SPR-Part B
MD-21-SHA/UM/5-23

MARYLAND DEPARTMENT
OF TRANSPORTATION
STATE HIGHWAY
ADMINISTRATION

Larry Hogan
Governor
Boyd K. Rutherford
Lt. Governor
Gregory Slater
Secretary
Tim Smith, P.E.
Administrator

# MARYLAND DEPARTMENT OF TRANPORTATION
# STATE HIGHWAY ADMINISTRATION

## RESEARCH REPORT

# IMPLEMENTING MACHINE LEARNING WITH HIGHWAY DATASETS

Yunfeng Zhang, Professor, Principal Investigator
Ross Cutts, P.E., Technical Lead
Jianshu Xu, Graduate Research Assistant

UNIVERSITY OF MARYLAND

## FINAL REPORT

May 2021

This material is based upon work supported by the Federal Highway Administration under the State Planning and Research program.  Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Federal Highway Administration or the Maryland Department of Transportation.  This report does not constitute a standard, specification, or regulation.

# TECHNICAL REPORT DOCUMENTATION PAGE

| | | |
|---|---|---|
| **1. Report No.**<br>MD-21-SHA/UM/5-23 | **2. Government Accession No.** | **3. Recipient's Catalog No.** |
| **4. Title and Subtitle**<br>Implementing Machine Learning with Highway Datasets | | **5. Report Date**<br>May 2021<br>**6. Performing Organization Code** |
| **7. Author(s)**<br>Dr. Yunfeng Zhang, Professor, University of Maryland (UMD)<br>Ross Cutts P.E., Technical Lead, MDOT SHA<br>Jianshu Xu, Graduate Research Assistant, UMD | | **8. Performing Organization Report No.** |
| **9. Performing Organization Name and Address**<br>University of Maryland<br>Department of Civil and Environmental Engineering<br>College Park, Maryland 20742 | | **10. Work Unit No.**<br>**11. Contract or Grant No.**<br>SHA/UM/5-23 |
| **12. Sponsoring Agency Name and Address**<br>Maryland Department of Transportation (SPR)<br>State Highway Administration<br>Office of Policy & Research<br>707 North Calvert Street<br>Baltimore MD 21202 | | **13. Type of Report and Period Covered**<br>Final Report<br>(October 2019-April 2021)<br>**14. Sponsoring Agency Code**<br>(7120) STMD - MDOT/SHA |
| **15. Supplementary Notes** | | |

**16. Abstract**

Every year MDOT invests millions of dollars into testing geomaterials, digitizing historic records and capturing inventory and condition data. Massive amounts of tabular data, documentation and imagery, which are relevant for planning and engineering purposes, continue to be accumulated. The engineering characteristics of pavement or other materials can be estimated in the early phase of the project. Additionally, scheduling and construction can be optimized by smart decision-making assistance enabled by such machine learning models. This project enhanced the existing machine learning models with newly available data and developed and tested new machine learning models for datasets of interest including drilling and pavement data, project duration and highway right-of-way (ROW) image datasets. Various machine learning models were developed and trained for the selected highway datasets including drilling data, pavement falling weight deflectometer (FWD) data, scheduling estimates using reinforcement learning, ROW image QA/QC processing and object detection for three types of image objects, pavement core thickness datasets. Tabular data neural network models for drilling data and pavement data were trained and used for dependent variable prediction. A state-of-art object detection model using YOLO (You Only Look Once) v3 algorithm were also trained and tested for detecting traffic barrier end treatments and insect blocking in ROW images. Reinforcement learning models for drilling project schedule estimation were developed and tested using historical data records. Furthermore, random forest model was also trained and tested for one type of drilling data – groundwater depth. These machine learning models can potentially be used to assist with the decision-making process in project planning and construction and some of these models have been integrated with existing working process in MDOT SHA. Improved cost effectiveness of the agency can be achieved by enhancing analysis capabilities and improving decision-making through incorporating machine-learning into planning and engineering work processes.

| | | | |
|---|---|---|---|
| **17. Key Words**<br>Artificial intelligence, data processing, machine learning, neural networks, object detection, random forest, reinforcement learning | | **18. Distribution Statement**<br>This document is available from the Research Division upon request. | |
| **19. Security Classif. (of this report)**<br>None | **20. Security Classif. (of this page)**<br>None | **21. No. of Pages**<br>56 | **22. Price** |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

**TABLE OF CONTENTS**

LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1: INTRODUCTION

## RESEARCH PROBLEMS & BACKGROUND

Every year MDOT invests millions of dollars into testing geomaterials and thus massive amounts of engineering datasets as well as other data such as pavement and construction history data have been accumulated over a long time period, which creates an excellent opportunity for establishing deep learning models to enable reliable prediction of engineering characteristics and other desired features from the massive datasets. If implemented, not only the engineering characteristics of pavement or other materials can be estimated in the early phase of the project, but also scheduling and construction can be optimized by smart decision-making assistance enabled by validated machine learning models. Improved cost effectiveness of the agency can be achieved by enhancing analysis capabilities and improving decision-making by incorporating machine-learning into planning and engineering work processes.

For traditional supervised learning algorithms, suitable features need to be selected from raw data according to engineering experience and professional knowledge. A classifier can then be constructed, and the prediction task is converted into a classification problem which can be solved by machine learning algorithms such as artificial neural networks. Deep learning is a machine learning technique that allows computational models to learn representation of massive and complex datasets without the need for explicit identification of prevalent features. Convolutional Neural Network (CNN) is a successful deep learning algorithm that has achieved record-striking performance especially in image classification and pattern recognition in the last decade. Deep learning models can be trained to represent high-dimensional data by automatically capturing the complex relations inherent with the datasets which traditional mathematical models are difficult to describe (Goodfellow et al. 2016; LeCunn et al. 2015). Since explicit feature definition is not required in advance, deep learning fits well with the need of automated tools for highway data modeling and prediction. As an evolved form of neural networks, deep learning models such as the CNN with deeper and more sophisticated structures provide promising tools for reliable representation and prediction of highway datasets. YOLO (You Only Look Once) v3 model for object detection in this study uses CNN.

## RESEARCH OBJECTIVES

By building on MDOT SHA initial development of drilling data neural network models, this project further optimized and updated the existing neural network models with newly available data and developed and tested new machine learning models by continually updating, retraining, and optimizing the machine learning structures and hyper-parameter values for the selected highway datasets. A literature review of machine learning algorithms for such highway data modeling and prediction has also been conducted. Survey and identifying areas of opportunity for machine learning in MDOT SHA has also been conducted, which included the following datasets: drilling data, pavement FWD data, project schedule estimation using reinforcement learning, Maryland precipitation data and modeling, ROW (right-of-way) images processing and object detection for desired highway objects (e.g.,  traffic barrier end treatments in guardrail; insect blocking detection in ROW images), pavement construction history (pavement thickness data), and geologic datasets. These machine learning models can be used and updated with new data to assist with the decision-making process of MDOT SHA in project planning and construction.

## RESEARCH APPROACH

To achieve the objectives of this study, the following tasks were undertaken. The general outline for the machine learning methods adopted in this research is shown in Figure 1. This research considered the following three types of machine learning algorithms: supervised learning, machine vision, and reinforcement learning. A brief description of each machine learning method is given below and more details on why they were chosen can be found in the corresponding chapters of this report.

### *Task 1. Project Management*

The research team coordinated closely with MDOT SHA throughout the project in order to establish machine learning models for the selected highway datasets including drilling data, pavement data (FWD), scheduling estimates using reinforcement learning, QA/QC sample location identification, Maryland precipitation data and modeling, ROW image processing and object detection for desired highway objects (e.g.,  traffic barrier end treatments in guardrail), pavement construction history (thickness) datasets (Task 2) and validation and test of the newly trained machine learning models (Task 3). Quarterly progress reports were prepared and submitted. Participation in project meetings coordinated by MDOT SHA with OMT staffs were attended regularly for machine learning model application needs and data requirements.

*Task 2: Neural network model development for highway tabular datasets*

The research team (here force defined as UMD and MDOT team members) trained tabular data neural network models for selected highway datasets of interest to MDOT SHA including drilling data, pavement datasets (pavement FWD data, construction history/pavement thickness data), and used the trained neural network models for target variable prediction. The research team also did literature review to ensure the tabular data neural network model is current and has been updated for the drilling data based on parametric study of hyper-parameter values.

*Task 3: QA/QC processing ROW image data and object detection using YOLO v3 models*

The research team reviewed example ROW images and has developed deep learning based object detection models for detecting the traffic barrier end treatments and insect blocking in ROW images using transfer learning and custom training data creation.  Data preparation work including quality assurance (QA) and quality control (QC) on ROW image data, bounding box creation and labeling, and converting image data into a format suitable for YOLO v3 model use has been conducted. The research team also did literature review of recent publications in this field to ensure the adopted object detection model has the state of art performance.

The research team also developed processes to automate the QA/QC of ROW images. The research developed processes to automate the detection of blurry photos, over and under exposure, lost signal, image corruption, and insects blocking the camera lenses. The research team have tested this QA/QC method with the ROW images collected from one county in Maryland.

*Task 4: Review & development of reinforcement learning model for scheduling estimation*

The research team developed and tested a simple reinforcement learning model for drilling project schedule estimation with 212 historical records in the Keras library environment. Q-learning algorithm was adopted to build the reinforcement learning model for project duration estimation. The research team also did literature review on reinforcement learning for project schedule estimation.

*Task 5: Developing Random Forest model for Maryland precipitation data*

The research team trained random forest model for groundwater depth tabular datasets with precipitation data included as feature variable and used the trained models to predict groundwater depth at a given location. Tuning the Random Forest model for optimal hyper-parameter values was conducted. Comparison of the predicted results from Fast.ai neural network model and Random Forest model was also made by calculating the corresponding confusion matrices in this study.

*Task 6: Final Report*

The research team developed this final report that includes all deliverables and analyses as described in Tasks 2 to 5.

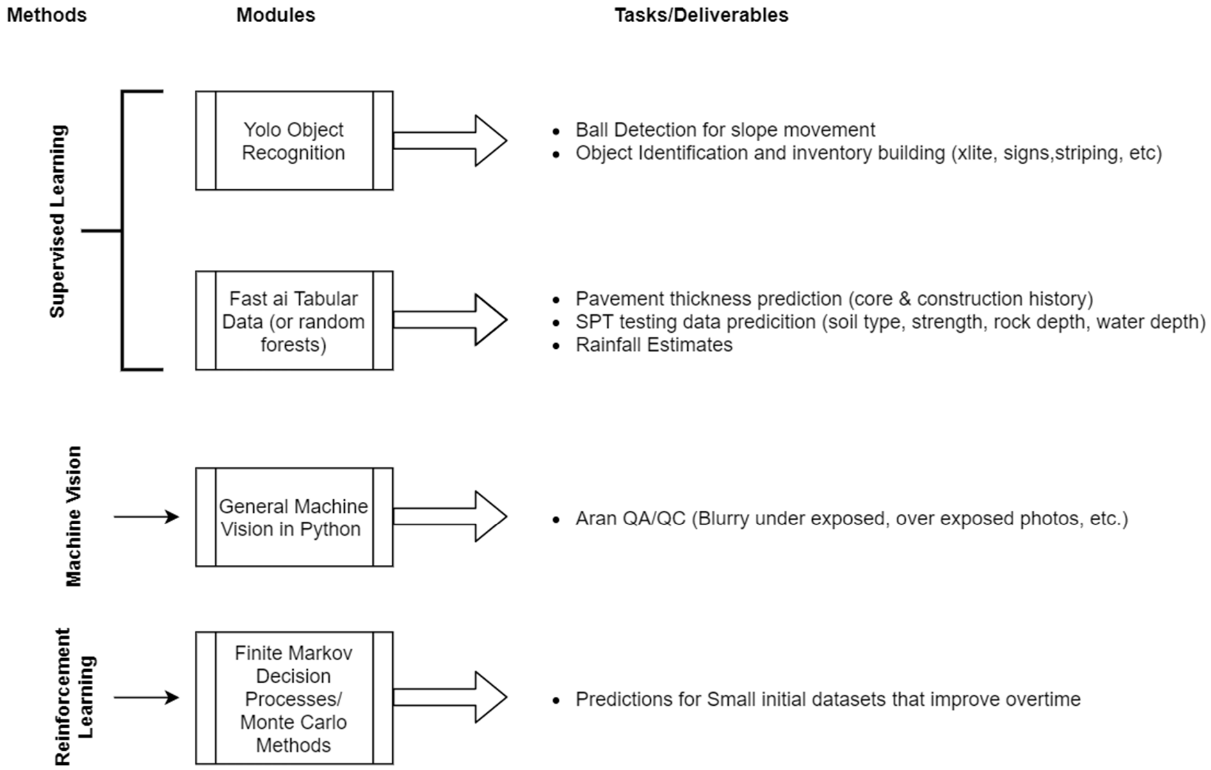

Methods
Modules
Tasks/Deliverables
Supervised Learning
Yolo Object Recognition
Ball Detection for slope movement
Object Identification and inventory building (xlite, signs,striping, etc)
Fast ai Tabular Data (or random forests)
Pavement thickness prediction (core & construction history)
SPT testing data predicition (soil type, strength, rock depth, water depth)
Rainfall Estimates
Machine Vision
General Machine Vision in Python
Aran QA/QC (Blurry under exposed, over exposed photos, etc.)
Reinforcement Learning
Finite Markov Decision Processes/ Monte Carlo Methods
Predictions for Small initial datasets that improve overtime

Figure 1. General outline for machine learning methods adopted in this research

## CHAPTER 2. NEURAL NETWORK MODEL DEVELOPMENT FOR HIGHWAY TABULAR DATASETS

The objective of this task was to train neural network models for selected highway tabular datasets of interest to MDOT SHA including drilling data, pavement datasets (pavement FWD data, pavement core thickness data), and then use the validated neural network models for dependable variable prediction. Data preparation including extracting relevant data entries from existing datasets, removing null data, filling missing values, discarding redundant data samples, normalization and converting data into acceptable format by the Fast.ai neural network models have been conducted.

In this study, the research team reviewed drilling (SPT) data based neural network model training and predictions and investigated optimization approach to the drilling data models. The research team also did literature review to ensure the tabular data neural network model is current and has been updated for the newly available drilling data.

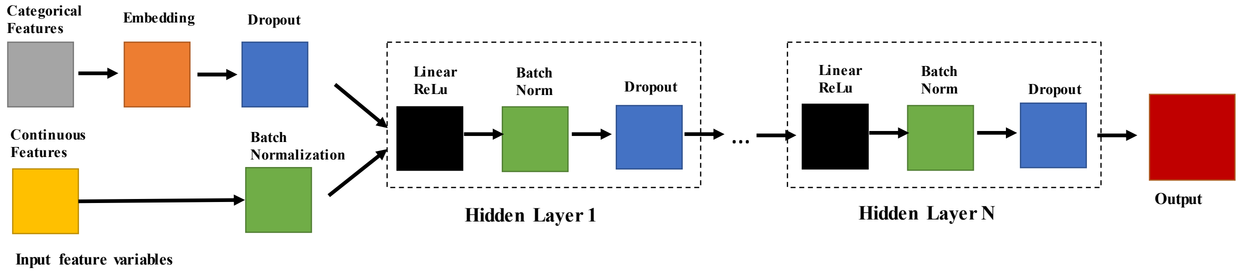
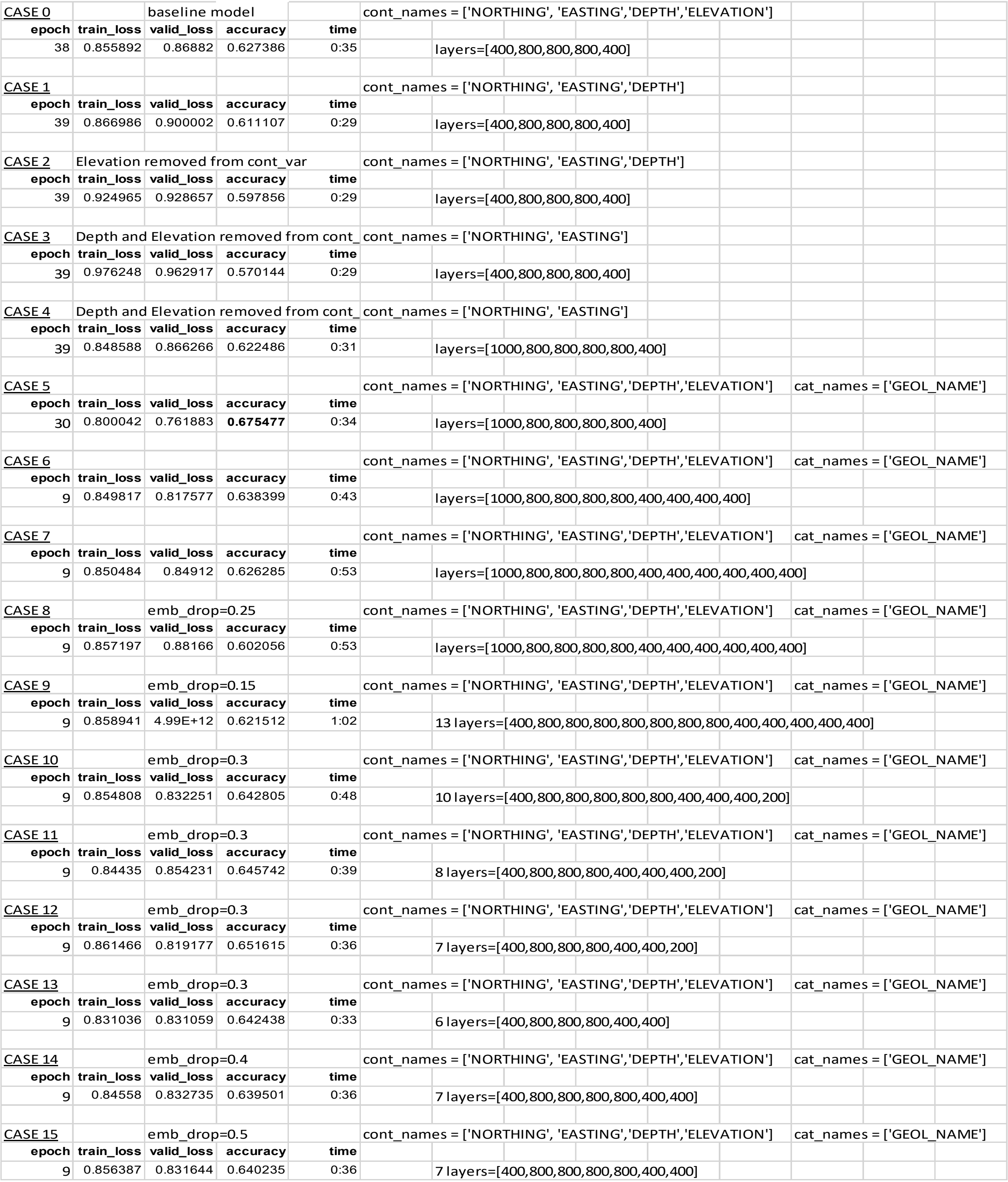


Figure 2. Architecture of feedforward neural network model for tabular data

The architecture of the adopted feedforward neural network model is shown in Figure 2. A typical hidden layer with ReLU (rectified linear unit) activation function, batch normalization, and dropout is the state of art for feedforward neural network model. Batch normalization proposed by Ioffe and Szegedy (2015) normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This ensures that the gradients are more predictive and thus allows for use of larger range of learning rates and faster network convergence (Santurkar et al. 2018). Therefore, Fast.ai-adopted neural network model is already the state of art model for tabular data.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CASE 0 | | baseline model | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 38 | 0.855892 | 0.86882 | 0.627386 | 0:35 | | layers=[400,800,800,800,400] | |
| CASE 1 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH'] | | |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 0.866986 | 0.900002 | 0.611107 | 0:29 | | layers=[400,800,800,800,400] | |
| CASE 2 | Elevation removed from cont_var | | | | cont_names = ['NORTHING', 'EASTING','DEPTH'] | | |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 0.924965 | 0.928657 | 0.597856 | 0:29 | | layers=[400,800,800,800,400] | |
| CASE 3 | Depth and Elevation removed from cont_ | | | | cont_names = ['NORTHING', 'EASTING'] | | |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 0.976248 | 0.962917 | 0.570144 | 0:29 | | layers=[400,800,800,800,400] | |
| CASE 4 | Depth and Elevation removed from cont_ | | | | cont_names = ['NORTHING', 'EASTING'] | | |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 0.848588 | 0.866266 | 0.622486 | 0:31 | | layers=[1000,800,800,800,800,400] | |
| CASE 5 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 30 | 0.800042 | 0.761883 | **0.675477** | 0:34 | | layers=[1000,800,800,800,800,400] | |
| CASE 6 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.849817 | 0.817577 | 0.638399 | 0:43 | | layers=[1000,800,800,800,800,400,400,400,400] | |
| CASE 7 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.850484 | 0.84912 | 0.626285 | 0:53 | | layers=[1000,800,800,800,800,400,400,400,400,400,400] | |
| CASE 8 | | emb_drop=0.25 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.857197 | 0.88166 | 0.602056 | 0:53 | | layers=[1000,800,800,800,800,400,400,400,400,400,400] | |
| CASE 9 | | emb_drop=0.15 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.858941 | 4.99E+12 | 0.621512 | 1:02 | | 13 layers=[400,800,800,800,800,800,800,800,400,400,400,400,400] | |
| CASE 10 | | emb_drop=0.3 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.854808 | 0.832251 | 0.642805 | 0:48 | | 10 layers=[400,800,800,800,800,800,400,400,400,200] | |
| CASE 11 | | emb_drop=0.3 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.84435 | 0.854231 | 0.645742 | 0:39 | | 8 layers=[400,800,800,800,400,400,400,200] | |
| CASE 12 | | emb_drop=0.3 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.861466 | 0.819177 | 0.651615 | 0:36 | | 7 layers=[400,800,800,800,400,400,200] | |
| CASE 13 | | emb_drop=0.3 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.831036 | 0.831059 | 0.642438 | 0:33 | | 6 layers=[400,800,800,800,400,400] | |
| CASE 14 | | emb_drop=0.4 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.84558 | 0.832735 | 0.639501 | 0:36 | | 7 layers=[400,800,800,800,800,400,400] | |
| CASE 15 | | emb_drop=0.5 | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEVATION'] | | cat_names = ['GEOL_NAME'] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 9 | 0.856387 | 0.831644 | 0.640235 | 0:36 | | 7 layers=[400,800,800,800,800,400,400] | |

Figure 3. Output and feature variables of the neural network model training for Grainsize data

In training the neural network model, it was optimized through an iteration process termed back-propagation. The prediction accuracy derives from the neural network models trained with large number of data samples that best represents the inherent complex relationship in the selected highway datasets. In this study, deep learning model is treated as a classification model and the training target is to classify the data samples into corresponding output values (or buckets with a range of values or bins) of highway datasets based on the given input. The output (e.g., engineering characteristics of geomaterials or pavement) of the machine learning model are thus discrete classes. This can be tackled as a classification problem with each class representing one possible value (or range) for the output of interest. Forecasting from the well-trained neural network model for a given data sample (x) is made by choosing the class with the highest probability. The training of the neural network model is to search for the optimal model parameters that has the least cost function value. This is achieved through an iteration process called gradient descent and its extensions. One extension of the gradient descent for enhanced training performance is the stochastic gradient descent (SGD) with momentum. The term of stochastic means the calculation of the cost function, is based on a randomly selected sub-dataset X rather than the entire training set. The sub-dataset X is called mini-batch, and the number of individual data samples in X is called the mini-batch size. In this project, the influencing factors (feature variables) for the selected highway datasets were investigated. Data cleansing procedure was also performed to discard redundant data points as well as removing bad data for each feature table.

Figure 3 shows the training outputs from a parametric study of the neural network models for the Grainsize data. Outputs include validation accuracy, train loss, validation loss, and training time. In the parametric study, different combination of feature variables including site location (northing, easting) information, depth, elevation, and geologic name, were considered. Dependent variable is the grainsize bucket. Different number of neural network layers were also tested. 90% of this dataset was randomly allocated for training and the remaining 10% data was used for validation. The validation accuracy level from this parametric study varied from 0.602 to 0.675, as shown in Figure 3.

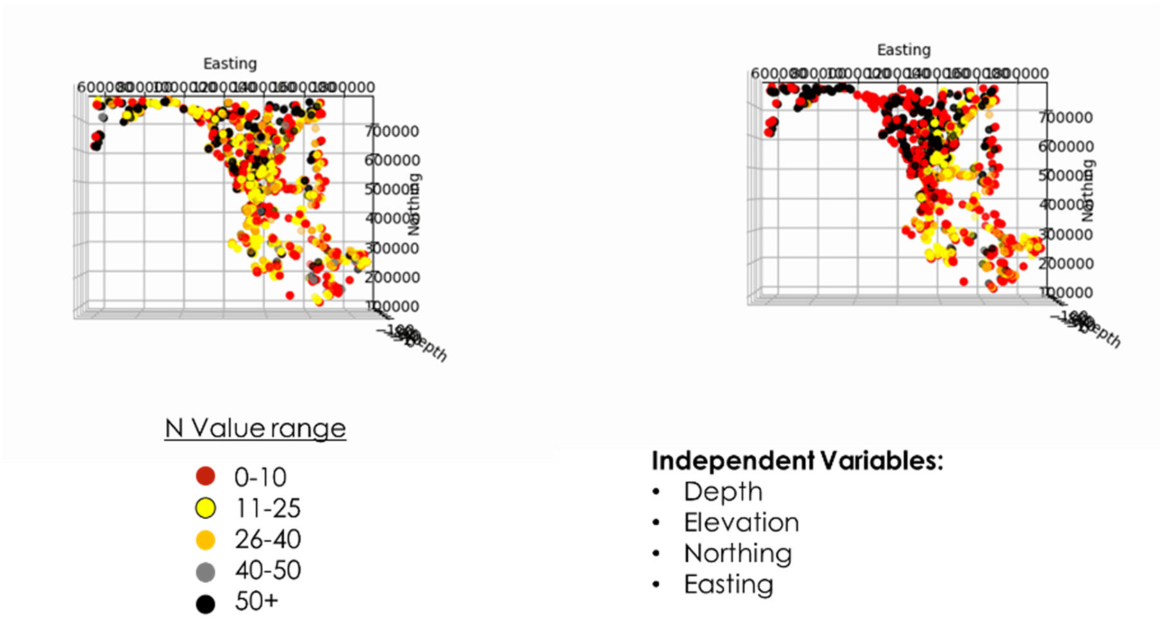
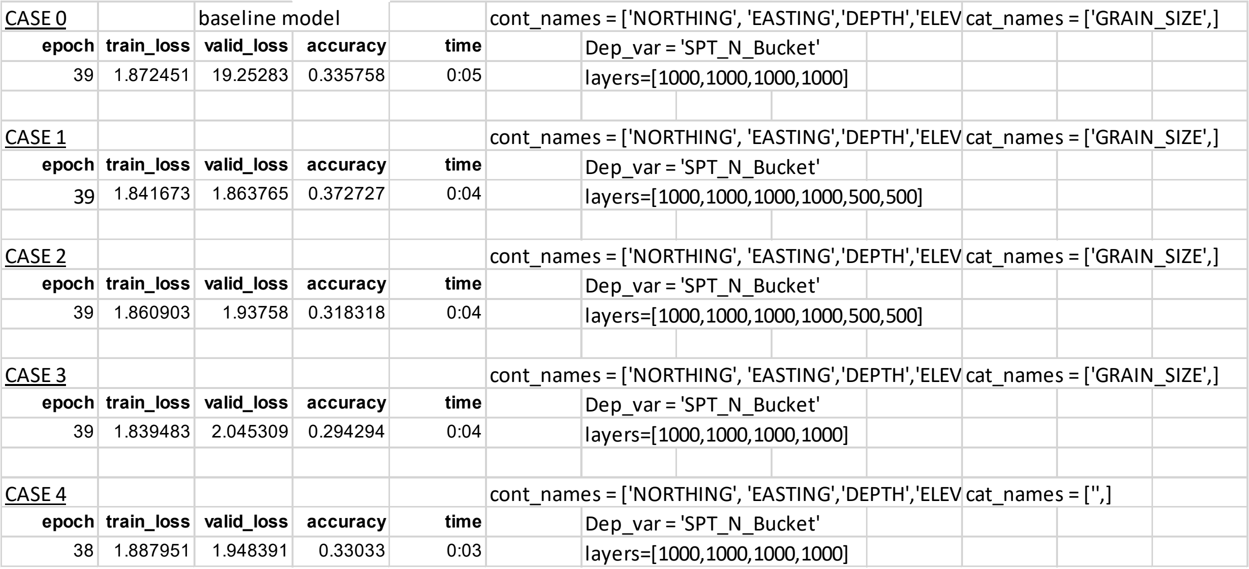


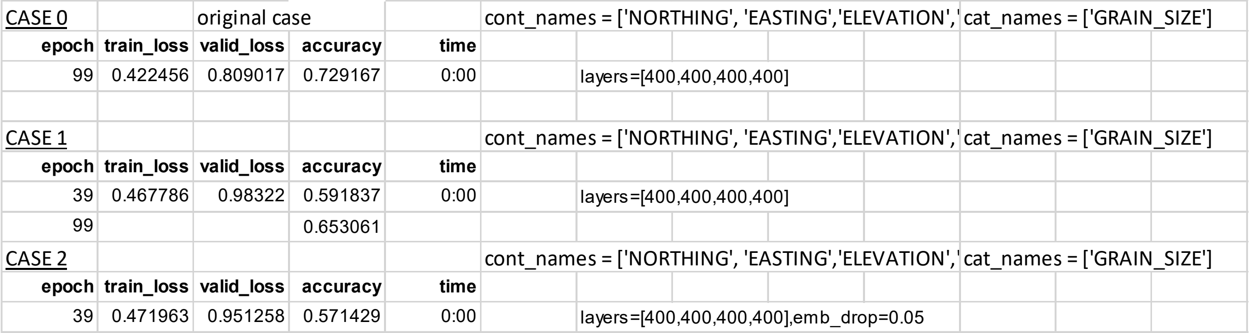
Figure 4. Comparing the real SPT N bucketed data compared to the model derived SPT N buckets.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CASE 0 | | baseline model | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEV | | cat_names = ['GRAIN_SIZE',] |
| epoch | train_loss | valid_loss | accuracy | time | | Dep_var = 'SPT_N_Bucket' | |
| 39 | 1.872451 | 19.25283 | 0.335758 | 0:05 | | layers=[1000,1000,1000,1000] | |
| | | | | | | | |
| CASE 1 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEV | | cat_names = ['GRAIN_SIZE',] |
| epoch | train_loss | valid_loss | accuracy | time | | Dep_var = 'SPT_N_Bucket' | |
| 39 | 1.841673 | 1.863765 | 0.372727 | 0:04 | | layers=[1000,1000,1000,1000,500,500] | |
| | | | | | | | |
| CASE 2 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEV | | cat_names = ['GRAIN_SIZE',] |
| epoch | train_loss | valid_loss | accuracy | time | | Dep_var = 'SPT_N_Bucket' | |
| 39 | 1.860903 | 1.93758 | 0.318318 | 0:04 | | layers=[1000,1000,1000,1000,500,500] | |
| | | | | | | | |
| CASE 3 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEV | | cat_names = ['GRAIN_SIZE',] |
| epoch | train_loss | valid_loss | accuracy | time | | Dep_var = 'SPT_N_Bucket' | |
| 39 | 1.839483 | 2.045309 | 0.294294 | 0:04 | | layers=[1000,1000,1000,1000] | |
| | | | | | | | |
| CASE 4 | | | | | cont_names = ['NORTHING', 'EASTING','DEPTH','ELEV | | cat_names = ['',] |
| epoch | train_loss | valid_loss | accuracy | time | | Dep_var = 'SPT_N_Bucket' | |
| 38 | 1.887951 | 1.948391 | 0.33033 | 0:03 | | layers=[1000,1000,1000,1000] | |

Figure 5. Output and feature variables of the neural network model trained for SPT N data

Figure 5 shows the training outputs from a parametric study of the neural network models for the SPT N data in the Fast.ai environment (Fast.ai 2020). Outputs include validation accuracy, train loss, validation loss, and training time. In the parametric study, different combination of feature variables including site location (northing, easting) information, depth, elevation, and grain size, were considered. Dependent variable is the SPT N bucket. Different number of neural network layers were also tested. 90% of this dataset was randomly allocated for training and the remaining 10% data was used for validation. The validation accuracy level from this parametric study varied from 0.294 to 0.373, as shown in Figure 5. Figure 4 above shows a comparison of the real value and neural network model derived value of SPT N bucketed data by visualizing the data points in the Northing-Easting map of Maryland.

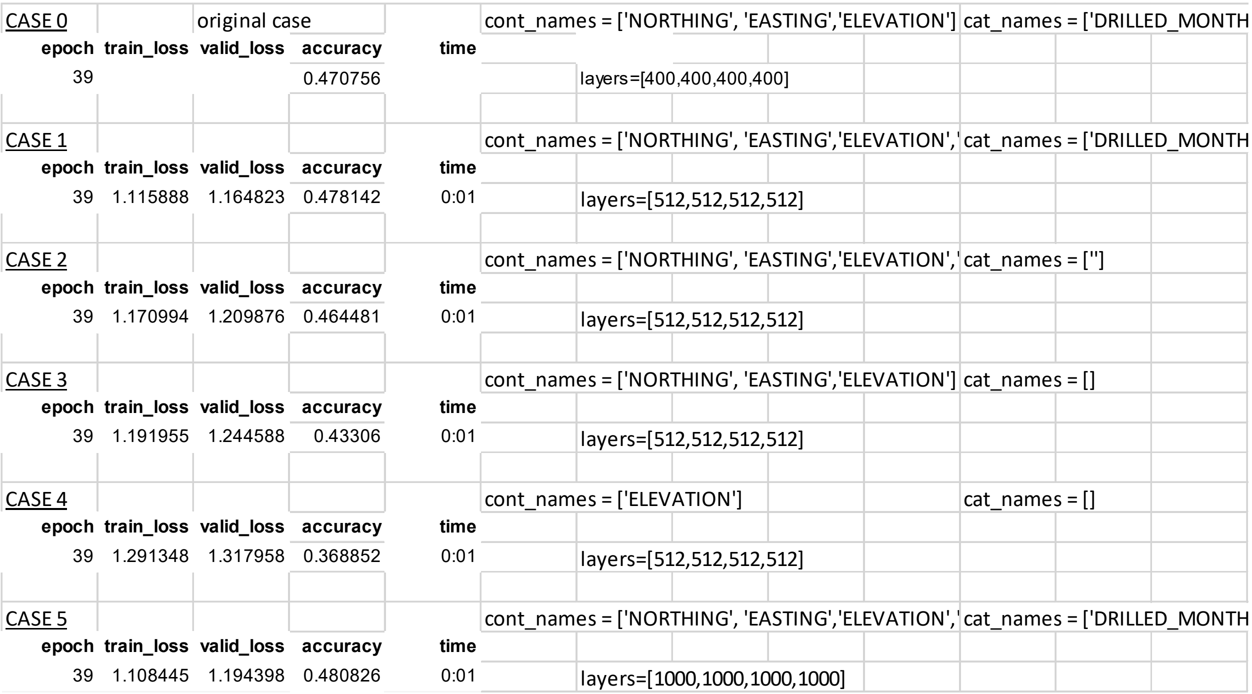


| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CASE 0 | | original case | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | | | cat_names = ['GRAIN_SIZE'] | | |
| **epoch** | **train_loss** | **valid_loss** | **accuracy** | **time** | | | | | | | |
| 99 | 0.422456 | 0.809017 | 0.729167 | 0:00 | | layers=[400,400,400,400] | | | | | |
| | | | | | | | | | | | |
| CASE 1 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | | | cat_names = ['GRAIN_SIZE'] | | |
| **epoch** | **train_loss** | **valid_loss** | **accuracy** | **time** | | | | | | | |
| 39 | 0.467786 | 0.98322 | 0.591837 | 0:00 | | layers=[400,400,400,400] | | | | | |
| 99 | | | 0.653061 | | | | | | | | |
| CASE 2 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | | | cat_names = ['GRAIN_SIZE'] | | |
| **epoch** | **train_loss** | **valid_loss** | **accuracy** | **time** | | | | | | | |
| 39 | 0.471963 | 0.951258 | 0.571429 | 0:00 | | layers=[400,400,400,400],emb_drop=0.05 | | | | | |

Figure 6. Output and feature variables of the neural network model trained for SWM Infiltration Pass/Fail data

Figure 6 shows the training outputs from a parametric study of the neural network models for the SWM infiltration data. Outputs include validation accuracy, train loss, validation loss, and training time. Four hidden layers were used. Dependent variable is the SWM infiltrates Pass or Fail (two classes). 90% of this dataset was used for training and the remaining 10% data was randomly allocated for validation. It is noted that this dataset has only 491 samples, fewer than other datasets. The validation accuracy level from this parametric study varied from 0.571 to 0.729, as shown in Figure 6.

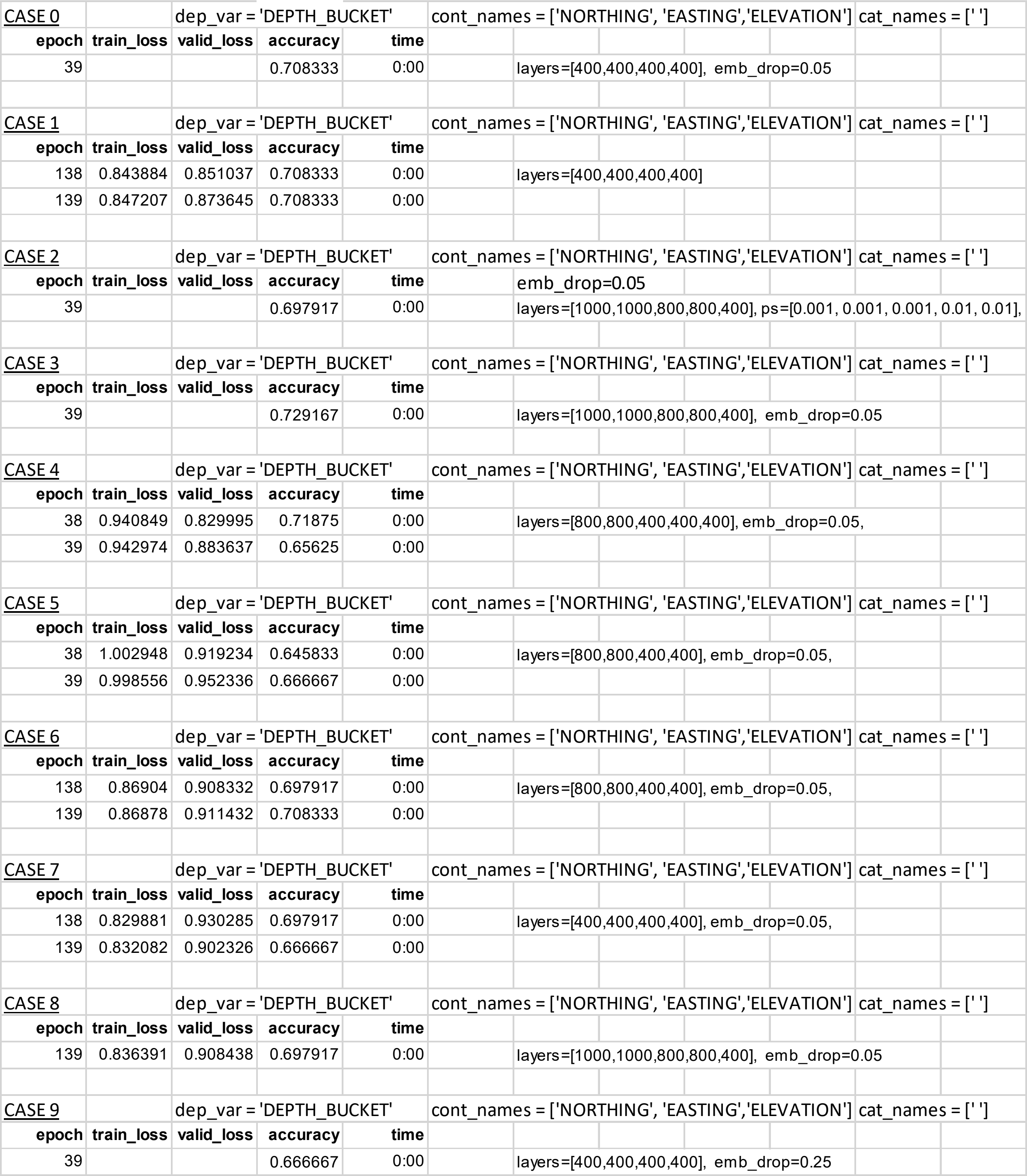
Figure 7 shows the training outputs from a parametric study of the neural network models for the water depth data. Outputs include validation accuracy, train loss, validation loss, and training time. In the parametric study, different combination of feature variables including site location (northing, easting) information, elevation, and drilled month, were considered. Dependent variable is the water depth bucket. Different number of neural network layers were also tested. 90% of this dataset was used for training and the remaining 10% data was randomly allocated for validation. The validation accuracy level from this parametric study varied from 0.367 to 0.481, as shown in Figure 7.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CASE 0 | | original case | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = ['DRILLED_MONTH |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | | | 0.470756 | | | layers=[400,400,400,400] | |
| | | | | | | | |
| CASE 1 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | cat_names = ['DRILLED_MONTH |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 1.115888 | 1.164823 | 0.478142 | 0:01 | | layers=[512,512,512,512] | |
| | | | | | | | |
| CASE 2 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | cat_names = [''] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 1.170994 | 1.209876 | 0.464481 | 0:01 | | layers=[512,512,512,512] | |
| | | | | | | | |
| CASE 3 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 1.191955 | 1.244588 | 0.43306 | 0:01 | | layers=[512,512,512,512] | |
| | | | | | | | |
| CASE 4 | | | | | cont_names = ['ELEVATION'] | | cat_names = [] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 1.291348 | 1.317958 | 0.368852 | 0:01 | | layers=[512,512,512,512] | |
| | | | | | | | |
| CASE 5 | | | | | cont_names = ['NORTHING', 'EASTING','ELEVATION',' | | cat_names = ['DRILLED_MONTH |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | 1.108445 | 1.194398 | 0.480826 | 0:01 | | layers=[1000,1000,1000,1000] | |

Figure 7. Output and feature variables of the neural network model trained for Water depth data

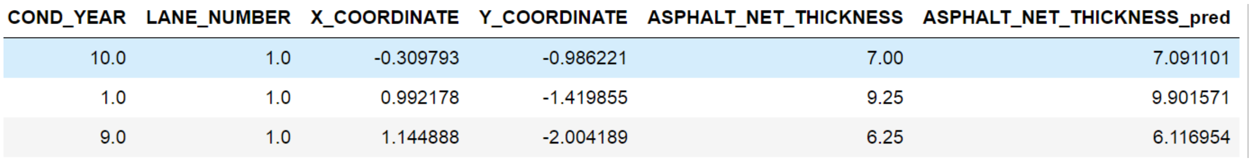
Figure 8 shows the training outputs from a parametric study of the neural network models for the refusal depth data. Outputs include validation accuracy, train loss, validation loss, and training time. In the parametric study, a single combination of feature variables including site location (northing, easting) information, elevation, were considered. Dependent variable is the refusal depth bucket. Different number of neural network layers were also tested. 90% of this dataset was randomly allocated for training and the remaining 10% data was used for validation. The validation accuracy level from this parametric study varied from 0.645 to 0.729, as shown in Figure 8.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CASE 0 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | | | 0.708333 | 0:00 | | layers=[400,400,400,400], emb_drop=0.05 | |
| | | | | | | | |
| CASE 1 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 138 | 0.843884 | 0.851037 | 0.708333 | 0:00 | | layers=[400,400,400,400] | |
| 139 | 0.847207 | 0.873645 | 0.708333 | 0:00 | | | |
| | | | | | | | |
| CASE 2 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | emb_drop=0.05 | |
| 39 | | | 0.697917 | 0:00 | | layers=[1000,1000,800,800,400], ps=[0.001, 0.001, 0.001, 0.01, 0.01], | |
| | | | | | | | |
| CASE 3 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | | | 0.729167 | 0:00 | | layers=[1000,1000,800,800,400], emb_drop=0.05 | |
| | | | | | | | |
| CASE 4 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 38 | 0.940849 | 0.829995 | 0.71875 | 0:00 | | layers=[800,800,400,400,400], emb_drop=0.05, | |
| 39 | 0.942974 | 0.883637 | 0.65625 | 0:00 | | | |
| | | | | | | | |
| CASE 5 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 38 | 1.002948 | 0.919234 | 0.645833 | 0:00 | | layers=[800,800,400,400], emb_drop=0.05, | |
| 39 | 0.998556 | 0.952336 | 0.666667 | 0:00 | | | |
| | | | | | | | |
| CASE 6 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 138 | 0.86904 | 0.908332 | 0.697917 | 0:00 | | layers=[800,800,400,400], emb_drop=0.05, | |
| 139 | 0.86878 | 0.911432 | 0.708333 | 0:00 | | | |
| | | | | | | | |
| CASE 7 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 138 | 0.829881 | 0.930285 | 0.697917 | 0:00 | | layers=[400,400,400,400], emb_drop=0.05, | |
| 139 | 0.832082 | 0.902326 | 0.666667 | 0:00 | | | |
| | | | | | | | |
| CASE 8 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 139 | 0.836391 | 0.908438 | 0.697917 | 0:00 | | layers=[1000,1000,800,800,400], emb_drop=0.05 | |
| | | | | | | | |
| CASE 9 | | dep_var = 'DEPTH_BUCKET' | | | cont_names = ['NORTHING', 'EASTING','ELEVATION'] | | cat_names = [' '] |
| epoch | train_loss | valid_loss | accuracy | time | | | |
| 39 | | | 0.666667 | 0:00 | | layers=[400,400,400,400], emb_drop=0.25 | |

Figure 8. Output and feature variables of the neural network model trained for Refusal depth data

The research team developed tabular regression model and performed training for hot mix asphalt (HMA) and concrete thickness prediction and used these model predicted HMA and concrete thickness values (not buckets like other classification models) for inclusion into the FWD dataset. In training the pavement core thickness model, the following feature variables were adopted: Categorical variables: 'ROUTE_NAME', 'COUNTY', 'COND_YEAR', 'LANE_NUMBER'; Continuous variables: 'X_COORDINATE’, 'Y_COORDINATE’; Dependent variables (i.e., target variable to predict): ’CONCRETE_NET_THICKNESS’ or 'ASPHALT_NET_THICKNESS’. It is worth noting that for the initial pavement thickness classification model, the following bucket sizes were used for HMA as ['0-2','2-4','4-6','6-8','8-10','10-12','12-14', '14-16','16-18','18-20'] (unit: inch), while the bucket size for concrete pavement thickness classification model is ['4-5','5-6','6-7','7-8','8-9','9-10'] (unit: inch). These lists were created to cover the range of data and also keep the accuracy of predictions at a high level. For the regression model, the dependent variable is a continuous variable and thus the output value from the model prediction was round off to nearest 0.25 inches. The prediction accuracy after 20 training epochs was 0.964 and 0.998 for HMA and Concrete pavement thickness models respectively. This high accuracy value over 0.94 was because of the large data size used for training and relatively simple relationship hidden in the pavement thickness data. Figure 7 below demonstrate this accuracy by showing three samples of HMA pavement true value and predicted value from trained neural network tabular regression model. Predicted thickness values from these pre-trained regression model for HMA and concrete thickness were intended to be used as Feature Variables in subsequent falling weight deflectometer (FWD) data model training. The distribution is similar between pavement thickness file and FWD data file.



| COND_YEAR | LANE_NUMBER | X_COORDINATE | Y_COORDINATE | ASPHALT_NET_THICKNESS | ASPHALT_NET_THICKNESS_pred |
|---|---|---|---|---|---|
| 10.0 | 1.0 | -0.309793 | -0.986221 | 7.00 | 7.091101 |
| 1.0 | 1.0 | 0.992178 | -1.419855 | 9.25 | 9.901571 |
| 9.0 | 1.0 | 1.144888 | -2.004189 | 6.25 | 6.116954 |

Figure 9. Comparison of HMA pavement true value and predicted value from trained neural network tabular regression model

The following metrics were also used for evaluation of the regression model performance,

- RMSE: The Root Mean Squared Error is the standard deviation of the errors/residuals. It tells us the ‘Goodness of Fit’ of a model. The lower the value of RMSE the better the model.

- R2score: The R-Squared metric also called the coefficient of determination is used to understand the variation in the dependent variable(y) and the independent variable(X). The closer the value of R-Squared is to one, the better the model

| | RMSE | R2_SCORE |
|---|---|---|
| HMA | 0.514 | 0.964 |
| Concrete | 0.059 | 0.997 |

Figure 10. RMSE and R2score values of HMA and Concrete pavement thickness regression models

It is clear that the high accuracy observed above in the training outputs of HMA and Concrete pavement thickness regression model are demonstrated in Figure 11 again for the high RMSE and R2score values greater than 0.96. Therefore, it is concluded that the tabular data regression neural network model can be used for HMA and Concrete thickness prediction and the model provides fairly good predictions in comparison to real data. This can also be seen in Figure 11, which shows a strong correlation between the predicted concrete pavement thickness values and corresponding true values.

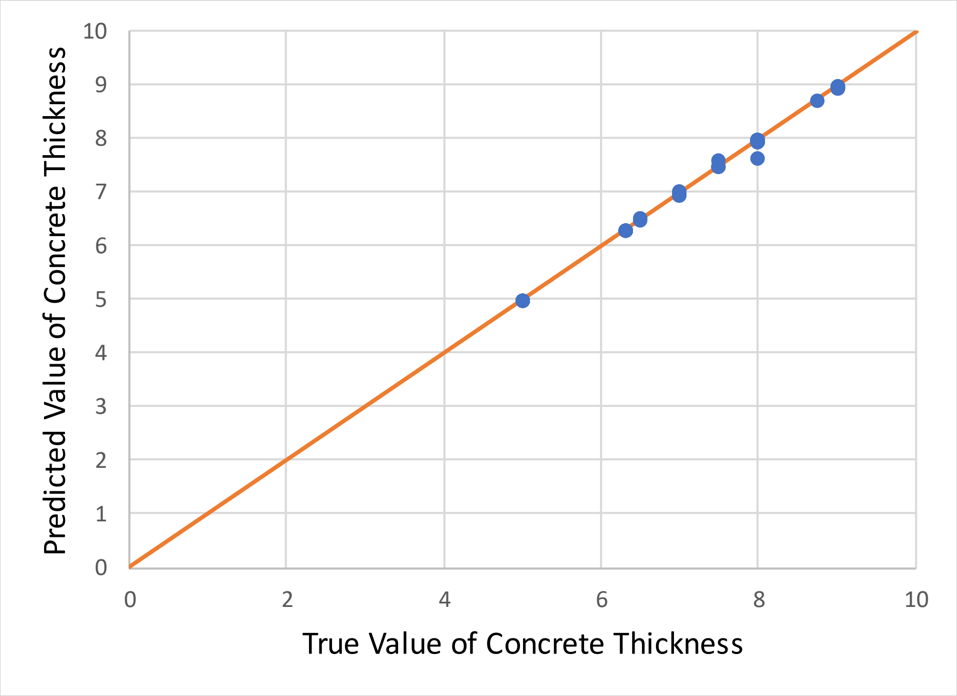
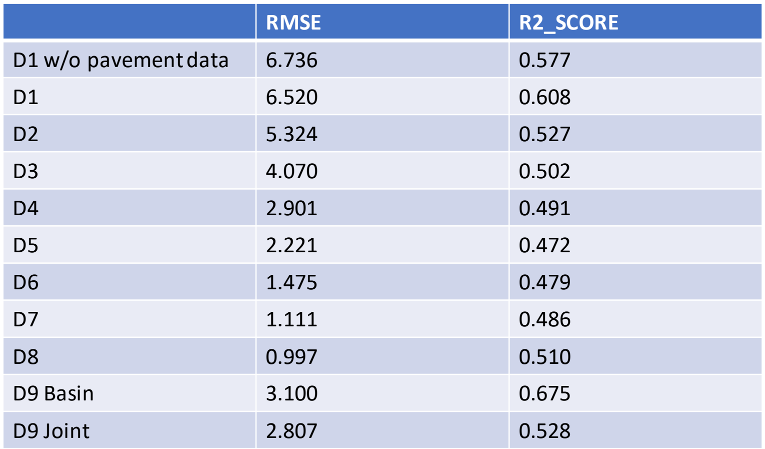


Figure 11. Scatter plot of Concrete Net Thickness prediction (horizontal axis is true value)

The research team developed tabular data neural network regression model and training for FWD data and also updated the prediction of tabular classification model training and prediction for FWD data with predicted pavement thickness data included. For training the FWD prediction regression model, the following variables were included: Categorical variables: ‘DIRECTION', 'ROUTE_NAME', 'TEST_SETUP’; Continuous variables: 'X_COORDINATE', 'Y_COORDINATE’, 'HMA_Pred', 'Concrete_Pred’; Dependent variables: D1-D8, D9 Basin, D9 Joint. From Figure 12 below, the FWD regression models showed RMSE values lower than 7 and had R2_score values falling within a range of 0.48 to 0.60. The model achieved a moderate accuracy with regard to the RMSE and R2 score. This moderate accuracy can also be seen in Figure 13, which shows some correlation between the predicted FWD D1 to D3 values and corresponding true values.



| | RMSE | R2_SCORE |
|---|---|---|
| D1 w/o pavement data | 6.736 | 0.577 |
| D1 | 6.520 | 0.608 |
| D2 | 5.324 | 0.527 |
| D3 | 4.070 | 0.502 |
| D4 | 2.901 | 0.491 |
| D5 | 2.221 | 0.472 |
| D6 | 1.475 | 0.479 |
| D7 | 1.111 | 0.486 |
| D8 | 0.997 | 0.510 |
| D9 Basin | 3.100 | 0.675 |
| D9 Joint | 2.807 | 0.528 |

Figure 12. RMSE and R2score values of FWD regression models

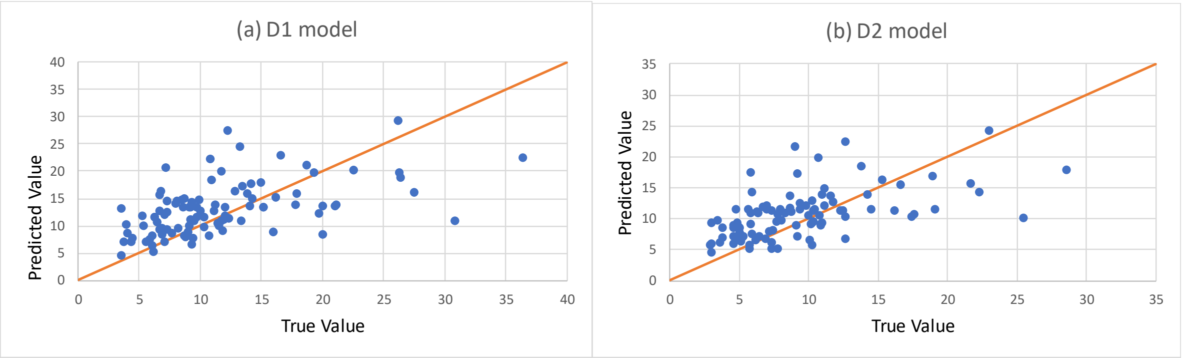
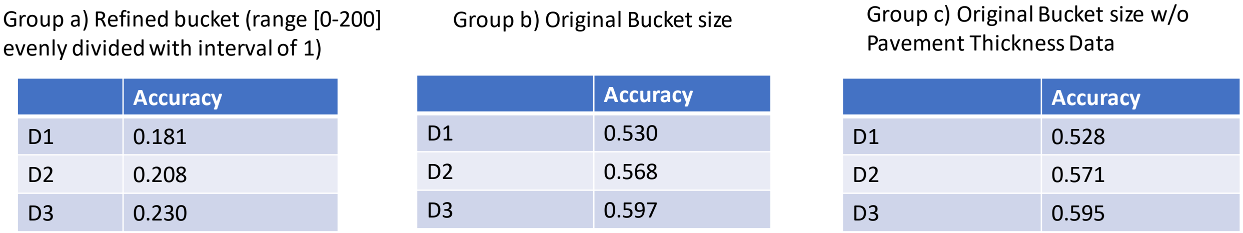


Figure 13. Scatter plot of predicted and true value of FWD data (only D1, D2 values shown)

FWD classification models were retrained after adding the predicted pavement thickness to feature variable list. In the FWD classification model, the following variables were adopted: Categorical variables: 'DIRECTION','ROUTE_NAME','TEST_SETUP'; Continuous variables: 'X_COORDINATE', 'Y_COORDINATE', 'HMA_Pred',' Concrete_Pred'; Dependent variables: D1, D2, D3 respectively. Two bucket sizes were considered: Group a) with refined bucket size: evenly divided bucket in range [0-200] with small interval of 1; Group b) Original Bucket size. Training accuracy results for these FWD classification models after 20 epochs are shown in Figure 14.



Group a) Refined bucket (range [0-200] evenly divided with interval of 1)

| | Accuracy |
|---|---|
| D1 | 0.181 |
| D2 | 0.208 |
| D3 | 0.230 |

Group b) Original Bucket size

| | Accuracy |
|---|---|
| D1 | 0.530 |
| D2 | 0.568 |
| D3 | 0.597 |

Group c) Original Bucket size w/o Pavement Thickness Data

| | Accuracy |
|---|---|
| D1 | 0.528 |
| D2 | 0.571 |
| D3 | 0.595 |

Figure 14. Training accuracy of three FWD classification models after adding predicted pavement thickness to the training data feature list

From these results, the following conclusions can be drawn: Shrinking the bucket size to '1' is not recommended as it causes significant drop in accuracy. Including the HMA thickness and Pavement thickness as continuous feature variables did not appear to improve the accuracy of the FWD classification model.

# CHAPTER 3: QA/QC PROCESSING IMAGE DATA AND OBJECT DETECTION USING YOLO V3 MODEL

In this study, the research team reviewed example ROW images and remote monitoring camera data. The research team developed deep learning based object detection models for detecting the traffic barrier end treatment and insect blocking in ROW images using transfer learning and custom training data creation. The research team also did literature review of recent publications in this field to ensure the adopted object detection model has the state of art performance.

Table 1. Comparison of object detection algorithms (adapted from Redmon and Farhadi 2018)

| | backbone | AP | $AP_{50}$ |
|---|---|---|---|
| **Two-stage methods** | | | |
| Faster R-CNN+++ | ResNet-101-C4 | 34.9 | 55.7 |
| Faster R-CNN w FPN | ResNet-101-FPN | 36.2 | 59.1 |
| Faster R-CNN by G-RMI | Inception-ResNet-v2 | 34.7 | 55.5 |
| Faster R-CNN w TDM | Inception-ResNet-v2-TDM | 36.8 | 57.7 |
| **One-stage methods** | | | |
| SSD513 | ResNet-101-SSD | 31.2 | 50.4 |
| DSSD513 | ResNet-101-DSSD | 33.2 | 53.3 |
| RetinaNet | ResNeXt-101-FPN | 40.8 | 61.1 |
| YOLOV3 608 x 608 | Darknet-53 | 33.0 | 57.9 |

Note: AP = Average Precision. Details of AP can be found later in this chapter.

Object detection involves identifying the presence, location, and type of one or more specified objects in a given picture or video images. It builds upon methods for object recognition, object localization, and object classification. In recent years, deep learning techniques are achieving state-of-the-art results for object detection, such as on standard benchmark datasets and in computer vision competitions. The computer based statistical model created by deep learning with convolutional neural networks can attain state-of-the-art accuracy, sometimes exceeding human-level performance with proven outstanding in image classification, segmentation, and object detection (Lawal 2020). Notable is the "You Only Look Once," or YOLO, family of Convolutional Neural Networks that achieve near state-of-the-art results with a single end-to-end model that can perform object detection in real-time (Redmon and Farhadi 2018). YOLO v3 makes detection at three different scales and extracts features from those scales using a similar concept to feature pyramid networks. Table 1 compares the accuracy of various object detection models. In evaluating object detection algorithm, two main performance metrics were used: accuracy and computing time. Based on the comparative study by Redmon and Farhadi (2018), YOLOv3 is much better than SSD (single shot detector) variants and comparable to other state-of-the-art models like RetinaNet in accuracy but very fast (inference time is shorter).

YOLO combines the region proposal network branch and classification stage into a single network, leading to more concise architecture, state of the art performance in object detection with high computation speed and better computational efficiency, making them the true sense of real-time detectors (Redmon and Farhadi 2018). YOLO v3 is a masterpiece in the rising era of artificial intelligence, and also an excellent example of the power of Convolution Neural Network techniques. Deep learning based object detection algorithm is very useful in ensuring the transportation safety. If implemented, not only the engineering and construction information can be accurately estimated in the early phase of the project, but also defective materials and components in existing highway infrastructures can be rapidly identified and replaced/repaired by using such machine learning models.

Transfer learning refers to the situation where what has been learned in one setting (base dataset) is exploited to improve the generalization in another setting (target dataset). When the target dataset is significantly smaller than the base dataset, transfer learning enables training a large target network without overfitting; recent studies that have taken advantage of this fact obtained state-of-the-art results from transfer learning (Liu and Zhang 2019, 2020). In YOLO v3, DarkNet-53 is used as a backbone feature extractor. DarkNet-53 has less billion floating point operations than the ResNet-152, but achieves two times faster with the same classification accuracy. Thus, YOLOv3 shows significant improvement for small objects detection and performs very well with speed involvement (Lawal 2021). The use of transfer learning to facilitate the training of deep learning based object detection models was adopted in this study because the number of available data samples usually falls below a threshold value (e.g., this minimum number of data records required for well-trained deep learning models usually on the order of millions of images for good prediction accuracy). Apparently, millions of images for a specific image classification or object detection application is not presently available in most civil engineering datasets, therefore transfer learning can significantly reduce the required data size for training deep learning models with acceptable prediction accuracy) only requires several hundreds of image data samples to achieve a reasonably good prediction accuracy. Transfer learning is also appealing to the proposed project because the sophisticated design of the architecture of those best-performing deep learning based object detection models validated by other professionals can be taken advantage of.

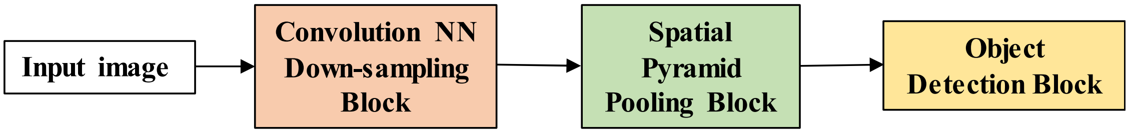

Input image
Convolution NN Down-sampling Block
Spatial Pyramid Pooling Block
Object Detection Block

Figure 15. Architecture of YOLO v3 spp (spatial pyramid pooling) model

In this study, the UMD team used YOLO v3 model to detect a number objects of potential interests to MDOT SHA, including traffic barrier end treatment end treatment in highway guardrails and insects blocking the ROW vehicle cameras. After comparing several open source models, the YOLO v3 spp open-source model from github (Ultralytics, 2020, https://github.com/ultralytics/YOLOv3) was found to have more features and higher accuracy based on testing results and thus adopted for this study. Figure 15 shows the architecture of YOLO v3 spp model. As a demonstration of the general procedure for object detection, ball detection was conducted and first presented here. It is worth noting that this ball detection can be potentially used for monitoring rock slope movement in field by attaching the color ball to the desired location on the rock slope using metal studs and polyurethane glue. First, raw images were prepared for training images by adding bounding boxes and labels to the identified objects in the training images, as shown for the two orange balls in Figure 16. To train the YOLO v3 model for object detection, a proper epoch number needs to be determined first before training. Several factors contribute to the proper number of training epochs, including the amount of training images, number of object classes for detection, and etc. It was found in this study from trial tests that good training weights was reached after training for about 150 epochs, therefore the training epoch value was set as 200 here.

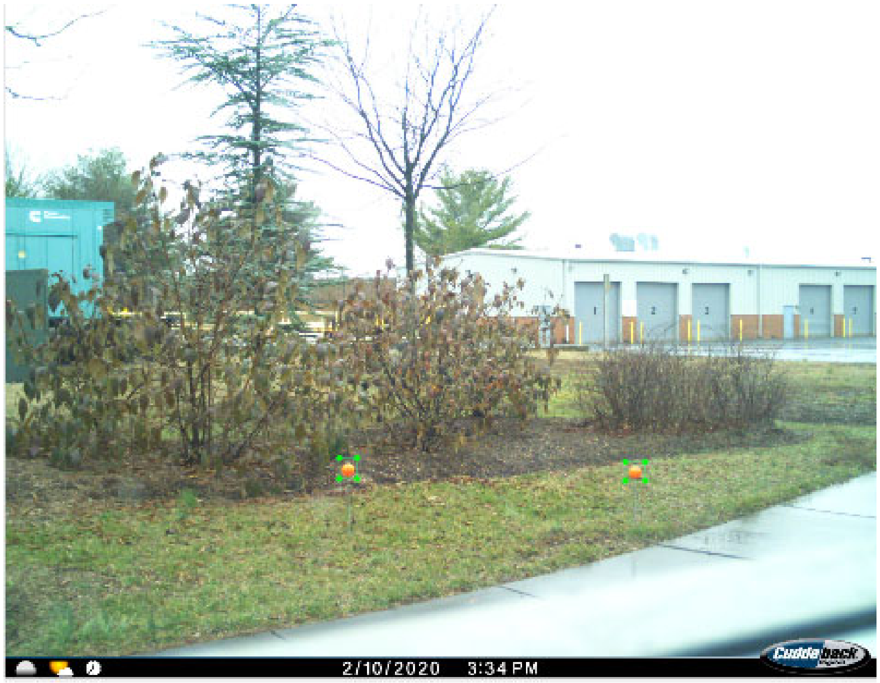


Figure 16. Prepared training images from remote cameras with bounding boxes around objects to be identified

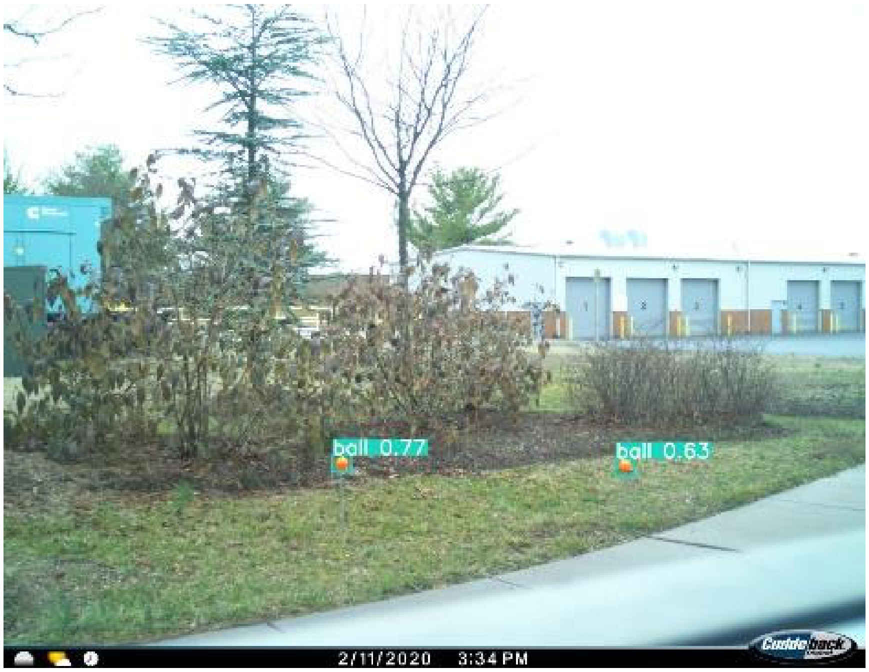


Figure 17. Sample output image with ball objects identified by YOLO v3 model

The output confidence value above each identified box can be used to assess the confidence probability of the identified object using the trained model. In order to not miss any real detections, the recommended confidence threshold for object detection is between 0.1 to 0.15. In this study, confidence threshold value was set as 0.1. Example images of detected orange balls with boxes and corresponding confidence value are shown in Figures 17 and 18. It is seen that photo quality (e.g., with exposure) had some negative effect on object detection performance.

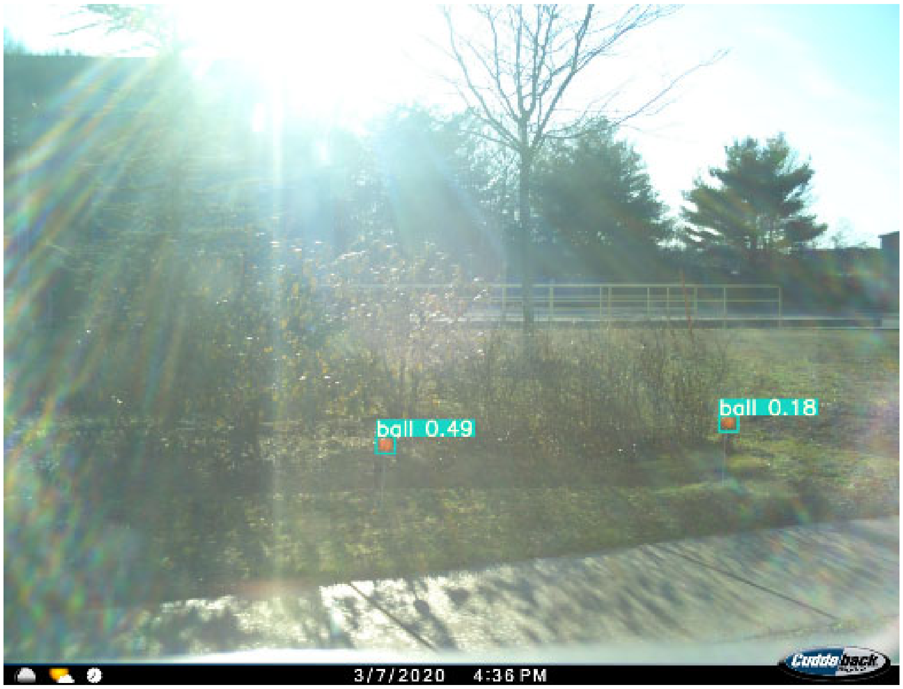


Figure 18. Sample output image with ball objects identified by YOLO v3 model in an overexposed photo from remote cameras

To study the feasibility of using machine learning based object detection method to capture objects of interest in ROW images, the YOLO v3 object detection model was first trained with huge amount of ROW image data collected by MDOT SHA. Figure 19 shows a typical ROW image with objects of interest (e.g., trucks, cars, traffic lights) being successfully identified by a trained YOLO v3 object detection model. To further test the performance of YOLO v3 algorithm in quickly identifying other objects of interest such as traffic barrier end treatment in guardrail, the research team trained YOLO v3 models with a goal of rapidly screening ROW image datasets in MDOT SHA archive. Preliminary results have shown the deep learning based YOLO v3 spp algorithm was very effective and achieved over 90% accuracy in identifying the traffic barrier end treatments in the scanned images.

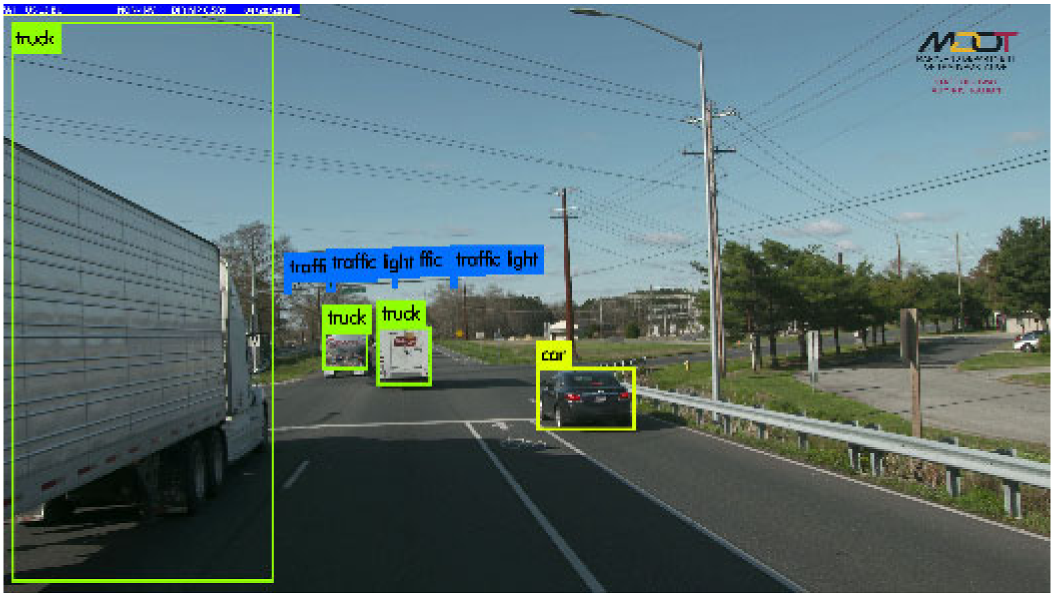


Figure 19. Objects of interest identified in ROW images

To train the YOLO v3 model for detecting the traffic barrier end treatment in highway guardrails, a selected number of ROW images were first prepared for training images by adding bounding boxes and labels to the identified objects in the image, as shown in Figure 20. The number of training images was determined to be around fifty on a trial and error base by considering the computing time and diversity of image background and object scales. In the current model, 47 training images and 5 validation images were included for the training model. Batch size was set to be 16. The training epoch value was set as 300 here. In order to not miss any real detections, the confidence threshold value was set as 0.1 in this study. Training of the YOLO v3 spp model was completed in 4.3 hours using a single Nvidia Titan X GPU card. Important output parameters from training the YOLO v3 spp (Ultralytics 2020) model are shown in Figure x, which track the right training path, including GIOU, Objectness score and Classification score for the training set and validation set. In this study, since only one class (i.e., traffic barrier end treatment) was defined in the training image set, the classification score would remain zero. Other controlling output parameters are ‘Precision’, ‘Recall’, ‘m-AP@0.5’ and ‘F1’ score. The precision of a class define how trustable is the result when the model answer that a point belongs to that class. The recall of a class expresses how well the model can detect that class. The F1 score of a class is given by the harmonic mean of precision and recall (2×precision×recall / (precision + recall)), it combines precision and recall of a class in one metric.

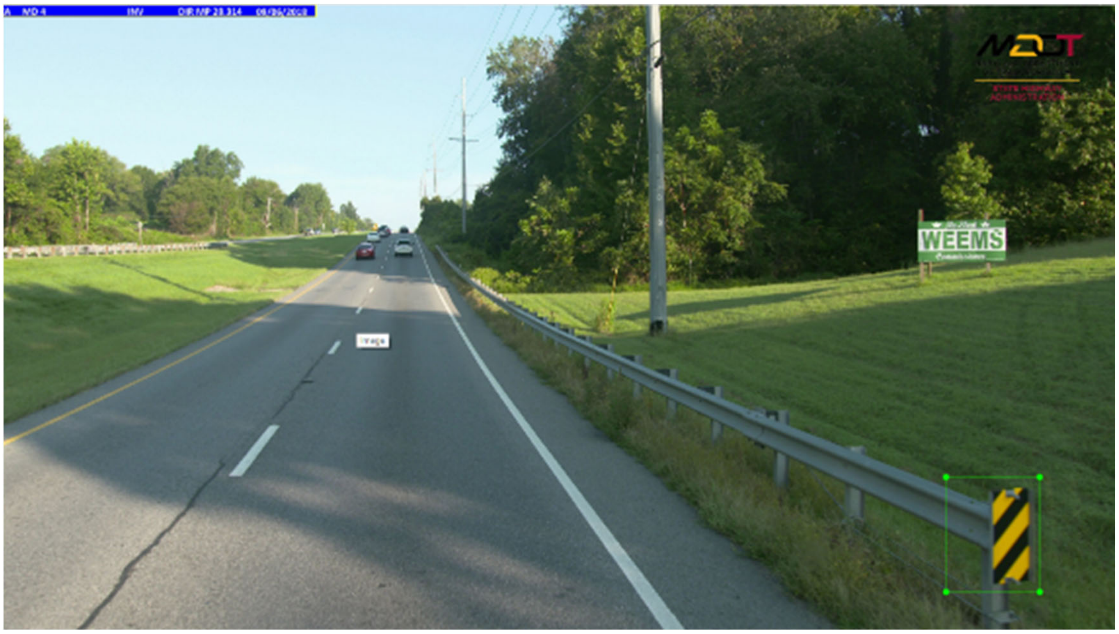


Figure 20. Prepared training images with bounding boxes around objects to be identified

To quantitatively evaluate the deep learning model performance in forecasting, several performance metrics were defined and used in this study. The AP is computed based on the Precision-Recall value. The general definition for the Average Precision (AP) is finding the area under the precision-recall curve. The IOU threshold is set as 0.5. For the single class training, the mean average precision (m-AP) is the same as average precision (AP) with its best possible value equal to one.  GIOU reflects the error overlapped area between trained bounding box and real bounding box. Objectness score represents the probability that an object is contained inside a bounding box. Recall of a class expresses how well the model can detect that class and best possible value is one. The F1 score of a class is given by the harmonic mean of precision and recall (2×precision×recall / (precision + recall)), it combines precision and recall of a class in one metric and its best possible value is one.

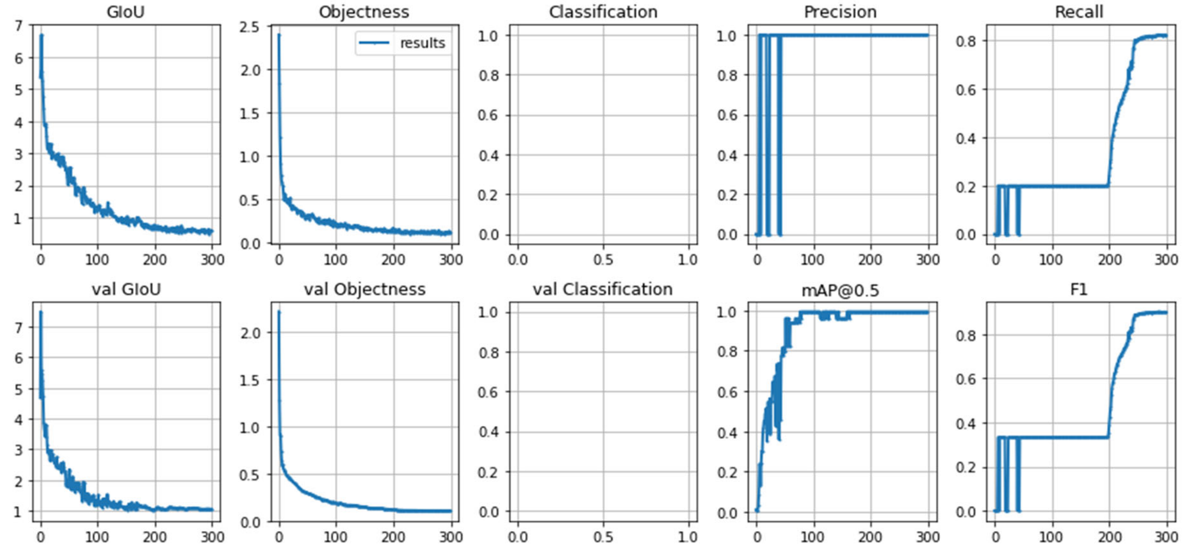


Figure 21. Training outputs from YOLO v3 spp model for traffic barrier end treatment detection

Table 2. Performance of YOLO v3 spp model for traffic barrier end treatment detection in ROW images

| | **Detection Rate** | **Error Rate** | **Accuracy Rate** |
|---|---|---|---|
| Garett County I | 25/2500=1% | (25-6+6)/2500=1% | 1-1%=99.00% |
| Garett County II | 9/2500=0.36% | (9-2)/2500=0.28% | 1-0.28%=99.72% |
| Garett County III | 11/2500=0.44% | (11-1)/2500=0.4% | 1-0.4%=99.60% |
| Wicomico I | 6/2500=0.24% | (6-0)/2500=0.24% | 1-0.24%=99.76% |
| Wicomico II | 7/2500=0.28% | (7-5+4)/2500=0.24% | 1-0.24%=99.76% |
| Wicomico III | 5/2500=0.2% | (5-0)/2500=0.2% | 1-0.2%=99.80% |

For testing the trained deep learning models, separate dataset other than the training dataset need to be used. The trained YOLO v3 model took approximately 40 minutes to finish the detection task by scanning one image data subset (each containing 2,500 images), which translates to about 0.96 seconds per image. The performance of the YOLO v3 spp model for detecting the traffic barrier end treatments in ROW images collected in two Maryland counties (Garrett County and Wicomico County) is summarized in Table 2. It is clear that a high accuracy was achieved by the YOLO v3 spp model in detecting traffic barrier end treatments. Example images of detected traffic barrier end treatments (End T. in each image) with boxes and corresponding confidence value are shown in Figures 22 to 32. These sample images were selected for different background and scale of traffic barrier end treatments. It is seen that scale of the traffic barrier end treatment (i.e., close or far view) has some effect on the detection confidence values.

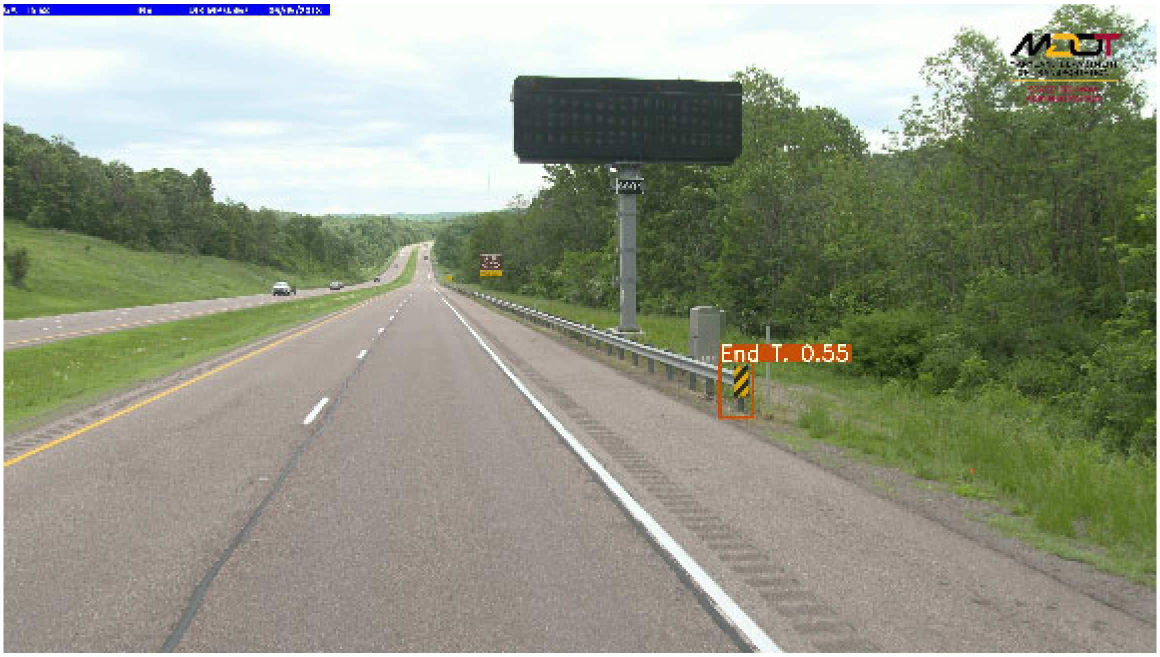


Figure 22. Sample output image with traffic barrier end treatment identified by YOLO v3 model

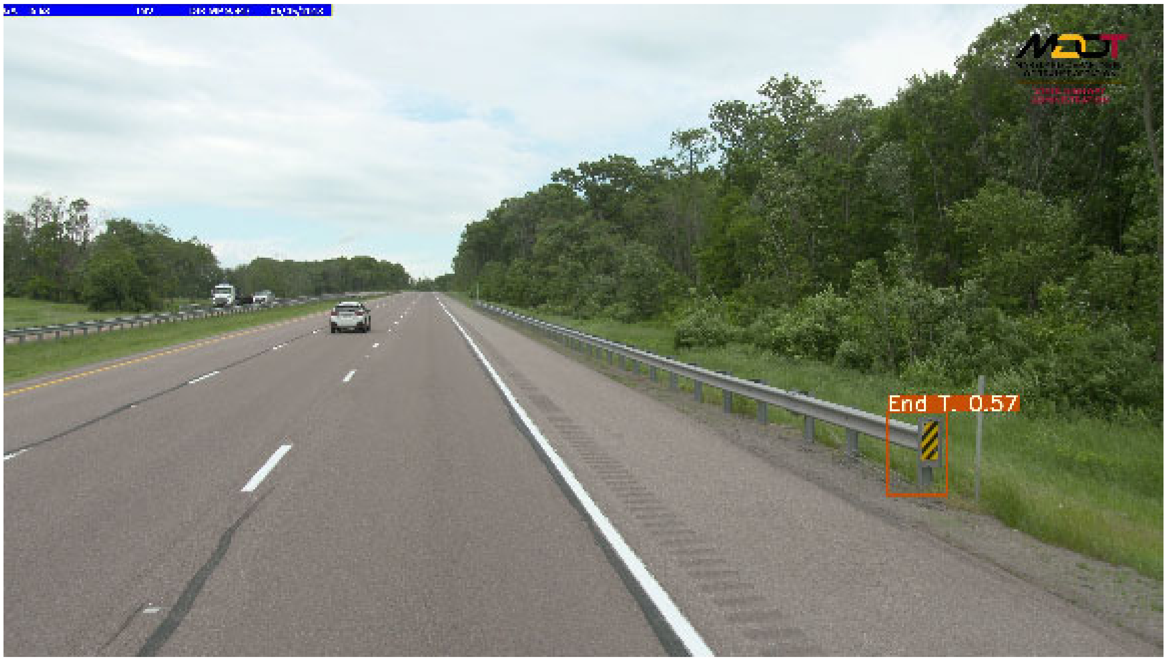


Figure 23. Sample output image with traffic barrier end treatment identified by YOLO v3 model

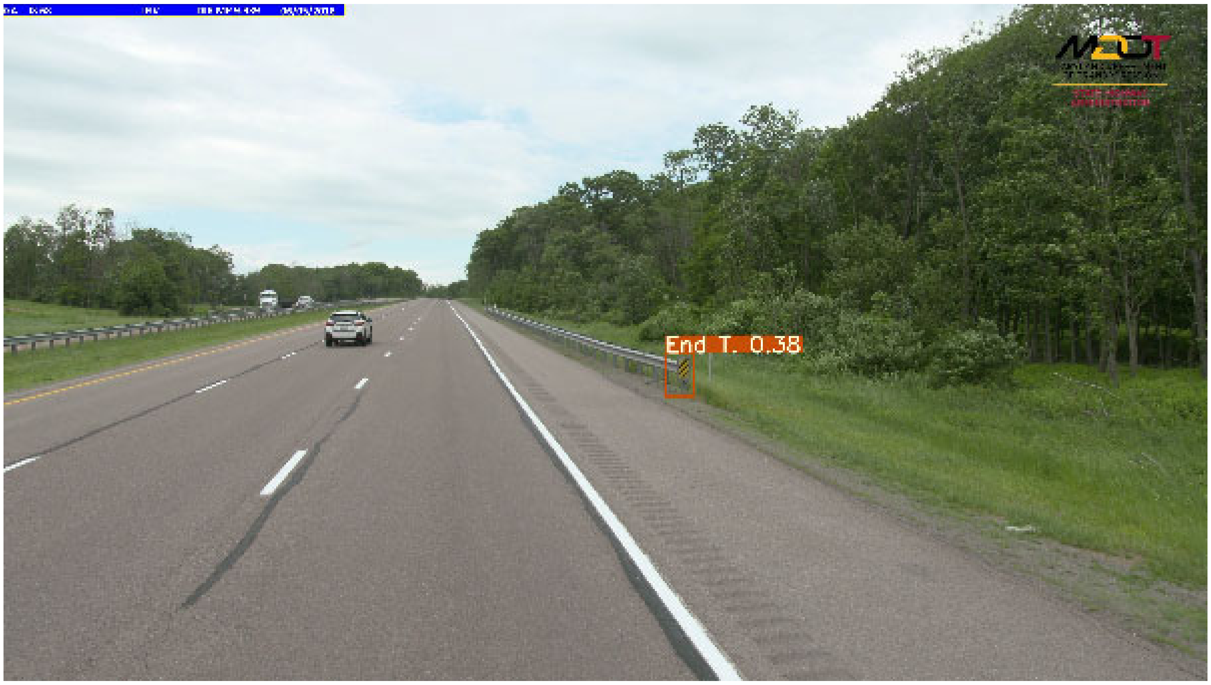


Figure 24. Sample output image with traffic barrier end treatment object identified by YOLO v3 model

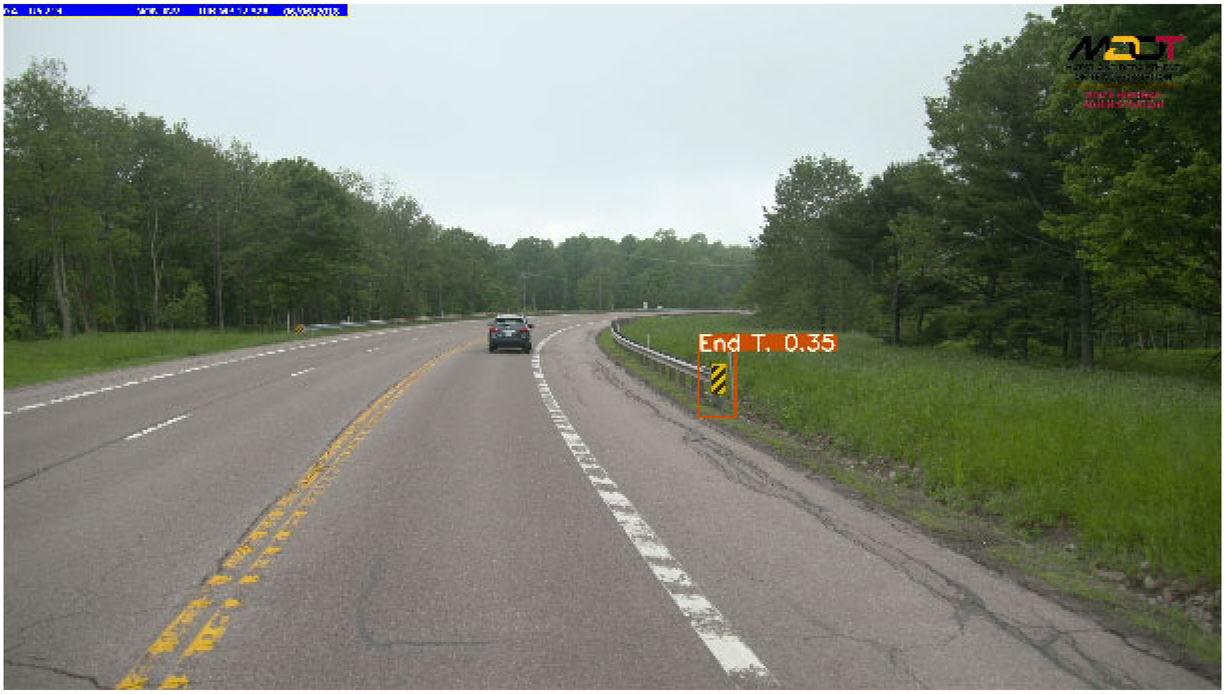


Figure 25. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with tall grass in the background and dashed yellow line marking on the road

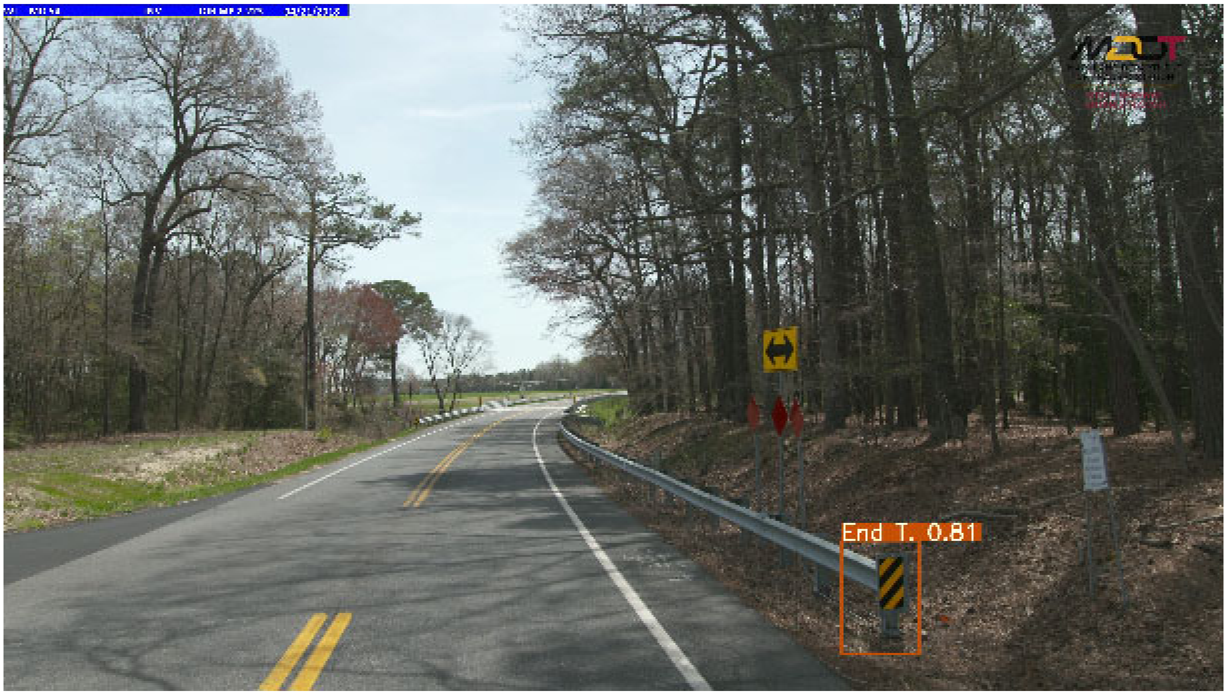


Figure 26. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with woods background and with close-up view of traffic barrier end treatment

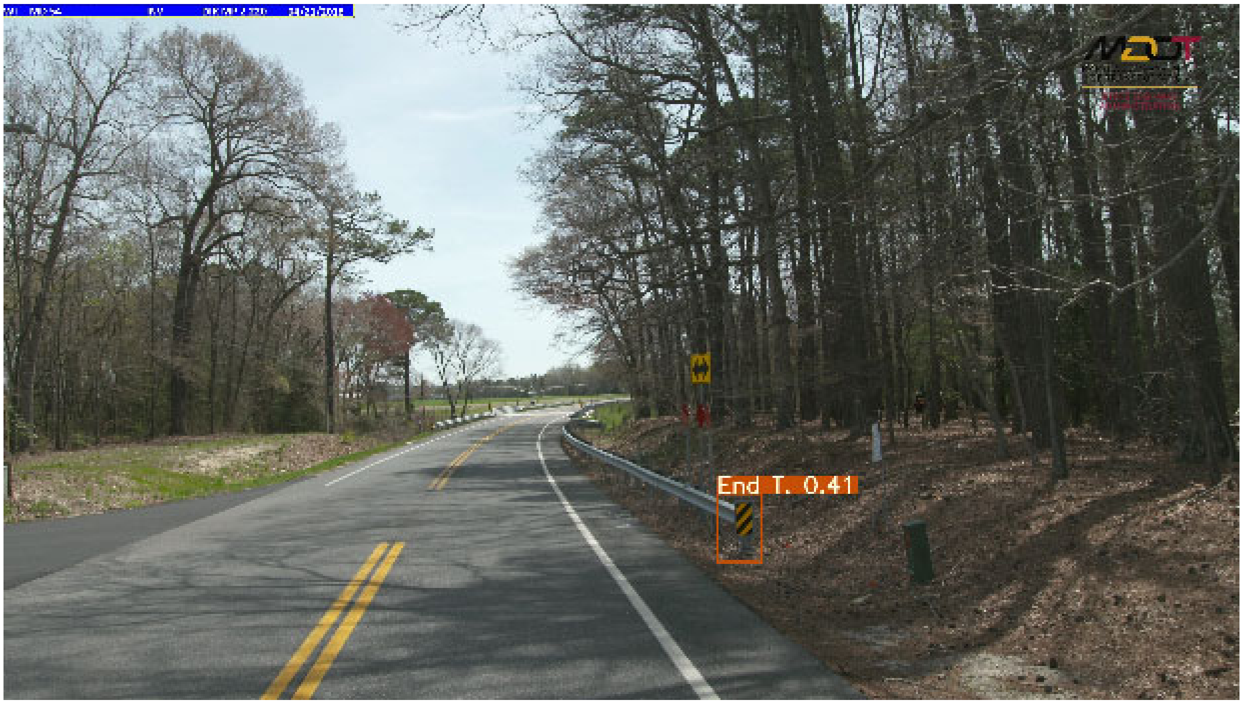


Figure 27. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with medium-scaled traffic barrier end treatment in the picture

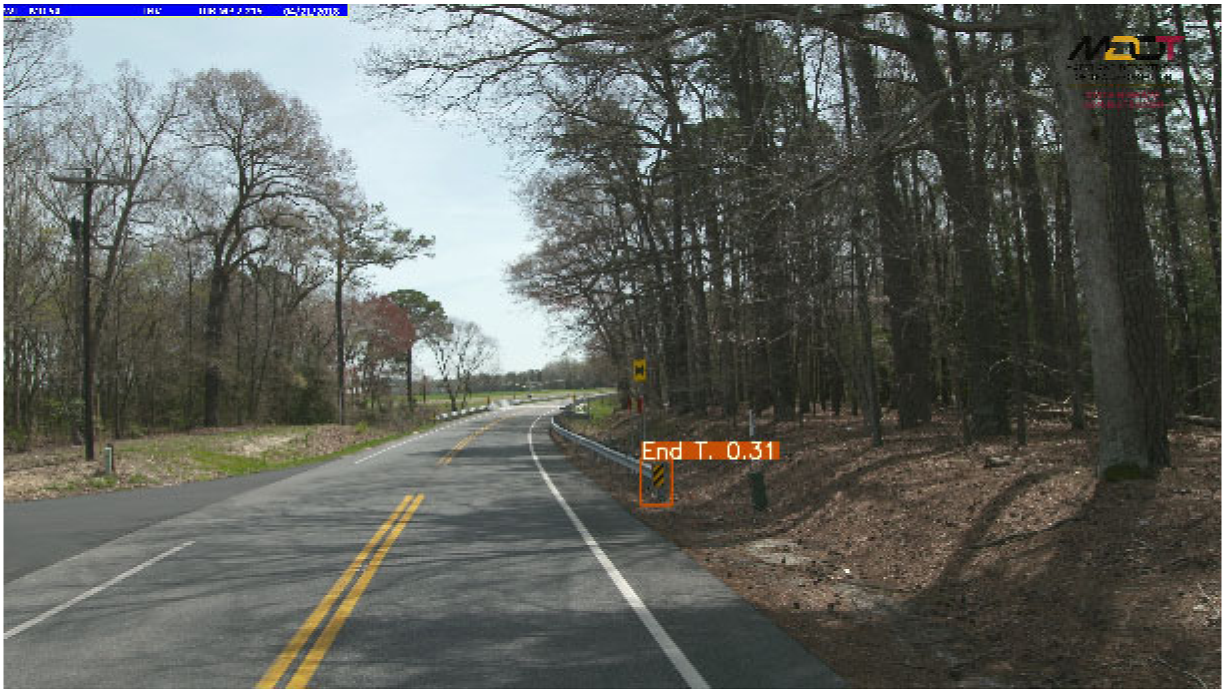


Figure 28. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with small-scaled traffic barrier end treatment in the picture

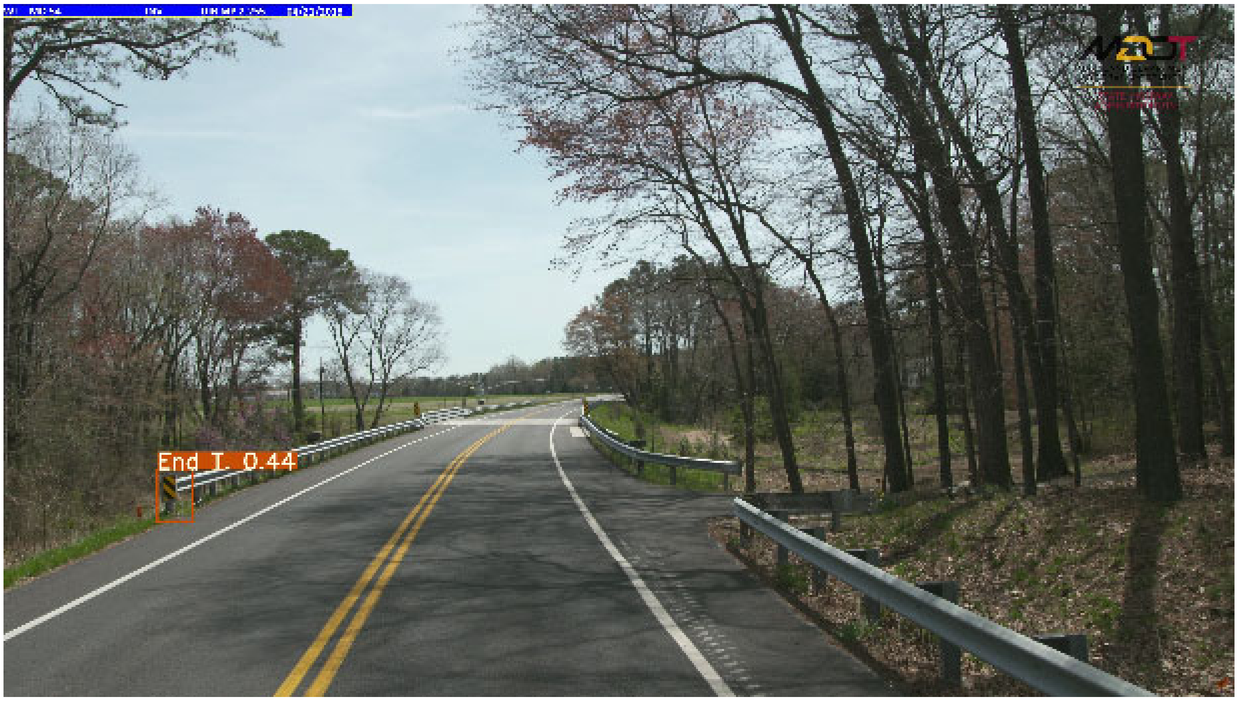


Figure 29. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with traffic barrier end treatment on the left side of the roadway

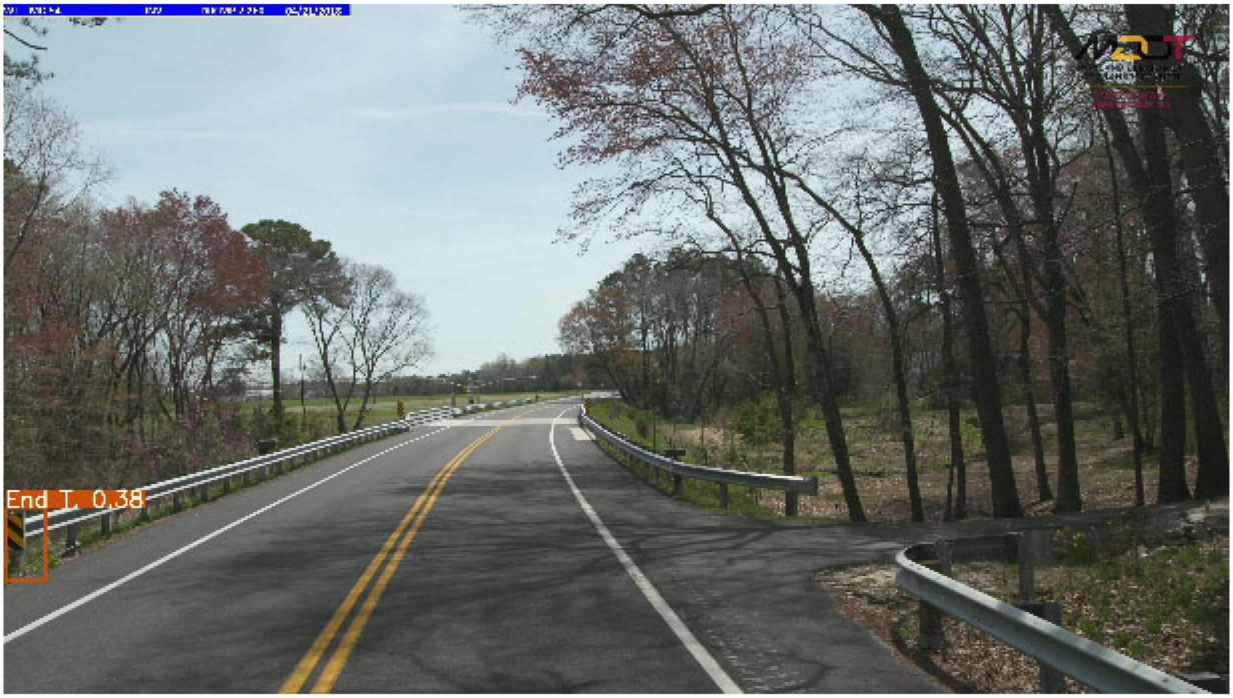


Figure 30. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with partial view of traffic barrier end treatment on the left

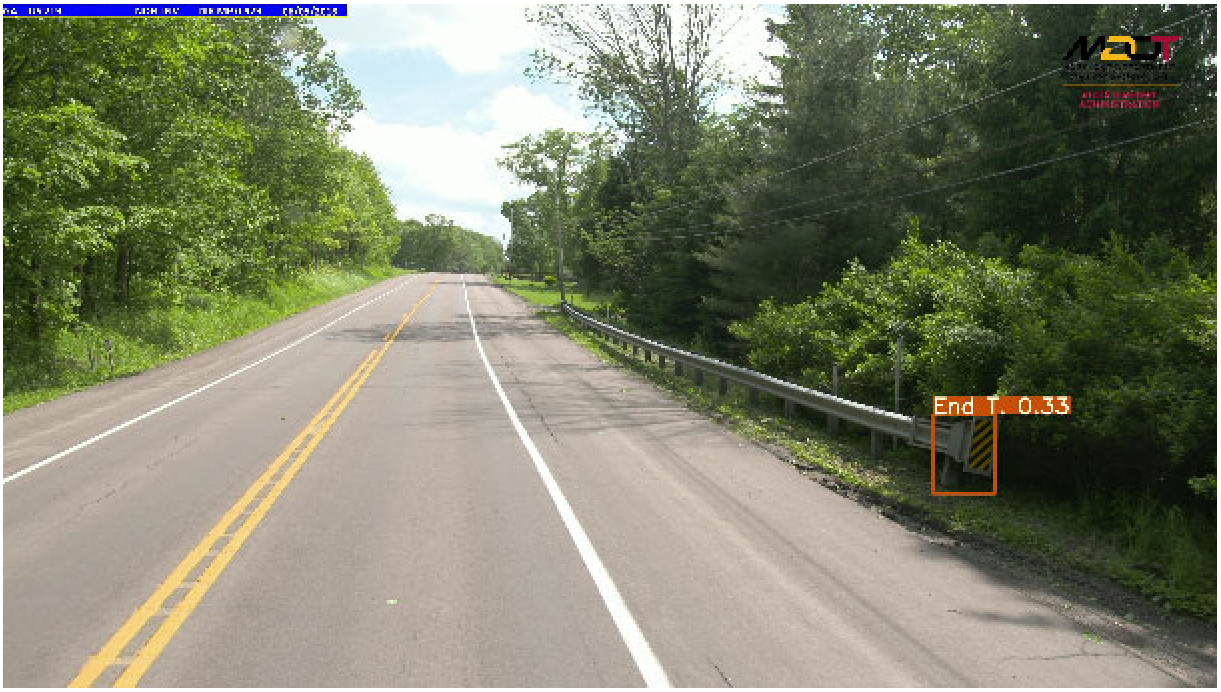


Figure 31. Sample output image with traffic barrier end treatment object identified by YOLO v3 model with inclined traffic barrier end treatment and shrubs in the background

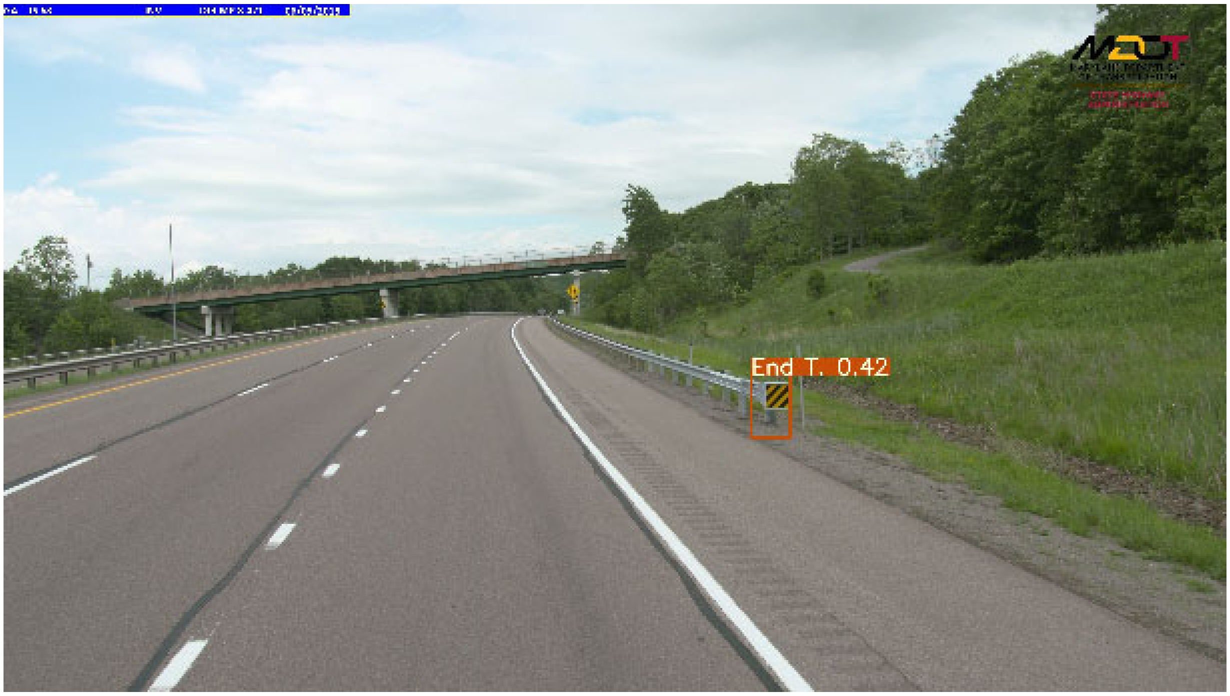


Figure 32. Sample output image with traffic barrier end treatment identified by YOLO v3 model with a bridge in the background

Currently MDOT SHA does QA/QC on all of the ROW imagery, and part of this QA/QC work involves identifying photos that are out of focus, covered lense (bugs trash), and etc. The research team was tasked to optimize the existing QA/QC tool to rapidly scan all the ROW imagery and find images with different types of defects including blurriness, underexposure, overexposure, no signal, discolored, and signal issues. A python script has been developed that can run any set of images for a particular year and county in Maryland. As shown in Figure 33, this QA/QC tool builds a csv file with the flagged errors and the shortcut to that photo, and a copy of the photo is then saved to another folder. The flags are found via algorithmic methods (e.g., variance of Laplacian for the blurriness) so processed images can be used for object detection such as insect blocking detection. The research team have tested the basic QA/QC method by runing this through the ROW images collected from one county in Maryland and it was found this method can effectively find potential issues. Optimization of the blurry and no signal thresholds was also done to catch blurry photos and minimize false positives.

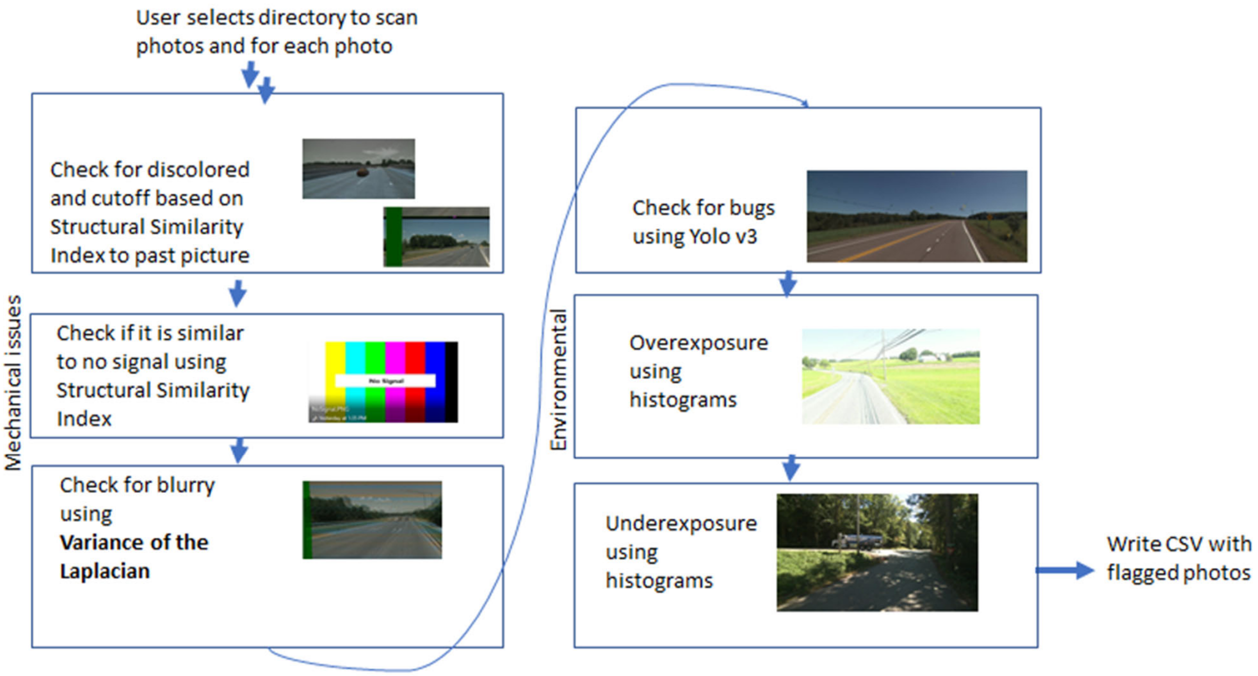
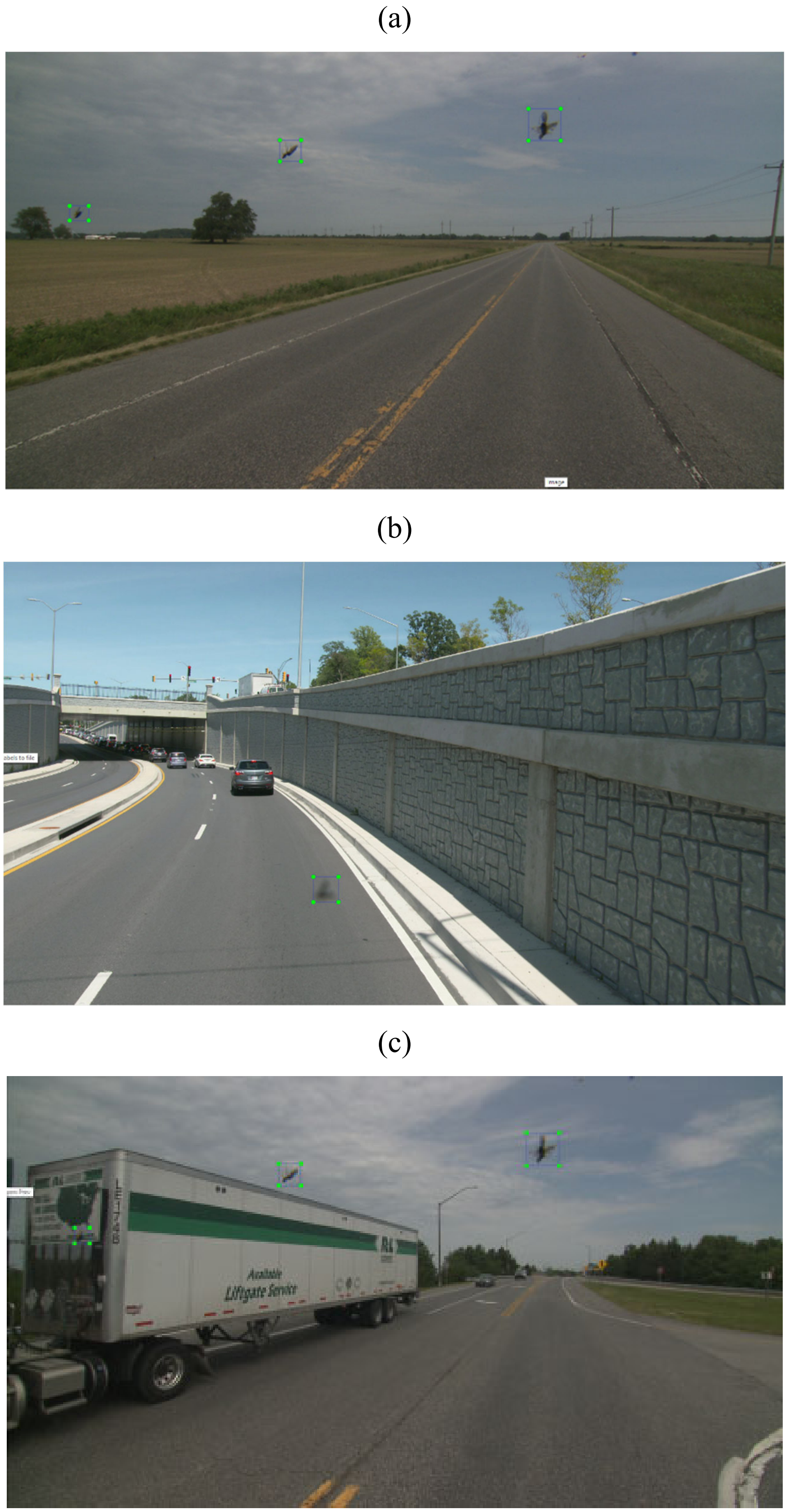

User selects directory to scan photos and for each photo
Check for discolored and cutoff based on Structural Similarity Index to past picture
Check if it is similar to no signal using Structural Similarity Index
No Signal
Check for blurry using Variance of the Laplacian
Mechanical issues
Environmental
Check for bugs using Yolo v3
Overexposure using histograms
Underexposure using histograms
Write CSV with flagged photos

Figure 33. Schematics of QA/QC tasks for ROW images

Lastly, the research team also did YOLO v3 spp object detection for insect blocking detection in ROW images as part of the task on QA/QC of ROW images. To train the YOLO v3 model for detecting the insect blocking, a selected number of ROW images were first prepared for training images by adding bounding boxes and labels to the identified objects in the image, as shown in Figure 34. For single batch image set, insects were almost same location in image and pixel sizes were fixed are each insect. The number of training images was determined to be around seventy on a trial-and-error base by considering the computing time and diversity of image background and object scales. In selecting these training images, images with insects mixed into diverse background types (see Figure x for three types of background) were preferred for model robustness and improved detection accuracy. Additional types of insects (shapes, sizes, location in image) would be helpful to make object detection robust for insects unseen here. A total of 70 training images were selected, 45 from Group One (first batch of images, sample shown in Figure 31(a)), and 25 from group two (2nd batch of images, sample shown in Figure 31(b)) were included for the training model.



(a)

(b)

(c)

Figure 34. Three prepared training images with bounding boxes around insect objects to be identified: (a) cloud background; (b) roadway and wall background; (c) truck background

Batch size was set to be 16. The training epoch value was set as 200 here. In order to not miss any real detections, the confidence threshold value was set as 0.1 in this study. Training of the YOLO v3 spp model was completed in 6.5 hours using a single Nvidia Titan X GPU card. Important output parameters from training the YOLO v3 spp model are shown in Figure 35, which track the right training path, including GIOU, Objectness score and Classification score for the training set and validation set.

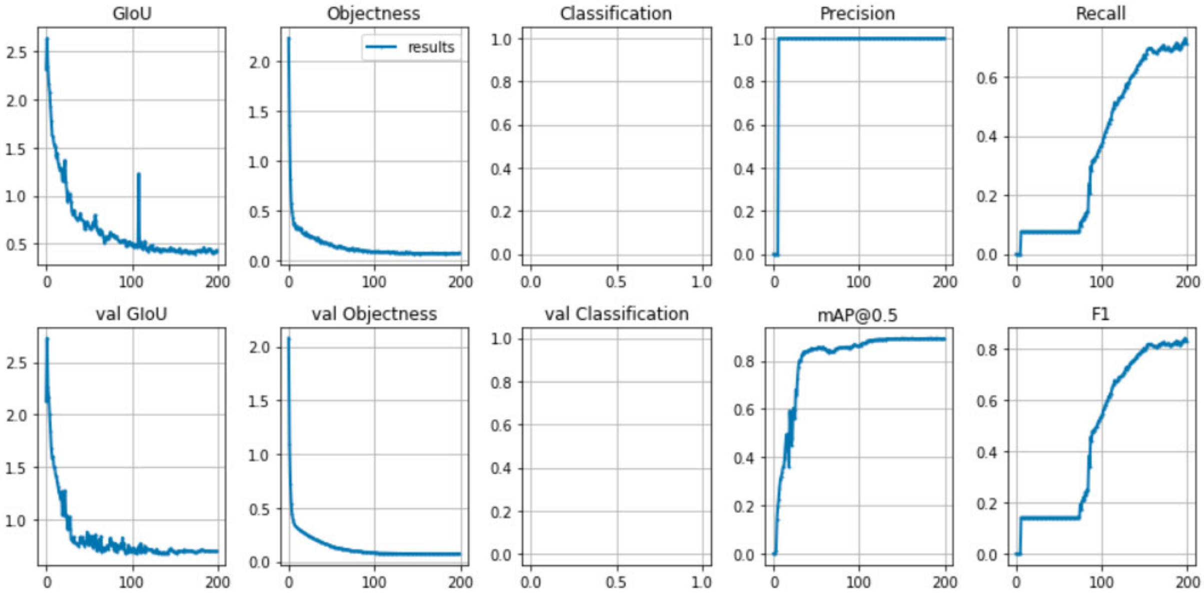


Figure 35. Training outputs from YOLO v3 spp model for insect blocking detection

The remaining 3,199 pictures (no training image included) were used for test, and it took 68.38 seconds total to complete the object detection scanning tasks for all these images (thus each image took about 0.16 seconds for insect detection using the trained YOLO v3 spp model for insect detection). The model could not detect insects against tree background because tree background was not included in training set. Example images of detected insect blocking with boxes and corresponding confidence value are shown in Figures 36 to 40.

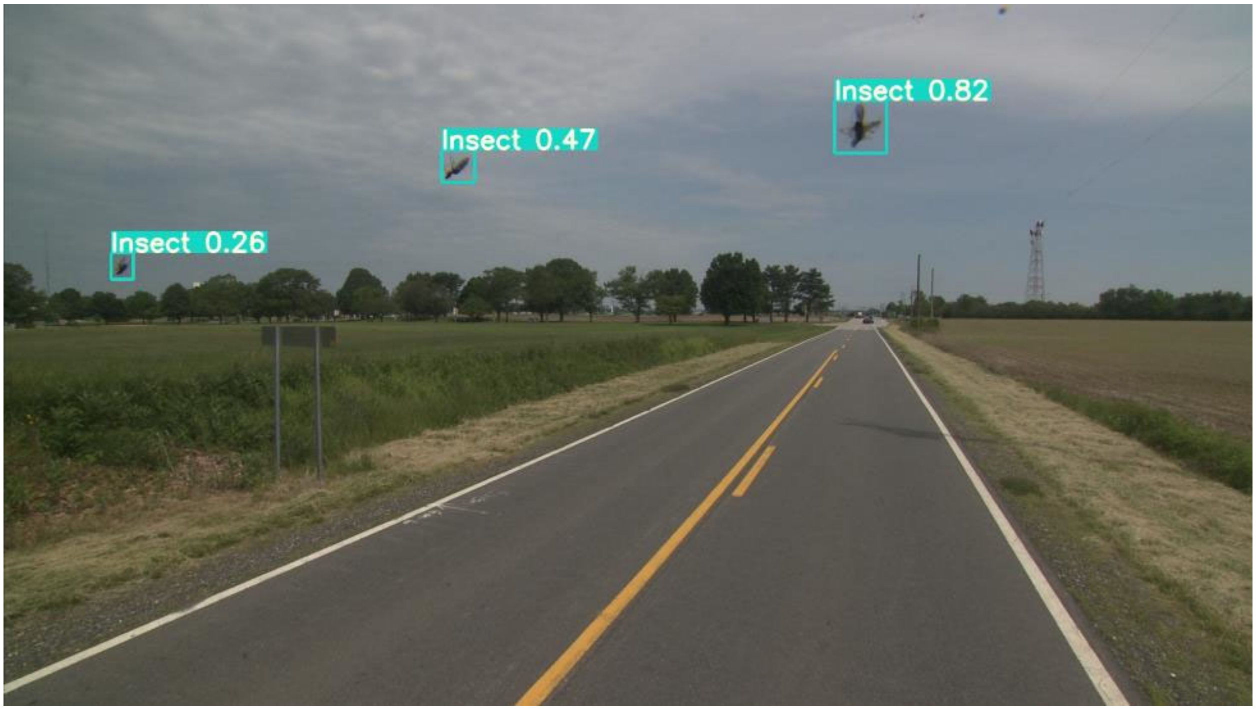


Figure 36. Sample output image with insect objects identified by YOLO v3 model

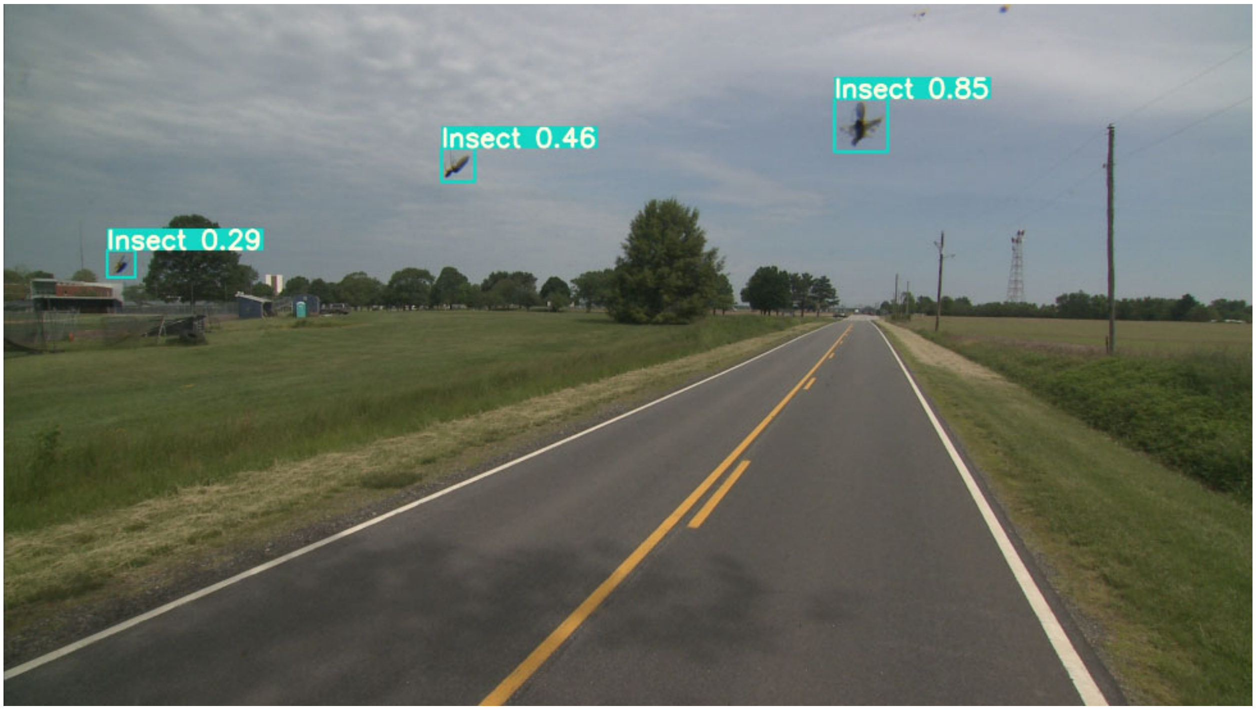


Figure 37. Sample output image with 3 insect objects identified by YOLO v3 model

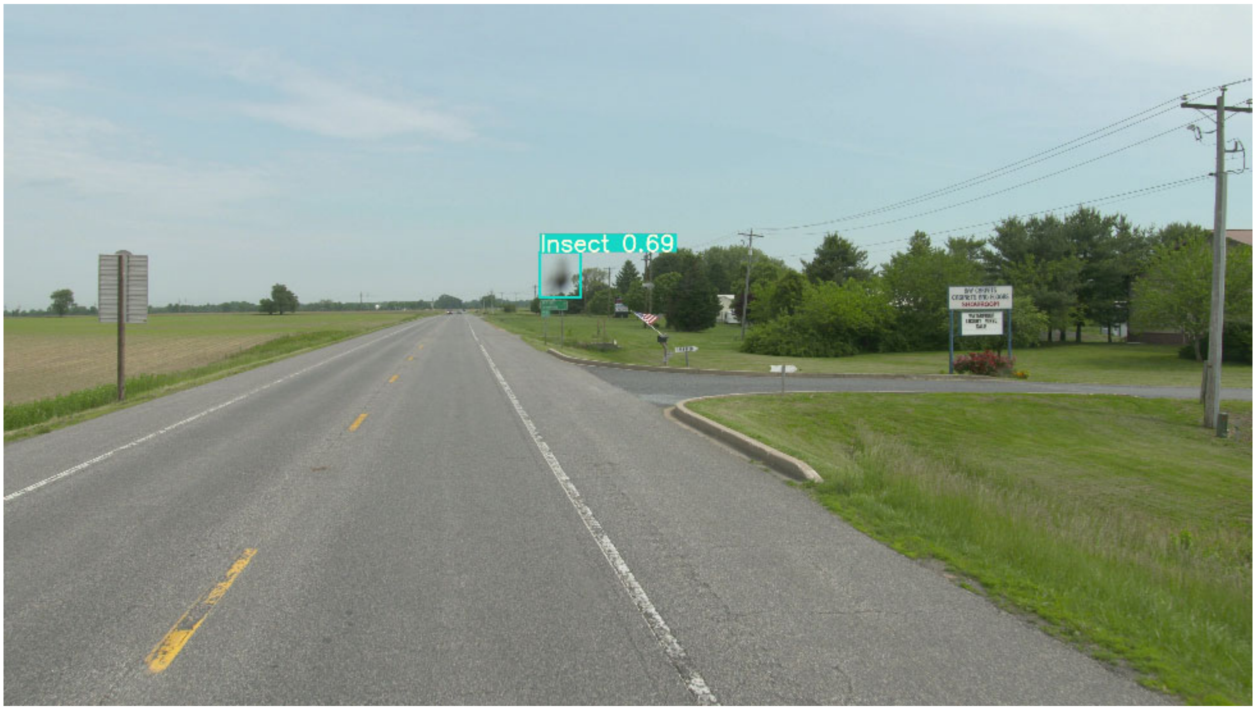


Figure 38. Sample output image with 1 insect object identified by YOLO v3 model

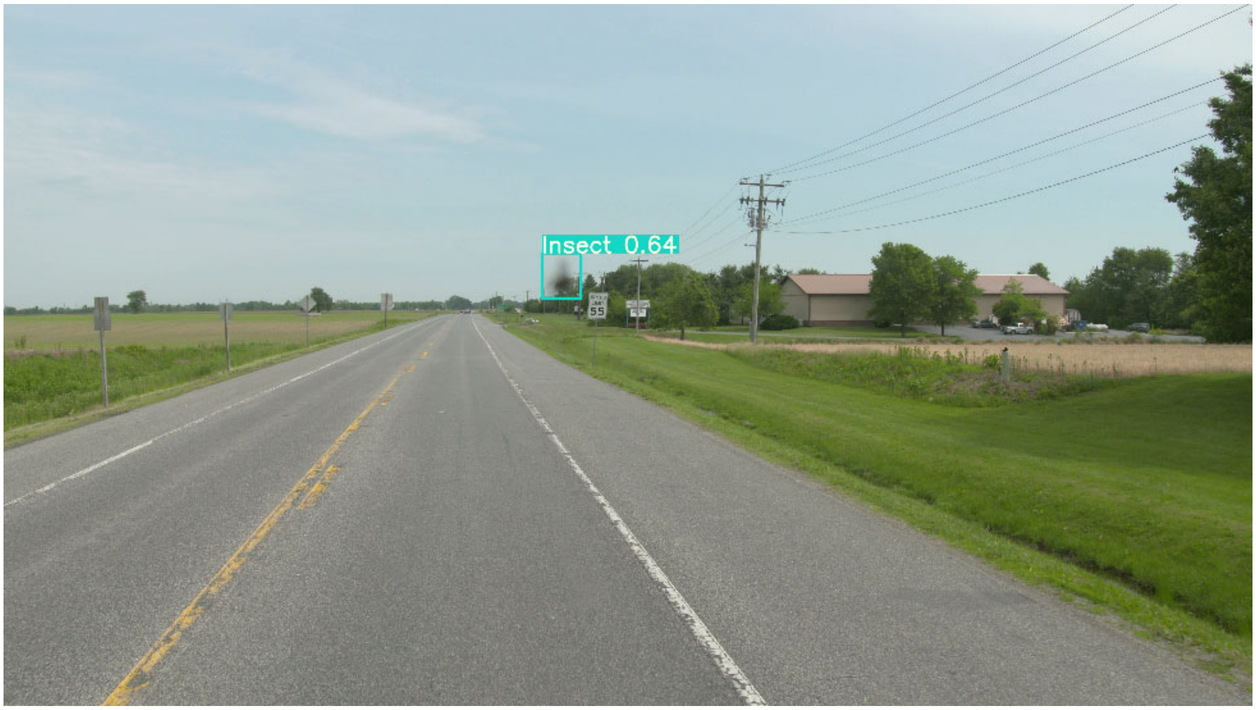


Figure 39. Sample output image with 1 insect object identified by YOLO v3 model

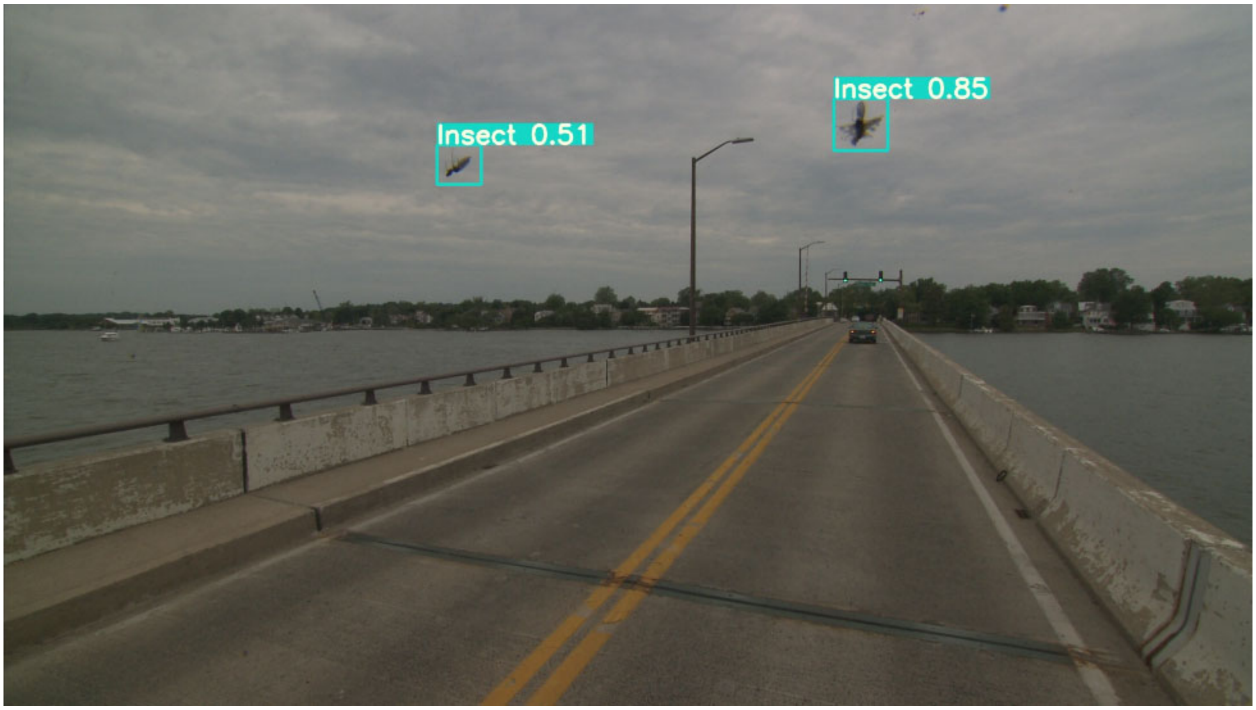


Figure 40. Sample output image with 2 insect objects identified by YOLO v3 model

# CHAPTER 4: REVIEW & DEVELOPMENT OF REINFORCEMENT LEARNING MODEL FOR SCHEDULING ESTIMATION

In this study, the research team developed and tested reinforcement learning models for drilling project schedule estimation with 212 historical records in Keras library environment. The research team also did literature review on reinforcement learning for project schedule estimation.

Reinforcement Learning (RL) is a machine learning algorithm that trains an agent to learn in an interactive environment using feedback from its own actions and experiences (Sutton and Barto 2018). The RL framework contains a decision maker (agent) that takes actions and interact with an environment to maximum the total rewards. The agent explores policies and chooses actions until an optimal policy is reached, and rewards are returned to the agent for each adopted action, as shown in Figure 41. Environment denotes the physical world in which the agent operates; Policy here refers to instructions to agent on how to take actions based on current state; Reward provides the feedback from the environment after taking a specific action.

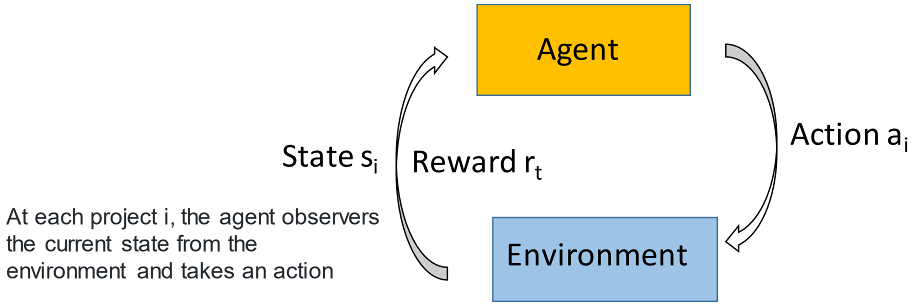
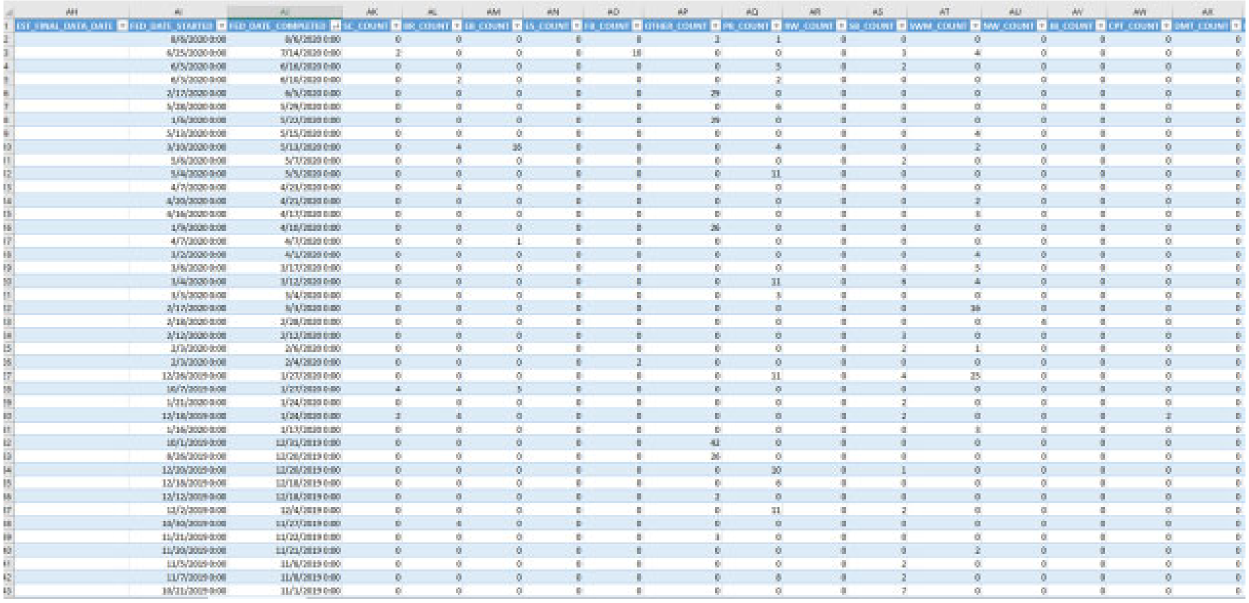

Agent
Action $a_i$
State $s_i$
Reward $r_t$
At each project i, the agent observers the current state from the environment and takes an action
Environment

Figure 41. The agent-environment interaction in reinforcement learning

Resource management problems are of special interest in many real-world applications such as equipment allocation and project scheduling. However, designing algorithms to allocate limited resources to different tasks is challenging and often requires human-generated heuristics. RL approaches are especially well-suited to resource management systems (Mao et al. 2016). First, decisions made by these systems are often highly repetitive, thus generating an abundance of training data for RL algorithms. Second, RL can model complex systems and decision-making policies as deep neural networks analogous to the models used for game-playing agents. Mao et al. (2016) showed how to use RL to automatically learn to allocate and schedule computer resources to waiting jobs, with the objective to minimize the average job slowdown.

RL problems can usually be modeled as a Markov Decision Process, which consists of a set of finite environment states S, a set of possible actions A(s) in each state, a real valued reward function R(s) and a transition model P(s', s|a). However, real world environments are more likely to lack any prior knowledge of environment dynamics. Model-free RL methods come handy in such cases (Bhatt 2018). Q-learning is a commonly used model-free approach which can be used for building the environment by training an agent. In Q-learning, a Q-table provides guidance to the best action at each state by looking at updated Q values which denotes the maximum expected future reward value for performing action $a_i$ in state $s_i$. The Q-function uses the Bellman equation to optimize the Q-value in making reward and action decisions. RL is theoretically able to alleviate the curse of dimensionality related to the state space, either under model-free approaches that do not utilize prior offline environment information on transition dynamics, or model-based approaches that also try to learn the underlying transition model of the environment.

In this study, Q-learning algorithm was adopted to build an RL model for project duration estimation. Possible input parameters to the RL model included: Project Location X, Y (potentially related to the distance of the drilling site from OMT office), number of SPT Borings, Linear Feet (LF) of SPT boring, number of Auger Borings (ABLF of AB). Example is [15, 8, 3.6, 3, 2]. Estimated Delivery Date of Each project checkpoint involves IE Approval, Field testing start date, Field testing end date, project delivery. The input: A vector of five values. distance from OMT, # of SPT Borings, Linear Feet LF of SPT, # of Auger Borings, LF of Auger); The Q table from the agent based on the input and action state selected using existing data (but the input space is limited to a few hundred historical data records available to the research team.



| | AH | AI | AJ | AK | AL | AM | AN | AO | AP | AQ | AR | AS | AT | AU | AV | AW | AX |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EST_FINAL_DATA_DATE | FLD_DATE_STARTED | FLD_DATE_COMPLETED | SC_COUNT | BR_COUNT | LB_COUNT | LS_COUNT | FB_COUNT | OTHER_COUNT | PE_COUNT | RW_COUNT | SB_COUNT | SWM_COUNT | MW_COUNT | BI_COUNT | CPT_COUNT | DMT_COUNT |
| 2 | | 8/6/2020 0:00 | 8/6/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | | 6/25/2020 0:00 | 7/14/2020 0:00 | 2 | 0 | 0 | 0 | 18 | 0 | 0 | 0 | 3 | 4 | 0 | 0 | 0 | 0 |
| 4 | | 6/5/2020 0:00 | 6/18/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 5 | | 6/5/2020 0:00 | 6/18/2020 0:00 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | | 2/17/2020 0:00 | 6/5/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | | 5/28/2020 0:00 | 5/29/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | | 1/6/2020 0:00 | 5/22/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | | 5/13/2020 0:00 | 5/15/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
| 10 | | 3/10/2020 0:00 | 5/13/2020 0:00 | 0 | 4 | 16 | 0 | 0 | 0 | 4 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 11 | | 5/6/2020 0:00 | 5/7/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 12 | | 5/4/2020 0:00 | 5/5/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | | 4/7/2020 0:00 | 4/23/2020 0:00 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | | 4/20/2020 0:00 | 4/21/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 15 | | 4/16/2020 0:00 | 4/17/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 16 | | 1/9/2020 0:00 | 4/10/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | | 4/7/2020 0:00 | 4/7/2020 0:00 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | | 3/2/2020 0:00 | 4/1/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
| 19 | | 3/6/2020 0:00 | 3/17/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 |
| 20 | | 3/4/2020 0:00 | 3/12/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 6 | 4 | 0 | 0 | 0 | 0 |
| 21 | | 3/3/2020 0:00 | 3/4/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | | 2/17/2020 0:00 | 3/3/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 |
| 23 | | 2/18/2020 0:00 | 2/28/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 |
| 24 | | 2/12/2020 0:00 | 2/12/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 25 | | 2/3/2020 0:00 | 2/6/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 |
| 26 | | 2/3/2020 0:00 | 2/4/2020 0:00 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | | 12/26/2019 0:00 | 1/27/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 4 | 25 | 0 | 0 | 0 | 0 |
| 28 | | 10/7/2019 0:00 | 1/27/2020 0:00 | 4 | 4 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | | 1/21/2020 0:00 | 1/24/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 30 | | 12/16/2019 0:00 | 1/24/2020 0:00 | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 |
| 31 | | 1/16/2020 0:00 | 1/17/2020 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 32 | | 10/1/2019 0:00 | 12/31/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | | 8/26/2019 0:00 | 12/20/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | | 12/20/2019 0:00 | 12/20/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 35 | | 12/16/2019 0:00 | 12/18/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 36 | | 12/12/2019 0:00 | 12/18/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | | 12/3/2019 0:00 | 12/4/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 38 | | 10/30/2019 0:00 | 11/27/2019 0:00 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 39 | | 11/21/2019 0:00 | 11/22/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | | 11/20/2019 0:00 | 11/21/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 41 | | 11/5/2019 0:00 | 11/6/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 42 | | 11/7/2019 0:00 | 11/8/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 43 | | 10/21/2019 0:00 | 11/1/2019 0:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 |

Figure 42. Raw drilling project duration data

In this study, the following RL policy definition was adopted to generate the Q table and guide action decisions.

1. First read in all 212 available samples and sort them in ascending order of a predefined resource demand metrics that ideally would be strongly correlated to the target of interest (i.e., drilling project duration or completion time in days). In this pilot study, sum of AB LF and SPT LF was used, however, other more sophisticated sorting metrics were also tested such as random forest of predicted drilling project completion days. After sorting, the first one is the sample with least resource demand value, and the last sample is the one with largest demand value. This sorted data bunch was used later in the environment. For simplicity, data sample format uses this format of [AB LF, SPT LF], however, other features such as drilling site distance from OMT office could also be used (e.g., AB bucket, SPT bucket, distance). A request (new) input sample was fed into the model to decide which sample is the closest with regard to resource demand, and then the duration of the closest sample (from 212 total samples) was used as estimate of the duration for the request input.

2. Reward definition
    a. reward = 1/[abs(input demand - current state demand)+0.1]-0.1; this reward enables that after each simulation, it gradually moves towards the true value (or closet sample) termed destination. The reward for each action is defined as (demand can be any custom-defined combo features related to the target variable):
    b. Q-table value is based on summation of rewards accumulated in the search process for each cell.

3. Action Policy
    a. For stage 1 (exploration stage), the action is determined by the difference between input demand and current state demand, moving the right if positive, left if negative.
    b. For stage 2 (knowledge utilization stage), the action is determined by comparing Q value of the state right and left of the current state.

4. Exit policy
    a. For stage 1, if the product of current action difference and previous action difference is negative, exit the loop.
    b. For stage 2, if the Q-value of the current state is higher than either of the adjacent state, exit the loop.

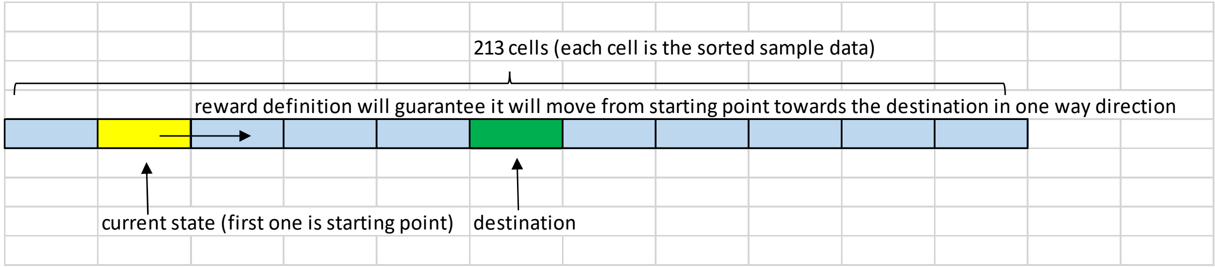

213 cells (each cell is the sorted sample data)
reward definition will guarantee it will move from starting point towards the destination in one way direction
current state (first one is starting point)
destination

Figure 43. Illustration of sorted data samples in RL

Flowchart of reinforcement learning model realizing the above policy defined for project schedule estimation is shown in Figure 44.

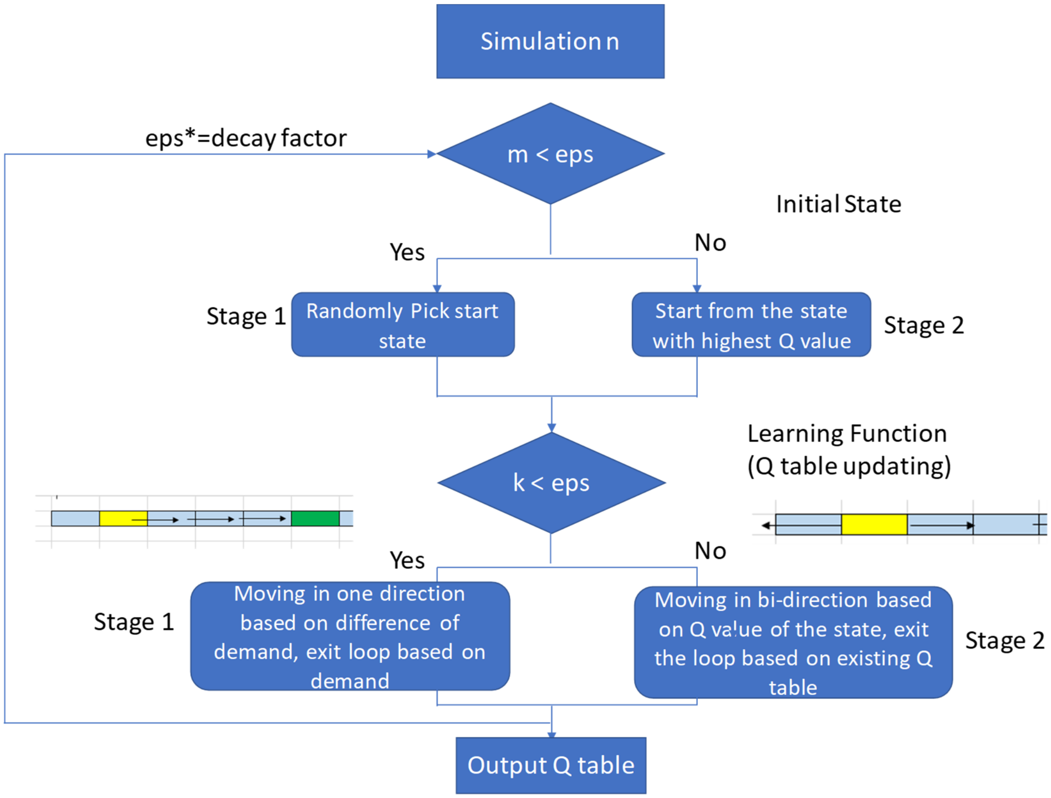
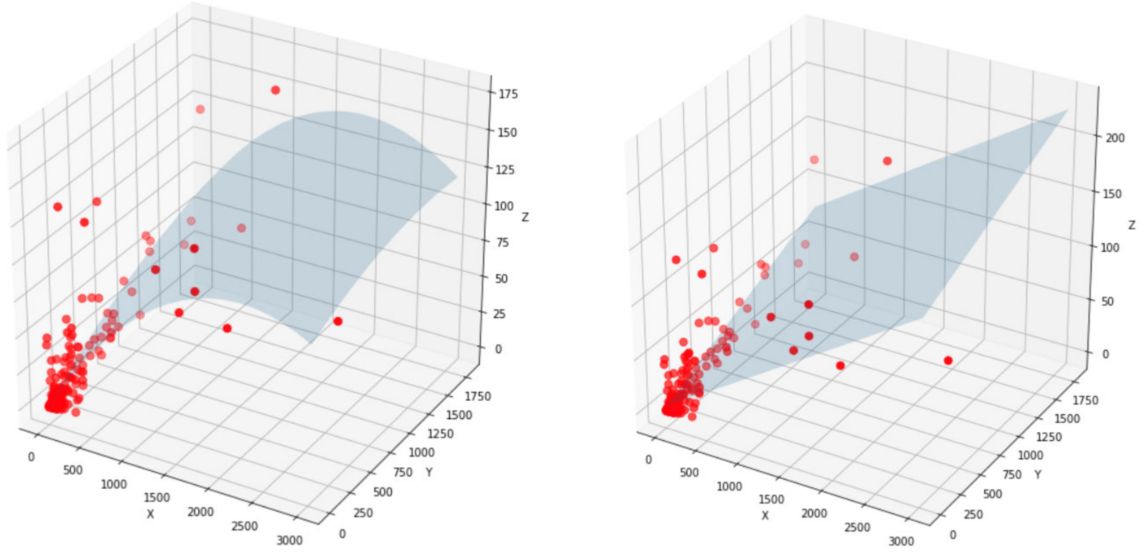

Simulation n
eps*=decay factor
m < eps
Initial State
Yes
No
Stage 1
Randomly Pick start state
Start from the state with highest Q value
Stage 2
Learning Function
(Q table updating)
k < eps
Yes
No
Stage 1
Moving in one direction based on difference of demand, exit loop based on demand
Moving in bi-direction based on Q value of the state, exit the loop based on existing Q table
Stage 2
Output Q table

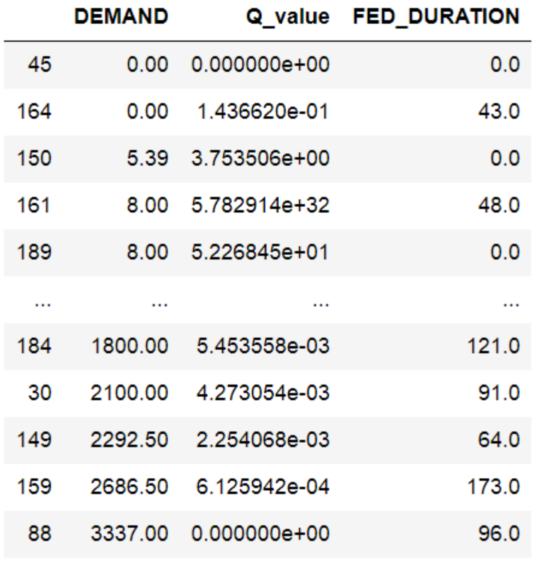
Figure 44. Flowchart of reinforcement learning model for project schedule estimation

As shown in Figure 43, this model was to generate a Q-table by finding the closest sample of an input among all available historical records. The historical record has been sorted in ascending order based on corresponding resource demand value. Resource demand is defined as the combination of features. A challenging task here is to find a combo feature that can best correlate the input feature variables with the project duration in days. In this model, it was defined as the summation of Auger LF and SPT LF. A more complex function such as advanced machine learning models can also be defined if that proves to be more closely match the resource demand with project duration days. For example, Figure 45 shows two definitions for this resource demand metric regressed from real data using the least mean squares technique in comparison with actual project duration data (in red dots). In the figure, AB LF is the x axis, SPT LF is the y axis, and duration (days) of job completion time as z axis. The real data samples were fitted by linear plane and quadratic surface using least mean squares regression. Quadratic function has this form:  z=a+bx+cy+dxy+ex^2+fy^2. However, these two resource demand functions seem to be dominated by outliers and the residual error is fairly large.



(a) (b)

Figure 45. Comparison of resource demand metrics (gray plane) vs. actual project duration data points (z axis or vertical axis = project duration; x axis, y axis = AB LF and SPT LF):
(a) Regressed Quadratic plane; (b) Regressed linear plane



| | DEMAND | Q_value | FED_DURATION |
|---|---|---|---|
| 45 | 0.00 | 0.000000e+00 | 0.0 |
| 164 | 0.00 | 1.436620e-01 | 43.0 |
| 150 | 5.39 | 3.753506e+00 | 0.0 |
| 161 | 8.00 | 5.782914e+32 | 48.0 |
| 189 | 8.00 | 5.226845e+01 | 0.0 |
| ... | ... | ... | ... |
| 184 | 1800.00 | 5.453558e-03 | 121.0 |
| 30 | 2100.00 | 4.273054e-03 | 91.0 |
| 149 | 2292.50 | 2.254068e-03 | 64.0 |
| 159 | 2686.50 | 6.125942e-04 | 173.0 |
| 88 | 3337.00 | 0.000000e+00 | 96.0 |

Figure 46. Sample Q-table values after convergence of RL simulation

.

Each simulation loop in the RL contains one realization to populate the Q-table. In Stage 1, current state is generated by random drawing; while initially the starting point is randomly selected, as simulation accumulates and Q-table is populated by the Q-learning, maximum Q number in the 212 cell is selected first as the starting point. A sample converged Q table generated by running the RL script (based on input demand=7) is shown in Figure 46 above. Since the input demand is closest to the 4th row of the record, the Q value in the 4th row generated by RL is much larger than other rows in the table.

Random Forest as a machine learning technique has also been applied to investigate if Random Forest predicted value can be potentially used for the resource demand metric that better map the features to duration days. In this Random Forest model, AB LF, SPT LF were selected as feature variables and duration days are dependent variable to predict. 80% records of the total 212 samples were used for training, while the remaining 20% records were used for validation. The results and plot of predicted value from the Random Forest model vs actual data (true project duration in days) are plotted in Figure 47 below. However, the Random forest model does not seem to be able to capture the complex relationship between these variables, mostly due to insufficient feature variables and sparse data size.

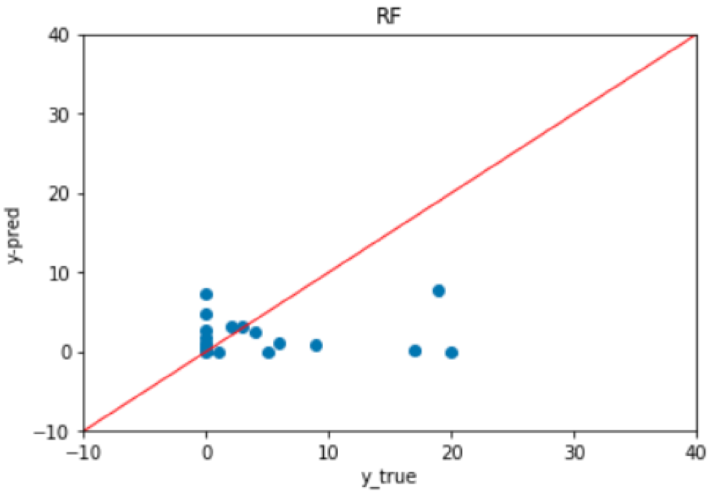


Figure 47. Comparison of predicted value from the Random Forest model (y axis) vs actual data of true project duration in days (x axis)

## CHAPTER 5: DEVELOPING RANDOM FOREST MODEL FOR MARYLAND GROUND WATER DATA

The objective was to train random forest models for selected highway tabular datasets of interest to MDOT SHA including groundwater data and drilling project schedule data, and then use the trained random forest models for dependable variable prediction. Data preparation including extracting relevant data entries from existing datasets, removing null data, filling missing values, normalization and converting data into acceptable format by the random forest models have been conducted. In this study, the research team reviewed groundwater depth data training and predictions and Maryland precipitation data extracted from NOAA data repository.

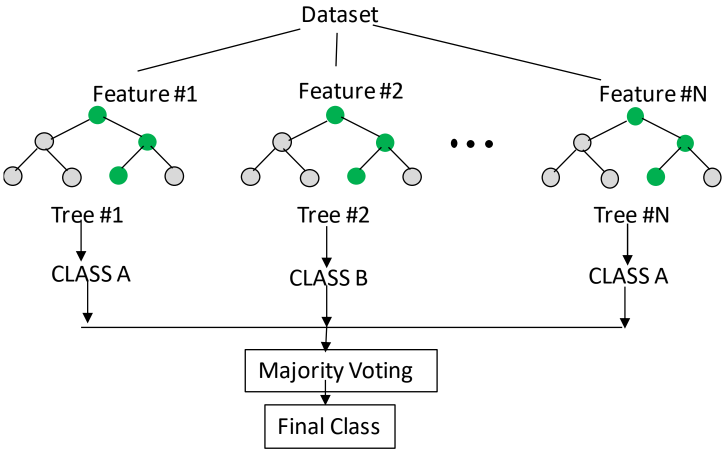
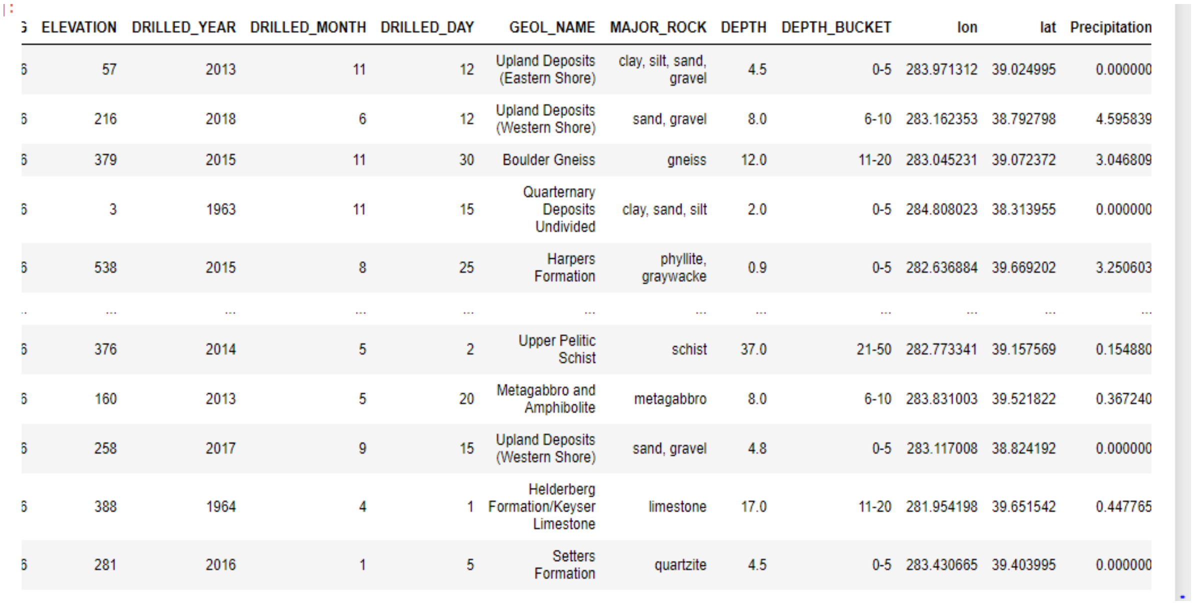


Figure 48. Schematics of Random Forest classifier

Random Forest Classifier is an ensemble tree-based learning algorithm that consists of many decision trees from randomly selected subset of training set, as shown in Figure 48. It aggregates the votes from an uncorrelated forest of decision trees to predict the class. In Breiman's approach (Breiman, 2001), each tree is formed by first selecting at random, at each node, a small group of features to split on and, secondly, by calculating the best split based on these features in the training dataset. The tree is grown using CART methodology to maximum size, without pruning. This subspace randomization scheme is blended with bagging to resample, with replacement, the training data set each time a new individual tree is grown (Biau, 2010). Random forest algorithm usually outperforms neural networks in small datasets (e.g., < 1K samples) according to literature search results. Random forest method was tested on two drilling dataset with relatively small size and low prediction accuracy using the fast.ai tabular model: groundwater depth data and SPTN data.

To utilize the precipitation data as a feature input to the Random Forest model training, the research team first developed scripts to extract NOAA precipitation data (daily, weekly, monthly) in a specified time window. The source of the NOAA data is from the CPC Unified Precipitation Project underway at NOAA Climate Prediction Center (CPC). The land mask for the CPC dataset is such that actual data resides between 20N to 49.5N and 233.75E to 292.75, which basically covers continental US data. The daily precipitation data values are accumulated from 12z of the day before to 12z of the present day. Precipitation data is based on the grids of geographic 0.25x0.25-degree cell. Drilling site GPS coordinates were converted and assigned to the corresponding data by Nearest neighbor algorithm. Each ground water point was assigned to a geographic 0.25x0.25degree cell for both daily accumulated precipitation and monthly mean precipitation.



| G | ELEVATION | DRILLED_YEAR | DRILLED_MONTH | DRILLED_DAY | GEOL_NAME | MAJOR_ROCK | DEPTH | DEPTH_BUCKET | lon | lat | Precipitation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 57 | 2013 | 11 | 12 | Upland Deposits (Eastern Shore) | clay, silt, sand, gravel | 4.5 | 0-5 | 283.971312 | 39.024995 | 0.000000 |
| 6 | 216 | 2018 | 6 | 12 | Upland Deposits (Western Shore) | sand, gravel | 8.0 | 6-10 | 283.162353 | 38.792798 | 4.595839 |
| 6 | 379 | 2015 | 11 | 30 | Boulder Gneiss | gneiss | 12.0 | 11-20 | 283.045231 | 39.072372 | 3.046809 |
| 6 | 3 | 1963 | 11 | 15 | Quarternary Deposits Undivided | clay, sand, silt | 2.0 | 0-5 | 284.808023 | 38.313955 | 0.000000 |
| 6 | 538 | 2015 | 8 | 25 | Harpers Formation | phyllite, graywacke | 0.9 | 0-5 | 282.636884 | 39.669202 | 3.250603 |
| .. | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6 | 376 | 2014 | 5 | 2 | Upper Pelitic Schist | schist | 37.0 | 21-50 | 282.773341 | 39.157569 | 0.154880 |
| 6 | 160 | 2013 | 5 | 20 | Metagabbro and Amphibolite | metagabbro | 8.0 | 6-10 | 283.831003 | 39.521822 | 0.367240 |
| 6 | 258 | 2017 | 9 | 15 | Upland Deposits (Western Shore) | sand, gravel | 4.8 | 0-5 | 283.117008 | 38.824192 | 0.000000 |
| 6 | 388 | 1964 | 4 | 1 | Helderberg Formation/Keyser Limestone | limestone | 17.0 | 11-20 | 281.954198 | 39.651542 | 0.447765 |
| 6 | 281 | 2016 | 1 | 5 | Setters Formation | quartzite | 4.5 | 0-5 | 283.430665 | 39.403995 | 0.000000 |

Figure 49. Groundwater depth data samples

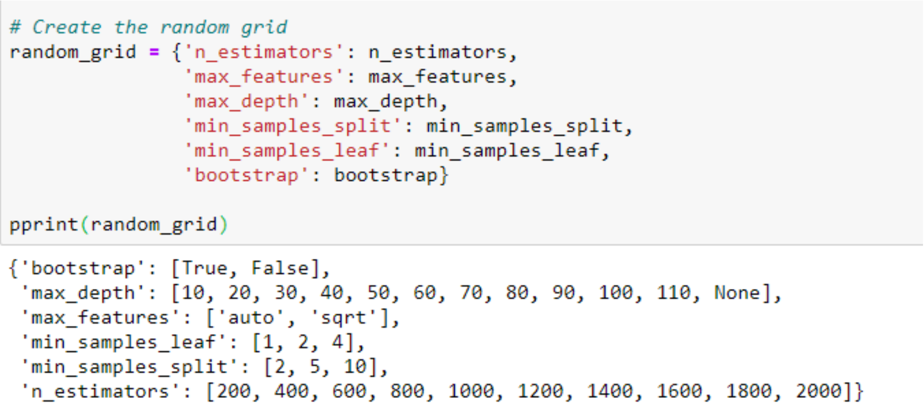
The groundwater depth data from drilling datasets has a total of 7,732 data samples and 10% allotted to validation dataset. Four feature variables were included: 'NORTHING', 'EASTING', 'ELEVATION', 'DRILLED_MONTH'. Three features from Maryland precipitation dataset (described later) were also included in the training dataset for Random Forest model: 'State mean Rainfall': average monthly MD precipitation over a 20 year period (1981-2010); 'Station': weather station name; 'Station Rainfall' (SR): average precipitation at specific station over a 20-year period (from 1981 to 2010). Normalization was applied to variables in the data pre-processing step. It is seen in Table 3 that random Forest performed better.

To see if further improvement of learning results can be achieved or not, tuning the Random Forest model by optimizing hyperparameter values was investigated. Sklearn provides useful tools for hyperparameter tuning: 'RandomizedSearchCV' and 'GridSearchCV'. Documentation for sklearn RandomForest function suggests considering two most important hyperparameters:

- number of trees in the forest (n_estimators)
- number of features considered for splitting at each leaf node (max_features).

Tuning the default Random Forest model was conducted to see if better results can be achieved by applying 'RandomizedSearchCV' and 'GridSearchCV'. Since hyper-parameter tuning is a trial-and-error procedure, a random hyper-parameter grid was created to consider different combinations. An initial trial was used for distributions of selected hyper-parameters as listed in 'random grid' while all other parameters remain unchanged. For classification problem, it is necessary to transform the labels to integers for the function by using 'LabelBinarizer' to transform string labels to NumPy arrays. Then the process of searching for the best parameters for the dataset started, and the original classifier was used as estimator, the parameter distribution is the random grid just created. One hundred combinations of parameters and three sub dataset were considered for cross validation.  To deal with multi-class classification problem, micro_f1 was selected for evaluation metric. The 'best' parameter generated from RandomizedSearchCV is shown in Figure 50. These 'optimal' parameter values generated by 'RandomizedSearchCV' were then used for the classifier.

The hyperparameter values currently used (before optimization) showed an accuracy of 0.6167. The above optimization steps resulted in an accuracy of 0.6233, which had only trivial improvement of 0.66% (small though, but proves the limit of the adopted model with existing dataset) over the original model.



```
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

Figure 50. Hyper-parameter tuning for Random Forest model

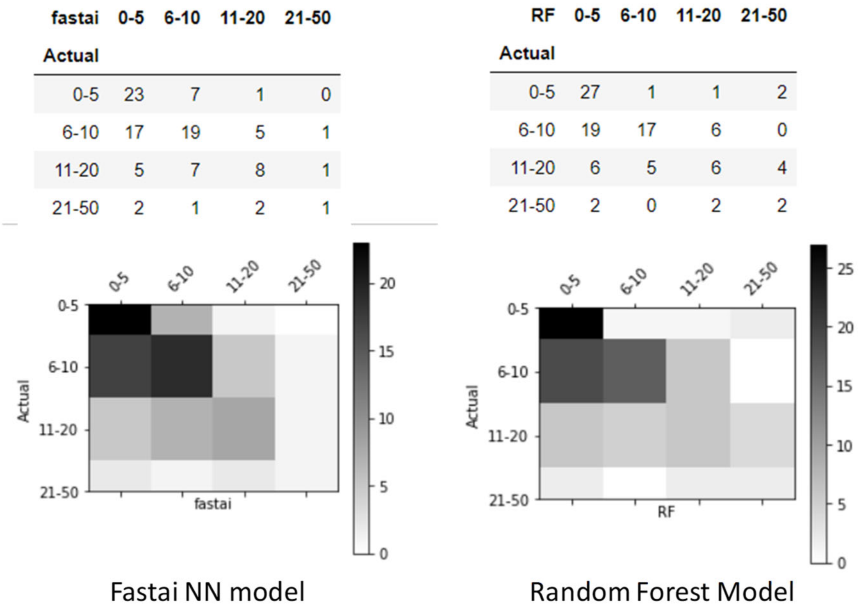
The results suggested adding daily accumulated precipitation data didn't help improve the accuracy for ground water depth prediction, but 'month' and 'monthly mean precipitation' helped improve the accuracy.

Table 3. Comparison of accuracy of 5 test data groups with randomly allocated data

| **Categorical Variables** | | | **Continuous Variables** | | | | **Accuracy (average of five randomly selected validation sets)** |
|---|---|---|---|---|---|---|---|
| **Drilled Year** | Drilled Month | Drilled Day | Northing | Easting | Elevation | Precipitation (12z of drilled day) | |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 0.607 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | 0.608 |
| | | ✓ | ✓ | ✓ | ✓ | ✓ | 0.621 |
| | | ✓ | ✓ | ✓ | ✓ | | 0.623 |

Table 3 shows the mean accuracy of different variable combinations, each accuracy is the average accuracy of five groups with randomly selected 10 percent of training set. The results suggested that adding more variables won't help increase the accuracy. It suggested that fewer categories of variables seemed to have better prediction accuracy, indicating too many individual uncorrelated decision trees might confuse the Random Forest model.

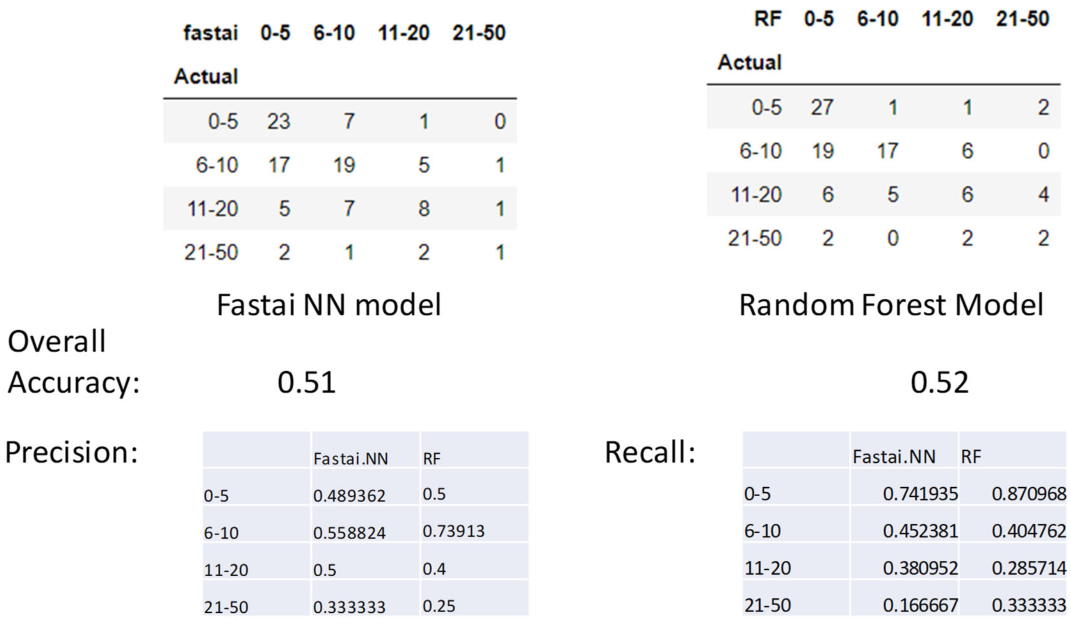
Finally, comparison study of Fast.ai neural network model and Random Forest model was made by calculating confusion Matrix shown in Figures 51 and 52. Same dataset was used for fast.ai neural network model (Fast.ai 2020) and Random Forest model evaluation here. Confusion matrix offers a metric for evaluating a classifier model. For all confusion matrices predicting any arbitrary number of n classes, an n x n matrix is developed in which the diagonals represent true predictions and any value off diagonal is an error in prediction. An independent test set has been created to evaluate the performance of fastai.tabular learner (Fast.ai 2020) and RandomForest model. Testing set size is 100, and training set size is 7,436. The prediction from each model was compared by confusion matrix which shows the distribution. It is concluded that Fast.ai neural network model and Random Forest model provides similar performance results for the groundwater data under consideration.



| fastai | 0-5 | 6-10 | 11-20 | 21-50 |
|---|---|---|---|---|
| **Actual** | | | | |
| 0-5 | 23 | 7 | 1 | 0 |
| 6-10 | 17 | 19 | 5 | 1 |
| 11-20 | 5 | 7 | 8 | 1 |
| 21-50 | 2 | 1 | 2 | 1 |

| RF | 0-5 | 6-10 | 11-20 | 21-50 |
|---|---|---|---|---|
| **Actual** | | | | |
| 0-5 | 27 | 1 | 1 | 2 |
| 6-10 | 19 | 17 | 6 | 0 |
| 11-20 | 6 | 5 | 6 | 4 |
| 21-50 | 2 | 0 | 2 | 2 |

Figure 1. Confusion matrices of Fast.ai neural network model and Random Forest model for groundwater data



| fastai | 0-5 | 6-10 | 11-20 | 21-50 |
|---|---|---|---|---|
| Actual | | | | |
| 0-5 | 23 | 7 | 1 | 0 |
| 6-10 | 17 | 19 | 5 | 1 |
| 11-20 | 5 | 7 | 8 | 1 |
| 21-50 | 2 | 1 | 2 | 1 |

Fastai NN model

| RF | 0-5 | 6-10 | 11-20 | 21-50 |
|---|---|---|---|---|
| Actual | | | | |
| 0-5 | 27 | 1 | 1 | 2 |
| 6-10 | 19 | 17 | 6 | 0 |
| 11-20 | 6 | 5 | 6 | 4 |
| 21-50 | 2 | 0 | 2 | 2 |

Random Forest Model

Overall Accuracy: 0.51 0.52

Precision:

| | Fastai.NN | RF |
|---|---|---|
| 0-5 | 0.489362 | 0.5 |
| 6-10 | 0.558824 | 0.73913 |
| 11-20 | 0.5 | 0.4 |
| 21-50 | 0.333333 | 0.25 |

Recall:

| | Fastai.NN | RF |
|---|---|---|
| 0-5 | 0.741935 | 0.870968 |
| 6-10 | 0.452381 | 0.404762 |
| 11-20 | 0.380952 | 0.285714 |
| 21-50 | 0.166667 | 0.333333 |

Figure 51. Performance results of Fast.ai neural network model and Random Forest model for groundwater data

# CHAPTER 6: SUMMARY & CONCLUSIONS

In this study, data-driven machine learning models that can be used to represent a variety of highway datasets such as drilling and pavement data, pavement thickness data, object detection in highway image data and drilling project schedule estimation have been investigated and their prediction performance were tested for validity. Team effort and coordination with MDOT SHA staffs and engineers has been recognized as a very important factor to the success of this research, especially in identifying the research opportunities in highway datasets, retrieving and converting raw datasets from database servers to make them suitable for machine learning use, and providing professional guidance on machine learning model feature variables and desired output formats (e.g., FWD data bucket sizes in prediction). Four types of machine learning algorithms were studied, including neural network model for tabular data, deep learning based object detection model, reinforcement learning and random forest models for different applications. Specifically, the research team developed and tested a simple reinforcement learning model to demonstrate drilling project schedule estimation using 212 historical data records. The research team also developed and tested random forest model for one type of drilling data – groundwater depth. A state-of-art object detection model using YOLO v3 algorithm were also tested for detecting objects of interest such as traffic barrier end treatments and insect blocking in ROW images. The YOLO v3 spp model exhibited faster training and testing speed, as well as increased accuracy in detection and full performance metrics output, compared with other open-source object detection models tested in this study.

The research team also developed processes to perform QA/QC on ROW imagery in order to quickly scan and find images with different types of defects including blurriness, underexposure, overexposure, no signal, discolored, and signal issues. A python script has been developed that can run any set of images for a particular year and county in Maryland. The research team have tested this basic QA/QC tool with the ROW images collected from one county in Maryland and it was found this method can effectively find potential issues. Optimization of the blurry and no signal thresholds was also done to catch blurry photos and minimize false positives.

In general, machine learning models are trained to represent high dimensional data (including images) or complex relationship hidden in the dataset for which traditional mathematical models are ill to describe. The accuracy of this representation relies on the amount and quality of the data available for model training in order to remove the uncertainty and noise from a variety of factors. It is found that data size and quality was critical to the performance of machine learning models under consideration. For the drilling datasets considered, the data size ranged from a few hundred to nearly 270K data samples for SPT (standard penetration test) Grainsize data. For the Fast.ai neural network models trained with these drilling datasets, the following accuracy values were observed: For the SWM Infiltration dataset with only 491 data samples, the highest accuracy achieved was 0.729, partly because this model involved only two classes for the target variable to predict and thus only a simple relationship is expected to be revealed by the model between the feature variables and the target variable. However, for the SPT Grainsize data with 274K data samples, slightly lower accuracy value of 0.67 was observed in training, due to more grainsize buckets (classes) for the target variable and thus a more complex relationship. For the SPT N (SPT counts) data with approximately 33K data samples, a fairly low accuracy of 0.37 was seen, likely due to the even more complex relationship to be represented in this data. Therefore, the machine learning model performance depends not only on the data size and data quality of the highway dataset under consideration, but also the complexity of the inherent relationship hidden in the data which machine learning model attempts to represent. As more data samples become available in the future, it is expected that machine learning model performance will continue to improve. For data quality control, data preparation including extracting relevant data entries from existing datasets, removing null data, discarding redundant data samples, filling missing (null) values, normalization to make the data suitable for machine learning model use have been conducted as a standard practice.

For the reinforcement learning models developed for drilling project schedule estimation, only 212 historical records were available. This simple model was to generate a Q-table by finding the closest sample to the input among all available historical records. The historical records have been sorted in ascending order based on corresponding resource demand value. In this study, three types of resource demand functions were defined and their performance in correlation with the target variable – project duration was compared. More complex functions for the resource demand metrics such as predictions by advanced machine learning models can be adopted in the future to seek close correlation with the real data. Training data size and quality is also an important factor affecting the performance of the reinforcement learning model for project schedule estimation. To further improve this model, future work is required to collect additional real data since 212 data samples were found to be insufficient to train a reinforcement learning model with high accuracy for project schedule estimation.

For the YOLO v3 spp object detection models, training datasets with 50 to 70 images usually gave reasonably good accuracy for the objects of interests in this study including traffic barrier end treatments and insect blocking detection in the ROW images. The model exhibited fast training and testing speed. Training time took approximately 4.3 hours for the traffic barrier end treatment objects and 6.5 hours for insect blocking detection for around 200 training epochs. Once the YOLO v3 spp model was trained, using the model to execute the object detect task took less than 1 second in scanning each image with pixel size 1920 x 1080. For insect blocking detection, it took 68.4 seconds total to complete 3,199 pictures used for test (thus each picture took about 0.16 seconds to scan). The YOLO v3 spp model in detecting the three objects of interest in this study have shown fairly high accuracy over 90% for the test image datasets. Transfer learning is the recommended training strategy for object detection application; by doing so the training data size can be made far less than the super large data size that would be typically required for training a deep learning model from scratch.

Based on the findings from this research, the suitability and applicability of each machine learning model type considered in this study of highway datasets are summarized below,

- Fast.ai neural network model for tabular data: Fast.ai model has demonstrated the state-of-art performance in the tabular data modeling in this study. Tabular data here refers to text (or ascii data) data typically stored in a SQL database and spreadsheet file. Two types of feedforward neural network models are available in Fast.ai for tabular data modeling: regression model and classification model. For classification model, the training target is to classify the data samples into corresponding discrete classes. The output of a regression model is nonetheless a continuous variable. In this study, default feedforward neural network model architecture generally had four hidden layers and the 400 to 1,200 neurons in each layer depending on the quality and amount of data samples available for training. The prediction accuracy derives from the neural network models trained with large number of data samples to represent the relationship hidden in the selected dataset. Default number of training epochs was initially set as 40 in this study and optimal training of the neural network model should be done by stop training when the training loss value is close to validation loss value.

- Object detection model: YOLO v3 spp model adopted in this study demonstrated state of the art performance in object detection with high computation speed and better computational efficiency. YOLO v3 model can be used to detect multiple classes of objects in a single picture. Training datasets for a single class YOLO v3 model require 50 to 100 images that ideally have the object of interest presented at different scales and with different backgrounds. With commonly available computing hardware (e.g., Nvidia Titan GPU card), training time usually took a few hours with 200 training epochs.
- Reinforcement learning: reinforcement learning models are well-suited to construction resource management and construction schedule estimation applications. Reinforcement learning models have been used for decision-making policies as deep neural networks analogous to the models used for game-playing agents. Q-learning is a commonly used model-free approach which can be used for building the environment by training an agent. In this study, Q-learning algorithm was adopted to build a reinforcement learning model for project duration estimation. While this preliminary study showed initial promise of reinforcement learning model training, additional data samples which fully captures the influencing factors and complex relationship in drilling project schedule estimation are needed to further optimize the reinforcement learning model. In this reinforcement learning model trained with 212 data samples, 5,000 simulation cycles generally lead to convergence in training.
- Random forest model for tabular data: Random forest models are generally believed to outperforms neural networks if applied to small datasets less than 1K data samples. In this study, random forest method was tested on two drilling dataset with relatively small size: groundwater depth data and SPT N data. However, the test results showed that the Fast.ai neural network model and Random Forest model trained with these two datasets provide similar performance results. For tabular data modeling, it is thus recommended to use Fast.ai neural network model for tabular data to reduce learning curve and streamline

the machine learning modeling process by avoiding different algorithms for the same dataset.

# REFERENCES

Andriotis, C.P., K.G. Papakonstantinou. (2019). “Managing engineering systems with large state and action spaces through deep reinforcement learning,” Reliability Engineering & System Safety, Volume 191: 106483.

Bhatt, S. (2018). “Reinforcement Learning 101: Learn the essentials of Reinforcement Learning!” Towards data science, https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292 (posted March 19, 2018).

Biau, G. (2012). “Analysis of a Random Forest Model,” J. of Machine Learning Research, 13: 1063-1095.

Breiman, L. (2001). “Random forest,” Machine Learning, 45: 5-32.

Fast.ai (2020). Tabular model, https://docs.fast.ai/tabular.model.html.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. MIT Press, Boston, Massachusetts. ISBN: 9780262035613.

Glorot, X., & Bengio, Y. (2010). “Understanding the difficulty of training deep feedforward neural networks.” *In Aistats* (Vol. 9, pp. 249-256).

Ioffe, S., C. Szegedy, (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” https://arxiv.org/abs/1502.03167

Lawal, M.O. (2021). “Tomato detection based on modified YOLOv3 framework,” Scientific Reports, vol. 11, Article number: 1447.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). “Deep learning.” *Nature*, 521(7553), 436-444.

Liu, H. and Zhang, Y. (2020). “Bridge condition rating data modeling using deep learning algorithm,” Structure and Infrastructure Engineering, Published online, https://doi.org/10.1080/15732479.2020.1712610

Liu, H. and Zhang, Y. (2019). “Image-driven structural steel damage condition assessment method using deep learning algorithm.” Measurement, 133: 168-181.

Mao, H., Alizadeh, M., Menache, I, and Kandula, S. (2016). “Resource Management with Deep Reinforcement Learning,” *HotNets-XV,* November 9-10, 2016, Atlanta, GA, USA.

Redmon, J. and A. Farhadi. (2018). “YOLOv3: An Incremental Improvement,” arXiv:1804.02767 [cs.CV].

Santurkar, S., D. Tsipras, A. Ilyas and A. Madry. (2019). “How Does Batch Normalization Help Optimization?” NerurIPS’18, https://arxiv.org/abs/1805.11604

Sutton, R.S. and Barto, A.G. (2018). Reinforcement Learning: an Introduction, 2$^{nd}$ edition, The MIT Press, Cambridge, Massachusetts.

Ultralytics. (2020). YOLO v3 spp model for object detection, https://github.com/ultralytics/YOLOv3.