Semantic Segmentation
A Model Comparison on Cityscapes Dataset

Pietro Maria Sanguin
pietromaria.sanguin@studenti.unipd.it

Jacopo Magliani
jacopo.magliani@studenti.unipd.it

Marco Furlan
marco.furlan.13@studenti.unipd.it

Abstract

When dealing with tasks that require a basic understanding of a street scene like autonomous driving and video surveillance, semantic image segmentation represents the starting point, classifying each pixel of a high resolution image into one of the available semantic labels. Since these are video-based tasks, objects and people show frequent and notable changes of scale, and this represents a challenge for high-level features representation, since a model must be able to correctly encode multi-scale information. Numerous models have been developed over the years to solve this challenging task and in this report we present the results obtained experimenting with three of them: DeepLabV3, U-Net and DenseASPP. All these models adopt different strategies and architectural implementations to effectively capture multi-scale contextual information and efficient segmentations of the images. The dataset of images we used as benchmark for testing the different models is Cityscapes [10], which is specifically created for tasks of street scene segmentation. The three models have very similar performance but DenseASPP can achieve better results.

1. Introduction

In the street scene semantic segmentation scenario each pixel of an image of a street scene is labeled with one of the possible semantic classes like road, person, building, vehicle, traffic lights, and so on. Street scene semantic segmentation plays a fundamental role in several computer vision applications as: scene understanding, urban planning and infrastructure management, where a correct semantic segmentation can enable machines to understand the visual content of a street scene with applications in crowd counting and control, autonomous driving, urban planning, navigation systems, surveillance drones.
Among the numerous models that have been developed to solve one or more of these tasks, we have tested the following:

- DeepLabV3
- U-Net
- DenseASPP

The three models perform similarly using all the metrics except when treating each class equally regardless of the prevalence in the dataset. In this case performance drops for all the models, but DenseASPP performs much better than U-Net, while DeepLabV3 is in between. The common feature with DeepLabV3 and DenseASPP is the use of the ASPP, so we can say that the addition of dense connections within the ASPP is an effective one.

2. Related Work

Here we present briefly a high level overview of the models used. DeepLabV3 introduced in [7] is a model that incorporates the Atrous Spatial Pyramid Pooling (ASPP) module [8] and effectively captures multi-scale contextual information by utilizing parallel atrous convolutions with different dilation rates. Unet [11], as its name suggests, has a U-shaped architecture that combines a contracting path and an expanding path. The first path captures the context and reduces the resolution, the second recovers the resolution and refines the segmentation output. Using skip connections it can transfer detailed spatial information across different layers of the network, obtaining better segmentations. Last but not least there is the DenseASPP [12] which is a model that improves the ASPP module by dense connections between parallel atrous convolutions, facilitating the flow of information and the propagation of the gradient, capturing in this way multi-scale contextual information.

In section 4 we provide a deeper explanation of the models architecture and learning.

3. Dataset

The Cityscapes dataset is a common benchmark dataset for computer vision, in particular for semantic understanding of urban street scenes. It includes a collection of various high-resolution images captured from the perspective of a moving vehicle, taken from real-world driving scenarios. It is rich of urban scenes as streets, intersections, sidewalks, buildings, vehicles, pedestrians, and other objects commonly found in urban environments, in various weather conditions and times of the day. The dataset contains images captured from different cities in Germany such as: Berlin, Frankfurt, Munich and is diveded in train, validation and test. Unfortunately the test set does not provide the masks to do the evaluation, therefore in every application we tested our models on the validation set. To enable semantic segmentation, each pixel in the images is labeled with one of 30 class categories such as road, person, car etc... These annotations are used as ground truth for evaluating semantic segmentation models. In our applications we reduced the number of classes to 10 or 19 depending on the architecture used. The legends are shown in figure 10 and 9. This choice was due by time constraints or necessity of adapting pretrained models to our problem.

4. Method

Before describing the models we talk about some key aspects as ResNet50, Atrous Convolutional Filter (ACF) and Atrous Spatial Pyramid Pooling (ASPP).

ResNet50 (Residual Network) is a convolutional neural network architecture, a variant of the original ResNet architecture introduced by Microsoft Research in 2015, and it is used as a backbone network in many computer vision tasks, including semantic segmentation.

Its name comes from using a total of 50 layers, distributed between 1 MaxPool layer, 1 Average Pool layer and 48 convolutional layers

As shown in Figure 1, ResNet50 architecture begins with a convolutional layer that performs a convolution with 64 filters 7x7 and a stride of 2, followed by a rectified linear unit (ReLU) and a max pooling layer with a 3x3 pooling size and a stride of 2. ReLU introduces non-linearity and helps the network to learn complex representations.

Then, the remaining 48 convolutional layers are grouped into four stages, each containing multiple residual blocks. Each block as in Figure 2 has two or three convolutional layers with skip connections that bypass one or more layers and connect the input to the output of the block. These skip connections let information from previous layers to be preserved and reused in later layers, allowing the network to learn residual mappings, facilitating gradient flow and so improving the training process.

After the last stage, a global average pooling layer is applied to reduce the spatial dimensions of the feature maps to a 1x1 size to summarize the features.

In the end, a a fully connected layer with 1,000 neurons represents all the possible classes in the ImageNet dataset, on which ResNet50 was originally trained. Each neuron represents the probability of the input image belonging to a specific class.

In the context of semantic segmentation, the output of ResNet50 is modified to produce pixel-wise predictions rather than class probabilities. This is usually done removing the average pooling layer and the fully connected layer.

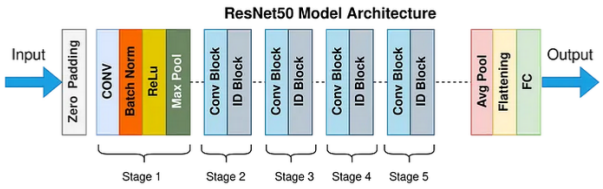

Figure 1. ResNet50 Architecture

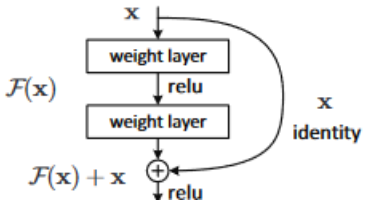

Figure 2. ResNet50 block

Atrous convolution is a variant of traditional convolutional operations. In standard convolution, a filter/kernel moves across an input image with a fixed stride, applying the filter at every location.

In one dimensional case, let $y[i]$ denote output signal and $x[i]$ denote input signal, atrous convolution can be formulated as follows:

$$y[i] = \sum_{k=1}^{K} x[i + d \cdot k] \cdot w[k] \tag{1}$$

where d is the dilation rate, $w[k]$ denotes the k-th parameter of filter, and K is the filter size. This equation reduces to a standard convolution when d = 1. Atrous convolution is equivalent to convolving the input x with up-sampled filters produced by inserting $d - 1$ zeros between two consecutive filter values. Thus, a large dilation rate means a large receptive field. This is done without downsampling the input nor increasing the numbers of kernel parameters.

By introducing these gaps in the convolutional filter, atrous convolution allows for the extraction of features from larger spatial contexts, capturing information from a broader area.

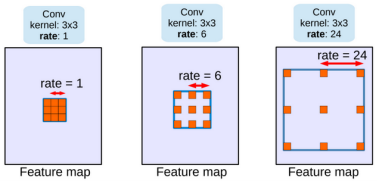

Figure 3. Atrous Convolutional filter

Atrous Spatial Pyramid Pooling (ASPP) is an extension of the Spatial Pyramid Pooling technique incorporating atrous convolutions. ASPP is commonly used in semantic segmentation tasks to capture multi-scale contextual information effectively. ASPP takes in input feature maps generated from a previous CNN architecture (eg ResNet50), then applies parallel atrous convolutions with differen dilatation rates to them in order to capture information at different scales. In parallel there is also a global average pooling layer that aggregates information across the entire input feature maps, allowing the network to capture global context. The results of the parallel atrous convolutions and the pooling operation are upsampled to match the original dimension of the input feature map, and then are concatenated together, creating a multi-scale representation that contains both local and global contextual information. Using $H_{K,d}(x)$ to term an atrous convolution we can consequently write ASPP as:

$$y = H_{3,6}(x) + H_{3,12}(x) + H_{3,18}(x) + H_{3,24}(x) \quad (2)$$

This multi-scale representation can be further processed with additional convolutional layers to reduce the dimensionality and extract more abstract features.

By using ASPP, the network can effectively capture context at different scales, incorporating both refined and raw information. The multi-scale representation obtained through ASPP enables more accurate and detailed predictions for semantic segmentation tasks, where pixel-level labeling is required.

ASPP module composes a feature pyramid using 4 atrous convolutional layers with dilation rate of 6, 12, 18, 24.

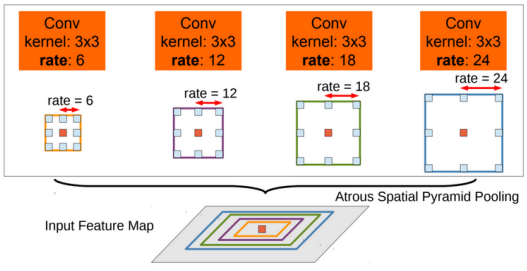

Figure 4. Atrous Spatial Pyramid Pooling

4.1. DeepLabV3

DeepLab is a family of convolutional neural network architectures designed for semantic image segmentation. DeepLabV3 is an advanced version that adds several improvements to previous DeepLabV1 [9] and DeepLabV2 [6]. DeepLabV1 introduced the concept of atrous convolution, which allows the network to have a larger receptive field without increasing the number of parameters, DeepLabV2 enhanced DeepLabV1 by adding a fully connected Conditional Random Field (CRF) as a post-processing step to refine the segmentation results. DeepLabV3, introduced in 2017, removed the CRF and introduced the Atrous Spatial Pyramid Pooling (ASPP module).

The original DeepLabV3 takes also advantage of a backbone architecture (eg ImageNet-pretrained ResNet50) to achieve high-quality pixel-level predictions.

An initial feature map is extracted from the input image by the backbone network ResNet50, but in the end the Average Pooling layer and FC layer are removed, since they would map the extracted features to specific classes. The extracted feature maps are fed to an ASPP network with different dilatation rates, then its output is passed through a 1 x 1 convolution to upsample to the actual size of the original input image, obtaining the final segmented mask for it.

The ASPP module can capture multi-scale contextual information by applying atrous convolutions at multiple dilation rates, leading the network to a better understand of objects and regions of the image.

4.2. UNet

U-Net [11] is a convolutional neural network that affirmed itself as one of the best in the field of biomedical image segmentation. It can be seen as an evolution of a Fully Convolutional Network.

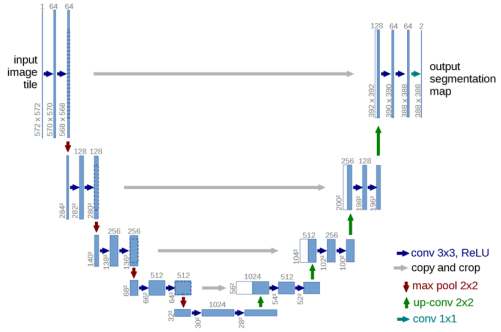

Figure 5. U-Net architecture (taken from [11])

The U-shaped structure of this algorithm is summed up in Figure 5. It consists of a contracting path (left side, the encoder) and an expanding path (right side, the decoder) .

The contracting path follows the typical architecture of a

convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded), each followed by a rectified linear unit (ReLU), and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels is doubled.

Each step in the expansive path consists of an upsampling of the feature map, followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the corresponding cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.The cropping is necessary due to the loss of border pixels in every convolution.

In the end a final layer 1x1 convolution reduces the number of feature channels to match the desired number of classes for segmentation. The final prediction output is a pixel-wise probability map or a label map.

4.3. DenseASPP

The DenseASPP architecture is built upon the ASPP module but expands it to capture multi-scale contextual information, using different dilatation rates for the atrous convolution layers and dense connections. The output of each atrous layer is concatenated with the input feature maps and all the outputs from previous layers, and the new concatenated feature maps are fed into the following layers.

Compared with the original ASPP, DenseASPP stacks all dilated layers together, and connects them with dense connections. Following equation 2, each atrous layer in DenseASPP can be formulated as follows:

$$y_l = H_{K,d_l}\left([y_{l-1}, y_{l-2}, \cdots, y_0]\right) \tag{3}$$

where d_l represents the dilation rate of layer l, and $[\dots]$ denotes the concatenation operation. $[y_{l-1}, \dots, y_0]$ means the feature map is formed by concatenating the outputs from all previous layers.

Dense connections allow each atrous layer within the module to receive in input the concatenation of the original feature maps and all previous outputs, allowing information to flow across different scales and forming a multi-scale feature representation.

Dense connections allows to have a denser feature pyramid, with better scale diversities, larger receptive field and more pixels involved in convolution than in ASPP.

This enables the model to capture both fine-grained details and global context, enhancing the performances particularly in scenarios where precise localization and accurate boundary delineation are crucial.

Overall, DenseASPP is an architecture that improves the original ASPP module by dense connections to capture multi-scale contextual information without increasing too much the overall complexity. This allows the model to produce more accurate and detailed semantic segmentation results.

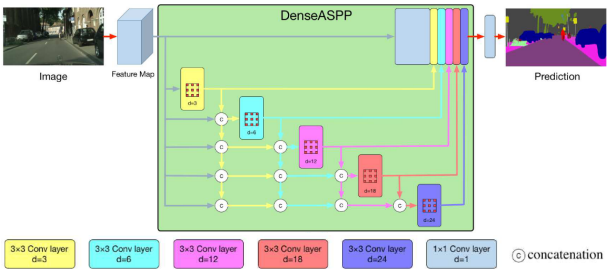

Figure 6. DenseASPP architecture

5. Experiments

5.1. DeepLabV3

The python library PyTorch offers some pretrained architectures, and among these there is the DeepLabV3. The implementation is originally trained on a subset of the dataset COCO which has different classes from Cityscapes. The original labels can identify cars and people, but not sky, road, sidewalk or vegetation.

We tried running the pretrained model on Cityscapes dataset, but of course it only detected the classes that it knew, like ”car” and ”person”, and all the rest was unlabeled. So we proceeded to write the code to train the model on the Cityscapes training images. First we rescaled all images by a factor of 0.5, going from size (2048,1024) to size (1024,512) with bilinear interpolation, which reduces considerably the amount of parameters and computations without a big impact on the performance (reducing further may worsen performance). The training was done on the 2975 raw images of the training set of Cityscapes, randomly shuffled with batch norm = 1, Adam optimizer, 10^{-5} learning rate, Cross Entropy loss, and we let it run for 72 epochs.

The Cityscapes dataset which is available for download online at [Cityscapes link][1] contains 33 classes at the moment, but since we used a model of DenseASPP pretrained on 19 classes we had to manually convert the entire label to those 19 classes, according to ”trainId” that can be found here [Cityscapes labels][2]. The 19 classes are shown in Figure 8.

In Figure 7 we can see some samples from the DeepLabV3 predictions on the validation set of Cityscapes.

5.2. U-Net

The U-Net we used is a PyTorch implementation of the original U-Net from [11], the code can be found in [UNet Link][4]. In order for the training to work successfully we had to modify several parts of the code. First we coded custom train and predict functions for convenience sake. The image loader had a section to count the number of masks in the labels which was poorly coded hence extremely slow despite the use of multiprocessing, so we coded a single function that counts the masks 100 times more efficiently (if the

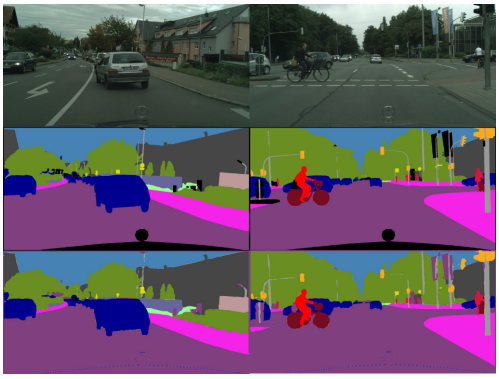

Figure 7. Top: original image, Center: true mask, Bottom: DeepLabV3 prediction

mask labels are known it can be skipped). We rescaled all images by a factor of 0.5, and we used the 2975 training images for training. We randomly shuffled with batch norm = 1, Adam optimizer, 10^{-3} learning rate (with ReduceLROnPlateau scheduler), 10^{-8} weight decay, Cross Entropy loss summed with Dice loss. The dataset was initially restricted to 19 classes. Furthermore, we implemented early stopping but we didn't use it for the final evaluation.

Despite his strong success in medical imaging field, the U-Net architecture is notably not great in segmentation tasks such as the one presented by the Cityscapes dataset, as documented here: [UNet test][5] (the best UNet variant is well below the best DeepLabV3 variant and the best DenseASPP variant in the ranking). This is also showed in the results we got, which were poor and not mirroring the potential of the UNet. So for the sake of showing the effectiveness of this algorithm we gave it an advantage: we decided to merge some of the less common classes with analogous classes to balance the labels. In the end we used for the final training the 10-class colormap shown in Figure 8.

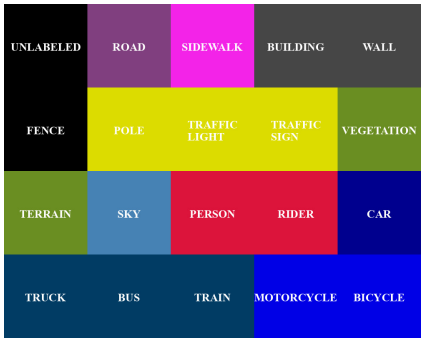

Figure 8. The 10 classes (colors) identified by U-Net

The results were surprisingly good at the end. We show some examples from the validation set of Cityscapes dataset in Figure 9.

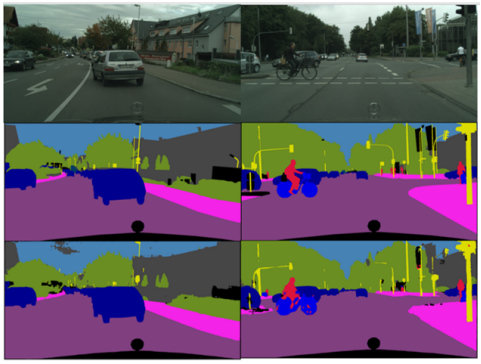

Figure 9. Top: original image, Center: true mask, Bottom: U-Net prediction

5.3. DenseASPP

In this case we already had a Python implementation and a model pretrained on Cityscapes from [DenseASPP link][3].

The challenge in this case was adapting the code to the newer versions of Python and PyTorch, since the code is 5 years old and uses PyTorch 0.3.1 and Python 3.7. After unsuccesfully trying to create an environment analogous to the one they used, we settled on updating the code and managed to run it with PyTorch 2.0 (current latest version) and Python 3.10.10. The biggest modifications consisted in creating an external function to change some names in the pretrained weights dictionary which had some characters that weren't accepted (and accordingly correct any part of code referring to those names); furthermore we had to update all the obsolete functions with modern ones.

For DenseASPP we use the full set of 19 classes (see Figure 10).

Some sample results are shown in Figure 11.

Figure 10. The 19 classes (colors) identified by DeepLabV3 and DenseASPP

6. Conclusion

We present the results of the different models on the Cityscapes validation dataset in Table 2. We computed the

	Truck	Car	Motorcycle	Bicycle	Bus	Train	Building	Rider	Person	Road	Wall	Sky	Vegetation	Sidewalk	Pole	Fence	Traffic Light	Traffic Sign	Terrain
DenseASPP	0.21	0.84	0.20	0.48	0.38	0.13	0.81	0.42	0.55	0.78	0.19	0.78	0.85	0.67	0.52	0.22	0.38	0.58	0.29
DeepLabV3	0.11	0.88	0.07	0.44	0.20	0.04	0.90	0.29	0.51	0.71	0.27	0.74	0.90	0.85	0.57	0.27	0.29	0.57	0.32

Table 1. IOU of the models for every class. mIOU DenseASPP = 0.52, mIOU DeepLabV3 = 0.45

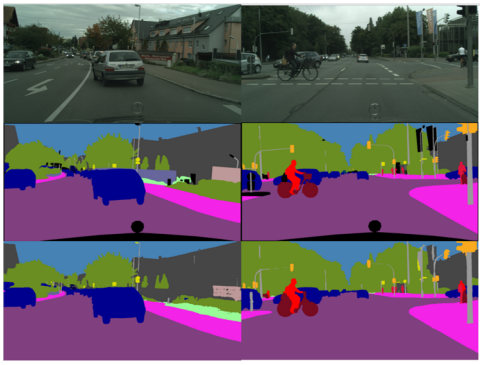

Figure 11. Top: original image, Center: true mask, Bottom: DenseASPP prediction

IOU with the function of the Jaccard Similarity from sklearn using different parameters for the 'average' setting.

'Micro average' calculates the Jaccard score by considering all the samples and their respective true and predicted labels as a single binary classification problem. It sums up the true positive, false positive, and false negative counts across all classes and then computes the Jaccard score, giving equal weight to each sample.

'Macro Average' calculates the Jaccard score separately for each class and then takes the average across all classes, treating each class equally regardless of the class's sample size or prevalence in the dataset. This approach can be useful to evaluate the performance of the model for each class individually and then average the scores across all classes.

'Weighted Average' calculates the Jaccard score for each class and then takes the average, weighted by the the number of true instances for each class. This gives higher weight to classes with more samples, providing a balance between the macro and micro averages. This approach is suitable when the dataset is imbalanced, as it considers both the performance of each class and their prevalence in the dataset.

Furthermore, we defined our personal metrics 'PerPixel' which simply compares if the pixels of the predicted mask correspond to the pixels of the true mask.

The three models have very similar performances using all the metrics except for when using 'Macro Average', when the performances drop for all the models. This can be explained by looking at the Table 1 where we have reported the meanIOU for each possible different class. In the table we can notice the drop of the mean is due to the performances of the models on 'Train', 'Truck' and 'Motorcycle', probably because these types of items are less common to appear in the images.

	DeepLabV3	Unet	DenseASPP
# classes	19	10	19
PerPixel	0.85	0.85	0.84
Micro	0.75	0.75	0.73
Macro	0.45	0.38	0.52
Weighted	0.75	0.78	0.74

Table 2. Results of the models computed using different Jaccard similarity scores

Unfortunately we could not apply this technique to U-Net since this model was adapted to work with just 10 classes instead of 19.

Despite this performance decline, it can be seen that DenseASPP performs better than Unet, while DeepLabV3 is in between.

The common feature with DeepLabV3 and DenseASPP is the use of the ASPP, so we can say that its use in itself is effective, but the addition of dense connections within the ASPP gives a significant improvement.

References

[1] Cityscapes dataset: https://www.cityscapes-dataset.com/.

[2] Cityscapes labels: https://github.com/mcordts/cityscapesscripts/blob/master/cityscapesscripts/helpers/labels.py.

[3] Denseaspp link: https://github.com/deepmotionairesearch/denseaspp.

[4] Unet link: https://github.com/milesial/pytorch-unet.

[5] Unet test: https://paperswithcode.com/ sota/semantic-segmentation-on-cityscapes-val.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele Jonathan Long, Evan Shelhamer, and Trevor Darrell. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[12] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. *CoRR*, 2018.