Semantic Segmentation in the Traffic Environment using the DeepLabv3+ Model

Sheng Li
Stanford University
Stanford, CA
lisheng@stanford.edu

Jin Woo Park
Stanford University
Stanford, CA
jinwoop@stanford.edu

Soyeon Jung
Stanford University
Stanford, CA
soyeonj@stanford.edu

Abstract

Semantic segmentation in the traffic environment is an important perception task for autonomous driving. This project performed semantic segmentation on the Mapillary Vistas Dataset [13], a novel street scene dataset containing 66 object categories. We conducted transfer learning based on the DeepLabv3+ [6] (link [1]) developed by Google TensorFlow. The DeepLabv3+ model employs the spatial pyramid pooling module and the encode-decoder structure. Our fine-tuned model is able to score 23.6% mIoU on the test set of the Mapillary Vistas Dataset (being able to rank No.6 on the MVD Challenge Leaderboard as of Jun. 7, 2018).

The code of the project based on the DeepLabv3+ code is available on GitHub [2].

1. Introduction

Deep convolutional neural networks (CNN) has achieved success in various computer vision tasks including semantic segmentation. Semantic segmentation is targeted towards partitioning the image into semantically meaningful parts with various applications [16]. Different from object detection, semantic segmentation is more challenging since it involves labeling each pixel in the image to some class. Sharp and accurate edges between classes are usually hard to achieve. It is therefore often computational intensive to train a semantic segmentation model. Semantic segmentation is an important task for autonomous driving, which has been applied in lane, vehicle, pedestrian and obstacle detection. It would be meaningful to build a CNN model that is able to effectively perform semantic segmentation in the traffic environment.

The DeepLabv3+ [6] model is the state-of-the-art semantic segmentation model developed by the Google TensorFlow team (ranking number one on PASCAL VOC 2012 Challenge Leader Board as of Feb 9, 2018 [1]). It is a CNN model that employs the spatial pyramid pooling module and the encode-decoder structure. The spatial pyramid pooling module is able to encode multi-scale contextual information by probing the incoming features with pooling operations at multiple rates and multiple effective fields-of-view, while the encode-decoder structure can capture sharper object boundaries by gradually recovering the spatial information [6]. Detailed technical approach is introduced in Section 3.

We performed transfer learning from the DeepLabv3+ by fine-tuning the model and retraining its classifier on the Mapillary Vistas Dataset [13], which is a street scenes specific dataset. The initial model checkpoint candidates are the DeepLabv3+ model pretrained on the PASCAL VOC 2012 Dataset [9] and the model pretrained on the Cityscapes Dataset [8].

The input to the DeepLabv3+ model is RGB images (in jpg format) of arbitrary size. The output prediction of the model is gray scale images (in png format) of the same size as the input with each pixel directly indexed by the class number. For a better visual effect, the output can be further rendered with colormap that designates vivid and distinct colors for different classes.

The dataset we used is the Mapillary Vistas Dataset [13]. It is a large-scale street-level image dataset containing 20,000 high-resolution and academically available images annotated into 66 object categories. We extracted images and labels from the original dataset to form smaller subsets to speed up training process. Detailed introduction to the dataset is covered in Section 4.

Experiments were performed on multiple initial model checkpoints. Hyperparameter sweeping was conducted in search of better models. Results shows that the best model from transfer learning can perform fairly satisfying semantic segmentation on the Mapillary Vistas Dataset. Main objects in the traffic environment can be clearly identified. Experiment setup, hyperparameter sweeping, results and failure case analysis are covered in Section 5.

[1] DeepLabv3+ code by Google TensorFlow is available at https://github.com/tensorflow/models/tree/master/research/deeplab

[2] Project code is available at https://github.com/parachutel/deeplabv3plus_on_Mapillary_Vistas

2. Related Work

Before deep learning techniques became widely in use, semantic segmentation was done with flat classifiers such as Random Forest [15] and Support Vector Machine [11]. With the success of deep learning techniques on other higher-level computer vision tasks (e.g. the image classification and the object detection), researchers adopted the classification networks (e.g. CNNs) to semantic segmentation. These networks worked successfully with the advantages of having feature representations learned by itself without a hand-crafted feature set which requires domain-specific knowledge and a lot of fine-tuning.

The key difference of semantic segmentation from tasks like classification and object detection is that each pixel is labeled with the class of its enclosing object or region. One of the earliest popular approaches was the patch classification [7, 10], where each pixel is labeled with a class using a patch of image around it. The reason for using patches was that the fully connected layers of the CNNs require fixed size images.

Long *et al.* proposed Fully Convolutional Networks (FCNs) [12], which replace fully connected layers with convolutional layers and upsample using deconvolutions to obtain spatial heatmaps. This enabled dense predictions for semantic segmentation on images of any size. FCNs significantly improved the segmentation accuracy, and most of the state-of-the-art approaches stem from this paradigm.

While FCN model has its power and flexibility, it still struggles with an issue arising from the pooling layers. The pooling layers, which induce the spatial invariance and decrease the resolution of CNNs, do not align with semantic segmentation task, where the exact spatial information has to be preserved. One of the approaches that can handle this problem is using dilated convolutions, also called the atrous convolution in DeepLab [17]. This enables enlarging the field of view while preserving the spatial dimensions. Another approach is incorporating a post-processing stage using Conditional Random Fields (CRFs). CRFs allow the model to consider both short and long range interactions between pixels. Still, there are other approaches such as multi-scale aggregation or trying different models like Recurrent Neural Networks (RNNs).

In 2015 and 2016, Chen *et al.* proposed v1 and v2 of the DeepLab (Deep Labeling) model [3, 4]. Both models used the fully connected CRF and hole algorithm / atrous convolutions. There are two main differences. Firstly, the multi-scale processing is achieved by using either image pyramid (DeepLabv1) or atrous spatial pyramid pooling (ASPP) (DeepLabv2). Secondly, DeepLabv2 improved the performance by adopting the ResNet-101 as its feature extractor, not the VGG-16 as in DeepLabv1. They achieved 71.6% and 79.7% mIOU (mean intersection-over-union), respectively, on the PASCAL VOC 2012 dataset.

DeepLabv3 [5], published in 2017, improved the performance to 85.7% mIOU on the PASCAL VOC 2012 Dataset without using CRF post-processing. It introduced modules which employ atrous convolutions in cascade or in parallel. It also improved ASPP by concatenating image-level features and operating batch normalization. In 2018, Chen *et al.* proposed v3+ model [6] where they introduced an encoder-decoder structure. It used DeepLabv3 model as the encoder module and add a new decoder module. Also, it adopted the Xception model as its backbone and applied depth-wise separable convolution to both the ASPP and decoder modules. The performance was improved up to 89% on the PASCAL VOC 2012 Dataset.

3. Technical Approach: DeepLabv3+

In this paper, we adopt Google's DeepLabv3+ [6] model for semantic image segmentation. Fig. 2 shows the network architecture of the DeepLabv3+ model. This model is extended from the DeepLabv3 [5], which uses ImageNet pre-trained ResNet as the feature extractor, with a new residual block for multi-scale feature learning. DeepLabv3+ mainly consists of four parts: atrous convolution, atrous spatial pyramid pooling (ASPP), multi-grid method, and the encoder-decoder modules. In the following text, we will provide a detailed explanation on each of them.

3.1. Atrous Convolution [5]

Atrous convolution, also called dilation convolution, works the same way as regular convolutions, except that it has atrous rate, i.e., dilation rate. The atrous rate r is the stride applied to the filter when sampling the input signal. Implementing larger rates expands the filter's field-of-view, and therefore enables the filter to capture objects in different scales. Note that the standard convolution corresponds to atrous convolution with $r = 1$. Figure 1 shows an atrous convolution with rates of 1, 6 and 24 on a 3×3 filter. Note that the standard convolution corresponds to atrous convolution with $r = 1$.

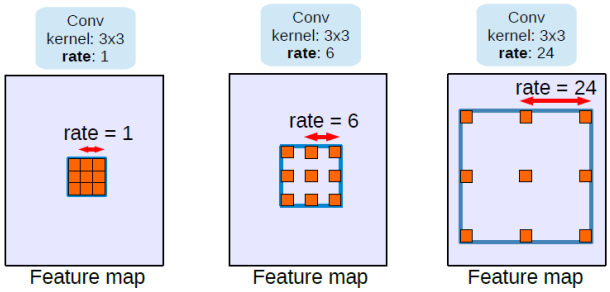


Figure 1: Atrous convolution with various atrous rates on a 3×3 filter, i.e., kernel [5].

Consider a 2-dimensional input feature map x and a filter

w with the atrous rate r. Then, the value for the ith location of the output y is computed as follows

$$y[i] = \sum_k x[i + r \cdot k]w[k].$$

This implies the unsampled filter filled with $r - 1$ zeros between filter values is convolved with input x. Hence the atrous convolution allows to adaptively alter filter's field-of-view by changing the dilation rate, r. Furthermore, atrous convolution enables us to compute feature responses without requiring learning parameters. The layer before FC layers or global pooling (i.e., final feature responses of DCNNs) is 32 times smaller than the size of input image for this model which corresponds to $output\_stride = 32$. The $output\_stride$ is the ratio of input image spatial resolution to the final output resolution (before FC layer). The spatial density could be quadrupled ($output\_stride = 8$) by setting stride to 1 for the last convolution or pooling layer to prevent vanishing signal and perform atrous convolutions with $r = 2$ and $r = 4$ for subsequent convolution layers.

The consecutive striding without atrous convolution negatively affects the semantic segmentation, because the spatial information, which is crucial for the purpose, is decimated after consecutive convolution due to reduced dimension. However, the atrous convolution captures long range information in the deeper blocks, thus it is crucial for DCNNs.

3.2. Atrous Spatial Pyramid Pooling (ASPP) [5]

In order to robustly classify objects of an arbitrary scale, atrous spatial pyramid pooling (ASPP) applies multiple parallel atrous convolutions with different atrous rates with a batch normalization. DeepLabv3+ applies four parallel convolutions, where each has a different atrous rate and they are applied on top of the feature map. It also includes batch normalization.

Although ASPP captures multi-scale information with different atrous rates, it experiences a decrease in the number of weights applied to the valid feature region as the atrous rate increases. For example, applying 3×3 atrous filter with $r = 21$ on 65×65 feature map (i.e., the filter has the same size as feature map), then filter acts as a simple 1×1 filter, which results in only center filter weight being effective. It is not capable of capturing the whole image context.

To overcome aforementioned problem, ASPP module can be augmented with image-level features to incorporate more global context information to the model. It first applies global average pooling (GAP) to the output features of the last atrous block, feeds the resulting features into a 1×1 convolution with 256 filters, performs batch normalization, then bilinearly upsamples the features to the desired spatial dimension. All the resulting features from different branches are concatenated into a single vector, fed into a 1×1 convolution with 256 filters and a batch normalization, and finally another 1×1 convolution which generates the final segmentation logits. For $output\_stride = 16$, ASPP consists of 1×1 convolution and three 3×3 convolutions with $rates = (6, 12, 18)$ (shown in Figure 3). The rates are doubled for $output\_stride = 8$.

3.3. Multi-grid Method [5]

"DeepLabv3" implements different atrous rates within three consecutive blocks from block4 to block7. The unit rates for three convolutional layers are defined as $Multi\_Grid = (r_1, r_2, r_3)$. The rates are chosen such that the atrous rate for the convolution layer, $output\_stride$, is equal to the product of unit rate (i.e., $Multi\_Grid$) and corresponding rate of 2. For example, whe $output\_stride = 16$ and $Multi\_Grid = (1, 2, 4)$, then three convolutions will have $rates = 2 \cdot (1, 2, 4) = (2, 4, 8)$ respectively in each block.

3.4. Encoder-Decoder [6]

The semantic segmentation is slightly different from a typical image classification. The model is determining every pixel on image to one of the pre-determined classes. Thus the resulting layer is a tensor with shape of [W, H, 1] for the input image shape of [W, H, 3] preserving the input dimension. The DCNN implements convolution striding and pooling operations to downsample neural nets which outputs a dense (non-spatial) vector only containing the probabilities for each pre-determined classes. Hence transpose convolution is required to increase the spatial resolution (i.e., upsample) to achieve the same dimension as input. The upsampling is basically transposing deep and narrow layers to wider and shallower layers. The encoder and decoder are the terms equivalent to downsampling and upsampling respectively.

DeepLabv3+ implements atrous convolution to extract abitrary resolution of features from the DCNN. The typical $output\_stride$ is 32 and atrous convolutions are applied in last one or two blocks for adopting $output\_stride = 16$ or 8 respectively. For $output\_stride = 8$, last two blocks apply atrous rate of 2 and 4 respectively. In the end, the encoder output feature map contains 256 channels and rich semantic information.

The encoder features of DeepLabv3+ are typically computed with $output\_stride = 16$. Hence the features have to bilinearly upsampled by a factor of 16 (i.e., naive decoder), but this may not successfully recover object segmentation details. Thus, the model proposes an effective decoder that first bilinearly upsampels by a factor of 4 and then concatenate with corresponding low-level features with the same spatial resolution from the network backbone. Consecutively, the model applies 1×1 convolution on the low-level

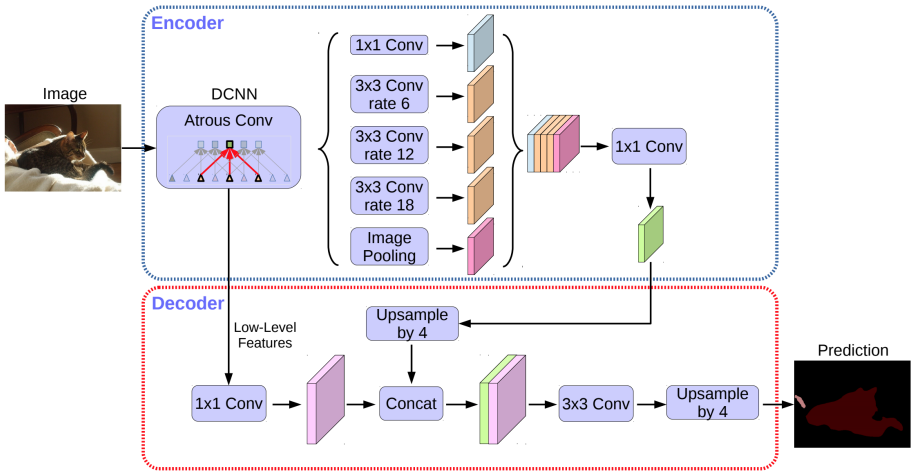


Figure 2: DeepLabv3+ Network Architecture [6].

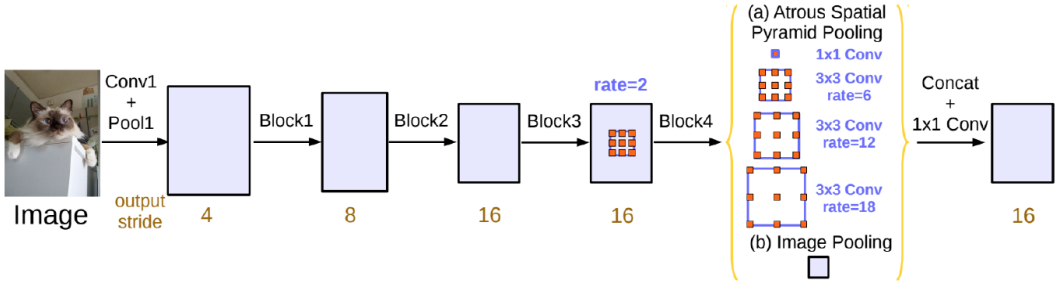


Figure 3: ASPP module, augmented with image-level features [5].

features to reduce the number of channels followed by concatenation, a few 3×3 convolutions, and another bilinear upsampling by a factor of 4 (shown in Figure 2). Chen *et al.* assert that encoder module using $output\_stride = 16$ shows the best trade-off between speed and accuracy. The performance improvement for $output\_stride = 8$ is marginal given the cost of extra computation complexity.

4. Mapillary Vistas Dataset

We performed semantic segmentation on the Mapillary Vistas Dataset [13]. It is a novel, large-scale street-level image dataset containing 20,000 high-resolution and academically available images annotated into 66 object categories. The dataset is $5\times$ larger than the total amount of fine annotations for Cityscapes and contains images from streets all around the world, captured at various conditions regarding weather, season and daytime. Images come from different imaging devices (mobile phones, tablets, action cameras, professional capturing rigs) and differently experienced photographers. In such a way, the Mapillary Vistas Dataset has been designed and compiled to cover diversity, richness of detail and geographic extent. As default benchmark tasks, developers of the dataset define semantic image segmentation, aiming to significantly further the development of state-of-the-art methods for visual road-scene understanding [13].

An example of the semantic segmentation label from the Mapillary Vistas Dataset is presented in Fig. 4. An example of the semantic segmentation overlaying with the original image is shown in Fig. 5.

(a) Image (b) Ground Truth Label

Figure 4: An example of an image and its complex semantic segmentation label from the Mapillary Vistas Dataset. Note that every pixel is labeled with a class specified by different colors according to the colormap prescribed by Mapillary.

Figure 5: An example of semantic segmentation overlaying with the original image from the Mapillary Vistas Dataset.

4.1. Dataset Splits

The Mapillary Vistas Dataset has 18,000 images/labels in its original training split and 2,000 images/labels in the original validation split. In order to speed up training, we sampled images from the original training split to form our training and validation set. We then sampled images from the original validation set to form the test set. The number of images in the used dataset and their resolutions are listed in Table 1.

4.2. Data Augmentation and Label Preprocessing

The input images and labels to the model are augmented by random scaling. The scaling factor is within the range of $0.5 \sim 2$. With the scaling factor step size of 0.25, the scaling factor range is discretized. For every input feed to the model, we randomly sample a scaling factor from the range and rescale the images and their labels.

The labels are preprocessed to gray scale images so that each pixel is directly indexed by the class number from 0 to 65. For instance, Fig. 4 after being preprocessed by removing the colormap is shown in Fig. 6.

Figure 6: An example of the preprocessed ground truth label (Fig. 4). The preprocessed label is in gray scale.

5. Experiments and Results

Since the DeepLabv3+ model has a massive amount of parameters, we use the pretrained models provided by Google TensorFlow Team [2] as our initial checkpoints for transfer learning. To be specific, we rely on the two initial checkpoints that were first pretrained on the ImageNet Dataset [14] and then further trained on the Cityscapes Dataset [8] and the PASCAL VOC 2012 Dataset [9]. Heuristically, the initial model pretrained on the Cityscapes Dataset is used more often since the contents of Mapillary Vistas Dataset are similar to those of the Cityscapes Dataset (both being street level images).

We modified number of channels of the final layer of the DeepLabv3+ model, i.e., the classifier, so that it is compatible with number of classes of the Mapillary Vistas Dataset.

5.1. Performance Evaluation Metrics

The main metric used for semantic segmentation is the mean intersection over union (mIoU) score [9], also known as averaged Jaccard index. IoU is defined as $TP/(TP + FP + FN)$, where TP, FP and FN are true positives, false positives and false negatives, respectively. The mIoU score can be calculated throughout all the classes as well as separately for each class. For the current stage of the project, we use the mIoU score across all the classes.

5.2. Hyperparameters

Hyperparameter sweeping was performed in the experiments. Important hyperparameters as well as the reasoning of how the values were determined are listed:

- **Base learning rate**: since we are performing transfer learning on a fairly different dataset from the datasets the model is pretrained on, a large starting learning rate is preferred. We sweep between 1×10^{-4} and 5×10^{-3}, results shows that larger learning rate typically gives better performance.

- **Learning rate decay rate**: a linear learning rate decay schedule is used in the implementation of the model. The learning rate is multiplied by 0.9 very x steps, where x varies from 200 to 1000. For larger learning rate, a faster decay rate is applied (i.e., a smaller x).

- **L2 regularization strength**: due to the nature of the semantic segmentation task, i.e., predicting labels for each pixel, the risk of overfitting is relatively low when comparing with simpler tasks. Therefore, we swept low regularization strength values from 1×10^{-5} to 4×10^{-5}.

- **Batch size**: according to the notes in the original DeepLabv3+ code [6] commented by its original authors, a large batch size is essential when performing transfer learning on different datasets so that the batch normalization parameters can be effectively tuned. The recommended batch size is at least over 12. Due to the large number of classes of the Mapillary Vistas Dataset, one Nvidia Tesla K80 GPU with 11GB memory can only process 4 images in a mini-batch. There-

Dataset	# Training Images	# Validation Images	Resolution Range
Small train/val	2533	281	$640 \leq w \leq 2700, 480 \leq h \leq 2700$
Medium train/val	6844	760	$w = 3264, 2176 \leq h \leq 2448$
Large train/val	12323	1367	$640 \leq w \leq 4000, 480 \leq h \leq 4000$
Test	331		$800 \leq w \leq 3072, 600 \leq h \leq 2976$

Table 1: Dataset split and the corresponding resolution.

fore, we typically used 4 K80 GPUs to train a batch size of 16.

- **Optimizer**: we used the default momentum optimizer with momentum $= 0.9$ as provided by the original DeepLabv3+ code [6] since it has fairly reliable performance.

5.3. Baseline

According to the experiments and the corresponding results demonstrated in the DeepLabv3 [5] and DeepLabv3+ [6], they mainly based on the PASCAL VOC 2012 Dataset. We believed that training the Mapillary Vistas Dataset based on the initial checkpoint pretrained on the PASCAL VOC 2012 Dataset could set a reasonable baseline. Two experiments were run based on the PASCAL VOC 2012 initial checkpoint using the medium split of the Mapillary Vistas Dataset. Their hyperparameter settings and mIoU results on the validation sets are listed in Table 2.

5.4. Device

We used 1 Tesla K80 GPU for a minibatch of 4 images, and 4 Tesla K80 GPUs for a minibatch of 16 images. The whole training/evaluation/visualization process was completed on Google Could Platform.

5.5. Results and Discussion

Experiments were run on the Cityscapes Dataset pretrained initial checkpoint. The experiments covered all the datasets (small/medium/large); and we mainly focused on the medium dataset due to the limitation on the computational resources and the time available. We performed hyperparameter sweeping on the base learning rate, the learning rate decay rate, as well as the regularization strength. The hyperparameter settings and the results of the experiments are listed in Table 2.

We discovered that the learning rate played an essential role in lowering the loss effectively and improving the prediction performance on the validation set. When using a low learning rate (in the order of 10^{-4}), the learning curve (the loss curve) tend to oscillate and easily plateau. The resulting mIoU scores are therefore typically low (see Exp. 8, 9, 10, 12 in Table 2). When the learning rate increases to the order of 10^{-3}, the mIoU score increases accordingly (see Exp. 2, 11, 13, 14, 15 in Table 2). Fig. 7 compares the loss histories of two experiments (Exp. 10 and 13) with same regularization strengths but different learning rates (the loss histories are smoothed by the exponentially weighted moving average (EWMA) with the smoothing strength = 0.95). The loss histories show that the experiment with the higher learning rate can reach a lower loss.

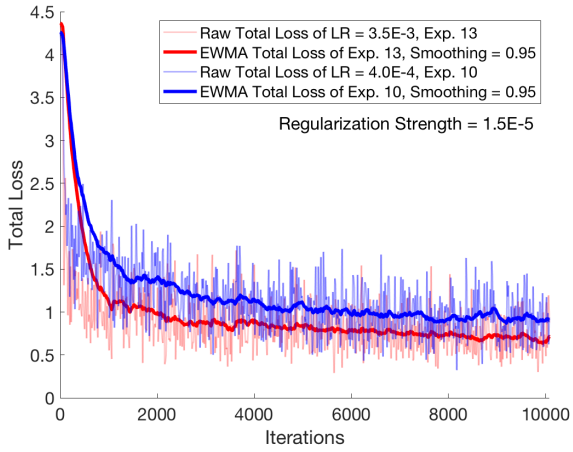


Figure 7: The loss histories of Exp. 10 and 13. Training process with lower learning rate plateaus earlier.

Due to the extremely high complexity of the features of the input (i.e., pixel-wise labels and large number of classes), it is almost impossible to overfit the model to the training set. We discovered that a high regularization strength did not help increase the validation performance. The best model (Exp. 15 in Table 2) is trained with the lowest regularization strength.

The performance of the baseline models and some best models from experiments on the test set are listed in Tabel 3.

5.6. Prediction Demonstration and Failure Analysis

To better understand the performance of the model, we visualized the prediction results of the model. Fig. 9 shows a comparison between the original images, the ground truth labels, the prediction of the baseline, the prediction of the second best model (from Exp. 13), and the prediction of the best model (from Exp. 15).

Judging by visual effects, the baseline model based on

Exp. No.	Initial Checkpoint	Base Learning Rate	LR Decay Steps	Reg. Strength	Batch Size	Dataset	mIoU (%)
Baseline 1	PASCAL VOC 2012	3.50E-03	250	1.50E-05	16	Medium	13.9
Baseline 2	PASCAL VOC 2012	5.00E-03	200	1.00E-05	16	Medium	**17.1**
1	Cityscapes	2.50E-03	500	2.00E-05	4	Small	20.7
2	Cityscapes	2.00E-03	500	2.00E-05	16	Small	**22.4**
3	Cityscapes	2.50E-03	500	1.50E-05	16	Small	21.9
4	Cityscapes	1.00E-03	500	1.50E-05	16	Small	16.8
5	Cityscapes	1.00E-04	1000	4.00E-05	16	Large	8.03
6	Cityscapes	8.00E-04	1000	4.00E-05	16	Large	**15.8**
7	Cityscapes	2.50E-03	500	2.00E-05	16	Medium	20.6
8	Cityscapes	1.50E-03	1000	1.50E-05	16	Medium	14.5
9	Cityscapes	8.00E-04	1000	1.50E-05	16	Medium	15.5
10	Cityscapes	4.00E-04	250	1.50E-05	16	Medium	15.9
11	Cityscapes	2.00E-03	500	2.00E-05	16	Medium	22.0
12	Cityscapes	1.00E-04	1000	2.50E-05	16	Medium	10.3
13	Cityscapes	3.50E-03	250	1.50E-05	16	Medium	23.4
14	Cityscapes	3.00E-03	500	1.50E-05	16	Medium	22.8
15	Cityscapes	5.00E-03	200	1.00E-05	16	Medium	**24.5**

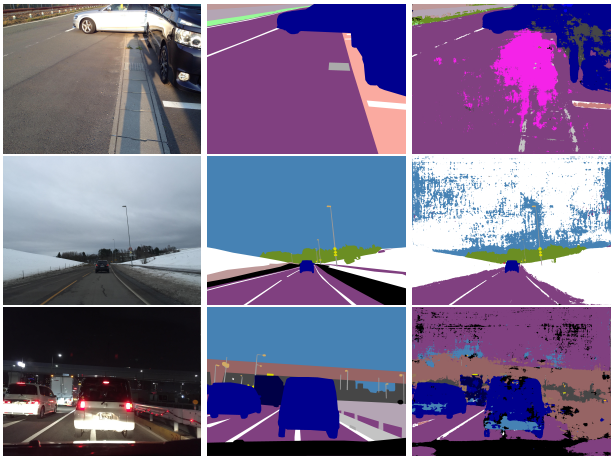
Table 2: Experiment settings and results split by the dataset size and the initial checkpoint. Note the mIoU's listed are obtained on the validation split of each dataset.

Exp. No.	mIoU on Val. Set (%)	mIoU on Test Set (%)
Baseline 1	13.9	11.9
Baseline 2	17.1	**13.2**
13	23.4	20.0
15	24.5	**23.6**

Table 3: Performance of the baselines and a selection of the best models on the test set.

the PASCAL VOC 2012 pretrained initial checkpoint performs poorly in prediction in most of the cases. While the best model based on the Cityscapes pretrained checkpoint from Exp. 15 shows a clearly better performance in general. The best model performs well in labeling the large-continuous regions, e.g. the sky, the building blocks, the vehicles and most of the road surfaces. However, it has weakness in identifying details like the walkway, the traffic lights and the pedestrian crossings. They often got mixed with the background on the boundaries.

The prediction failed dramatically when some extreme cases are encountered. As shown in Fig. 8, the best model failed in some scenarios. For example, when the highlight and shadow present with a high contrast, when there are snow on the ground making the ground have the same texture and color as the sky, and when it is dark at night. For the first failure case, the model predicted the shadow as a walkway. In the latter two failure cases, the model labeled the sky as the road/ground due to the small difference between them in the sense of pixel.

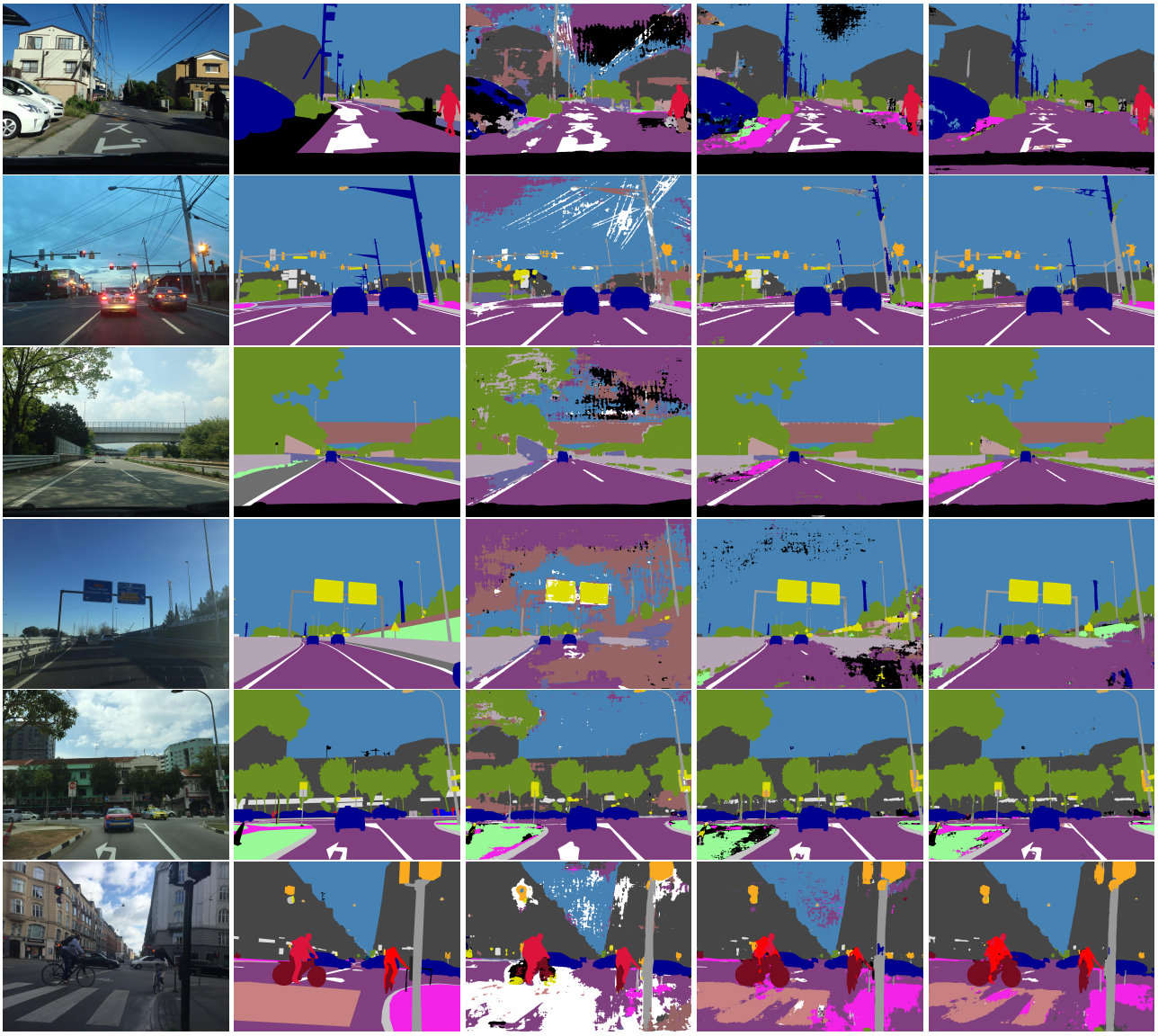


(a) Image (b) Ground Truth (c) Pred. (Exp. 15)

Figure 8: Typical failure analysis for the best model. 1. shadow and highlight; 2. snow on the ground; 3. night scene

6. Conclusion and Future Work

The project implemented transfer learning based on the DeepLabv3+ model by Google for performing semantic segmentation on the Mapillary Vistas Dataset. Hyperparameter sweeping. The best model results from high learning rate and low regularization strength mainly due to the high complexity of the pixel-wise features. The best model is able to achieve 24.5% mIoU on the validation set and 23.6% mIoU on the test set. The visualizations of the pre-



(a) Image (b) Ground Truth (c) Baseline 2 (d) Second Best Model (e) Best Model

Figure 9: A comparison between (a) the original images from the Mapillary Vistas Dataset, (b) the ground truth labels and (c) the predictions from baseline 2 model, (d) second best model from Exp. 13 and (e) the best model from Exp. 15.

diction output show clear blocks and sharp edges for major parts of the images. However, there are some typical failure cases such as highlight, shadow, covering snow and night scene, which lead to poor segmentation quality. The issue is expected to be resolved by increasing the size of training set and training for longer time as future work.

We will try more hyperparameter sweeping, training with the full dataset as well as trying more data augmentation scaling schemes. In particularly, we are interested in participate in the Mapillary Vistas Dataset Semantic Segmentation Challenge using the fine-tuned DeepLabv3+ model. In addition, the Pyramid Scene Parsing Network (PSPNet) [18] is worth a try on the Mapillary Vistas Dataset. It is a recently published network structure exploiting different-region-based context aggregation through pyramid pooling [18]. The PSPNet shows cutting edge performance on semantic segmentation.

7. Contributions

S.L., setup the model for transfer learning and input preprocessing routines. S.L. and J.P., performed hyperparameter sweeping and training. S.J. did literature review. S.L., J.P. and S.J wrote the paper.

References

[1] Pascal voc 2012 semantic segmentation competition leader board, Feb 2018.

[2] Tensorflow deeplab model zoo, 2018.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.

[4] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018.

[7] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[11] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[12] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[13] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy*, pages 22–29, 2017.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[15] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[16] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. *arXiv preprint arXiv:1707.02432*, 2017.

[17] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[18] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.