Hazard Detection in Self-driving Vehicles

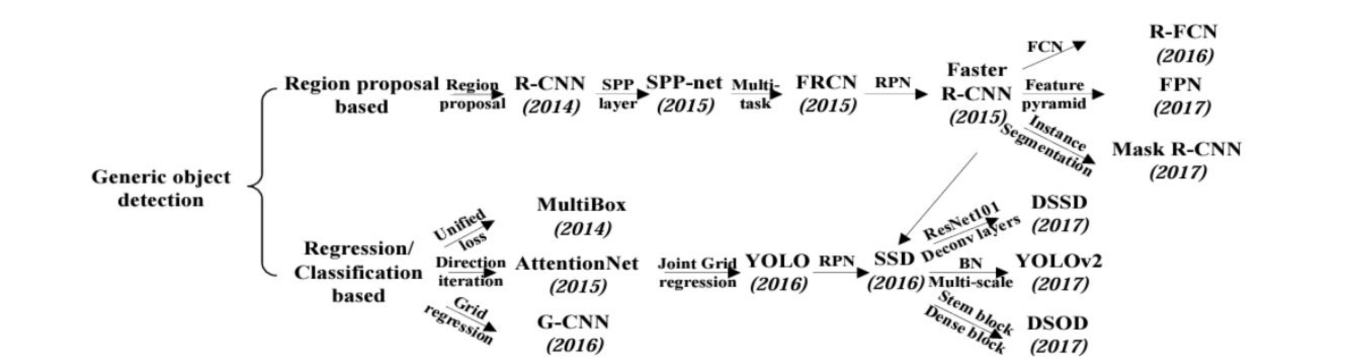
Viet Luong, Sabrina Wu, Matthew Zietz

Motivation

- Zero Emission Government Fleet Declaration (2022): Electric and self-driving vehicles are increasingly popular
- Electric vehicles (EVs) face various safety challenges, particularly regarding the detection of pedestrians and hazards on road.
- How these cars can further improve in detecting and avoiding hazards?
- How can the task be done through computer vision?
- Looking to make a simple model for broad classification of hazards
- After classification we wanted to go further and try to identify potential hazards

Prior Work

- Tons of prior work in object classification and detection
- Many libraries and CNN architectures devoted to object detection
  - PyTorch, Keras, Tensorflow
  - LeNet, AlexNet, ResNet, GoogleNet, MobileNet, VGG
- YOLO is a commonly used object detection model used in a variety of computer vision projects



Dataset and Preprocessing

Classification Model

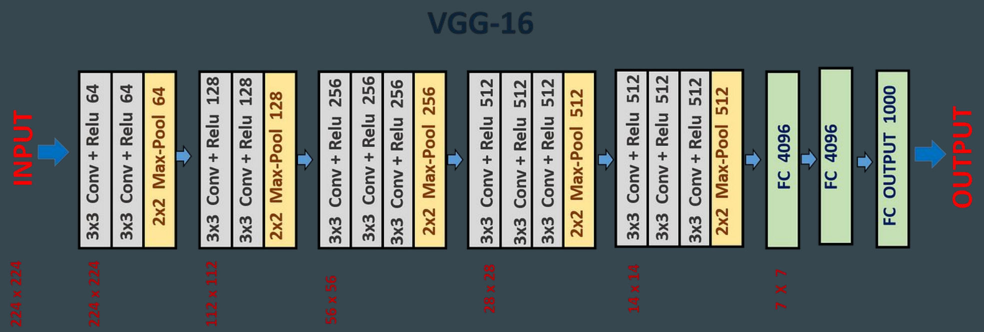
- Dataset
  - Found on [Kaggle]
  - 740 images of roads scraped from Google
- Preprocessing
  - Removed poor images
  - Added images for other class of hazards
  - Resized and scaled images using Keras for uniformity and ease of computing

Detection Model

- Dataset
  - Found from [BDD100K]
  - 70,000 images for training
  - 10,000 images for validation
  - 20,000 images for testing
- Preprocessing
  - Using the Fiftyone library to convert the dataset to YOLO format
  - Storing the top left and bottom right coordinates in a text file for each images.

Model Architecture - Classification

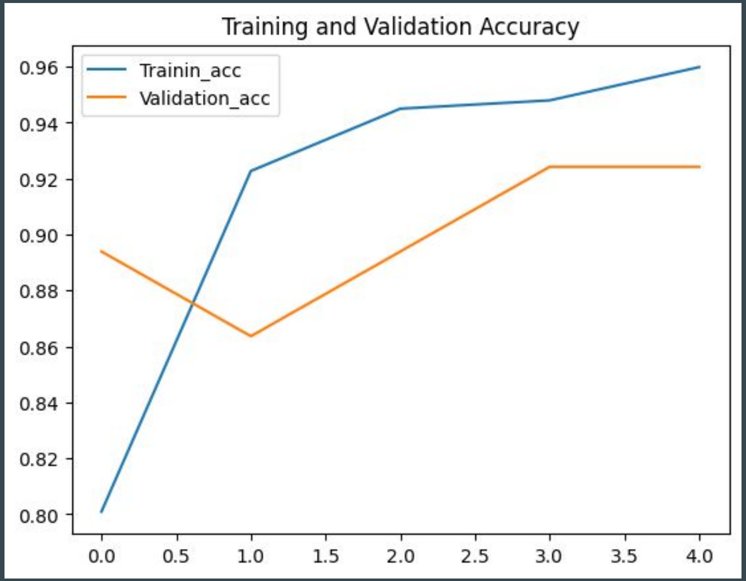
- VGG-16 architecture implemented using Keras library
    - CNN architecture specializing in object recognition
    - From Oxford in 2014
- 13 convolutional layers, 5 max pooling layers
- ~1,000,000 trainable parameters



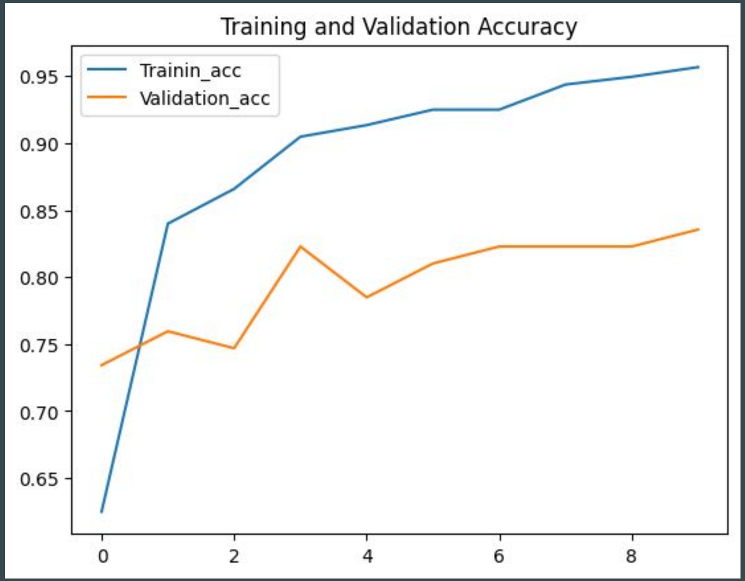

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 300, 300, 3)]     0

 block1_conv1 (Conv2D)       (None, 300, 300, 64)      1792

 block1_conv2 (Conv2D)       (None, 300, 300, 64)      36928

 block1_pool (MaxPooling2D)  (None, 150, 150, 64)      0

 block2_conv1 (Conv2D)       (None, 150, 150, 128)     73856

 block2_conv2 (Conv2D)       (None, 150, 150, 128)     147584

 block2_pool (MaxPooling2D)  (None, 75, 75, 128)       0

 block3_conv1 (Conv2D)       (None, 75, 75, 256)       295168

 block3_conv2 (Conv2D)       (None, 75, 75, 256)       590080

 block3_conv3 (Conv2D)       (None, 75, 75, 256)       590080

 block3_pool (MaxPooling2D)  (None, 37, 37, 256)       0

 block4_conv1 (Conv2D)       (None, 37, 37, 512)       1180160

 block4_conv2 (Conv2D)       (None, 37, 37, 512)       2359808

 block4_conv3 (Conv2D)       (None, 37, 37, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 18, 18, 512)       0

 block5_conv1 (Conv2D)       (None, 18, 18, 512)       2359808

 block5_conv2 (Conv2D)       (None, 18, 18, 512)       2359808

 block5_conv3 (Conv2D)       (None, 18, 18, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 9, 9, 512)         0

 average_pooling2d_3 (Avera  (None, 4, 4, 512)         0
 gePooling2D)

 flatten_3 (Flatten)         (None, 8192)              0

 dense_6 (Dense)             (None, 128)               1048704

 dropout_3 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 3)                 387

=================================================================
Total params: 15763779 (60.13 MB)
Trainable params: 1049091 (4.00 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Results - Classification

2 classes: Pothole and Plain

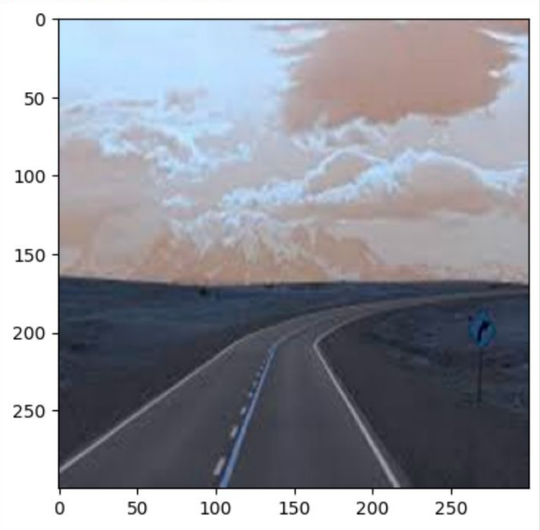


3 classes: Pothole, Animal, and Plain

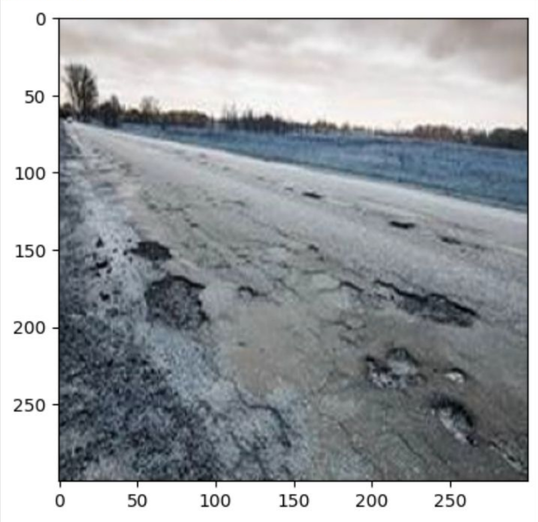


Results - Classification

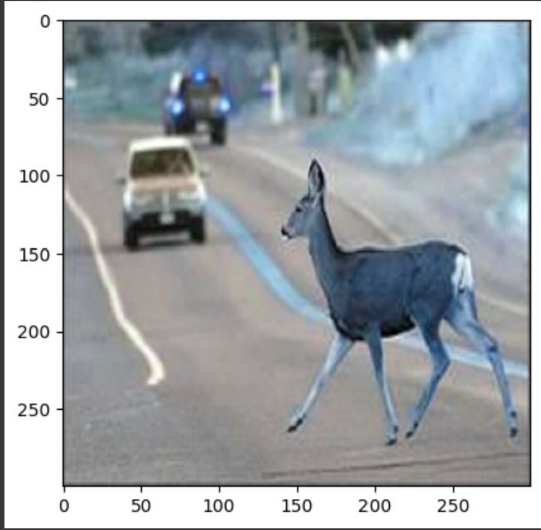
```
1/1 [==============================] - 1s 1s/step
prediction class:  Plain
```




```
1/1 [==============================] - 0s 42ms/step
prediction class:  Pothole
```

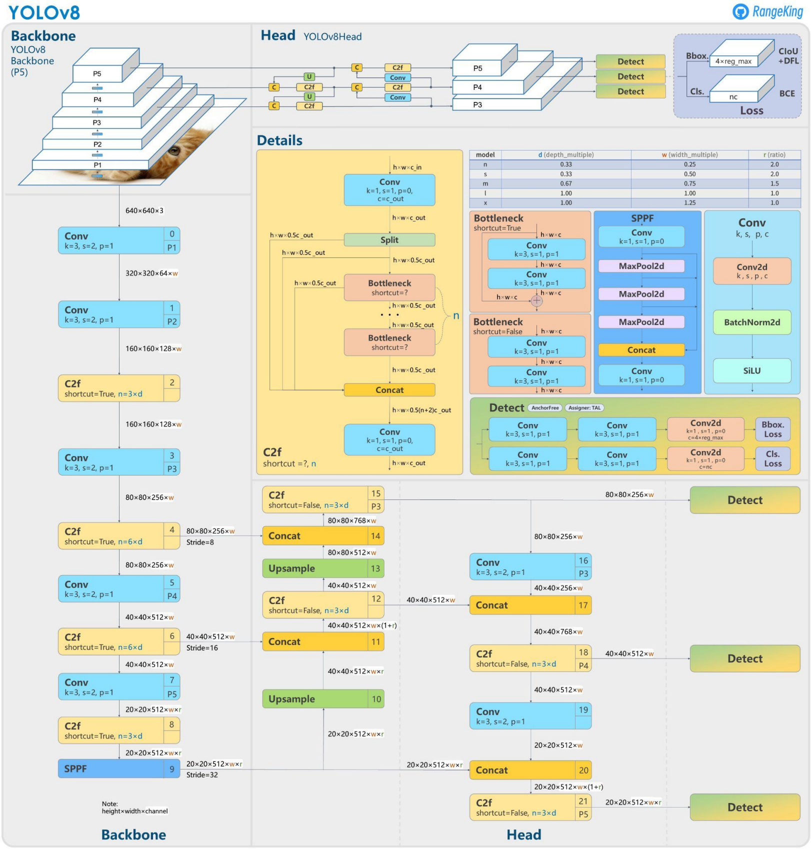



```
1/1 [==============================] - 0s 21ms/step
prediction class:  Animal
```



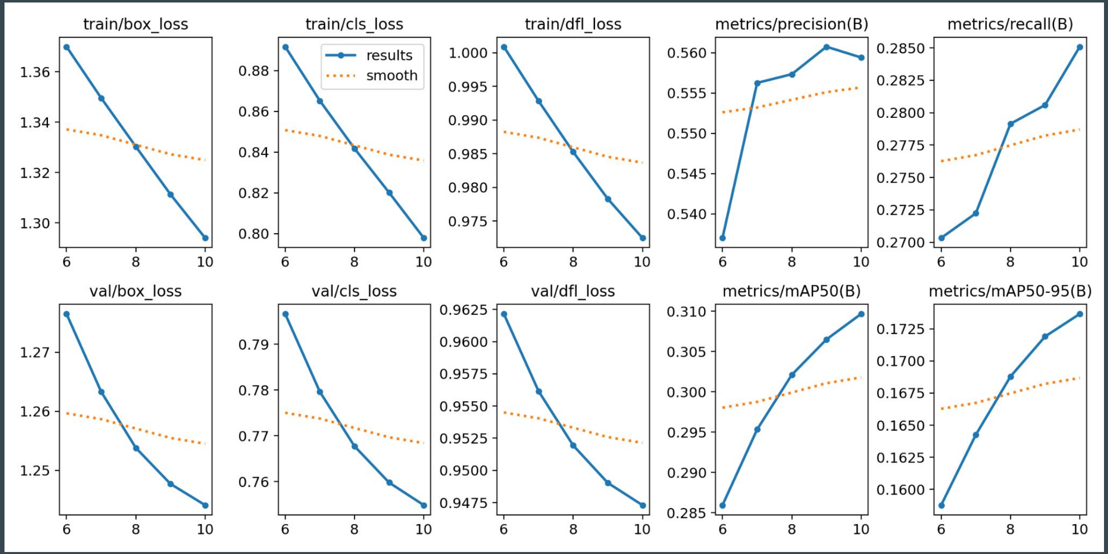
Model Architecture - Detection

- State-of-the-art deep learning model
- CNN used for Real-time object detection
- Used for its efficiency, accuracy, and easy of use
- One stage object detection and time of detection is faster and better for large datasets

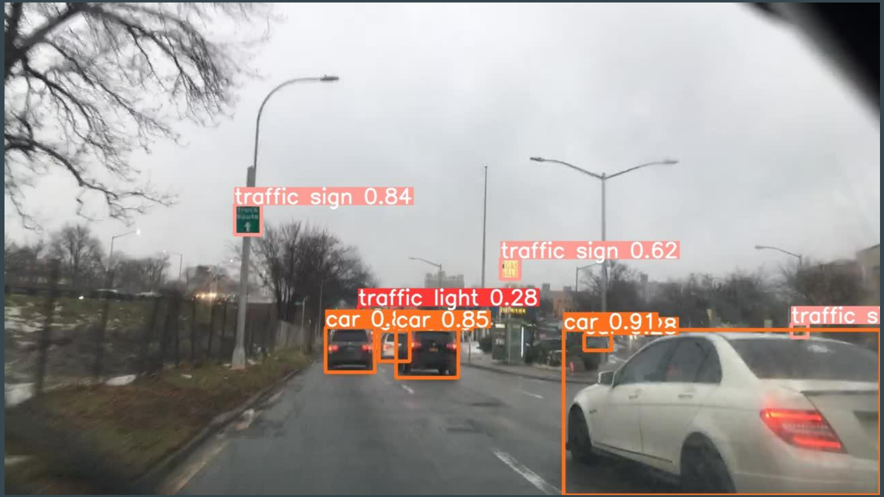


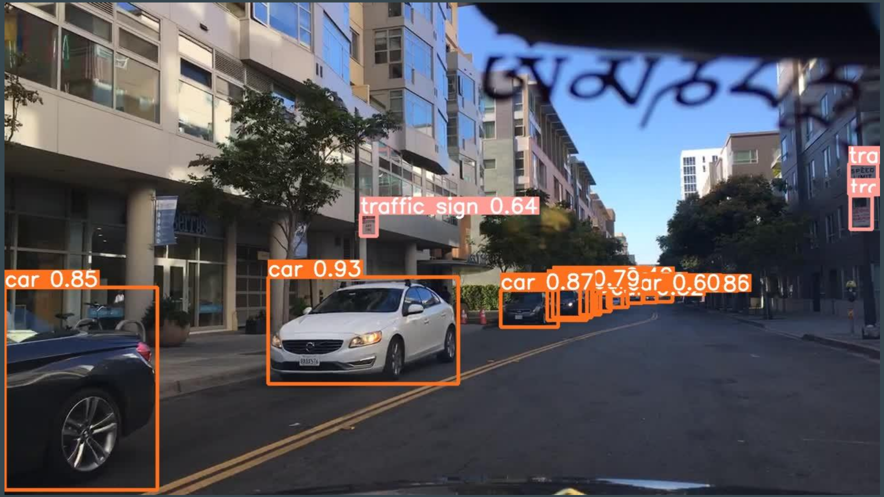
Results - Detection

More advanced results after training the YOLOv8 model using the BDD100K image datasets:



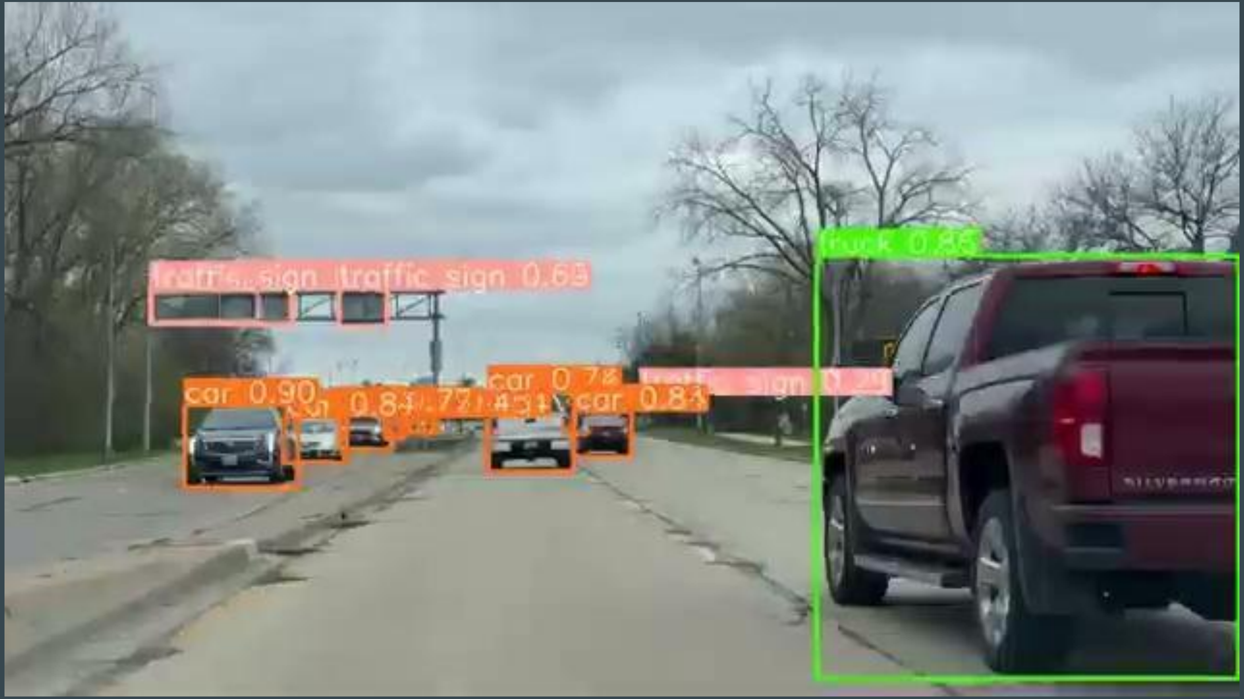
Results - Detection





Results - Detection

Verification conducted using a video captured by us on John Nolen Drive, Madison, WI.



Conclusions

- Baseline implementation through VGG-16 model
    - Achieves solid binary classification on pothole identification, and worse ternary classification
- With further hazard classification implemented by YOLOv8
    - One of the most accurate and user-friendly models, we achieve hazard classification on video efficiently.
- Possible future direction: 3D Object Detection

Thank you

Any questions?

References

[1] State of New Jersey. "Title 39: Motor Vehicles and Traffic Regulation."

[2] IEA (2023), Global EV Outlook 2023, IEA, Paris
https://www.iea.org/reports/global-ev-outlook-2023, Licence: CC BY 4.0

[3] arXiv:1807.05511 [cs.CV]