WAD Video Segmentation

Emre DOĞAN
Department of Computer Engineering
Hacettepe University, Ankara, TURKEY
emre.dogan@hacettepe.edu.tr

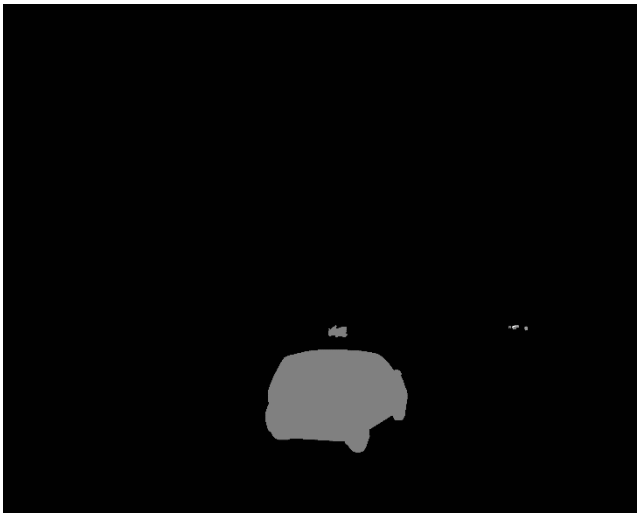
Tarık Ramazan BAŞOĞLU
Department of Computer Engineering
Hacettepe University, Ankara, TURKEY
tarik.basoglu@hacettepe.edu.tr

Abstract

In this paper,CVPR 2018 WAD (Workshop on Autonomous Driving) chalange on kaggle is handled under the scope of BBM 416 Computer Vision course. Today,Self-driving cars are being discussed as never before. It aims to get together researchers and engineers from academia and industries to discuss computer vision applications in autonomous driving.Video segmentation is important to detect object for secure driving. We will predict segmentations of different movable objects appearing in the view of a car camera

1.INTRODUCTION

In this Project,Sample of original Dataset which is 30GB(20GB for train,10GB for test)is used.It includes high resolution images which is taken from the top of the car.All photos has same resolution which is 3384x2710.It means that we have 9.2 megapixels photos taken sequentially. Labels are given in binary format and you can see sample below.



2.RELATED WORKS

It is huge subject on Computer Vision.There lots of data sets and research on this area. You can see some researches and their dataset on the table.

- ***DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving***

  Chenyi Chen, Ari Seff, Alain Kornhauser, Jianxiong Xiao

  http://deepdriving.cs.princeton.edu
  KITTI dataset(The KITTI dataset contains over 40,000 stereo image pairs taken by a car driving through European urban areas.)

- ***Semantic video segmentation: Exploring inference efficiency***
  CamVid: The Cambridge-driving Labeled Video Database was one of the first semantically segmented datasets to be released in the self-driving space in late 2007.

- ***Scharwächter, M. Enzweiler, S. Roth, and U. Franke. "Efficient Multi-Cue Scene Segmentation", In Proc. of the German Conference on Pattern Recognition (GCPR), 2013***

  DUS: The Daimler Urban Segmentation dataset is a dataset of 5000 grayscale images of which only 500 are semantically segmented

- ***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 Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.***

  CitySCapes:The 30 classes are split across 8 higher level categories as well. A unique feature of this dataset is that the authors have provided 20000 more images with coarse segmentation.

- **The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes**
  Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bul`o, Peter Kontschieder
  Mapillary Research

  Mapillary is a street-level imagery platform where participants collaborate to build better maps.

3.MODEL

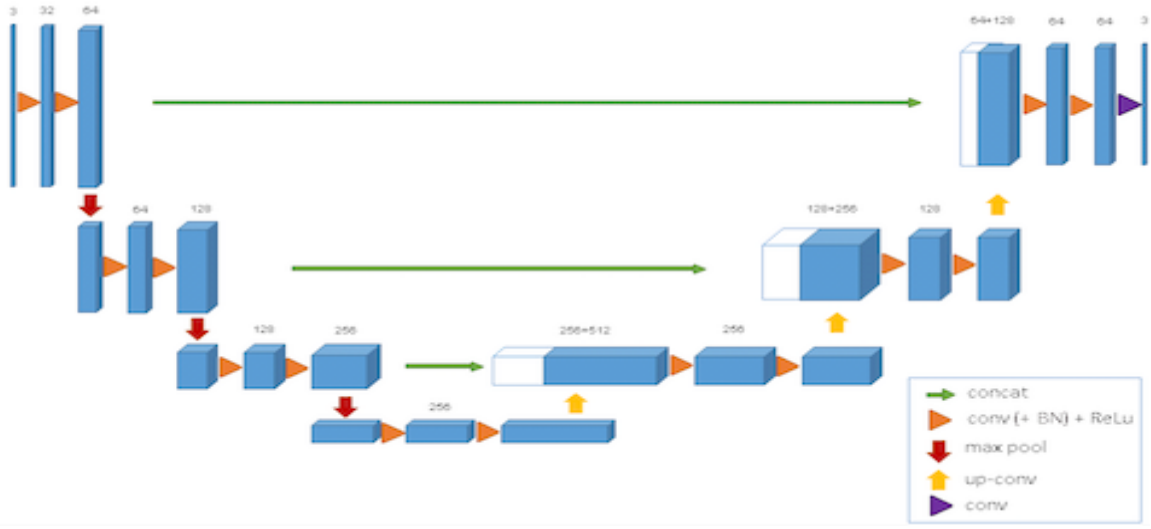
U-Net for Image Segmentation

The U-Net is one of the fast convolutional network architectures for precise segmentation of images. Up to now it has outperformed the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. It has won the Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI 2015, and it has won the Cell Tracking Challenge at ISBI 2015 on the two most challenging transmitted light microscopy categories (Phase contrast and DIC microscopy) by a large margin

In this project, we will go over using U-Net for detecting vehicles in the given dataset. U-net is a encoder-decoder type of network for pixel-wise predictions. U-net is unique because in U-net, the receptive fields after convolution are concatenated with the receptive fields in up-convolving process. This additional feature allows network to use original features in addition to features after up-convolution. This results in overall better performance than a network that has access to only features after up-convolution. Post-training, the network was correctly able to identify vehicles in an urban setting, and more *interestingly* performed better than humans *in cases where cars were not correctly annotated.*

Development Environment

Anaconda3	5.1.0
Python	3.6.4
Tensorflow-gpu	1.8.0
Keras	2.1.6
Scikit-image	0.13.1
Numpy	1.14.0



Structure

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
lambda_1 (Lambda)	(None, 224, 224, 3)	0	input_1[0][0]
conv2d_1 (Conv2D)	(None, 224, 224, 8)	224	lambda_1[0][0]
conv2d_2 (Conv2D)	(None, 224, 224, 8)	584	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 112, 112, 8)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 112, 112, 16)	1168	max_pooling2d_1[0][0]
conv2d_4 (Conv2D)	(None, 112, 112, 16)	2320	conv2d_3[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 56, 56, 16)	0	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 56, 56, 32)	4640	max_pooling2d_2[0][0]
conv2d_6 (Conv2D)	(None, 56, 56, 32)	9248	conv2d_5[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 28, 28, 32)	0	conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 28, 28, 64)	18496	max_pooling2d_3[0][0]
conv2d_8 (Conv2D)	(None, 28, 28, 64)	36928	conv2d_7[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 64)	0	conv2d_8[0][0]

conv2d_9 (Conv2D)	(None, 14, 14, 128)	73856	max_pooling2d_4[0][0]
conv2d_10 (Conv2D)	(None, 14, 14, 128)	147584	conv2d_9[0][0]
conv2d_transpose_1(Conv2DTrans)	(None, 28, 28, 64)	32832	conv2d_10[0][0]
concatenate_1 (Concatenate)	(None, 28, 28, 128)	0	conv2d_transpose_1[0][0] conv2d_8[0][0]
conv2d_11 (Conv2D)	(None, 28, 28, 64)	73792	concatenate_1[0][0]
conv2d_12 (Conv2D)	(None, 28, 28, 64)	36928	conv2d_11[0][0]
conv2d_transpose_2(Conv2DTrans)	(None, 56, 56, 32)	8224	conv2d_12[0][0]
concatenate_2 (Concatenate)	(None, 56, 56, 64)	0	conv2d_transpose_2[0][0] conv2d_6[0][0]
conv2d_13 (Conv2D)	(None, 56, 56, 32)	18464	concatenate_2[0][0]
conv2d_14 (Conv2D)	(None, 56, 56, 32)	9248	conv2d_13[0][0]
conv2d_transpose_3(Conv2DTrans)	(None, 112, 112, 16)	2064	conv2d_14[0][0]
concatenate_3 (Concatenate)	(None, 112, 112, 32)	0	conv2d_transpose_3[0][0] conv2d_4[0][0]
conv2d_15 (Conv2D)	(None, 112, 112, 16)	4624	concatenate_3[0][0]
conv2d_16 (Conv2D)	(None, 112, 112, 16)	2320	conv2d_15[0][0]
conv2d_transpose_4(Conv2DTrans)	(None, 224, 224, 8)	520	conv2d_16[0][0]
concatenate_4 (Concatenate)	(None, 224, 224, 16)	0	conv2d_transpose_4[0][0] conv2d_2[0][0]
conv2d_17 (Conv2D)	(None, 224, 224, 8)	1160	concatenate_4[0][0]
conv2d_18 (Conv2D)	(None, 224, 224, 8)	584	conv2d_17[0][0]
conv2d_19 (Conv2D)	(None, 224, 224, 1)	9	conv2d_18[0][0]

Total params: 485,817

Trainable params: 485,817

Non-trainable params: 0

Input Size : 224 x 224 x 3

Train on 8100 samples, validate on 900 samples

4.RESULTS

Batch	Epoch	Result
8	10	loss: 0.0249 - mean_iou: 0.6293 - val_loss: 0.0292 - val_mean_iou: 0.6347
8	20	loss: 0.0191 - mean_iou: 0.6880 - val_loss: 0.0249 - val_mean_iou: 0.6904
8	30	loss: 0.0158 - mean_iou: 0.7417 - val_loss: 0.0473 - val_mean_iou: 0.7426
16	20	loss: 0.0226 - mean_iou: 0.6072 - val_loss: 0.0269 - val_mean_iou: 0.6106
64	20	loss: 0.0180 - mean_iou: 0.6020 - val_loss: 0.0270 - val_mean_iou: 0.6066

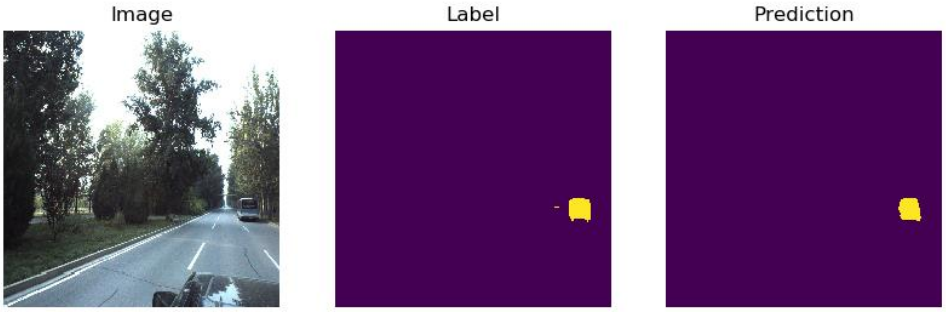
Advantages:

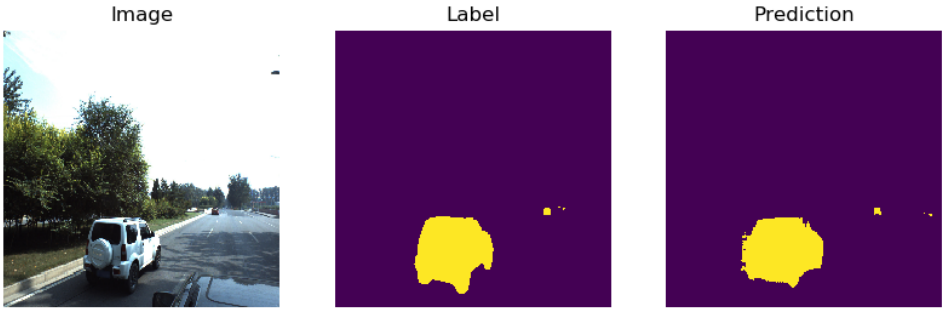
- Simple structure, easy implementation
- Fast train and computing
- Almost good results

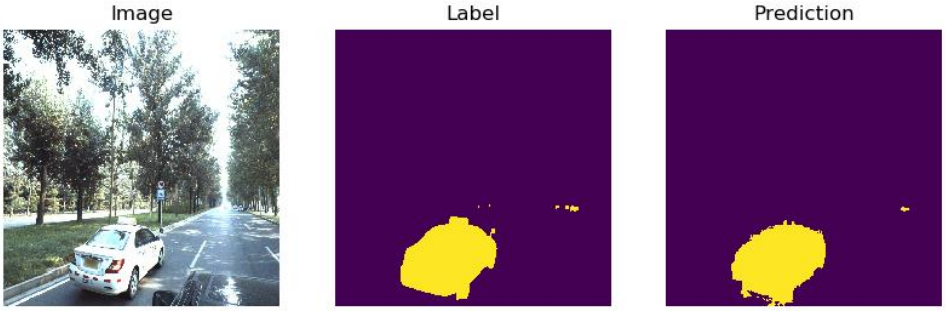
Disadvantages:

- Loss of detail due to low resolution
- Incorrect predictions due to low resolution
- Inadequate learning in large vehicles

Validation Results:



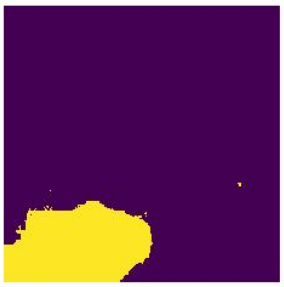




Image

Label

Prediction



Image

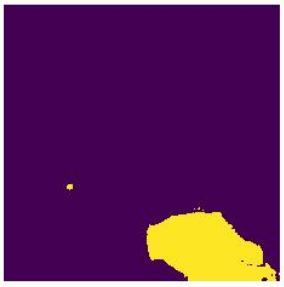
Label

Prediction

Image

Label

Prediction



Image

Label

Prediction

Image

Label

Prediction

Image

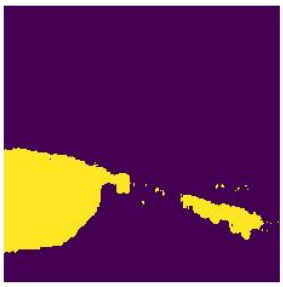
Label

Prediction

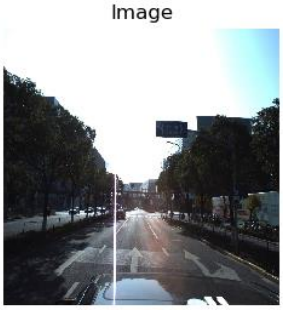
Image

Label

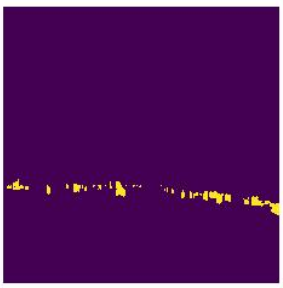
Prediction



Image



Label



Prediction

Image

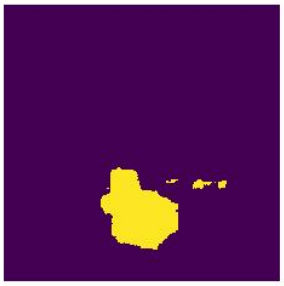
Label

Prediction

Image

Label

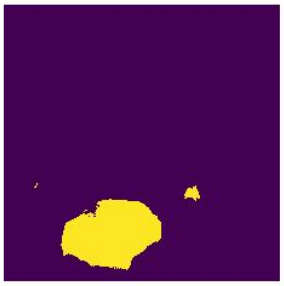
Prediction



Image

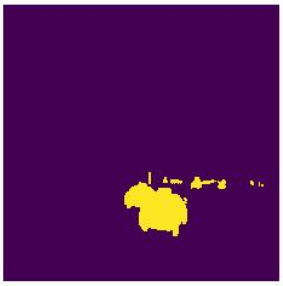
Label

Prediction

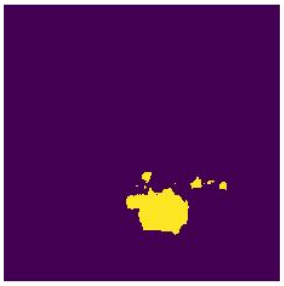


Image

Label



Prediction



Image

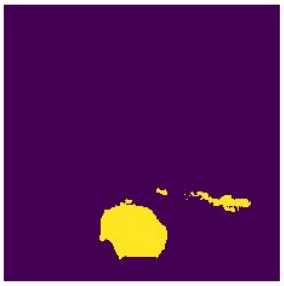
Label

Prediction

Image

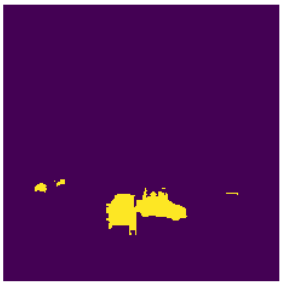
Label

Prediction



Image

Label



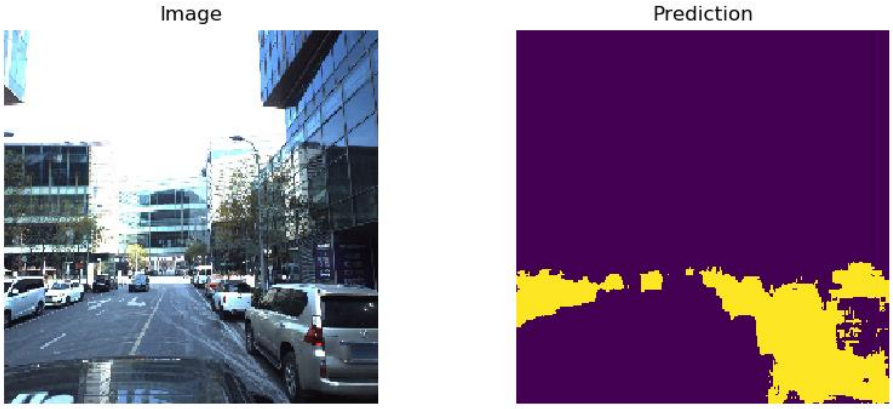
Prediction

Image

Label

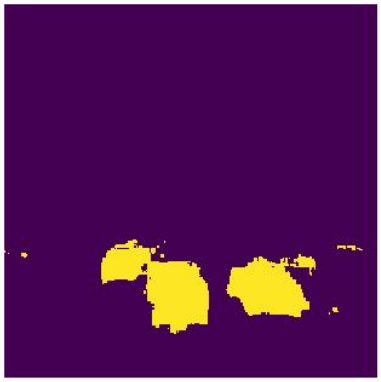
Prediction

Test Results:



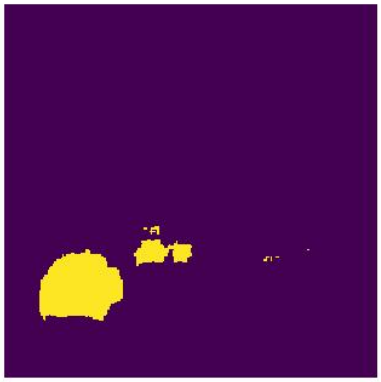
Image

Prediction



Image

Prediction

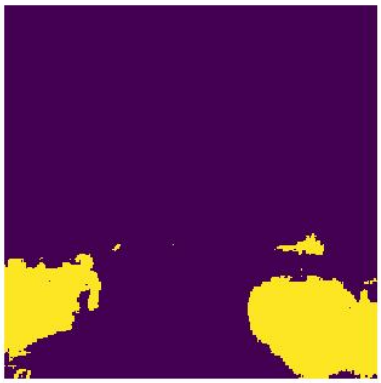


Image

Prediction

Image

Prediction

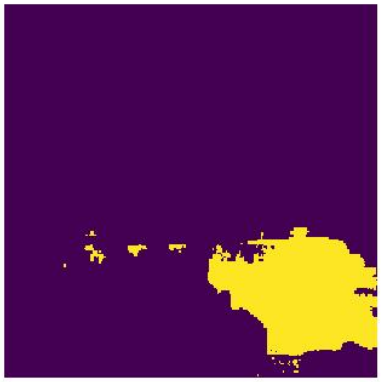


Image

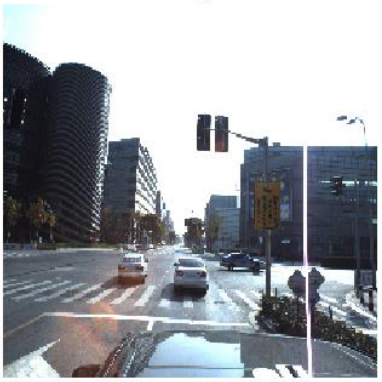
Prediction

Image

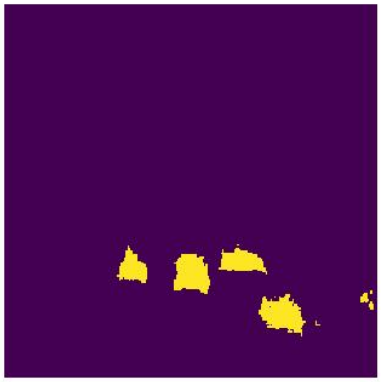
Prediction



Image

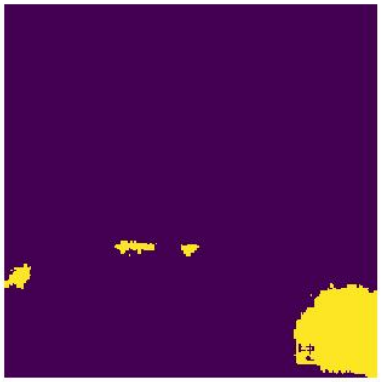


Prediction



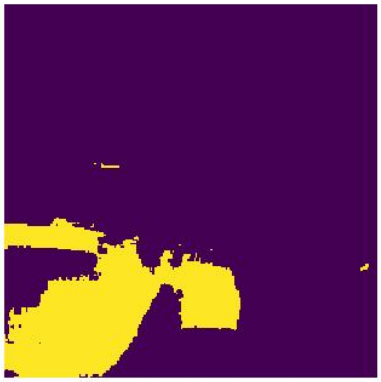
Image

Prediction



Image

Prediction

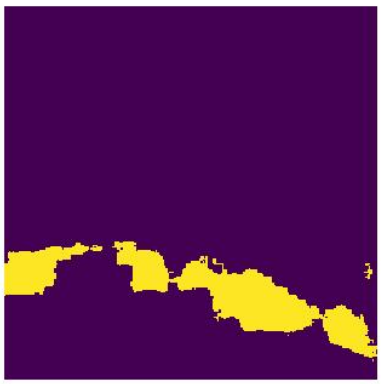


Image

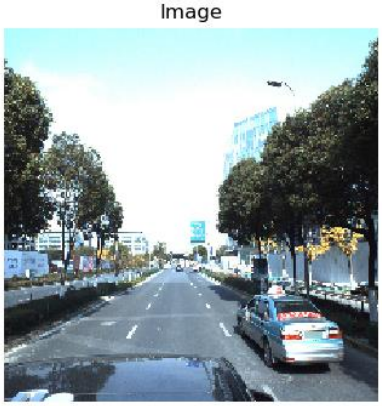
Prediction

Image

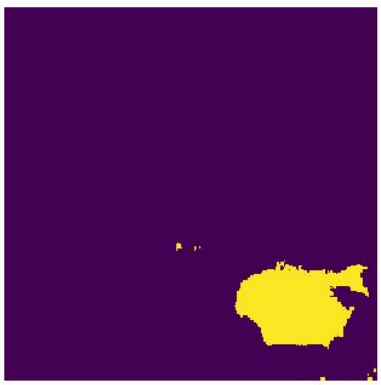
Prediction



Image

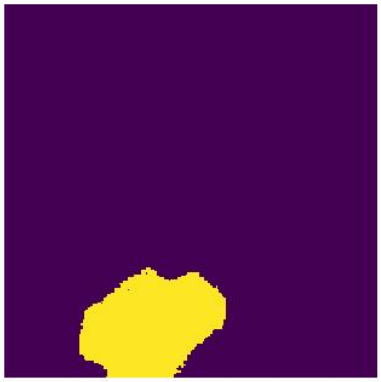


Prediction



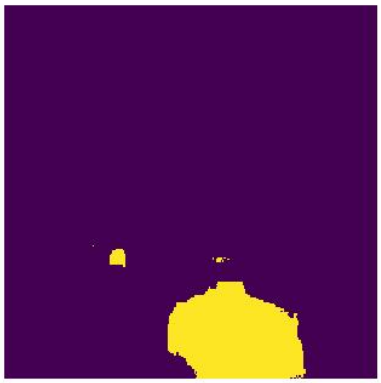
Image

Prediction



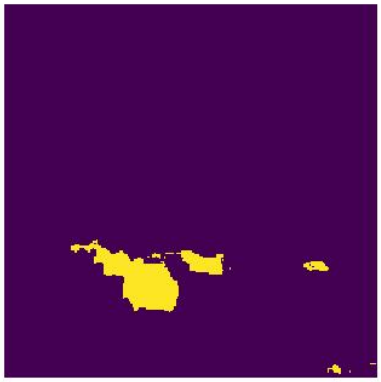
Image

Prediction



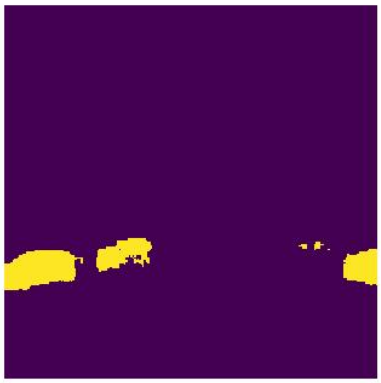
Image

Prediction



Image

Prediction



Image

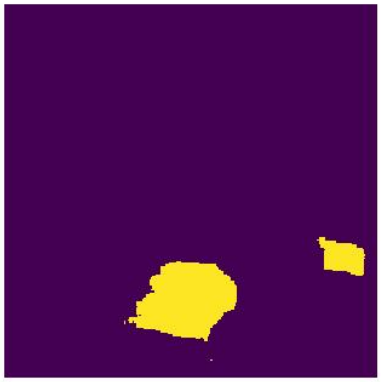
Prediction

Image

Prediction

Image

Prediction



Image

Prediction