Deep learning-based Lane Detection in Foggy weather Conditions

Final Presentation

Project Advisors:
Dr Rahul Rai (rrai@clemson.edu)
Shengli Xu (shenglx@clemson.edu)

Team Members:
Pranava Swaroopa(pswaroo@clemson.edu)
Sanskruti Jadhav (sanskrj@clemson.edu)
Siddharth Joshi (sajoshi@clemson.edu)

CLEMSON UNIVERSITY

Motivation

- According to NHTSA the fatality and fatality rate per 100 million miles traveled by the vehicle, is of the order of 30,000-40,000. In high-speed scenarios, human reaction is often not fast enough or accurate enough to avoid all forms of injury. Most of the collision that arises is due to some form of human error. (Fatality Analysis Reporting System (FARS) | NHTSA)
- Detecting and following lines using sensors like Camera, LiDAR, RADAR is difficult in inclement weather conditions.
- Using a network that is specifically trained with worse weather conditions help in accurate lane following algorithms

CLEMSON UNIVERSITY

Computation Platform and dependencies

HPC: Palmetto Cluster Clemson University

Compute Node:

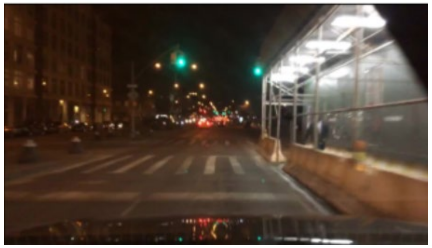
- GPU: Tesla V100
- RAM 125gb
- Cores 28

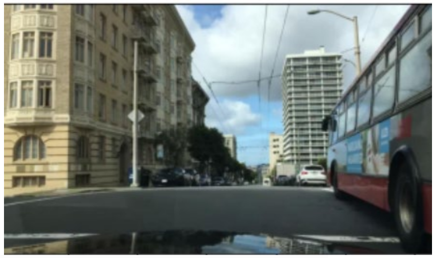
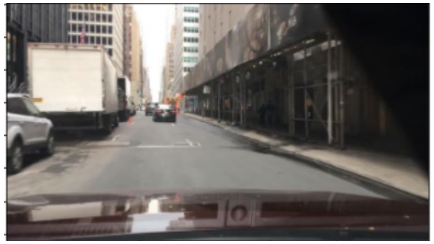
Package Dependencies:

- Python 3.6.4
- Tensorflow 2.4.0
- Pickle 5
- Open CV
- Numpy

Dataset

- Drivable area segmentation BDD100K consists of
- 70,000/10,000/20,000 of training, Validation, Testing
- Labels – Masks, Polygons, Colormaps



Frame attributes

```
- weather: "rainy|snowy|clear|overcast|undefined|partly cloudy|foggy"
- scene: "tunnel|residential|parking lot|undefined|city street|gas stations|highway|"
- timeofday: "daytime|night|dawn/dusk|undefined"
```

Dataset

Images Attribute	Number
Partly Cloudy	4881
Rainy	5070
Snowy	5549
Foggy	130

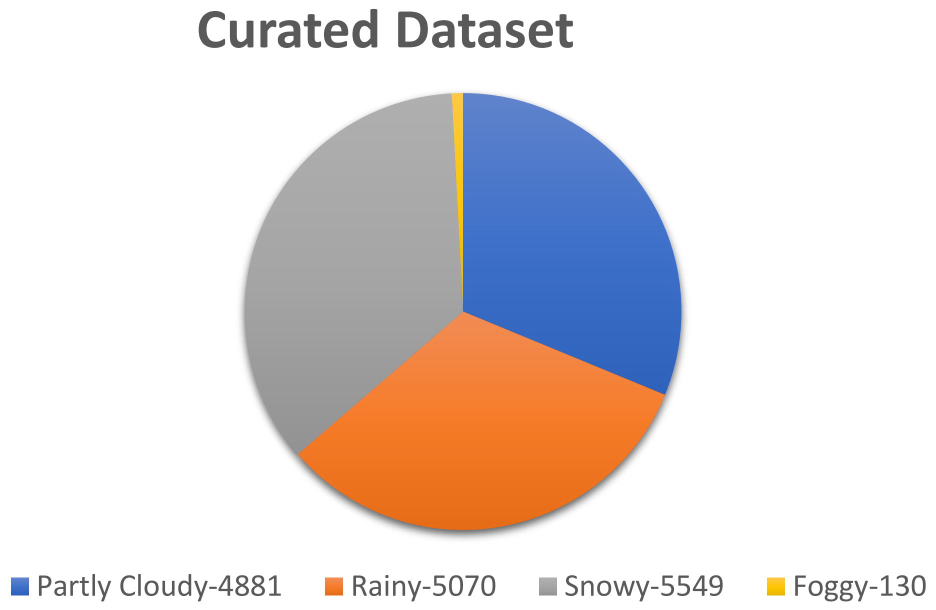
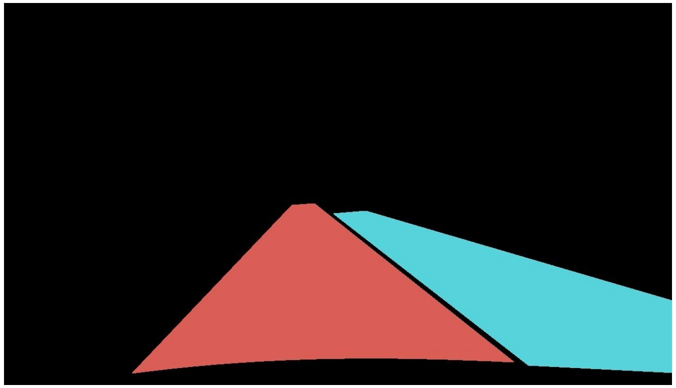


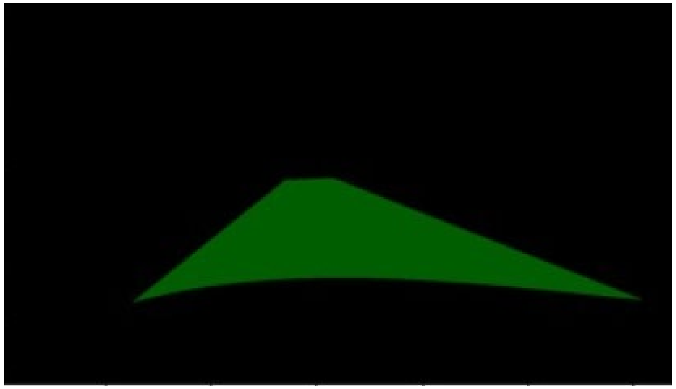
Image and Label Sizes	
Images	**Labels - Colormaps**
(15630,720,1280,3)	(15630,720,1230

CLEMSON UNIVERSITY

Data Processing

- Huge dataset of 15560 images in jpg format and Labels in PNG format
- Processed labels converting to HSV images to remove dual labels and only focusing on one type of label (Green Channel)
- Normalizing and Resizing the datasets
- Saving NumPy Arrays to facilitate reusing and avoiding long processing time
- Dead Kernel issues





OPTIMUM MODEL FRAMEWORK

Model: "sequential"

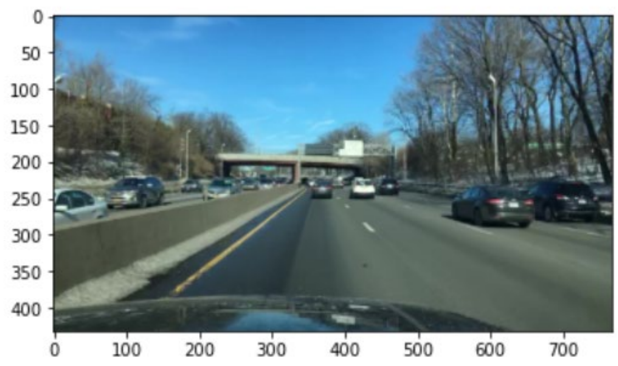
Layer (type)	Output Shape	Param #
Conv1 (Conv2D)	(None, 430, 766, 8)	224
Conv2 (Conv2D)	(None, 428, 764, 16)	1168
max_pooling2d (MaxPooling2D)	(None, 214, 382, 16)	0
Conv3 (Conv2D)	(None, 212, 380, 16)	2320
dropout (Dropout)	(None, 212, 380, 16)	0
Conv4 (Conv2D)	(None, 210, 378, 32)	4640
dropout_1 (Dropout)	(None, 210, 378, 32)	0
Conv5 (Conv2D)	(None, 208, 376, 32)	9248
dropout_2 (Dropout)	(None, 208, 376, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 104, 188, 32)	0
Conv6 (Conv2D)	(None, 102, 186, 64)	18496
dropout_3 (Dropout)	(None, 102, 186, 64)	0
Conv7 (Conv2D)	(None, 100, 184, 64)	36928
dropout_4 (Dropout)	(None, 100, 184, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 50, 92, 64)	0
up_sampling2d (UpSampling2D)	(None, 100, 184, 64)	0
Deconv1 (Conv2DTranspose)	(None, 102, 186, 64)	36928
dropout_5 (Dropout)	(None, 102, 186, 64)	0
Deconv2 (Conv2DTranspose)	(None, 104, 188, 64)	36928
dropout_6 (Dropout)	(None, 104, 188, 64)	0
up_sampling2d_1 (UpSampling2	(None, 208, 376, 64)	0
Deconv3 (Conv2DTranspose)	(None, 210, 378, 32)	18464
dropout_7 (Dropout)	(None, 210, 378, 32)	0
Deconv4 (Conv2DTranspose)	(None, 212, 380, 32)	9248
dropout_8 (Dropout)	(None, 212, 380, 32)	0
Deconv5 (Conv2DTranspose)	(None, 214, 382, 16)	4624
dropout_9 (Dropout)	(None, 214, 382, 16)	0
up_sampling2d_2 (UpSampling2	(None, 428, 764, 16)	0
Deconv6 (Conv2DTranspose)	(None, 430, 766, 16)	2320
Final (Conv2DTranspose)	(None, 432, 768, 1)	145

RESULTS

```
_________________________________________________________________
Final (Conv2DTranspose)      (None, 432, 768, 1)       145
=================================================================
Total params: 181,681
Trainable params: 181,681
Non-trainable params: 0
_________________________________________________________________
```

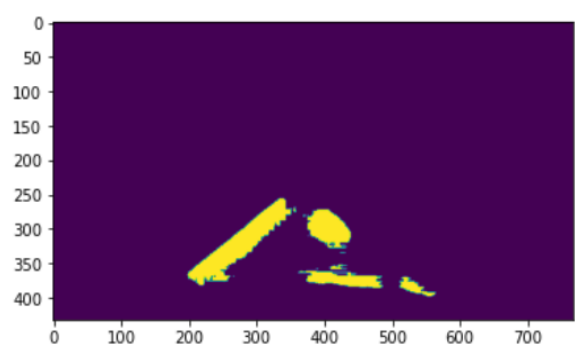
```
[45]: plt.imshow(x_test[5])
```

```
[45]: <matplotlib.image.AxesImage at 0x1516d8bfbf90>
```



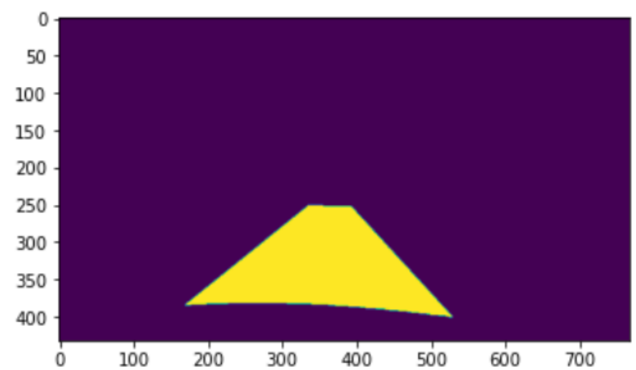

```
[43]: ## Visualize Images

      plt.imshow(y_pred[5])
```

```
[43]: <matplotlib.image.AxesImage at 0x1516d8b57a90>
```




```
[44]: plt.imshow(y_true[5])
```

```
[44]: <matplotlib.image.AxesImage at 0x1516d8fc5710>
```



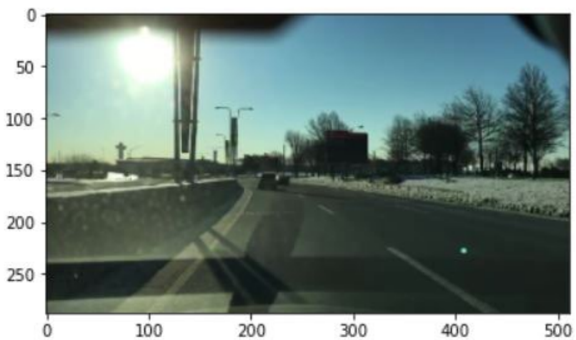
Optimum Model Hyperparameters

Sr. No	Hyperparameter	Values
1	Image size	60% of original value (432, 768, 3)
2	Training dataset: Testing dataset:	12504 images 3126 images
3	Batch size	32
4	Epoch	10
5	Optimizer	Adam (learning rate -0.001)
6	Loss Function	Mean Squared Error
7	Activation Function	ReLU
8	Layer sizes	7 Convolution Layers 7Deconvolution Layers 3 max pooling Layers 3 Up-sampling Layers

Poor Model Result

```
plt.imshow(x_test[0])
```

```
<matplotlib.image.AxesImage at 0x150ba807cd90>
```

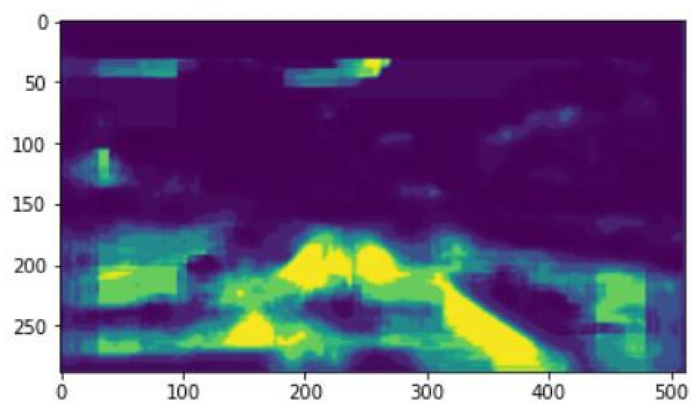


Hyper-params :

Epochs – 10
Batch size –16
No. of Layers –20
Learning rate –0.01

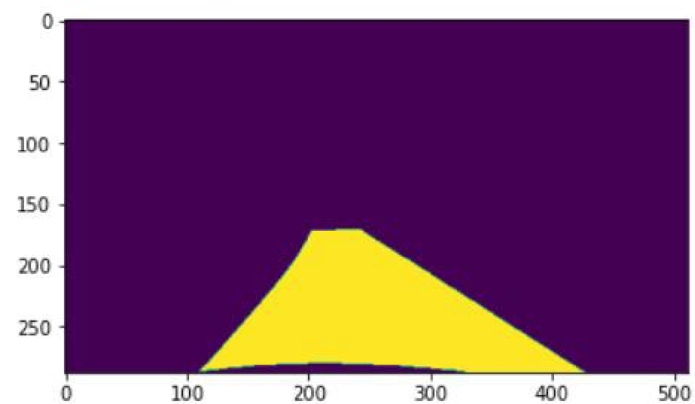
```
plt.imshow(y_pred[0])
```

```
<matplotlib.image.AxesImage at 0x150ab23ff7d0>
```




```
plt.imshow(y_true[0])
```

```
<matplotlib.image.AxesImage at 0x150ab2489b10>
```



Our Contribution

- Adding Convolutional layers with more filters at the beginning proved to effective for feature learning

Model: sequential

Layer (type)	Output Shape	Param #
Conv1 (Conv2D)	(None, 430, 766, 8)	224
Conv2 (Conv2D)	(None, 428, 764, 16)	1168
max_pooling2d (MaxPooling2D)	(None, 214, 382, 16)	0

- Corresponding Deconvolution Layers and Up-sampling Layers

dropout_9 (Dropout)	(None, 214, 382, 16)	0
up_sampling2d_2 (UpSampling2	(None, 428, 764, 16)	0
Deconv6 (Conv2DTranspose)	(None, 430, 766, 16)	2320
Final (Conv2DTranspose)	(None, 432, 768, 1)	145

- Training Image Size 60 % (432,768) was good enough compared to basic image size (720,1280)

Hyperparameter Tuning

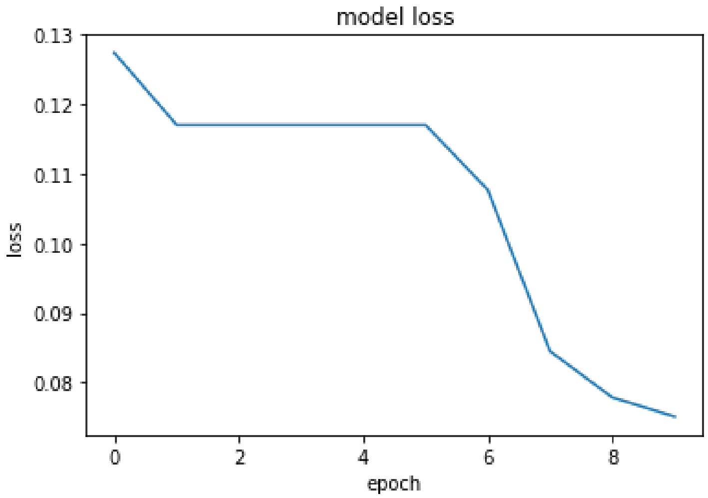
No	# Epochs	Image Resize Factor	Batch Size	Learning Rate	mIoU	Time (s)	Final Loss
1	10	0.6	8	0.001	0.41	1160	0.118
2	10	0.6	8	0.0001	0.43	1170	0.157
3	10	0.6	8	0.1	0.44	1165	0.121
4	10	0.6	16	0.001	0.44	2152	0.068
5	10	0.6	32	0.001	0.44	1230	0.0705
6	30	0.6	32	0.0001	0.4415	1490	0.0713

Hyperparameter Tuning

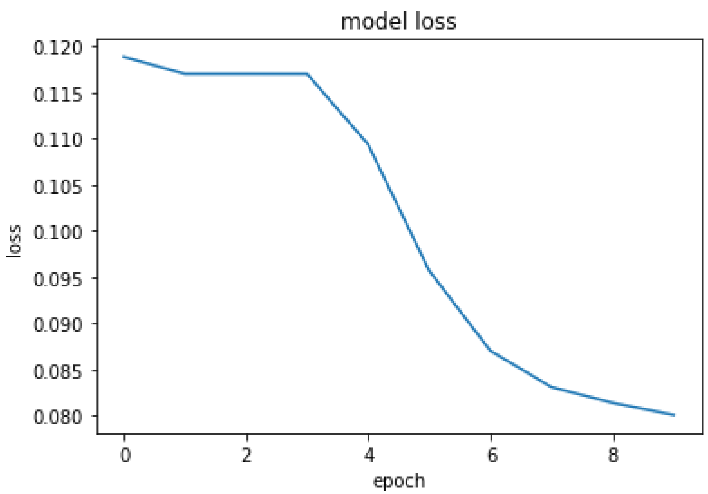
- Hyperparameters Selected:
  - Kernel Size – (3,3) & (5,5)
  - Epoch – 10, 6, 30
  - Learning Rate – 0.001, 0.1, 0.0001, 0.2
  - Batch Size – 8, 16, 32
  - Input Image Resizing – 60% of Original, 40% of original

INITIAL TRIAL RESULTS

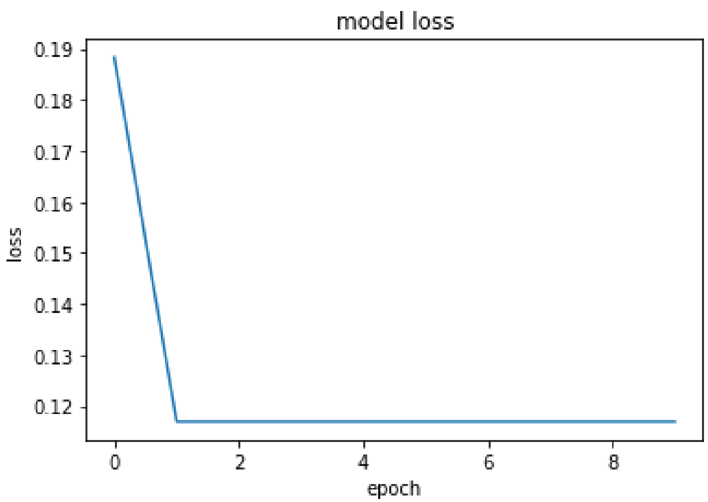
Epoch - 10; Batch Size -16;



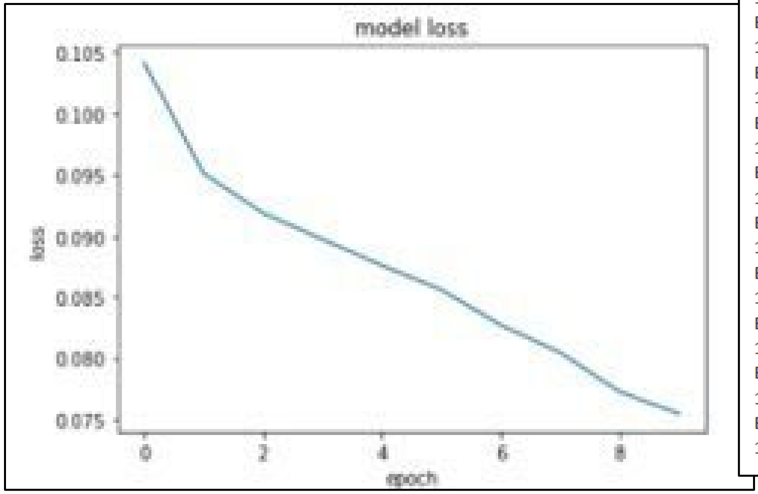
Epoch - 10; Batch Size -16; Some layers removed



Epoch - 10; Batch Size -16; Kernel - (5,5) 50% Drop with layers removed



Final Model Losses and Mean_IoU




```
12504/12504 [==============================] - 144s 12ms/sample - loss: 0.1042 - mean_io_u: 0.4415
Epoch 2/10
12504/12504 [==============================] - 137s 11ms/sample - loss: 0.0951 - mean_io_u: 0.4415
Epoch 3/10
12504/12504 [==============================] - 135s 11ms/sample - loss: 0.0919 - mean_io_u: 0.4415
Epoch 4/10
12504/12504 [==============================] - 134s 11ms/sample - loss: 0.0897 - mean_io_u: 0.4415
Epoch 5/10
12504/12504 [==============================] - 136s 11ms/sample - loss: 0.0876 - mean_io_u: 0.4415
Epoch 6/10
12504/12504 [==============================] - 137s 11ms/sample - loss: 0.0856 - mean_io_u: 0.4415
Epoch 7/10
12504/12504 [==============================] - 138s 11ms/sample - loss: 0.0827 - mean_io_u: 0.4415
Epoch 8/10
12504/12504 [==============================] - 139s 11ms/sample - loss: 0.0805 - mean_io_u: 0.4415
Epoch 9/10
12504/12504 [==============================] - 136s 11ms/sample - loss: 0.0773 - mean_io_u: 0.4415
Epoch 10/10
12504/12504 [==============================] - 137s 11ms/sample - loss: 0.0755 - mean_io_u: 0.4415
```

- Final Loss 0.0755
- mIOU = 0.4415

Conclusion

- Our project started with gathering and curating the data, where the data curation was the most tedious task
- Based on the results we achieved our model is not robust enough but gives you an idea about where the lane is and has a scope to lot of improvement

CLEMSON UNIVERSITY

TESTING: Video Output using Optimized Model