Formatting Instructions For NeurIPS 2020

Lingxi Li **Yejun Li** **Jiawen Zeng** **Yunyi Zhang** **Zachary Higgins**

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Semantic segmentation aims at assigning a categorical label to every pixel in an image, which is important in self-driving systems and image understanding[1]. One key issue of this task is to make use of local information, which indicates an object's position, and global information, which indicates an object's type [2]. A deep FCN [2] encodes the local and global information in a local-to-global pyramid, which is suitable for this problem. U-Net [3] is an alternative for FCN, it replaces pooling by upsampling and increases the resolution of output.

This project applies *India Driving Dataset* to perform the semantic segmentation task. We implement different network structures(e.g., FCN, U-Net and CNN) and use figure augmentation/transformation learning to improve the performance of networks.

We use Xavier uniform weights [4] to initiate the networks: $W = Unif(-a, a)$, here $a = \sqrt{6/(fan_{in} + fan_{out})}$, fan_{in} is the size of input layer; and fan_{out} is the size of output layer. By assuming the weights are initialized independent and the input has variance v, we have $Var(z_i) = v\prod_{j=0}^{i-1} fan_{in,j}Var(W_j)$; and $Var\left(\frac{Cost}{\partial a_i}\right) = Var\left(\frac{Cost}{\partial a_d}\right) \times \prod_{j=i}^{d} fan_{out,j}Var(W_j)$, here d is the depth of the network. This initiation procedure helps maintain $fan_{in,j}Var(W_j) \approx 1$ and $fan_{out,j}Var(W_j) \approx 1$, which keeps the activation variance and back-propagated gradients' variance as 1 when one performs forward and backward propagation.

We use batch normalization [5] to accelerate training: suppose the input of a network is $z = (z_1, ..., z_n)^T$, we normalize each dimension of the input by $\widehat{z}_i = (z_i - \mathbf{E}z_i)/\sqrt{Var(z_i)}$, and feed the network by $y_i = \gamma_i\widehat{z}_i + \beta_i$. Here $\gamma_i, \beta_i, i = 1, 2, ..., n$ are learned during training. This transformation makes training resilient to the parameter scale and allows high learning rate.

The rest of this report is organized as follows: section 2 introduces related works; section 3 describes the implementation details of proposed networks; section 4 proposes the experiment results; and section 5 makes discussion and conclusion.

2 Related work

Network architectures for semantic segmentation: Long et al. [2] introduced a fully connected network which achieved about 63% mean IoU on the PASCAL VOC 2011 dataset. Noh et al. [6] combined a convolution network with a deconvolution network and improved the mean IoU to about 70% on PASCAL VOC 2012 data set. Wang et al. [1] adopted ResNet for encoding features and dense upsampling convolution layers for decoding; they got 83% mean IoU on the PASCAL VOC 2012 dataset. Ronneberger et al. [3] proposed U-Net for biomedical image Segmentation. They allowed the network to propagate context information to the high resolution layers.

Address the imbalanced class problem: class imbalance brings two issues: 1. inefficient training, and 2. model degeneration. A common way to solve class imbalance is to down-weight the loss of well-classified examples. For example, Lin et al. [7] proposed focal loss to improve the improve the cross entropy loss. We refer Jadon [8] for a summary of loss functions in semantic segmentation.

3 Methods for experiments

3.1 General setting

This report applies average pixel accuracy and average IoU to evaluate the performance of a network. We calculate per-class IoUs and average all IoUs to get the average IoU. Without specified, we use mini-batch Adam with a learning rate 0.001 and batch size 32 to train a network. We center crop input figures to 256×256 pixels to adapt the environment. We use early stopping to store the best model.

3.2 Baseline model

The baseline model uses the architecture shown in table 1. In the last layer, we use the softmax activation function(i.e. $\sigma_i(x_1, ..., x_n) = \exp(x_i)/\sum_{j=1}^{n} \exp(x_j), i = 1, 2, ..., n$). We apply the cross entropy as a baseline loss criterion. Then we use the weighted cross entropy to solve the imbalance class problem. The weight for each category $w = (w_1, ..., w_{27})^T$ is given by

$$w_i = \frac{1/\sqrt{a_i}}{\sum_{j=1}^{27} 1/\sqrt{a_j}}, i = 1, 2, ..., 27 \tag{1}$$

here a_i is the number of pixels in the training set belonging to category i. In the experiment, this loss criterion down-weights the loss of well-labeled categories, but does not invalidate the loss of them.

We also augment the dataset to improve the performance of the baseline model. In our setting, each image has identical and independent probability for 1) random 10 degree rotation. 2) horizontal flip. 3) random crop at a different location.

Table 1: Architecture of the baseline model, '/' in padding means 'padding/output padding'

Layer	Dimensions						
	Type	in channels	out channels	kernel size	padding	stride	Activation
1	Conv2d	3	32	3	1	2	
2	BatchNorm2d	32	32				
3	Conv2d	32	64	3	1	2	
4	BatchNorm2d	64	64				
5	Conv2d	64	128	3	1	2	
6	BatchNorm2d	128	128				
7	Conv2d	128	256	3	1	2	
8	BatchNorm2d	256	256				
9	Conv2d	256	512	3	1	2	
10	BatchNorm2d	512	512				ReLU
11	ConvTranspose2d	512	512	3	1/1	2	
12	BatchNorm2d	512	512				
13	ConvTranspose2d	512	256	3	1/1	2	
14	BatchNorm2d	256	256				
15	ConvTranspose2d	256	128	3	1/1	2	
16	BatchNorm2d	128	128				
17	ConvTranspose2d	128	64	3	1/1	2	
18	BatchNorm2d	64	64				
19	ConvTranspose2d	64	32	3	1/1	2	
20	BatchNorm2d	32	32				
21	Conv2d	32	27	1			

3.3 Experimentation

In the experimentation, we develop our own CNN architecture; adopt transfer learning with ResNet-18; and use U-Net to improve the baseline model. The CNN architecture is demonstrated in table(fill);

the transfer learning network is described in table 3; and the U-Net's architecture is described in table 4.

In our base model, the 32-pixel stride at the final prediction layer limits the scale of details in the upsampled output. In order to compensate for that, we took inspiration from Long et al. [2]. We used the same encoding network as the baseline model, but allow the convolution layers to propagate information to the deconvolution layers directly.

We also tried to use transfer learning by using ResNet-18 (see [9] for details) as the encoder for our network. Specifically, we replaced the convolution layers of our network with the convolution layers of ResNet-18 (without the fully connected layers at the end of ResNet-18). We then trained the model as usual, with the weights in the ResNet-18 convolution layers unfrozen so that they update during gradient descent. During training, we used weighted cross entropy loss and also probabilistically transformed training images in each minibatch as described in our baseline model description.

We used Xavier weight initialization and Adam gradient descent with learning rate 0.001 in the U-Net. Since our dataset has a class imbalance problem(i.e., sky predominating the output labels), we address this problem by using a weighted loss cross entropy loss criterion. The weight $w = (w_1, ..., w_{27})^T$ is given by

$$w_i = \frac{1/a_i^2}{\sum_{j=1}^{27} 1/a_j^2}, i = 1, 2, ..., 27 \tag{2}$$

We do not use a regularization technique in this experiment.

Table 2: Architecture of the proposed CNN, '/' in padding means 'padding/output padding'

Layer	Dimensions							
	Type	in channels	out channels	kernel size	padding	stride	Activation	Extra connection
1	Conv2d	3	32	3	1	2		
2	BatchNorm2d	32	32					
3	Conv2d	32	64	3	1	2		
4	BatchNorm2d	64	64					
5	Conv2d	64	128	3	1	2		
6	BatchNorm2d	128	128	14				14
7	Conv2d	128	256	3	1	2		
8	BatchNorm2d	256	256					12
9	Conv2d	256	512	3	1	2		
10	BatchNorm2d	512	512				ReLU	
11	ConvTranspose2d	512	512	3	1/1	2		
12	BatchNorm2d	512	512					
13	ConvTranspose2d	512	256	3	1/1	2		
14	BatchNorm2d	256	256					
15	ConvTranspose2d	256	128	3	1/1	2		
16	BatchNorm2d	128	128					
17	ConvTranspose2d	128	64	3	1/1	2		
18	BatchNorm2d	64	64					
19	ConvTranspose2d	64	32	3	1/1	2		
20	BatchNorm2d	32	32					
21	Conv2d	32	27	1				

Table 3: Architecture of the ResNet-18 transfer learning model, '/' in padding means 'padding/output padding'

Layer	Dimensions						
	Type	in channels	out channels	kernel size	padding	stride	Activation
1	Conv2d	3	64	7	3	2	
2	BatchNorm2d	64	64				ReLU
3	Max Pooling	64	64 3	1	2		
2	ConvTranspose2d	512	512	3	1/1	2	
3	BatchNorm2d	512	512				
4	ConvTranspose2d	512	256	3	1/1	2	
5	BatchNorm2d	256	256				
6	ConvTranspose2d	256	128	3	1/1	2	
7	BatchNorm2d	128	128				
8	ConvTranspose2d	128	64	3	1/1	2	
9	BatchNorm2d	64	64				
10	ConvTranspose2d	64	32	3	1/1	2	
11	BatchNorm2d	32	32				
12	Conv2d	32	27	1			

Table 4: Architecture of the U-net model, '/' in padding means 'padding/output padding'

Layer	Dimensions						
	Type	in channels	out channels	kernel size	padding	stride	Activation
1	Conv2d	3	64	3	1	1	
2	BatchNorm2d	64	64				ReLU
3	Conv2d	64	64	3	1	1	
4	BatchNorm2d	64	64				ReLU
5	Max Pooling	64	64	2		2	
6	Conv2d	64	128	3	1	1	
7	BatchNorm2d	128	128				ReLU
8	Max Pooling	128	128	2		2	
9	Conv2d	128	256	3	1	1	
10	BatchNorm2d	256	256				ReLU
11	Conv2d	256	256	3	1	1	
12	BatchNorm2d	256	256				ReLU
13	Max Pooling	256	256	2		2	
14	Conv2d	256	512	3	1	1	
15	BatchNorm2d	512	512				ReLU
16	Conv2d	512	512	3	1	1	
17	BatchNorm2d	512	512				ReLU
18	ConvTranspose2d	512	256	2	0/0	2	
19	Concatenation: 12	256	512				
20	Conv2d	512	256	3	1	1	
21	BatchNorm2d	256	256				ReLU
22	Conv2d	256	256	3	1	1	
23	BatchNorm2d	256	256				ReLU
24	ConvTranspose2d	256	128	2	0/0	2	
25	Concatenation: 7	128	256				
26	Conv2d	256	128	3	1	1	
27	BatchNorm2d	128	128				ReLU
28	Conv2d	128	128	3	1	1	
29	BatchNorm2d 128	128				ReLU 30	ConvTranspose2d
128	64	2	0/0	2			
31	Concatenation: 4	64	128				
32	Conv2d	128	64	3	1	1	
33	BatchNorm2d	64	64				ReLU
34	Conv2d	64	64	3	1	1	
35	BatchNorm2d	64	64				ReLU
36	Conv2d	64	27	1			

4 Result

4.1 Baseline model

We first demonstrate the baseline model's performance in table 5 and figure 5. The validation set pixel-wise accuracy can reach 70% and the average IoU reaches 25.7%. However, some categories(like sidewalk and billboar) have a low IoU.

Table 5: Validation set pixel accuracy and IoUs of the baseline model, 'baseline' uses cross entropy and does not augment the dataset, 'improved' uses weighted cross entropy (1) and augments the dataset. 'Accuracy' indicates the average accuracy; 'IoU' indicates the average IoU; and Categorical IoU indicates IoU in each category.

Network	Accuracy	IoU	Categorical IoU				
			Road	Sidewalk	Car	Billboard	Sky
baseline	71.5%	25.7%	85.2%	2.6%	47.6%	8.9%	87.7%
improved							

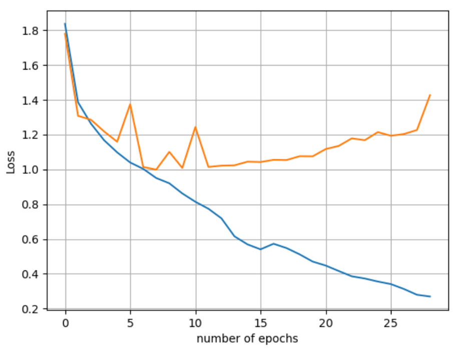
4.2 Experiment

We demonstrate the performance of the proposed CNN network, transfer learning network; and the U-Net in table 6 and figure

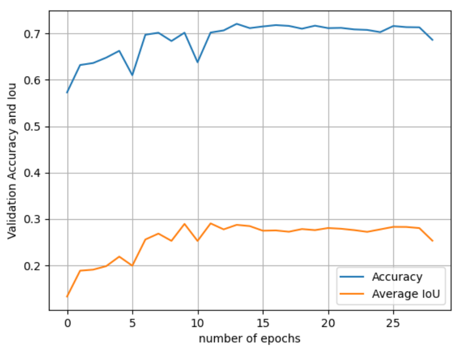
Table 6: Validation set pixel accuracy and IoUs of the proposed models. Meaning of each column coincides with table 5

Network	Accuracy	IoU	Categorical IoU				
			Road	Sidewalk	Car	Billboard	Sky
proposed CNN							
transfer learning	77.4%	42.8%	87.9%	nan	60.0%	23.7%	nan
U-Net	78.3%	47.5%	87.0%	nan	59.6%	nan	89.5%

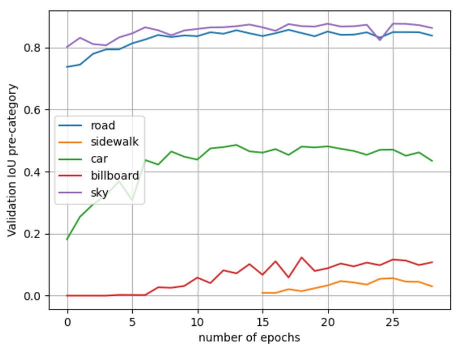
5 Discussion



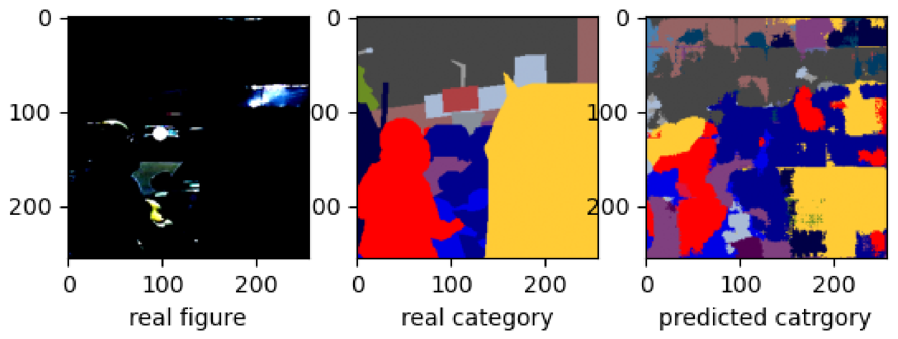
(a) Training and validation loss



(b) Validation accuracy and average IoU

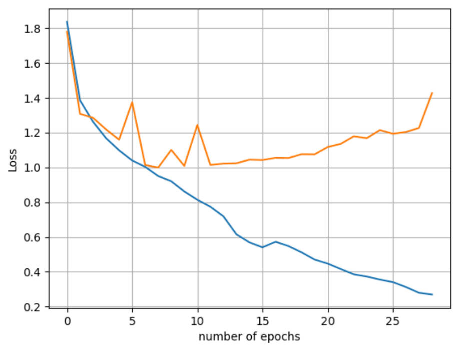


(c) Validation set categorical accuracy

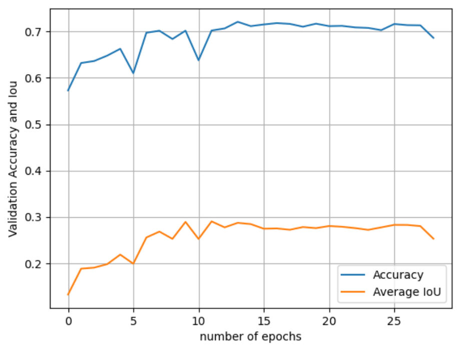


(d) Test set visualization

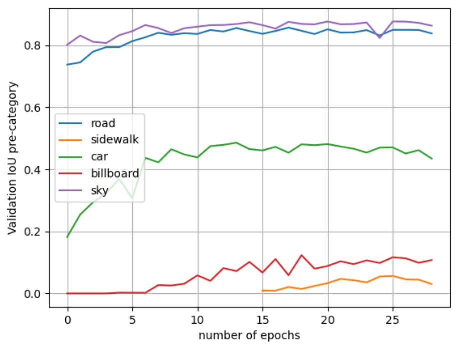
Figure 1: Performance of baseline model without data set augmentation and with cross entropy loss



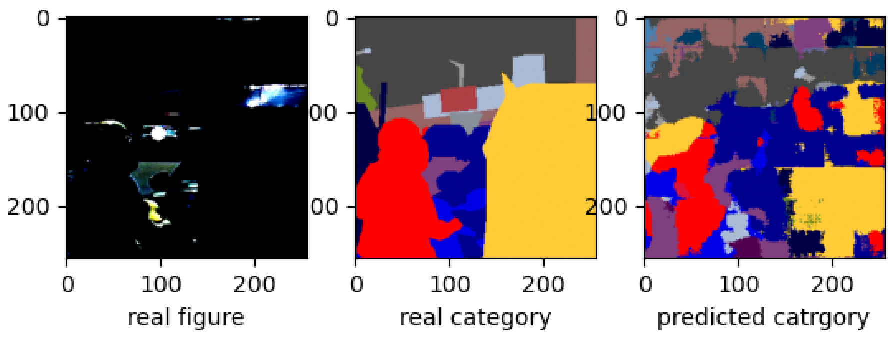
(a) Training and validation loss



(b) Validation accuracy and average IoU

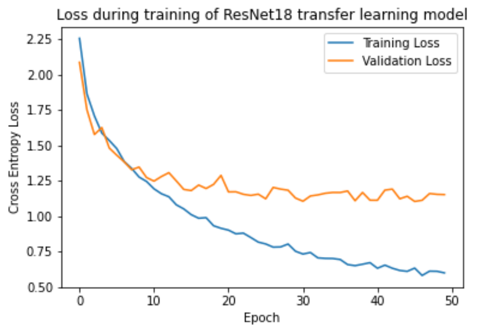


(c) Validation set categorical accuracy

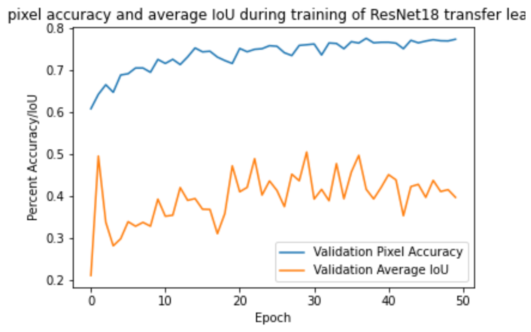


(d) Test set visualization

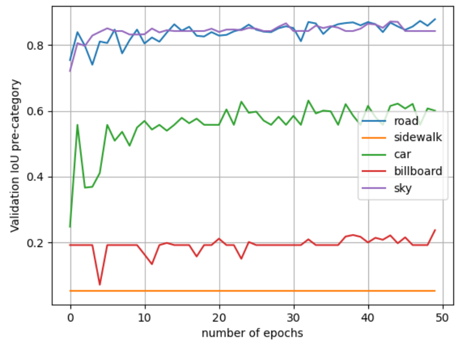
Figure 2: Performance of baseline model with data set augmentation and weighted cross entropy loss



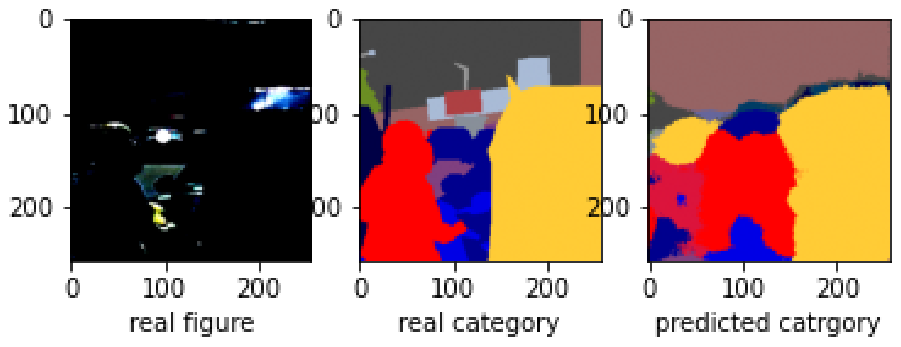
(a) Training and validation loss



(b) Validation accuracy and average IoU

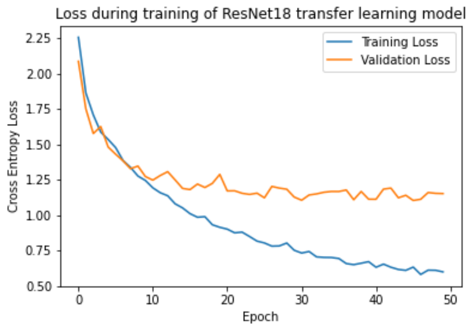


(c) Validation set categorical accuracy

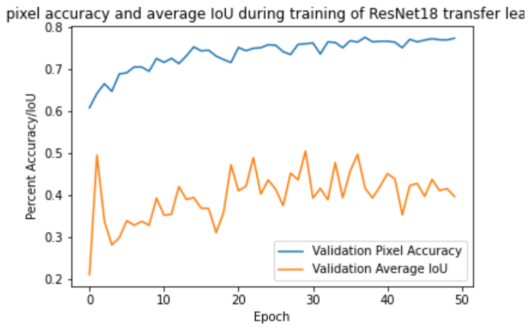


(d) Test set visualization

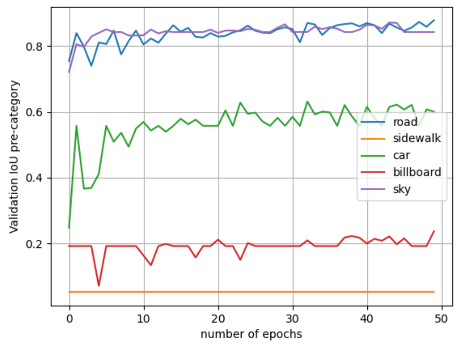
Figure 3: Performance of the proposed CNN network



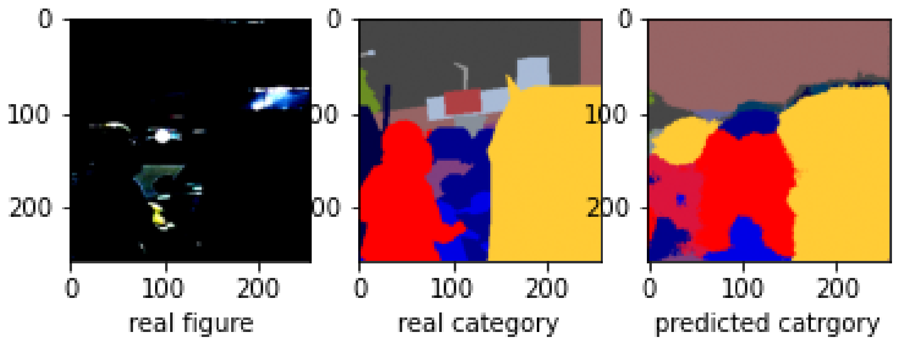
(a) Training and validation loss



(b) Validation accuracy and average IoU

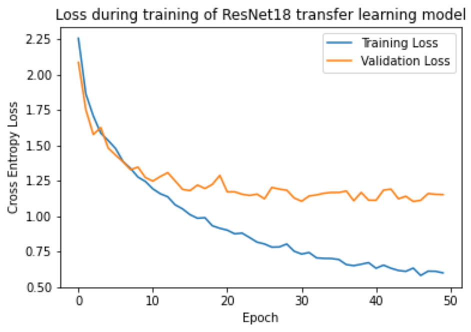


(c) Validation set categorical accuracy

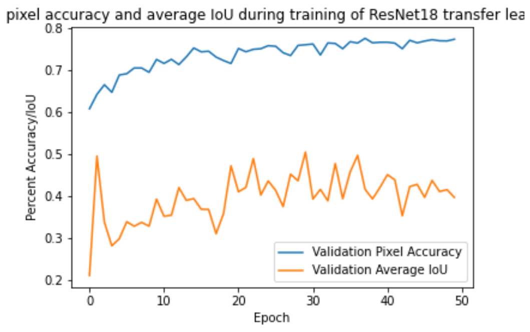


(d) Test set visualization

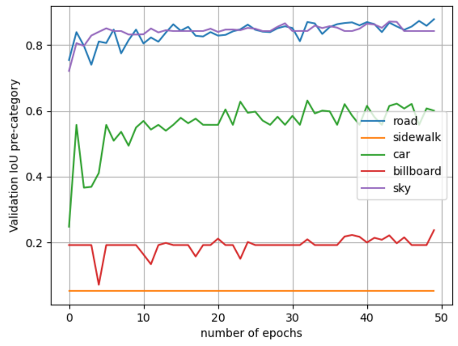
Figure 4: Performance of transfer learning network



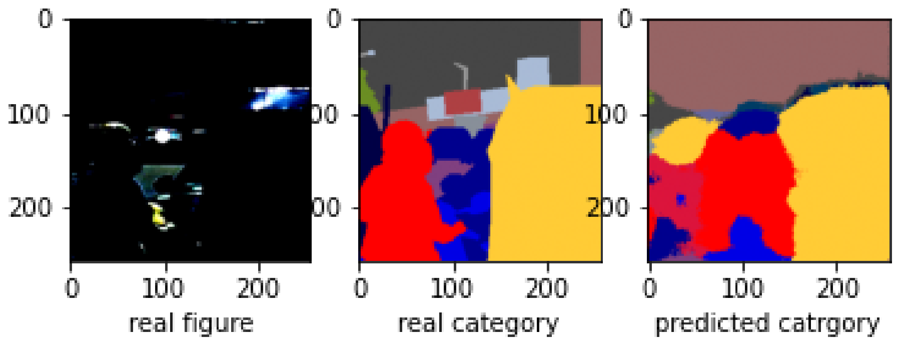
(a) Training and validation loss



(b) Validation accuracy and average IoU



(c) Validation set categorical accuracy



(d) Test set visualization

Figure 5: Performance of U-Net

References

[1] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451–1460, 2018.

[2] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[3] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[6] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015.

[7] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[8] S. Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.