architecture, we use 64 filters with the size 5×5 . These filter is initialized with random number using normal distribution. The filter convolved through its input with the stride 1.

Maxpooling. The maxpooling layer is used to add translation invariant property to the network. This layer also enlarge the receptive field on the following encoder, enabling the network to give coarse prediction of its input image. However, the use of maxpooling in the encoder resulting a feature map that is spatially smaller than its input size.

3.3 Decoder Architecture

The decoder is used to reconstruct the segmentation map created by the encoder part. This can be done by using deconvolution and unpooling operation.

The output of the last decoder has the same spatial size with the input image and the ground truth. We then add a cross-entropy loss function layer to calculate how far the difference of the segmentation result generated by the network with its respective ground truth. The comparison is done pixel wise, that is by comparing each pixel on the segmentation result with pixel on the same position on the ground truth image. This error value is then back propagated through the network to correct its weights and produce better segmentation.

Unpooling. Reconstructing from low resolution to high resolution image is not trivial because there are some information that is lost during the maxpooling step in the encoder part. To overcome this problem, [9] propose the use of unpooling layer to reconstruct high resolution image from low resolution image. The position of maximum value in maxpooling step is stored in a switch variable. The unpooling layer then use this information to place current pixel value to the position of its maximum value when maxpooled that is stored in the switch variable and fill the other position with 0. The maxpooling operation is shown in Fig. 2.

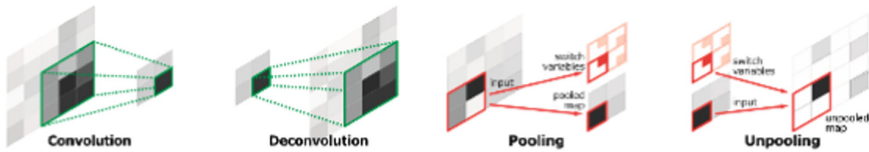


Fig. 2. Unpooling and deconvolution operation

Deconvolution. The deconvolution operation can be thought as inverse of convolution operation. It convolves to each single pixel on its input image with a $M \times N$ kernel, then multiply the current convolved pixel with the kernel. The multiplication result a $M \times N$ output matrix. A certain activation function then applied to the multiplication result. The $M \times N$ activated matrix then placed in the output space of the deconvolution layer. The deconvolution layer also works with certain stride value S . As the

layer move one step in the input space, it moves S steps in the output space. When two output overlap each other, those overlapping value is added. The deconvolution operation can be used to convert sparse image generated by unpooling layer to dense image. This operation is shown in Fig. 2.

As for recap, the unpooling layer is used to increase the spatial size of its input image. The sparse output generated by unpooling layer is then filled using deconvolutional layer using its learnable filter. By stacking several decoders, the segmentation map size can be reconstructed back to its input image size (Fig. 3).



Fig. 3. Sample input dataset and the segmentation ground truth

3.4 Training Deconvolution Neural Network

The encoder-decoder architecture is trained on 95 image and ground truth pair of The KITTI Vision Benchmark Database. This dataset is a collection 2D images of road lane photograph taken from driver perspective. The ground truth consist of binary images with the same spatial dimension as the dataset images with each pixel is labeled as ego-lane or not ego-lane.

The input images is roughly 1242×375 pixels by size. However, the exact size across images are not equal. In addition, the original size of the input images is too big for available computational power. Therefore, we reduce the spatial size of all input images and its corresponding ground truth to $640 * 192$.

We use Adam algorithm [10] with starting learning rate of 0.0001 to train our final architecture. The forward phase of the training algorithm will compute the predicted value of a every pixel. A cross-entropy loss function then computed using predicted value and its ground truth. The loss/error then backpropagated through the network's architecture to determine the update needed to be done to each of network's parameters (weights).

Since we introduce deep neural network with many parameters, we add dropout layer to each of our encoder and decoder to reduce overfitting. We also add batch normalization layer to each encoder and decoder to speed up the training process.

We employ 7-fold cross validation on the training process. The data were initially partitioned to 7 part. On each fold, six parts of the data is used for training and the remaining one is used for validation. The network is trained for 27 epochs within each fold.

4 Experimental and Evaluation Result

4.1 Overall Result

In this section, we present result of our method in twelve different architecture setups. On this experiment, we vary the architecture depth, that is the number of encoder and decoder used in the architecture, and the filter/kernel size. We varying the architecture depth with the number of encoder-decoder pair of 3, 5, 6, and 7 and varying the kernel size to be 3×3 , 5×5 , and 7×7 . Other hyperparameters are set to constant. Our experiments result is shown in Table 1.

Table 1 shows that with more encoder-decoder used, the network achieve a higher accuracy. Meanwhile the inference time is highly depends on the filter size used. The larger filter size gives longer inference time.

Table 1. Architecture experiments

Model name	Architecture	Kernel size	Loss	Accuracy	Inference time
Model3-K3	3 Encoder - 3 Decoder	3×3	0.1447	0.9233	0.024 s
Model3-K5	3 Encoder - 3 Decoder	5×5	0.1330	0.9214	0.029 s
Model3-K7	3 Encoder - 3 Decoder	7×7	0.1126	0.9396	0.035 s
Model4-K3	4 Encoder - 4 Decoder	3×3	0.1193	0.9333	0.024 s
Model4-K5	4 Encoder - 4 Decoder	5×5	0.0859	0.9697	0.028 s
Model4-K7	4 Encoder - 4 Decoder	7×7	0.0709	0.9772	0.035 s
Model5-K3	5 Encoder - 5 Decoder	3×3	0.0904	0.9673	0.024 s
Model5-K5	5 Encoder - 5 Decoder	5×5	0.0522	0.9838	0.028 s
Model5-K7	5 Encoder - 5 Decoder	7×7	0.0558	0.9834	0.035 s
Model6-K3	6 Encoder - 6 Decoder	3×3	0.1216	0.9560	0.024 s
Model6-K5	6 Encoder - 6 Decoder	5×5	0.1949	0.9559	0.029 s
Model6-K7	6 Encoder - 6 Decoder	7×7	0.1241	0.9576	0.035 s

The correlation between filter size and inference time is related to the number of parameter/weight of a filter. Filters on the convolution and deconvolution layer contains weight that will be multiplied with its input on inference process to produce segmentation. On a 3×3 filter, there are 9 weights on a filter. Meanwhile, on 5×5 there are 25 weights on a filter. Increasing the filter size to 7×7 produce 49 weights on a filter.

The different on that three configuration is not significant when we only sees it as a single filter. However, in our architecture, we use 64 filter for each encoder or decoder. Each encoder is always paired to a decoder. Thus, we use 128 filter for each encoder-decoder pair. Moreover, we use 3 up to 6 encoder-decoder pair in our experiment. This high number of filter that is used introduce significant different on number of weight that is used on different filter size. Table 2 shows comparison of number of weight that is used on different filter size. The difference of number of weight in a complete architecture shown on Table 3.

Table 2. Number of weight in an encoder-decoder pair

Kernel size	Number of weights on each Filter	Number of weights for each encoder-decoder pair
3×3	9	1152
5×5	25	3200
7×7	49	6272

Table 3. Number of weight in 5 encoder-decoder architecture

Architecture	Total number of weight (using 5 encoder-decoder pair)
3×3	5760
5×5	16000
7×7	31360

Although that the number of weights grow significantly as small increase done on the filter size, the networks depth (number of encoder-decoder pair used) shows only a little change on the number of weight that is used when it is varied. The reason is that by adding a new encoder-decoder pair on a network, we will only create $128 \times$ (filter size) new weight. Table 4 shows the number of weight that is used on a four different architecture with the same filter size.

Table 4. Number of weights on different architectures

Architecture	Total number of weight (using 3×3 filter size)
3 Encoder - 3 Decoder	3456
4 Encoder - 4 Decoder	4608
5 Encoder - 5 Decoder	5760
6 Encoder - 6 Decoder	6912

4.2 Qualitative Result

In this section we shows qualitative result on four different architecture depths, as shown in Fig. 4. This comparison suggest that as the number of encoder-decoder pair increased, the network produce better segmentation. The better segmentation can be identified by smaller false positive and false negative.

The better segmentation result is produced by employing more encoder-decoder pair. Initial layer on the encoder part tend to learn primitive feature such as edge and curve. Meanwhile, the later layer tend to learn more complex feature such as geometric shapes and other lane feature.

However, each encoder layer reduces the its input spatial size to into half of its original size. In our experiment, we only use up to 6 encoder-decoder pair.

Further addition of encoder-decoder pair will produce segmentation map that is too small and difficult to upsampled. This limitation is showed in Table 5.

**Fig. 4.** Qualitative result over architectures**Table 5.** Output size shrinking

Encoders layer	Output size
Encoder 1	310×96
Encoder 2	160×48
Encoder 3	80×24
Encoder 4	40×12
Encoder 5	20×6
Encoder 6	10×3

Not all images in the testing set is well segmented. Figure 4 shows several image that is difficult to segment. The three images share similar characteristic for having shadow present in the image. We found that the training set does not contain many image with such characteristic. This makes the network not able to give a good generalization for those image, and produce inferior segmentation result.

4.3 Inference Time Evaluation

In order to get a better understanding on how this architecture could process the segmentation task in short time frame, we conduct a comparative experiment with the convolutional patch based method as proposed in [6].

In this method, input images are broken down into patches. Each patch is 16×16 pixels. The 16×16 pixel will be used as context to predict 4×4 pixel on its center. Patches are created starting from top left of an image to its bottom right. The stride of 4 pixel is used when creating patches. Thus, from an input image with size of $640 * 192$, there are 7680 patches created. Summing up all 95 input images resulting 729600 image patches.

Table 6 shows the comparison of the two architecture in term of pixel accuracy and its inference time.

Table 6. Output size shrinking

Parameter	Convolutional patch	Deconvolutional neural network
Accuracy	47.50%	98.38%
Average inference time	7.936 s	0.028 s
Average training time on each epoch	320 s	15 s

On the convolutional patch method, classification is done on each 16×16 path. As the size of each patch is small, the inference time is only 2 ms. However, since there are 7680 different patches on a single input image that need to be evaluated, the total inference time of an image is 15 s.

On the other hand, as shown in Table 1 the average inference time for our architecture is 29 ms for each input image. On our architecture, the inference process only done once on each input image. This makes the inference time significantly faster.

5 Conclusion

We employ encoder-decoder architecture to solve road lane segmentation problem. The use of fully convolutional network coupled with gradual upsampling approach produce segmentation result that is not only accurate but also computationally feasible for real-time application. Our best model scores 98.38% accuracy on 7 fold cross validation and able to inference a single image frame in 28 ms.

The segmentation accuracy is related to the network depths, while the inference time is highly correlated to the filter size used in convolution and deconvolution operation. Deeper architecture produce better segmentation. However, the larger the filter size, the longer the time needed to inference a single image.

However, several optimization could be done to achieve better result. The small number of dataset used for training resulting bad segmentation on several test images. Our experiment reduce original size of the dataset to its half due to computational limitation. A better hardware could be used so that the training can be done in its original size. A skip connection, as introduced in [4] can also be employed so that the upsampling is not only takes account of its last segmentation map but also the prior segmentation maps.

References

1. World Health Organization: Global status report on road safety: time for action (2009). http://apps.who.int/iris/bitstream/10665/44122/1/9789241563840_eng.pdf. Accessed 2 Sept 2016

2. IFRCRCS: Practical Guide on Road Safety: A Toolkit for Red Cross and Red Crescent Societies (2007). <http://www.grsproadsafety.org/themes/default/pdfs/GRSPRed%20Cross%20Toolkit.pdf>. Accessed 7 Sept 2016
3. Kusano, K.D., dan Gabler, H.C.: Comparison of expected crash and injury reduction from production forward collision and lane departure warning systems. *Traffic Injury prevent.* **16** (sup2), S109–S114 (2015)
4. Jonathan, L., Shelhamer, E., dan Darrell, T.: Learning deconvolution network for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015)
5. Mohan, R.: Deep deconvolutional networks for scene parsing (2014). <https://arxiv.org/pdf/1411.4101.pdf>. 15 November 2014. Accessed 22 Oct (2016)
6. Brust, C.A., et al.: Efficient convolutional patch networks for scene understanding. In: *CVPR Scene Understanding Workshop* (2015)
7. Mendes, C.C.T., Fremont, V., dan Wolf, D.F.: Exploiting fully convolutional neural networks for fast road detection. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3174–3179 (2016)
8. Noh, H., Seunghoon H., dan Bohyung H.: Learning deconvolution network for semantic segmentation. In: *Proceedings of the IEEE International Conference on Computer Vision 2015*, pp. 1520–1528 (2015)
9. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014*. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
10. Kingma, D., Jimmy B.: Adam: a method for stochastic optimization (2014). <https://arxiv.org/pdf/1412.6980.pdf>. 30 Januari 2017. Accessed 31 Mar 2017
11. Mendes, C., Frémont, V., Wolf, D.: Vision-Based Road Detection using Contextual Blocks (2015). <https://arxiv.org/pdf/1412.6980.pdf>. 30 Januari 2017. Accessed 10 Mar 2017
12. He, B., Ai, R., Yan, Y., dan Lang, X.: Accurate and robust lane detection based on dual-view convolutional neural network. In: *Intelligent Vehicles Symposium (IV)*, pp. 1041–1046 (2016)
13. Laddha, A., Kocamaz, M.K., Navarro-Serment, L.E., dan Hebert, M.: Map-supervised road detection. In: *Intelligent Vehicles Symposium (IV)*, pp. 118–123 (2016)
14. Niu, J., Lu, J., Xu, M., Lv, P., dan Zhao, X.: Robust lane detection using two-stage feature extraction with curve fitting. *Pattern Recogn.* **59**, 225–233 (2015)
15. Aly, M.: Real time detection of lane markers in urban streets. In: *Intelligent Vehicles Symposium*, pp. 7–12 (2008)

Soft Computing in Data Science

Third International Conference, SCDS 2017, Yogyakarta,
Indonesia, November 27–28, 2017, Proceedings

Mohamed, A.; Berry, M.W.; Yap, B.W. (Eds.)

2017, XV, 317 p. 123 illus., Softcover

ISBN: 978-981-10-7241-3