**Automated Bot Using Semantic Segmentation and**

**Deep Learning**

**Introduction**

An automated driving system is a complex combination of various components that can be defined as systems where perception, decision making, and operation of the automobile are performed by electronics and machinery instead of a human driver, and as introduction of automation into road traffic. This includes handling of the vehicle, destination, as well as awareness of surroundings. While the automated system has control over the vehicle, it allows the human operator to leave all responsibilities to the system.

Segmentation is essential for image analysis tasks. Semantic segmentation describes the process of associating each pixel of an image with a class label, (such as flower, person, road, sky, ocean, or car). Applications for semantic segmentation include: Autonomous driving.

**Semantic segmentation** is one of the key problems in the field of computer vision. Looking at the big picture, semantic segmentation is one of the high-level tasks that paves the way towards complete scene understanding. The importance of scene understanding as a core computer vision problem is highlighted by the fact that an increasing number of applications nourish from inferring knowledge from imagery. Some of those applications include self-driving vehicles, human-computer interaction, virtual reality etc. With the popularity of deep learning in recent years, many semantic segmentation problems are being tackled using deep architectures, most often Convolutional Neural Nets, which surpass other approaches by a large margin in terms of accuracy and efficiency.

Convolutional Network (FCN) learns a mapping from pixels to pixels, without extracting the region proposals. The FCN network pipeline is an extension of the classical CNN. The main idea is to make the classical CNN take as input arbitrary-sized images. The restriction of CNNs to accept and produce labels only for specific sized inputs comes from the fully-connected layers which are fixed. Contrary to them, FCNs only have convolutional and pooling layers which give them the ability to make predictions on arbitrary-sized inputs.

In the modern day transportation, accidents due to the manual errors created by humans on a high rate which is further leading to the high in the death rates year by year, To overcome this the Autonomous steering provides a better solution by semantic segmenting the image captured by the camera and learning the ENet Neural Network so that it segments images captured based on different class labels and varied color representation so that the Automated vehicle detects the obstacles forward and steers the vehicles away from the obstacle. Hence the Automated vehicle avoids the accidents caused and eliminated the human intervention.

**Literature Survey**

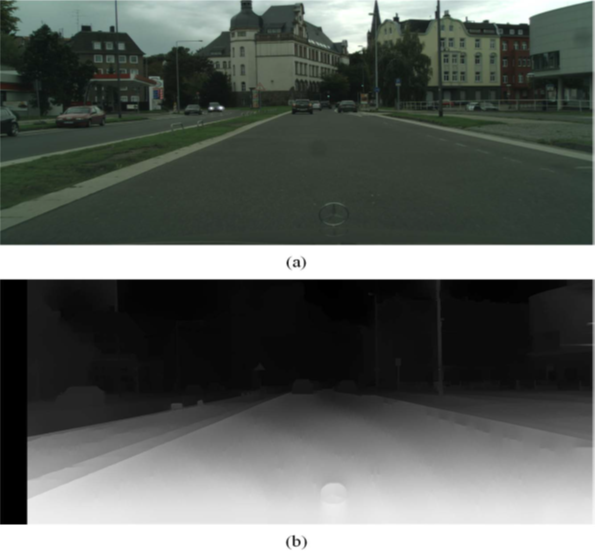
**Linhui Li, Bo Qian, Jing Lian, Weina Zheng, and Yafu Zhou, “Traffic Scene Segmentation Based on RGB-D Image and Deep Learning”,**

**IEEE Transaction on Intelligent Transportation Systems- 2017**

* **Obtaining Disparity Map**

In stereo vision, the disparity map can be obtained through the stereo matching step. From the disparity map, we can obtain object information such as depth, edges, etc. which are important for extracting rich features for the convolutional neural network. Developing a good disparity map can improve the segmentation accuracy for certain networks.

There are three stereo vision matching algorithms: local matching, semi-global matching and global matching. While the matching accuracies of the three matching algorithms increase in order, the time consumed by the algorithms also increase in the same order. For real-world applications of traffic scene segmentation, while the matching accuracy is important, the real-time factor is a more important metric. Due to this reason, while choosing the appropriate algorithm we need to balance the trade-off between accuracy and time complexity. We observe that the accuracy of the semi-global matching is close to the global matching while at the same time semi-global matching has better real-time performance. Therefore, we use the semi-global matching algorithm to obtain the disparity map. However, using semiglobal matching results in two issues: 1) there are many non-matched points in the disparity map and 2) the edges of objects in a disparity map are usually rough, which is bad for feature extraction. We use a method of fast global image smoothing to optimize the coarse disparity map and make the disparity value more continuous. This is the key to improving the segmentation accuracy of traffic scenes when using RGB-D images for training and testing the network. Below are shown with Matching Result. Subfigure (a) is the left colour image in the image pair. Subfigure (b) is the disparity map.



* **RGB-D Dataset Description**

In the representative outdoor traffic scene dataset, the cityscapes dataset is an annotated subset of the main dataset that contains semantic classes such as road, buildings, cars etc. In particular, it contains traffic scene images captured by stereo cameras that make it possible to verify the reliability of the stereo matching algorithm and train the network using RGBD images. Therefore, we use the cityscapes dataset as a benchmark to verify the network performance. The cityscapes dataset contains 5000 colour image pairs which were originally divided into the training set (2975 images), the validation set (500 images) and the test set (1525 images). Fine annotations are provided for the training and validation sets. The training set is used to train the network, the validation set is to verify the numerical accuracy of the trained network and the test set is used to assess the qualitative performance of the network.

First, we divide a traffic scene into 11 dominant classes: road, sidewalk, building, pole, traffic sign, tree, lawn, sky, person, vehicle, two-wheelers. The classes are labelled from 0 to 10, respectively. The other classes are all labelled as 11 and this group is not used during the computation for updating the weights in the back-propagation stage. Next, the disparity map D is acquired based on the stereo matching algorithm mentioned before. Finally, the left RGB image and the disparity map are fused into a four-channel RGB-D image. The images in the cityscapes dataset are 2048 × 1024 pixels in dimension; but we resize them to 400 × 200 pixels to avoid high memory costs at training time. The final generated dataset is split in the same proportion, with 2975 RGB-D images in the training set, 500 RGB-D images in the validation set and 1525 RGB-D images in the test set.

TRAFFIC SCENE SEGMENTATION BASED ON RGB-D IMAGE AND DEEP LEARNING



Below Figure illustrates the detailed configuration of the **Network Architecture**. The network architecture consists of an encoder network and a corresponding decoder network. The encoder network converts the input image to a set of feature maps and the decoder network produces the image segmentation from this set of generated feature maps.



**Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh2, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, Bernhard Nessler, Sepp Hochreiter, “Speeding up Semantic Segmentation for Autonomous Driving”-2016**

**Institute of Bioinformatics, Johannes Kepler University Linz, Austria**

**Audi Electronics Venture GmbH, Germany**

* **Network Architecture for Speeding Up of Semantic Segmentation**

Reducing the computational burden of semantic segmentation is essential to make it feasible for embedded systems and autonomous driving. Neural networks are trained on servers or workstations with powerful GPUs, and these GPU systems are subsequently used for inference on new data. However, these commodities do not exist in self-driving cars. A self-driving car needs to react to new events instantly to guarantee the safety of passengers and other traffic participants, while it is often acceptable if the borders of objects are not recognized perfectly down to a pixel resolution. To segment an image in real-time is a strong requirement in self-driving applications. Thus, it is critical that any convolutional neural network deployed in these systems fulfils strict requirements in execution speed.

There has been a vast amount of research in reducing the computation required for deep learning. Squeeze Net showed that it was possible to reproduce the image classification accuracy of powerful CNNs such as Alex Net using 50x less parameters by employing a more efficient architecture. ENet followed the same path and showed that semantic segmentation is feasible on embedded devices in real-time. Another line of research increases the efficiency of existing networks by deriving smaller networks from larger counterparts [1, 11], by pruning or quantizing weights [6, 8] or tweaking the network for execution on specific hardware designs. These methods can be applied on top of new architectures to speed up execution.

This trade-off in resolution is typically solved by using skip-connections from lower layers to the output which increase the resolution at layers close to the output. Skip-connection were introduced by the The Fully Convolutional Network (FCN), which still serves as a blue-print for most modern approaches. These approaches only differ in how they encode the object level information and how they decode these classifications to pixel-exact labels.

Architecture of the proposed network for semantic segmentation.



Input

Output

Convolution Pool

Fire module

Parallel dilated convolution

Transposed convolution

Refinement module

In order to achieve semantic segmentation in real time, we have to trade execution speed against achievable segmentation accuracy. Like most successful segmentation networks, our network is structured as an encoder-decoder pair. An encoder CNN detects higher-level objects such as cars or pedestrians in the input image. A decoder takes this information and enriches it with information from the lower layers of the encoder, supplying a prediction for each pixel in the original input.

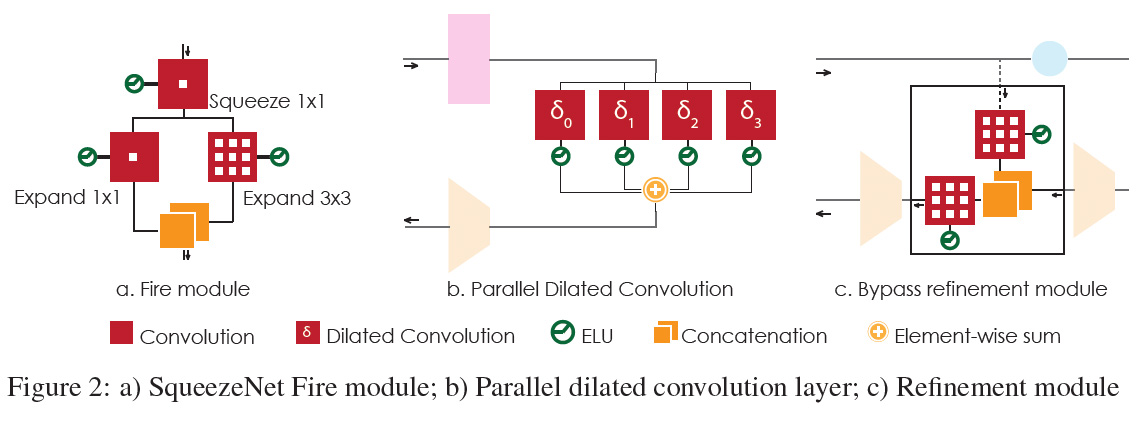
* **Encoder**

The encoder is a modified Squeeze Net 1.1 architecture [15], which was designed as a low-latency network for image recognition while retaining Alex Net [17] like accuracy. The main computational modules of Squeeze Net are the so-called “fire” modules consisting of three convolutional operations,

depicted in Figure 2a. The encoder consists of eight “fire” modules, interspersed with a total of three

max-pooling layers for down sampling. All rectified linear units (ReLUs) of the original architecture

are substituted with exponential linear units (ELUs) [4], which make more efficient use of parameters by also conveying information in the negative part of the activation.



* **Parallel Dilated Convolutions**

The decoder is based on a parallel dilated convolution layer [20] as depicted in Figure 2b. This dilated layer combines the feature maps at the encoder output at different receptive field sizes by using four dilated convolutions of kernel size 3 with different dilation factors. This is equivalent to sampling the layer input with different rates. The contributions from the four dilated convolutions are then fused by an element-wise sum. As a result, the receptive field size is increased and multiscale spatial dependencies are taken into account without having to resort to fully connected layers which would be computationally infeasible. Thus, the decoder can be realized by considerably less parameters while the high performance is kept.

* **Decoder and Bypasses**

Pooling layers in the encoder are used to ensure a degree of translational invariance when detecting

the parts of an object. However, they in turn reduce the spatial resolution of the output. Transposed

convolutions in the decoder are used to up sample the information back to its original size. To improve the up sampling, we don’t just use the data that comes directly from the layer below the transposed convolution layer, but combine it with low-level knowledge from lower layers of the encoder. These layers are responsible for detecting finer structures at a higher resolution, which helps with classifying the contours of objects more exactly. Each refinement module combines two streams of information, one coming from the previous up sampling layer, the other one from the encoder. The two convolutional layers in the refinement module learn how to weigh these two streams before passing the information on to the next up sampling layer. Right before every pooling layer in the encoder, a bypass branches off to the refinement module. Once there, a convolution layer weights knowledge from lower layers. Then, it is concatenated with semantic object information from the previous up sampling layer. A second convolutional layer combines the concatenated feature maps from both branches into the class map.