RESEARCH

UDOT

Keeping Utah Moving

Utah Department of Transportation - Research Division
4501 South 2700 West - P.O. Box 148410 - SLC,UT 84114-8410

Report No. UT-23.06

AUTOMATED SAFETY ASSESSMENT OF RURAL ROADWAYS USING COMPUTER VISION

Prepared For:

Utah Department of Transportation
Research & Innovation Division

Final Report
April 2023

DISCLAIMER

The authors alone are responsible for the preparation and accuracy of the information, data, analysis, discussions, recommendations, and conclusions presented herein. The contents do not necessarily reflect the views, opinions, endorsements, or policies of the Utah Department of Transportation or the U.S. Department of Transportation. The Utah Department of Transportation makes no representation or warranty of any kind, and assumes no liability, therefore.

ACKNOWLEDGMENTS

The authors acknowledge the Utah Department of Transportation (UDOT) for funding this research, and the following individuals from UDOT on the Technical Advisory Committee for helping to guide the research:

- Robert Chamberlin (consultant)
- Jeff Lewis
- Ivana Vladisavljevic
- Clancy Black
- Scott Jones
- Rudy Zamora
- Chris Siavrakas
- Derek Lowe

TECHNICAL REPORT ABSTRACT

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
UT-23.06	N/A	N/A

4. Title and Subtitle	5. Report Date
Automated Safety Assessment of Rural Roadways Using Computer Vision	April 2023
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Ali Hassandokht Mashhadi, Abbas Rashidi, Nikola Markovic	

9. Performing Organization Name and Address	10. Work Unit No.
The University of Utah Department of Civil and Environmental Engineering 201 Presidents Circle Salt Lake City, Utah 84112	74041 15D
	11. Contract or Grant No. 228094

12. Sponsoring Agency Name and Address	13. Type of Report & Period Covered
Utah Department of Transportation 4501 South 2700 West P.O. Box 148410 Salt Lake City, UT 84114-8410	Final August 2022 to April 2023
	14. Sponsoring Agency Code PIC No. UT21.315

15. Supplementary Notes

Prepared in cooperation with the Utah Department of Transportation and the U.S. Department of Transportation, Federal Highway Administration

16. Abstract

Roadside elements play an important role in the number and severity of crashes. Rigid obstacles (trees, rocks, embankments, etc.), guardrails, clear zones, and side slopes are among the factors that might affect roadside safety. The Federal Highway Administration (FHWA) presented a rating system to help DOTs and transportation agencies make better decisions about improving road segments. However, the manual process of rating road segments is time consuming, inconsistent, and labor intensive. To this end, this project proposed an automated rating system based on images taken from Utah roadways. Utilizing machine-learning algorithms and Mandli images, the developed approach employs the FHWA rating system as the primary standard for assessing roadside safety. To provide more detailed information about safety conditions on the roadside, various computer vision algorithms have been developed to detect each roadside feature. The pre-trained models for available clear zone detection and side slope classification have also been established. A shape-file has been generated by assigning a safety ranking to road segments on five state roads. This product can assist traffic engineers in decision-making to improve road safety by prioritizing projects that address problematic locations. The results show a promising approach to enhancing road safety and preventing crashes.

17. Key Words	18. Distribution Statement	23. Registrant's Seal
Roadside Safety, Automation, Machine Learning, Image Processing, Computer Vision	Not restricted. Available through: UDOT Research & Innovation Division 4501 South 2700 West P.O. Box 148410 Salt Lake City, UT 84114-8410 www.udot.utah.gov/go/research	N/A

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		N/A

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

UNIT CONVERSION FACTORS

NO UNIT CONVERSIONS

EXECUTIVE SUMMARY

Due to the importance of roadside hazards in the number and severity of rural crashes, this project is implemented in order to identify the most dangerous road segments based on the FHWA safety ranking. The images collected during pavement surface evaluations by Mandli Communications, Inc.[1] are a valuable resource for vision-based assessments. Mandli is a company dedicated to collecting data from U.S. highways using 3D pavement technology, mobile LiDAR, and geospatial technologies. The collected data can be used to evaluate safety parameters along each road segment. Before training the models, various preprocessing techniques were applied to the collected images, such as color correction, resizing, and cropping.

In this research, various models have been developed for detecting each criterion on the roadside. Two binary logistic regressions have been developed to detect guardrails and rigid roadside obstacles. In order to detect the distance between the obstacle and pavement edgeline, a multinomial logistic regression model has been developed. However, the developed model for detecting the distance of rigid obstacles had moderate accuracy. A third model has been trained to enhance the results, combining the previous two models for detecting rigid roadside obstacles, yielding a significant improvement in accuracy. The pre-trained models have been developed to detect the available clear zone, achieving an accuracy of 83%. Furthermore, a pretrained model has been established for classifying images based on the sideslope into three classes: low, mid, and high. The model demonstrated an accuracy of 94% in identifying the correct class for roadside slopes.

Using the extracted features and developed algorithm, a safety ranking has been assigned to road segments on five state roads, US-6, SR-10, SR-12, US-40, and SR-150. The final product is a shape file comprising safety ranking and roadside features at each given interval on these six roads. This final product can assist traffic engineers in the decision-making process for improving the safety level of each road segment. With this information, UDOT can prioritize projects that address problematic locations, such as removing overgrown trees and installing guardrails, thereby improving road safety and preventing crashes.

[1] https://www.mandli.com/, https://roadview.udot.utah.gov/utah/index.php

1.0 INTRODUCTION

1.1 Introduction

In 2020, the highest number of fatalities since 2007 occurred on U.S. roadways, with 38,824 people killed in motor vehicle traffic crashes, representing a 6.8 percent increase from 36,355 fatalities in 2019 (Stewart, 2022). According to the Federal Highway Administration (FHWA), roadway departures, which occur when a vehicle crosses the edge line or centerline or leaves the main road, accounted for about 50 percent of all traffic fatalities between 2016 and 2018 (Roadway Departure Safety, n.d.). In Utah, more than 40 percent of all fatalities are associated with roadway departures, even though they make up only 15 percent of crashes (*Utah SHSP*, n.d.). The higher occurrence of roadway departures in rural areas is most likely attributable to a number of contributing factors (Reducing Rural Roadway Departures, n.d.), including:

- Higher average roadway speeds
- Steeper embankments
- More hills and curves
- Less lighting

Figure 1 illustrates the primary crash categories for roadside departures and their respective proportions in rural areas in the United States. (Satterfield et al., n.d.).

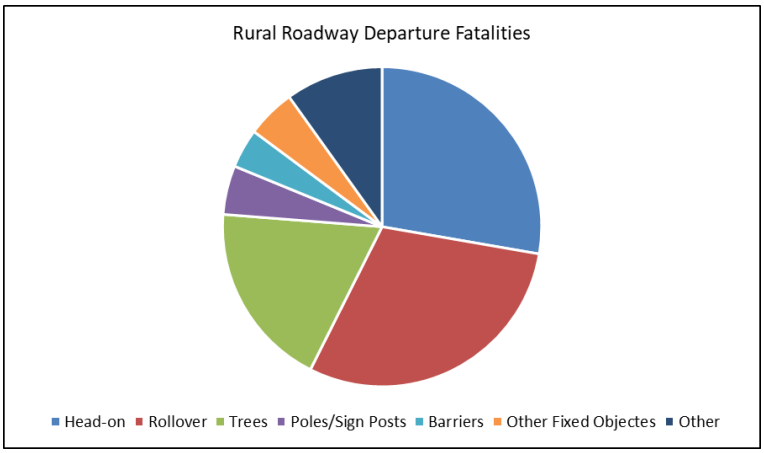

Figure 1. Rural Roadway Departure Fatalities Categories

Roadside condition plays a critical role in preventing potential crashes or minimizing the severity of crashes. For example, providing an adequate clear zone width helps drivers gain control

of the departed vehicle and stop safely. Icy road surfaces during the winter can cause collisions or diversion from the main road, leading to hitting the roadside hazards or crossing into the path of other vehicles that can be prevented using guardrails or concrete barriers. In order to evaluate roadside conditions, FHWA proposed a rating system ranked on a seven-point categorical scale from 1 (best) to 7 (worst). Considering the importance of roadside safety, an automated safety assessment system can inform safety engineers of problematic road segments, minimizing the number and severity of crashes caused by roadside safety conditions.

Several attempts have been made to address roadside safety conditions. The most commonly investigated factors are guardrail detection, safety barrier type detection, and clear zones. Moreover, LiDAR and point cloud technology were used to evaluate factors such as clear zones (Gouda et al., 2021). However, these methods have drawbacks in terms of accuracy, feasibility, and cost. Additionally, all criteria must be considered simultaneously to evaluate roadside safety conditions. The following describes the current problem with the existing methods for evaluating roadside safety in rural areas.

1.2 Problem Statement

Detecting roadside features that have a negative impact on traffic safety is imperative for all state DOTs. This can be achieved by manually inspecting videos and images collected by third-party data providers, such as Mandli Communications, Inc. However, this process is both time consuming and susceptible to human error. Figure 2 depicts a sample image taken from I-80 at the mileage point of 0.85.

Figure 2. A Sample Image From the Roadside of I-80

In recent years, many transportation-related problems have been tackled using computer vision. Pavement monitoring, vehicle detection, road safety, and asset management are topics researchers have addressed using various algorithms, including Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN) (Farhadmanesh et al., 2021a, 2021b; Farhadmanesh, Marković, et al., 2022; Farhadmanesh, Rashidi, et al., 2022; Matsumoto et al., 2021; Sherafat et al., 2022). Thanks to vast imagery and video data collected on Utah roads, these algorithms can also be used to facilitate the process of evaluating roadside features. To this end, we proposed developing an automated approach that leverages computer vision and machine learning to rate rural roadways based on different roadside safety criteria (e.g., side slopes, guardrails, and obstacles).

Various algorithms have been developed to detect each roadside safety criterion in this research. This approach does not need hardware other than an imagery dataset. The final product of this project is a map consisting of road segments with safety rankings and detailed information on the safety condition of the road segment. Figure 3 illustrates an example of the final product.

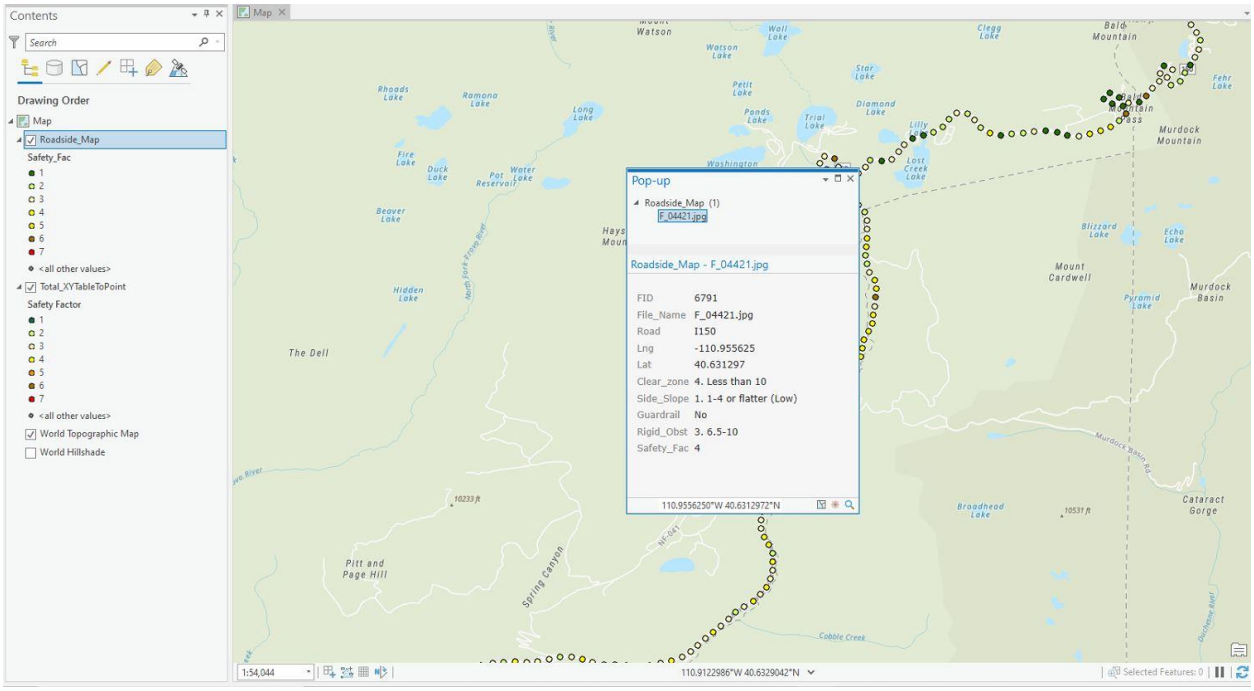

Figure 3. An Example of the Final Product

1.3 Background

Identifying the factors affecting the safety performance of rural roadways is essential for evaluating the current status and indicating problematic segments. The major factors contributing to the rural roadside safety risk are horizontal curve radius, longitudinal gradient, side slope grade, side slope height, the distance between roadway edge and fixed obstacles, the density of the fixed obstacles (like trees, utility poles), and density of continuous fixed objects like substandard roadside safety barriers (Tang et al., 2019). The Federal Highway Administration (FHWA) provided a rating of the safety of the roadside on a scale of 1 (best) to 7 (worst). Factors like clear zones, side slopes, and the presence of guardrails are the factors listed in the FHWA rating system. Due to the importance of roadside conditions, many studies have investigated the effect of roadside elements on the number and severity of crashes, which are summarized in Table 1.

Table 1. Summarizing the Literature on the Effect of Roadside Elements on Road Crashes

Study	Approach	Findings
(Lee & Mannering, 2002)	Zero-inflated count models and nested logit models	Frequency of run-off-roadway (ROR) crashes could be decreased by decreasing the number of trees along the road, avoiding cut side slopes, and increasing the distance of utility poles from the shoulder edge
(Gross et al., 2009)	Case-control method to check the effect of different lane-shoulder combinations	Provided crash modification factors (CMF) for different combinations of changes in lane and shoulder when total width of the road has to be kept constant.
(Lord et al., 2011)	Regression Analysis	Roadsides with wider shoulders at curves will have less frequency of roadside crashes
(Daniello & Gabler, 2011)	Exploratory data analysis	Collison with trees is 15 times more likely to be fatal than a collision with the ground
(Zou et al., 2014)	Binary logistic regression model	Find the risk of severity while hitting the guardrails; the risk of cable barrier was less than the rollover or hitting a pole or any fixed objects
(Manuel et al., 2014)	Negative binomial safety performance function for total collisions	Segment length, traffic volume, access-point density, and midblock changes were positively related, while the width is negatively related to collisions
(Roque et al., 2015)	Multinomial and mixed logit models using driver injury and severely injured occupant as the outcome variable	Critical slopes and horizontal curves without guardrail barriers significantly contributed to the run-off-roadway crashes
(J. Park & Abdel-Aty, 2015)	Naïve Bayes, generalized nonlinear models, multivariate adaptive regression spline	The number of crashes was reduced when the distance from the poles was increased
(Ewan et al., 2016)	Multivariate regression and correlation analysis	Roads with no shoulder have higher crash rates than roads with 4- to 5-foot shoulders.
(Haghighi et al., 2018)	Standard ordered logit model and multilevel order logit model (using hierarchical crash data) to evaluate the effect of roadway features on crash occurrence on a rural two-lane road	Lower risk of severe crashes in the presence of 10-ft-lane road, lower roadside hazards, higher driveway density, longer barrier length

As shown in Table 1, the number of studies focusing on roadside safety has increased over the last few years. This may be due to the importance of roadside elements on the number and severity of rural crashes and improvements in available technology and data.

Detection of locations with inadequate safety measures is of utmost importance to DOTs. The current practice is based on manual evaluation of roadside safety, which is time consuming and labor intensive. With the rapid advancement of computer vision algorithms, applying computer vision systems toward roadway safety is inevitable. LiDAR data, 2D images, videos, and simulation-based analysis are some of the most common data for evaluating roadside safety. For example, (Gao et al., 2020) used mobile laser scanning to detect urban guardrails using density-based spatial clustering of applications with noise (DBSCAN) and multilevel filtering techniques. However, the algorithm is unsuitable for detecting curve guardrails because of the straight-line fitting approach. In another study, (Zhong et al., 2019) used the point-cloud-based classification method for detecting roadside safety attributes and distances between them. They used classification and segmentation to detect the attributes and center approximation to find the centers of an object and applied Euclidean distance to calculate the distance between two objects. Though this method could detect the pole and tree with higher precision, it could only capture a limited number of objects compared to the ground truth (mean Intersection over Union (IOU) was only 63%).

Rezapour & Ksaibati (2021a) use the convolutional neural network (CNN) for roadside barrier detection using transfer and non-transfer learning. They used the transfer learning derived from the denseNet121, VGG19, and inception v3 algorithms and compared the accuracy of transfer learning with non-transfer learning in detecting various types of barriers (box beam, cable, concrete, hybrid). They found out that the accuracy of non-transfer learning was 85%, while transfer learning (Inception v3, Densenet 121, VGG 19) has an accuracy of 78%, 65%, and 97%, respectively. Only VGG19 has more accuracy than non-transfer learning. The author speculates that this might be due to a large number of weights in the architecture of the VGG19, which enabled the model to learn more complex things easily. The author confirms that non-transfer learning cannot accurately detect the presence of either box beam alone or box beam and guardrail configurations simultaneously. Due to the critical role of vegetation as a rigid obstacle on the roadside, many studies focused on detecting and classifying them. For instance, (Harbaš et al., 2018) used convolution networks for roadside vegetation detections.

Similarly, (Lau et al., 2015) used a shallow neural network, a radial basis function Neural Network (RBFNN), with two different training approaches for the recognition of Malaysian Traffic signs and compared those results with deep neural networks like convolution neural networks in the presence and absence of the Gaussian white noise in datasets. They used the incremental and batch training approaches for RBFNN and found that incremental training trains faster than batch training. The recognition rate performance of their CNN model was 99% for traffic road signs. Table 2 summarizes some of the most well-known CNN architectures.

Table 2. Summarizing Common CNN Architectures (Tested algorithms are specified with *)

Year	CNN Algorithm	Error Rate on ImageNet	No. of Parameters
1998	LeNet	-	60 thousand
2012	AlexNet	15.3%	60 Million
2013	ZFNet	14.8%	-
2014	GoogLeNet	6.67%	4 Million
2014	VGGNet*	7.3%	138 Million
2015	ResNet*	3.6%	
2015	Inception v3*	4.2%	-
2018	NasNet	2.4%	3.2 Million

Based on the literature, most of the existing studies focused on only one of the roadside safety parameters. Moreover, the current practice is based on using LiDAR data to classify and detect roadside safety issues. Although the LiDAR data is accurate, LiDAR data is expensive both on the data collection and the data processing sides. To this end, this project suggests using 2D images to classify roadside safety elements and accordingly rank them based on the FHWA rating system.

1.4 Objectives

This project aims to assist the Utah Department of Transportation (UDOT) in screening rural roadways and accordingly prioritize projects aimed at improving safety levels (e.g., removing trees and adding guardrails). The proposed method uses machine-learning algorithms and Mandli

images as input. The recommended criteria for evaluating roadside safety is the FHWA rating system, which is listed in the following table:

Table 3. FHWA Rating of the Safety of the Roadside

Rating	Criteria
1	• Wide clear zones greater than or equal to 30 ft from the pavement edge line • Side slope flatter than 1:4 • Recoverable
2	• Clear zone between 20 and 25 ft from the pavement edge line • Side slope about 1:4 • Recoverable
3	• Clear zone about 10 ft from the pavement edge line • Side slope about 1:3 or 1:4 • Rough roadside surface • Marginally recoverable
4	• Clear zone between 5 to 10 ft from pavement edgeline • Side slope about 1:3 or 1:4 • May have guardrails (5 to 6.5 ft from pavement edge line) May have exposed trees, poles, or other objects (about 10 ft from pavement edgeline) • Marginally forgiving, but increased chance of a reportable roadside collision
5	• Clear zone between 5 to 10 ft from the pavement edge line • Side slope about 1:3 • May have guardrails (0 to 5 ft from pavement edge line) • May have rigid obstacles or embankments within 6.5 to 10 ft of the pavement edge line • Virtually non-recoverable.
6	• Clear zone less than or equal to 5 ft • Side slope about 1:2 • No guardrail • Exposed rigid obstacles within 0 to 6.5 ft of the pavement edge line • Non-recoverable
7	• Clear zone less than or equal to 5 ft • Side slope 1:2 or steeper • Cliff or vertical rock-cut • No guardrail • Non-recoverable with a high likelihood of severe injuries from roadside collisions

The proposed system includes three phases: 1) Image Labeling, 2) Model Training, and 3) Application and Visualization. Detailed information about model training is provided in the research methods section. Also, more information about the final product could be found in the system evaluation section. The final product of this project will help traffic engineers in the decision-making process for improving the safety level of each road segment.

1.5 Outline of Report

- Introduction
- Research Methods
- Data Collection
- Data Evaluation
- Conclusion

2.0 RESEARCH METHODS

Computer vision is an area of machine learning concerned with the automatic interpretation, analysis, extraction, and understanding of images and video. These models are designed to translate visual data based on features and contextual information identified during training, enabling them to interpret images and videos and use those interpretations for predictive and decision-making tasks (*Deep Learning for Computer Vision*, n.d.). Computer vision tasks can be categorized into three main groups:

- **Image Classification**: labeling images into one of several predefined classes
- **Object Detection**: identifying and locating objects within an image or video
- **Image Segmentation**: partitioning an image into multiple parts or regions

In this section, first, image preprocessing operations will be discussed. Then, various image classification algorithms, advantages, and disadvantages will be explained. Finally, the selected algorithms for the image classification task will be elaborated on in Section 2.3.

2.1 Image Preprocessing

The term "preprocessing" refers to operations with images at the lowest level of abstraction before feeding into the machine or deep learning algorithm. This includes resizing, color correction, noise removal, and data augmentation.

2.1.1 Image Resizing

Because cameras can capture images of varying sizes, a standard size must be set up for all images fed into the machine-learning algorithms. 256×256 pixels are the dimensions used in this project.

2.1.2 Color Correction

Color correction refers to adjusting raw image data that converts a camera-dependent RGB color space to a standard color space.

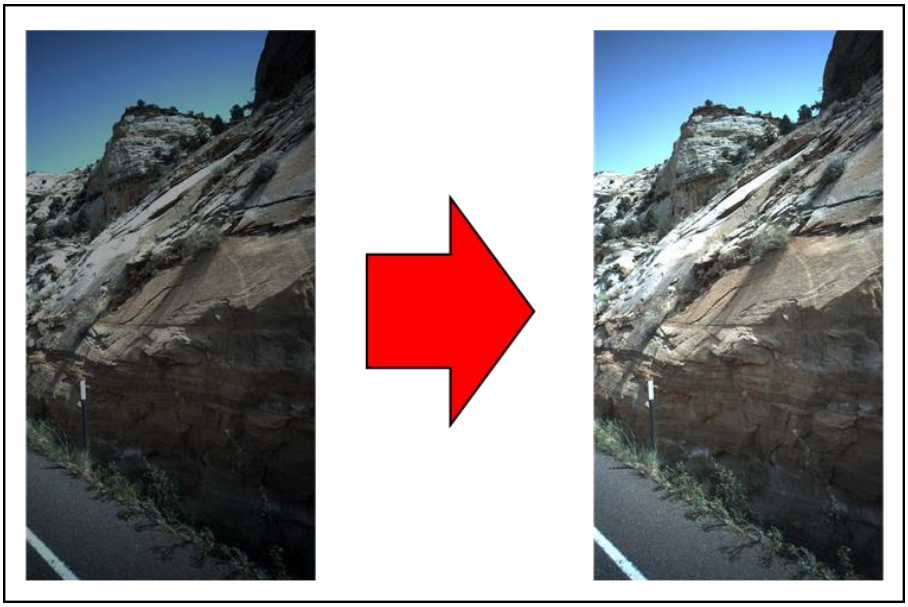

Figure 4. Color Correction Results

2.1.3 Noise Removal

Noise removal is the process of reducing or removing the visibility of noise by smoothing the image. This technique is particularly useful when dealing with low-quality images or images that were captured under poor lighting conditions.

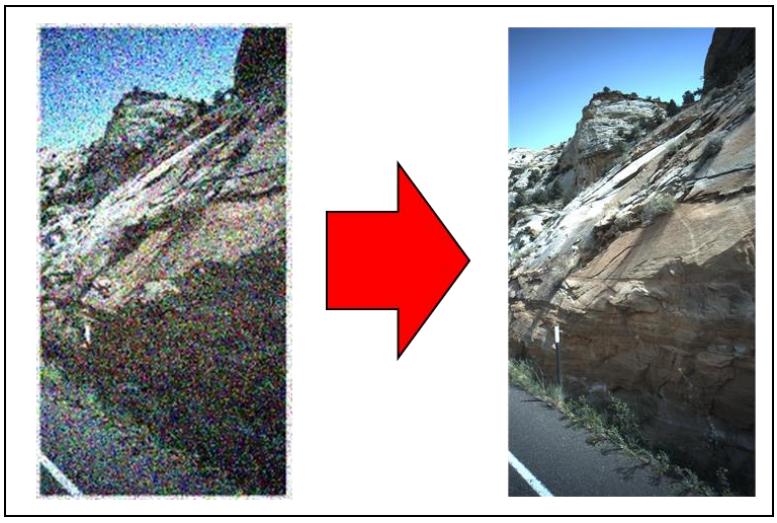

Figure 5. Noise Removal Results

2.2 Data Augmentation

Data augmentation is a technique used in machine learning and deep learning to increase the size of a dataset by creating additional training data from existing data. Data augmentation techniques vary depending on the type of data being used. In image data, common augmentation techniques include flipping, rotating, scaling, and adding noise or distortions to the image. By generating new training data from existing data, data augmentation can help to prevent overfitting, a common problem in machine learning where a model performs well on the training data but poorly on new, unseen data. Data augmentation can also help improve the model's robustness by exposing it to a wider range of variations in the data. Therefore, this process has two advantages: 1) generating more data; 2) preventing overfitting.

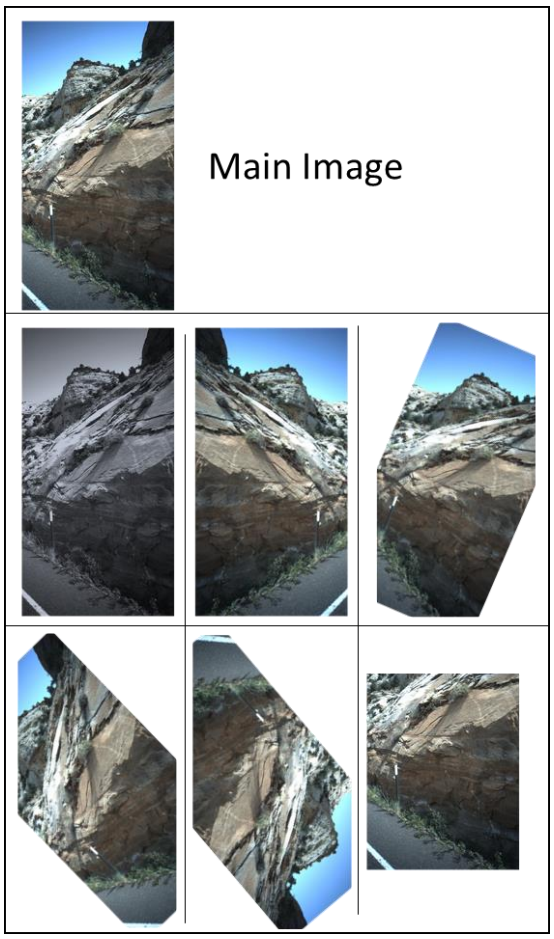

Figure 6. Sample Implementation of Data Augmentation

2.3 Image Classification Techniques

Classification is the process of labeling an image according to certain rules. One or more spectral or textural characterizations may be employed to determine the categorization rule. Image classification techniques can be categorized into two main groups: unsupervised and supervised.

2.3.1 Unsupervised Classification

Unsupervised classification technique is a fully automated method that uses machine learning algorithms to analyze and cluster unlabeled images. This task is being performed by discovering hidden image patterns without human intervention. K-means and ISODATA are the most popular unsupervised classification techniques.

2.3.1.1 K-means

K-means is a clustering algorithm used in unsupervised machine learning to partition a dataset into a pre-determined number of clusters based on the similarity of the data points. The algorithm works by iteratively assigning data points to the nearest cluster centroid and updating the centroids based on the mean of the data points assigned to each cluster. Figure 7 shows an example of clustering data points into three groups.

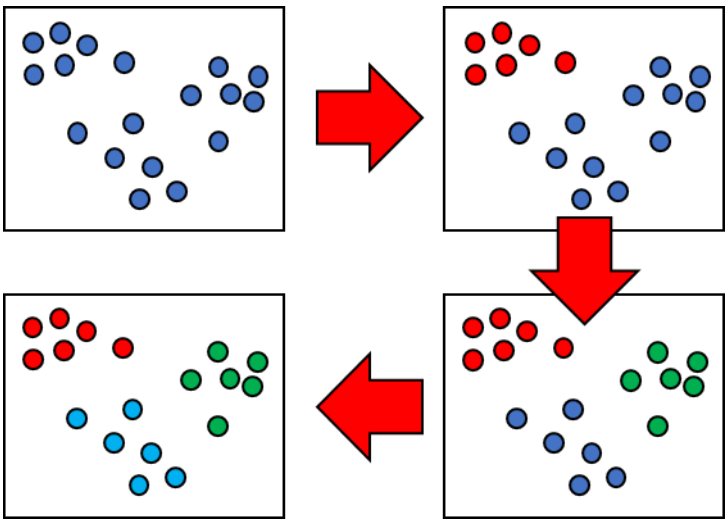

Figure 7. Data Points Clustering

2.3.1.2 ISODATA

The Iterative Self-Organizing Data Analysis Technique (ISODATA) is an unsupervised method that includes Euclidean distance iteratively as the similarity measure to cluster data points into different classes. Unlike K-means, the ISODATA algorithm does not assume the number of clusters a priori and allows for different numbers of clusters.

ISODATA has several advantages over other clustering algorithms, including its ability to handle datasets with varying cluster sizes and shapes and its adaptability to changing cluster structures. However, it also has some limitations, such as its sensitivity to the initial cluster assignments and difficulty determining the appropriate number of clusters for a given dataset.

2.3.2 Supervised Classification

Unlike unsupervised classification, supervised classification methods need previously labeled data (images) to train the classifier. In this technique, a part of the dataset (training set) is labeled manually and assigned to pre-chosen categories, such as car, boat, bicycle, and bus. This process allows the model to be learned and creates statistical measures that can be applied to the dataset. In the following, different image classifiers will be elaborated.

2.3.2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine-learning algorithm that can be used for both classification and regression tasks. SVM works by minimizing the distance between the hyperplane and two or more data classes. Figure 8 shows an example of multiclass classification with SVM hyperplanes. SVM can handle both linear and non-linearly separable data by transforming the data into a higher-dimensional space through kernel trick (Mashhadi et al., 2021a, 2021b). The kernel function calculates the dot product between the data points in the higher-dimensional space, which allows the algorithm to find the hyperplane that separates the data points even when they are not linearly separable in the original feature space (Cheng et al., 2017; Mohammadi et al., 2023). SVM has several advantages over other classification algorithms, such as its ability to handle high-dimensional data, its robustness to outliers, and its ability to handle non-linear decision boundaries. However, SVM can be sensitive to the choice of kernel function and the cost parameter, and it can be computationally expensive for large datasets.

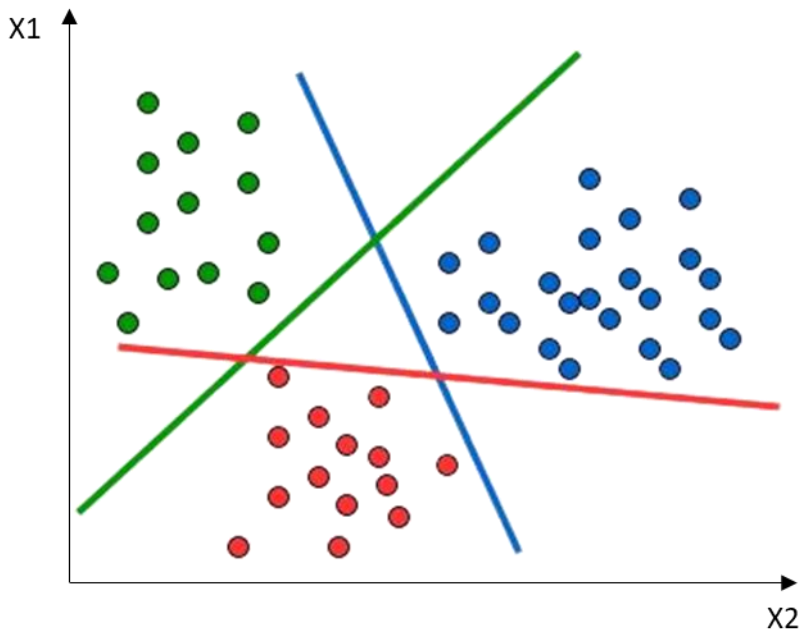

Figure 8. Multiclass Classification Using SVM

2.3.2.2 Logistic Regression

Binary or binomial logistic regression is a supervised algorithm that can be used to predict the probability of a binary target variable. The relationship between the predictor features and the target variable can be modeled using this approach. In this model, the linear function is used as an input to another function, such as g,

$$h_\theta(x) = g(\theta^T x) \text{ Where } 0 \leq h_\theta \leq 1 \quad (1)$$

where g is the logistic or sigmoid function which,

$$g(z) = \frac{1}{1+e^{-z}} \; where \; z = \theta^T x \quad (2)$$

Figure 9 shows the sigmoid function. It can be seen that the value of the y-axis lies between 0 and 1.

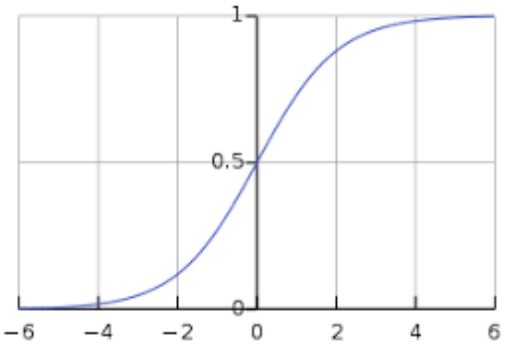

Figure 9. Sigmoid Curve

The labels can be assigned to each data point based on the output probability. For example, considering a threshold of 0.5, we interpret the hypothesis function output as positive if it is ≥ 0.5; otherwise, negative. Figure 10 visualizes a sample implementation. Other than binomial logistic regression, there are two more categories:

1. Multinomial: The target variable has three or more classes
2. Ordinal: The target variable has ordered categories, e.g., application popularity ranking from 1 to 5.

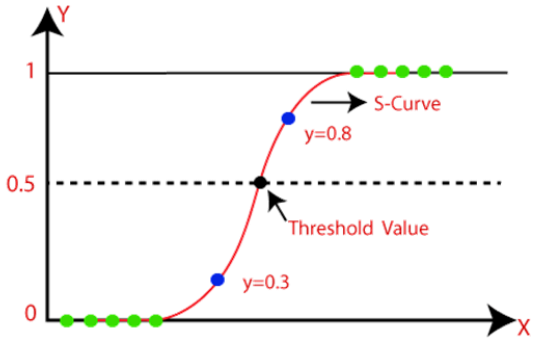

Figure 10. A Sample Implementation of Logistic Regression Classification

2.3.2.3 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) are composed of connected processing units, called nodes, which are functionally similar to biological neurons. Numerical connections between distinct nodes, called weights, enable the model to approximate the desired function by continuously changing weights. Figure 11 depicts the neuron's function.

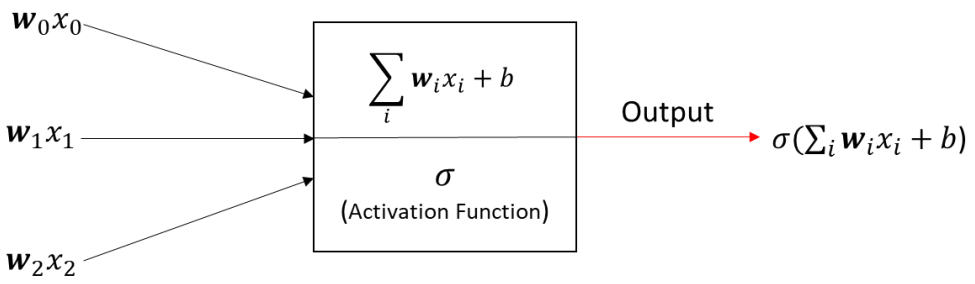

Figure 11. Nodes' Function in Neural Networks

The most common type of ANN is the feedforward neural network, where the data flows from the input layer to the output layer through one or more hidden layers. The activation function of each neuron determines the output of the neuron based on the weighted sum of the inputs. Figure 12 shows an example of neural network architecture with two hidden layers. The number of neurons in hidden layers can be modified based on the level approximation. However, the number of neurons in the input and output layers must be equal to the dimension of the dataset and the number of classes of the target attribute, respectively.

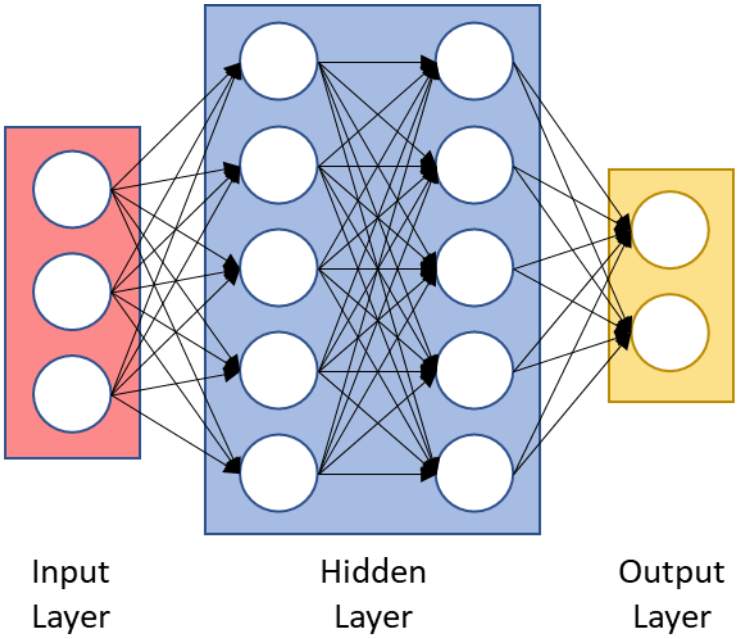

Figure 12. Neural Network Architecture

The hidden layers are discrete feature detectors that can help the model in pattern recognition. For example, if the model is supposed to recognize a car, the first hidden layer might

detect lines; the second one takes the lines as input and compiles them to form a car. The next layer may contribute to detecting head and tail lights until the whole car is finally constructed. This layered structure helps the model eventually recognize complex objects.

ANN has several advantages over traditional machine learning models, such as their ability to handle complex, non-linear relationships in the data and their ability to learn from large amounts of data. However, they require a lot of computational resources and can be prone to overfitting if not properly regularized.

2.3.2.4 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is the most commonly employed deep learning algorithm. CNNs consist of two main parts:

1) Convolution Layers: which aim at extracting features, and

2) Classification Layers: determine the class of each input image.

Feature extraction is the process of converting the input data (images) into a representative set of features (Yuan-Fu, 2019), while the classifier layer uses those features to assign a class to each input. Due to the robustness of CNN, they have been applied to many fields, including computer vision (Schneider et al., 2019; Xia et al., 2018), speech recognition (Pan et al., 2020; D. S. Park et al., 2019), and natural language processing (Alayba et al., 2018; Widiastuti, 2019).

CNNs consist of multiple convolutional, activation, pooling, and fully connected layers. Figure 13 shows an example of CNN architecture for image classification. In the following, the exact definition and function of each layer is explained:

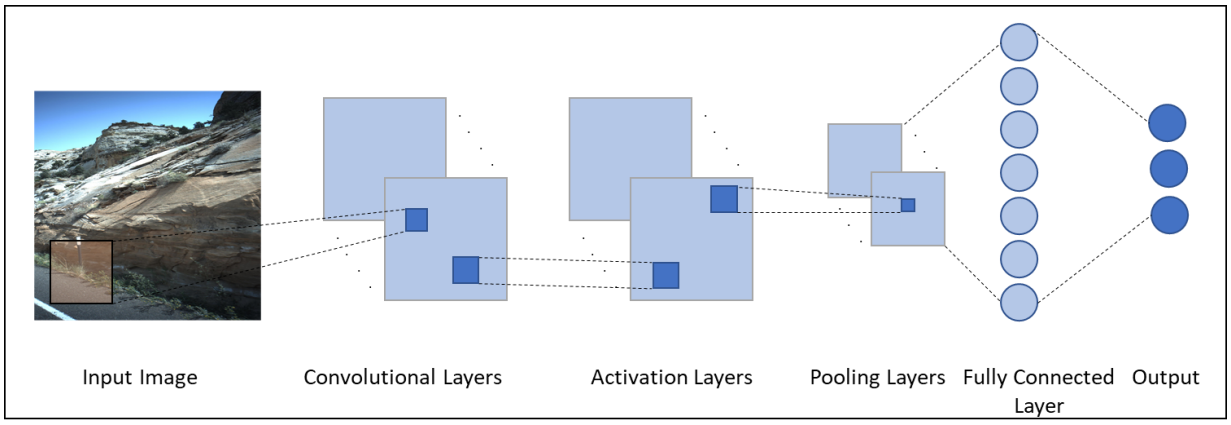

Figure 13. A Sample Architecture of CNN

- **Convolutional Layer:** These layers are the most significant component of CNN models. Each convolution layer includes multiple filters that convolve with the output of the previous layer to generate the output feature map. Figure 14 shows the calculation implemented at each step of the convolutional layers.

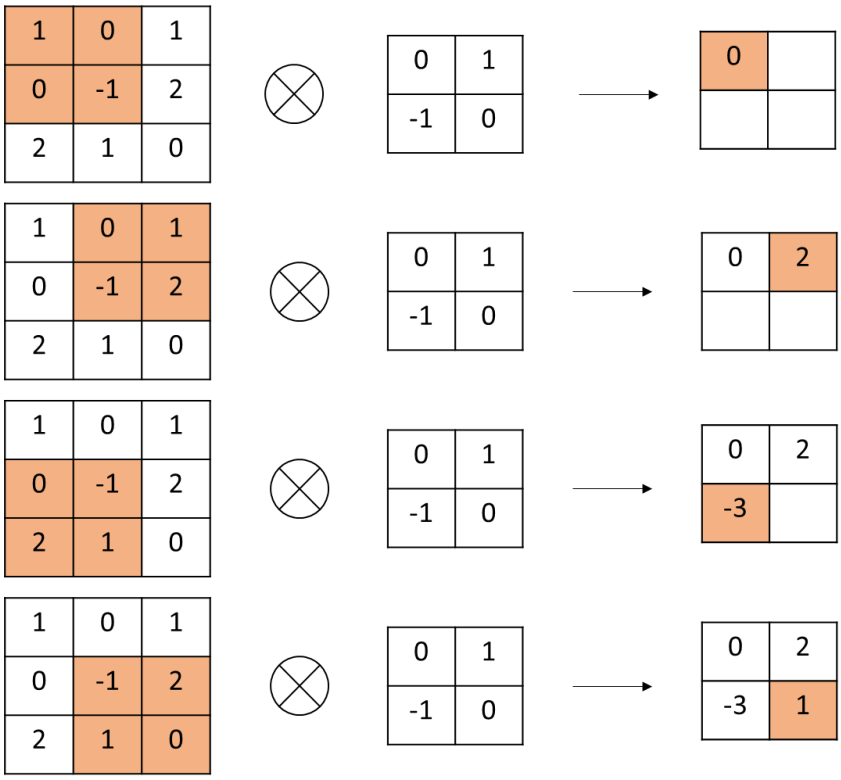

Figure 14. Convolution Calculation at Convolution Layers

- **Activation Layer:** Activation functions are added to neural networks in order to add nonlinearity to the model. This will help the model to learn more complex relationships in the data. The most common activation functions are listed below.

1. Sigmoid

This activation function is computationally expensive, not zero-centered, and also causes vanishing gradient problems. The term "vanishing gradient" refers to the problem of exponentially decreasing (or increasing) backpropagated error as a function of the distance from the output layer. Therefore, it is not usually used in real models. The sigmoid is defined as,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

2. Tanh

Tanh (hyperbolic tangent) is a popular activation function used in neural networks. It maps input values to a range between -1 and 1, effectively centering the data around zero. It is defined as,

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

3. Rectified Linear Unit (ReLU)

This function does not saturate in the positive region. It is computationally efficient and converges faster than sigmoid/tanh in practice. Also, ReLU overcomes the limitations of previous activation functions. However, it is not zero-centered, and also the gradient for negative values is always zero. It is defined as,

$$f(x) = \max(0, x) \tag{5}$$

4. Leaky ReLU

Leaky ReLU (Rectified Linear Unit) is a variant of the popular activation function ReLU used in neural networks. The main difference between Leaky ReLU and ReLU is that Leaky ReLU allows for a small, non-zero gradient when the input is negative. Leaky ReLU is effective in deep neural networks and is commonly used in tasks such as image classification, object detection, and natural language processing. However, the choice of activation function depends on the specific task and the structure of the neural network, and it may require experimentation to determine the best activation function for a given problem. It is defined as,

$$f(x) = \begin{cases} x & x > 0 \\ mx & x < 0 \end{cases} \tag{6}$$

5. Maxout

It generalizes ReLU and Leaky ReLU. Also, it does not die nor saturate. The term "die" means that the neuron output is zero for the input data, and the term "saturate" refers to the state in which a neuron predominantly outputs values close to the asymptotic ends (here one) of the bounded activation function. Nevertheless, it doubles the number of parameters. Therefore, it is expected to take more time for the model to learn.

$$f(x) = \max(w_1^T x + b_1, w_2^T x + b_2) \tag{7}$$

6. Exponential Linear Units (ELU)

ELU is another variant of the ReLU activation function used in neural networks. The ELU activation function is similar to the leaky ReLU in that it allows for non-zero outputs for negative input values to prevent the "dying ReLU" problem. Also, it does not die and is closer to zero mean outputs. However, it is more expensive computationally. It is defined as,

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases} \tag{8}$$

One of the main advantages of the ELU function over other activation functions is that it has been shown to produce better performance on some deep learning tasks, such as image classification and object detection. However, like other activation functions, the choice of ELU depends on the specific task and the architecture of the neural network.

One potential drawback of the ELU function is that it is computationally more expensive than other activation functions, such as ReLU, due to the use of the exponential function. However, this cost can be mitigated through hardware acceleration and optimization techniques.

- **Pooling Layer:** These layers aim to shrink the large feature matrices to create smaller ones. Pooling is achieved by partitioning the input feature map into non-overlapping rectangular or

square regions, called pooling regions, and applying a pooling function to each region to produce a single output value. The most commonly used pooling function is max pooling, which outputs the maximum value in each pooling region. Other pooling functions, such as average pooling or L2-norm pooling, can also be used. Figure 15 depicts the max-pooling operation.

Figure 15. Max Pooling Operation

- **Fully Connected Layer:** The last layer of a CNN model is a fully connected neural network that performs image classification based on the extracted features from prior layers. A fully connected neural network consists of multiple layers with neurons in each layer. The first layer is called the input layer, and the last layer represents the output layer, which is the model prediction. Hidden layers are placed in between. Each neuron is connected to all neurons from the previous and the layers after in this architecture. Figure 16 illustrates an example of a fully connected neural network.

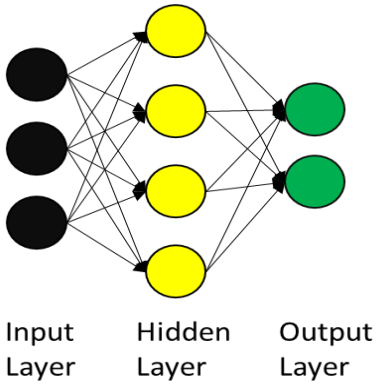

Figure 16. A Fully Connected Neural Network Model

2.4 Algorithm Selection

This section delves into the algorithms chosen for feature extraction and image classification. While deep learning models have proven effective, they are often plagued by issues

such as insufficient data, lengthy training times, and high computational costs. One approach to overcoming these challenges is transfer learning, which involves repurposing a pre-trained model as the foundation for building a new model on a new dataset. Rather than creating a new architecture and training the model from scratch, the pre-trained model's weights are utilized for feature extraction, and only a few final layers are modified. Despite the fact that the dataset used in this project differs from the one used to train the pre-trained models, it has been demonstrated that transferring features from remote datasets outperforms non-CNN models (Rezapour & Ksaibati, 2021). The following sections provide a detailed overview of the algorithm selected in this research for feature extraction and image classification.

2.4.1 Feature Extraction

Various pre-trained models, such as AlexNet, VGG16, VGG19, GoogleNet, and ResNet50, are trained on the ImageNet dataset, which includes over 14 million images from 1000 classes (Deng et al., 2009). This project uses VGG16, ResNet, and Inception v3 for the feature extraction step. The following paragraphs discusses each method's architecture, advantages, and disadvantages.

2.4.1.1 VGG16

Feature extraction is a critical step in image processing and computer vision tasks. It transforms raw input images into feature representations that capture relevant information and characteristics useful for subsequent tasks such as image classification, object detection, and segmentation. One popular feature extraction approach is using models that have already been trained on large datasets, such as ImageNet, and fine-tuned for a specific application. These pre-trained models can extract relevant features from input images, which can be used as inputs to other machine learning models.

The ImageNet dataset, which includes over 14 million images from 1000 classes, is commonly used to train pre-trained models. This project uses the VGG16, ResNet, and Inception v3 pre-trained models for feature extraction. Each of these models has a different architecture providing different advantages and disadvantages. For instance, VGG16 has a simple architecture and is easy to understand, but it is computationally expensive due to its large number of parameters (Figure 17). ResNet, on the other hand, is known for its skip connections, which allow it to be

deeper while avoiding the vanishing gradient problem. Inception v3 utilizes an Inception module that allows for efficient feature extraction at multiple scales, making it useful for object detection and segmentation tasks. By utilizing these pre-trained models, the feature extraction process can be accelerated, and the extracted features can be used as inputs to other machine learning models, thereby improving their performance.

Figure 17. VGG16 Architecture (Top 4 Pre-Trained Models, n.d.)

2.4.1.2 ResNet50

ResNet50 is a pre-trained deep learning model that is part of the ResNet family of convolutional neural networks (CNNs). ResNet50 has 50 layers, including shortcut connections, which allow for efficient training of deeper networks by alleviating the vanishing gradient problem. The pre-trained weights of ResNet50 can be fine-tuned on specific datasets, making it a popular choice for transfer learning in computer vision applications (Figure 18). ResNet50 could improve the efficiency of deep neural networks by employing more layers while minimizing errors. However, deeper network architectures require weeks of training due to the increased number of parameters that need to be trained.

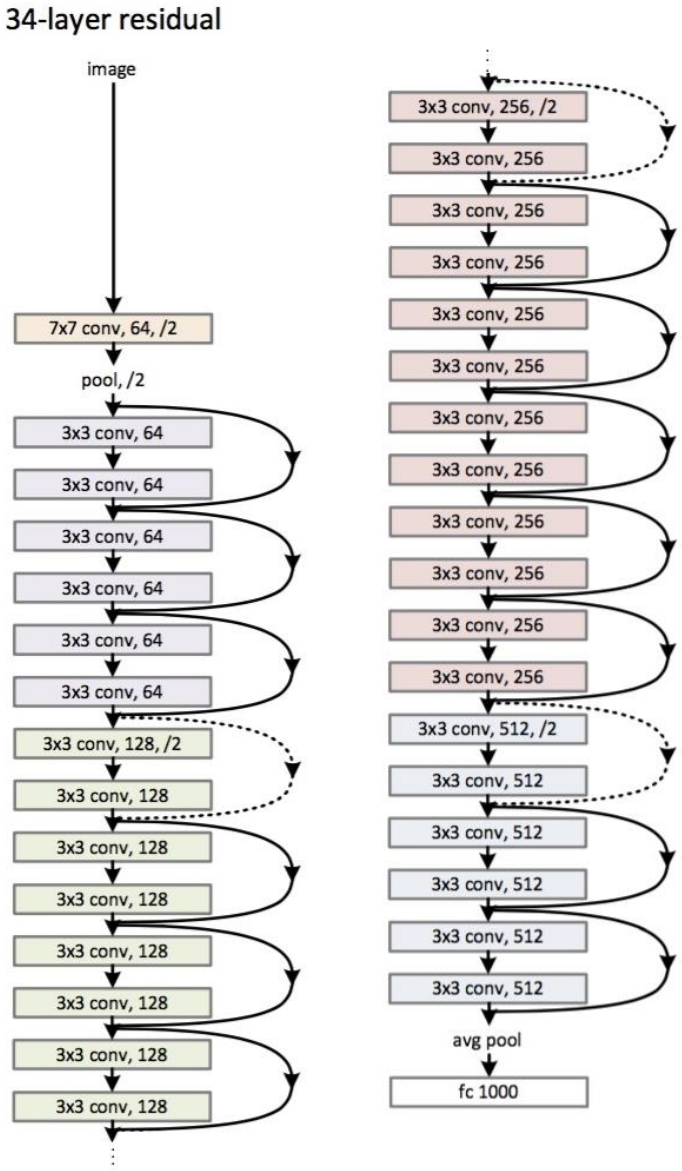

Figure 18. ResNet50 Architecture (MobileNet vs ResNet50, n.d.)

2.4.1.3 Inception v3

Inception v3 is a pre-trained deep learning model that was introduced by Google in 2015 (Figure 19). It is a convolutional neural network (CNN) that uses a combination of 1x1, 3x3, and 5x5 convolutional filters to extract features from images. Inception v3 was designed to be

computationally efficient while achieving state-of-the-art performance on the ImageNet dataset, which includes over 14 million images from 1000 classes.

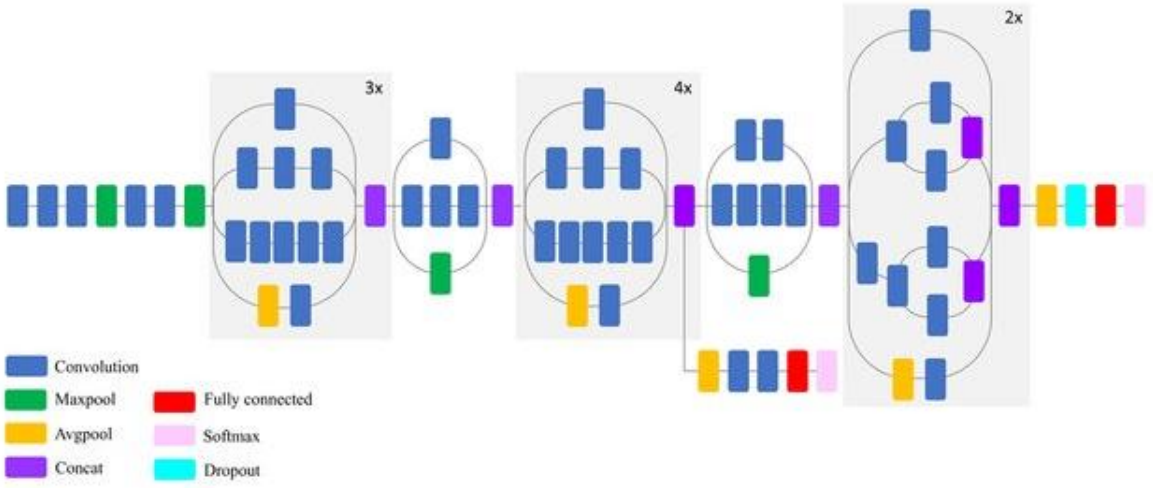

Figure 19. Inception v3 Architecture (MobileNet vs ResNet50, n.d.)

2.4.2 Image Classification

After feature extraction, the next step in image classification is to use an algorithm to classify the features extracted from the images. A CNN model consists of convolutional layers and a fully connected neural network. In order to improve the precision of the model, this project suggests combining CNN and eXtreme Gradient Boosting (XGBoost), which is a robust classification algorithm. The architecture of the suggested method is shown in Figure 20. The following paragraphs explain more about the XGBoost algorithm.

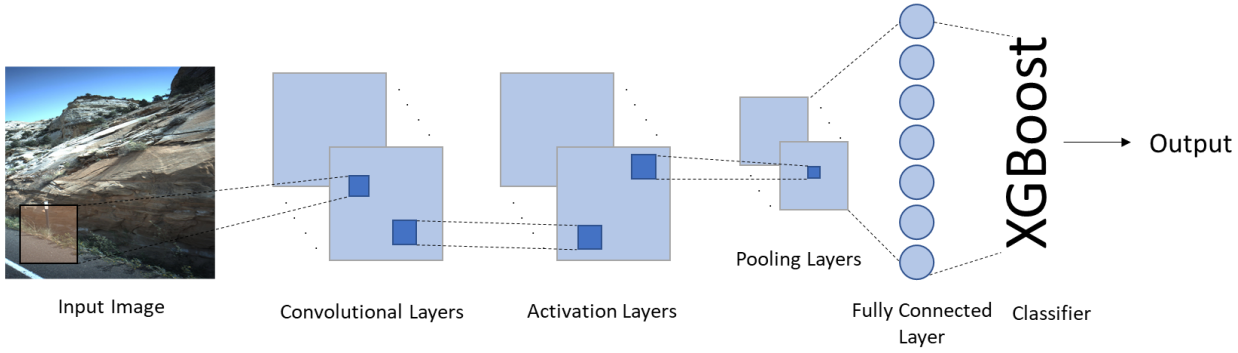

Figure 20. Combining CNN and XGBoost Algorithms

2.4.2.1 eXtreme Gradient Boosting (XGBoost)

XGBoost is a supervised learning algorithm that is based on the gradient-boosting decision tree (GBDT). A GBDT is an ensemble learning algorithm similar to a random forest that can be used for classification and regression. Both GBDT and random forest consist of multiple decision trees. However, the method of building trees is different in these two approaches. While random forest uses the bagging technique to build a full tree, GBDT iteratively trains an ensemble of shallow decision trees. The boosting ensemble technique has three steps:

- An initial model, F_0, is defined to predict the target variable y.
- This model will be associated with a residual ($y - F_0$). A new model, h_1, is fit to the residuals
- Now, F_0 and h_1 are combined to give F_1, the boosted version of F_0. The mean squared error from F_1 will be lower than that from F_0:

$$F_1(x) \leftarrow F_0(x) + h_1(x) \tag{9}$$

In order to improve the performance of the F_1, again, a new model is used to model the residuals of F_1.

$$F_2(x) \leftarrow F_1(x) + h_2(x) \tag{10}$$

This process can be done for "m' iterations until residuals have been minimized as much as possible:

$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x) \tag{11}$$

2.5 Summary

In order to classify images based on roadside safety factors, various computer vision algorithms have been developed using logistic regression, pre-trained CNN models, and XGBoost. Figure 21 depicts the process of image classification using the suggested approach. The next section discusses the process of data collection.

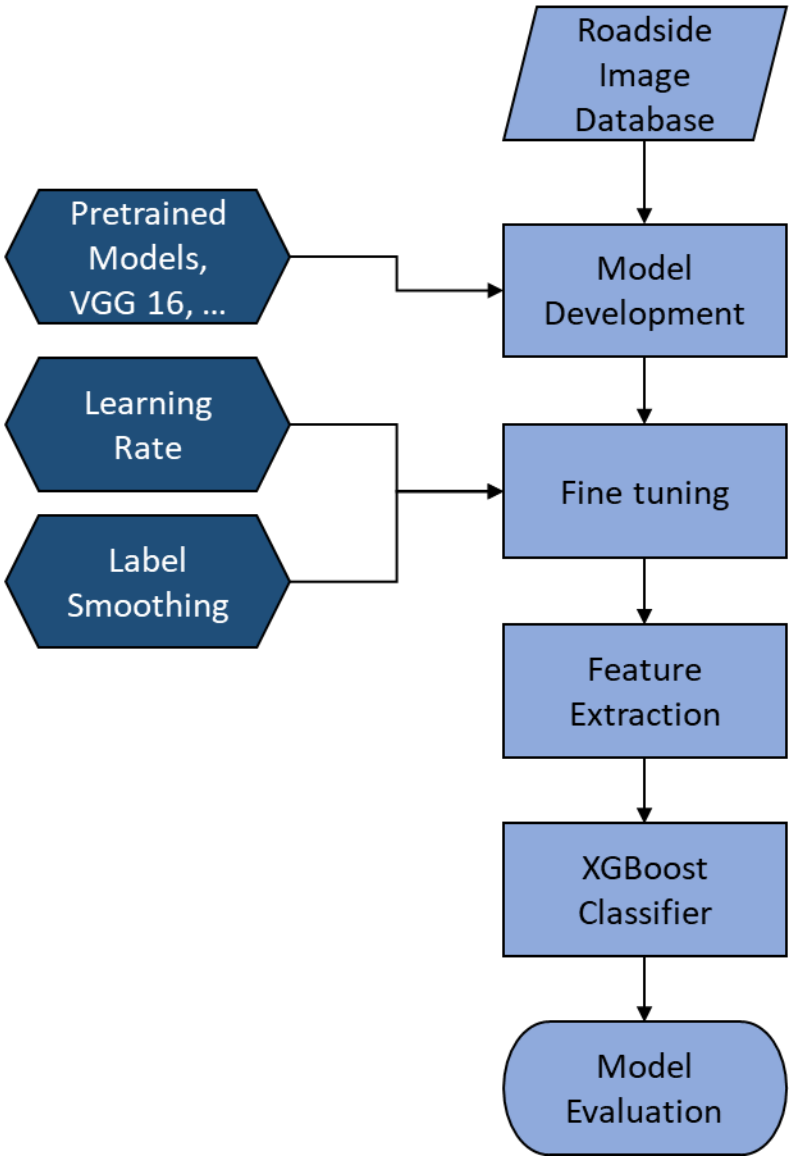

Figure 21. Flowchart of the Selected Approach

3.0 DATA COLLECTION

3.1 Overview

UDOT (Utah Department of Transportation) has a comprehensive data collection plan that regularly acquires road and roadside imagery and LiDAR data. This data is used for various purposes, including road safety analysis, traffic management, and infrastructure planning. The data is collected by Mandli Communications, Inc. at least every other year, which ensures that UDOT always has a new data set to work with. UDOT's approach involves collecting high-resolution images of the entire state road network, including urban and rural areas. The images are captured using specialized cameras mounted on vehicles driven along designated routes. The cameras capture both forward and backward-facing images, providing a comprehensive view of the road and surrounding areas. LiDAR data is also collected simultaneously, using laser scanning technology to measure the distance between the vehicle and objects on the ground.

The images collected by Mandli Communications, Inc. are a valuable resource for identifying unsafe road segments. Mandli Communications, Inc. is dedicated to collecting data from U.S. highways using 3D pavement technology, mobile LiDAR, and geospatial technologies. The collected data can be used to evaluate safety parameters along each road segment in Utah. One of the vehicles used by Mandli for capturing images and videos of roadways can be seen in Figure 22. As shown, there are six types of equipment attached to the vehicles, including:

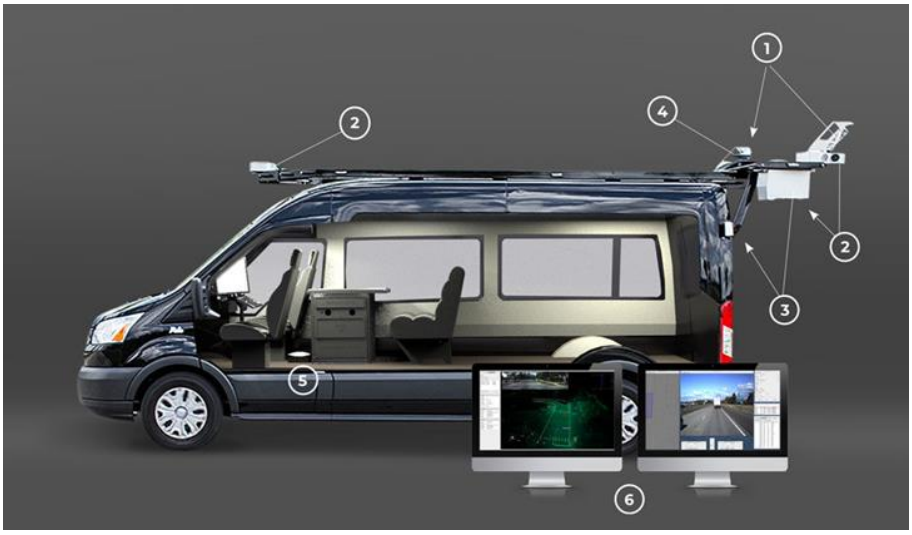

Figure 22. Mandli Vehicle Used for Data Collection

1. Dual Velodyne HDL-32 LiDAR sensors.
2. Nine 8.9 MP Cameras deliver nearly 80 megapixels of a 360° image
3. Dual LCMS Pavement Scanners
4. Position Orientation System
5. Advanced independent Power System
6. Processing/Post-Processing Software (*Mandli X-35 - Mandli Communications*, n.d.)

Using the cameras deployed in front of the vehicle, different angles of the roads can be seen (Figure 23-25).

Figure 23. A Sample View of Mandli Images from SR-10 at Mileage 50 (Year 2019)

Figure 24. A Sample View of Mandli Images from SR-10 at Mileage 50 (Year 2020)

Figure 25. A Sample View of Mandli Images from SR-10 at Mileage 50 (Year 2021)

3.2 Specifications

In order to cover different conditions, the dataset was collected from different roads and mileages. Moreover, in the proposed approach, we utilize specifically selected images captured in the year 2020 from the right-hand side to evaluate the roadside conditions of each road segment (the right portion in Figure 24). By focusing on the right-hand side images, we are able to obtain a more comprehensive assessment of the roadside situation, allowing for a more accurate and thorough evaluation process. Figure 26 shows different situations at different road segments.

Figure 26. Various Roadside Situations on Different Roads

3.3 Summary

In order to evaluate roadside safety within state roadways, thousands of images were collected from different roadways with various conditions. The next section evaluates each method’s performance using the collected data from Utah's roadways.

4.0 SYSTEM EVALUATION

4.1 Overview

FHWA proposed a seven-point categorical scale from 1 (best) to 7 (worst) for roadside safety, which is listed in Table 3. Considering the factors listed in Table 3, four major road parameters contribute to roadside safety, including:

1. **Clear zone**
2. **Guardrails**
3. **Side slope**
4. **Rigid obstacles (trees, embankments, rocks, …)**

In order to provide UDOT with detailed information about safety criteria at each road segment, different models have been developed for each parameter. Below, the results of each model will be presented.

4.2 Model Performance

4.2.1 Guardrail Detection

In order to detect guardrails on the roadside, a binary logistic regression model has been developed. Two sample images, taken from Utah roadways, are shown in Figure 27.

Figure 27. Sample Images With and Without Guardrails

The developed model had an accuracy of 93% in detecting guardrails on the roadside. The loss function is shown in the following diagram:

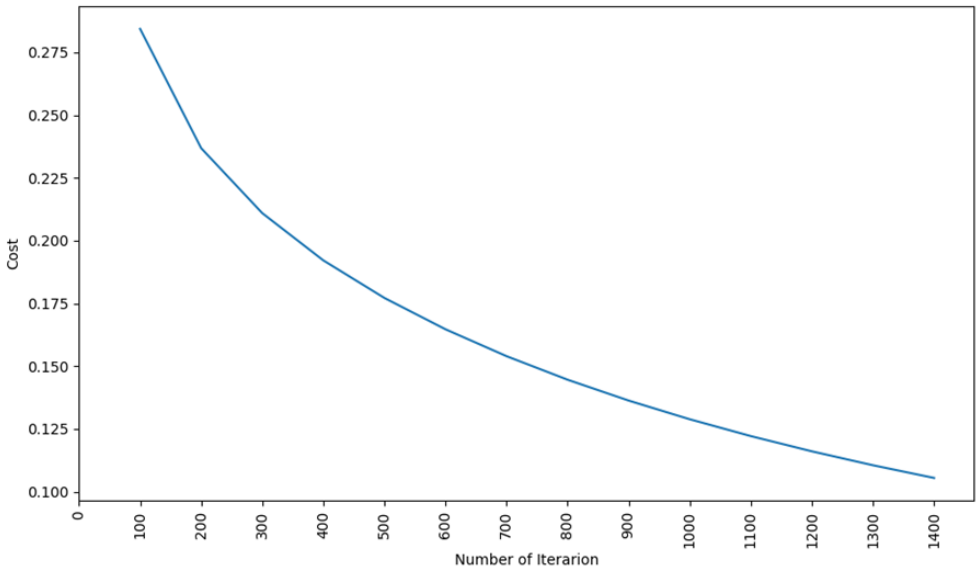

Figure 28. Loss Function of Detecting Guardrails on the Roadside

4.2.2 Rigid Obstacle Detection (Model 1)

In order to detect the presence of rigid obstacles (trees, mountains, …) on the roadside, another binary logistic regression model is developed. According to the discussion with FHWA representatives, poles and breakaway posts are not considered rigid obstacles. The following figures show examples of rigid obstacles on the roadside:

Figure 29. Images With Rigid Obstacles on the Roadside

Figure 30. Images Without Rigid Obstacles on the Roadside

The developed model had an accuracy of 100% in detecting rigid obstacles on the roadside. The process of detecting rigid obstacles is shown in the following.

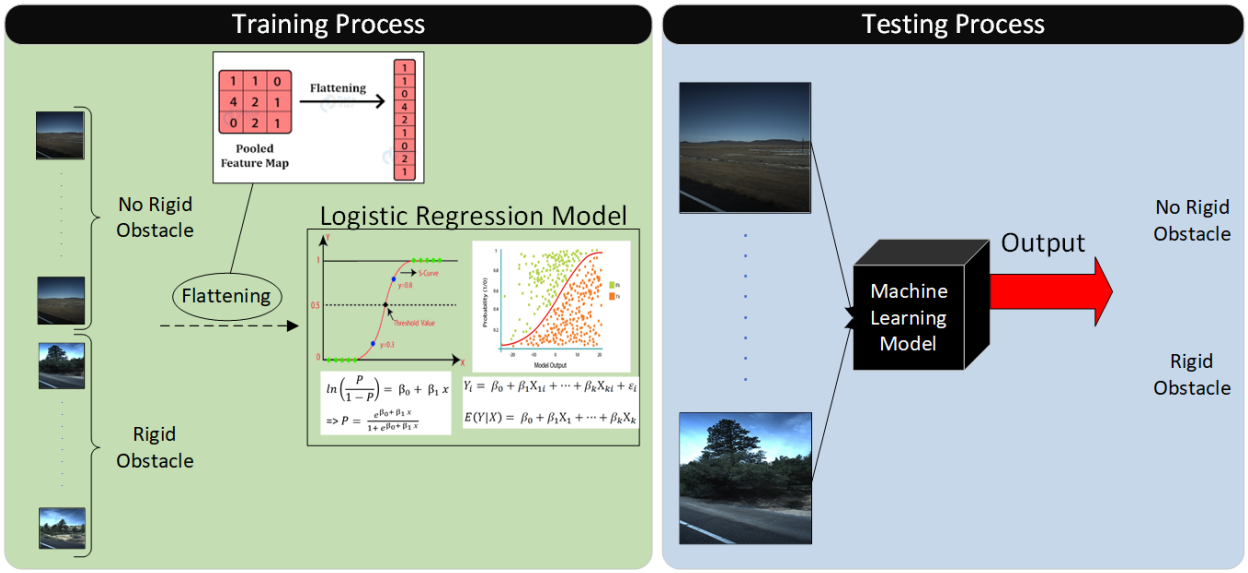

Figure 31. Process of Detecting Rigid Obstacles on the Roadside

4.2.3 Rigid Obstacle Distance Detection (Model 2)

After detecting obstacles on the roadside, it is necessary to detect the distance between the pavement edge line (pavement marking) and the obstacle. For this purpose, a multinomial logistic regression is developed with three classes:

1. Distance of 0-6.5 ft
2. Distance of 6.5-10 ft
3. Distance of more than 10 ft

Three sample images with various distances of obstacles are shown in Figure 32.

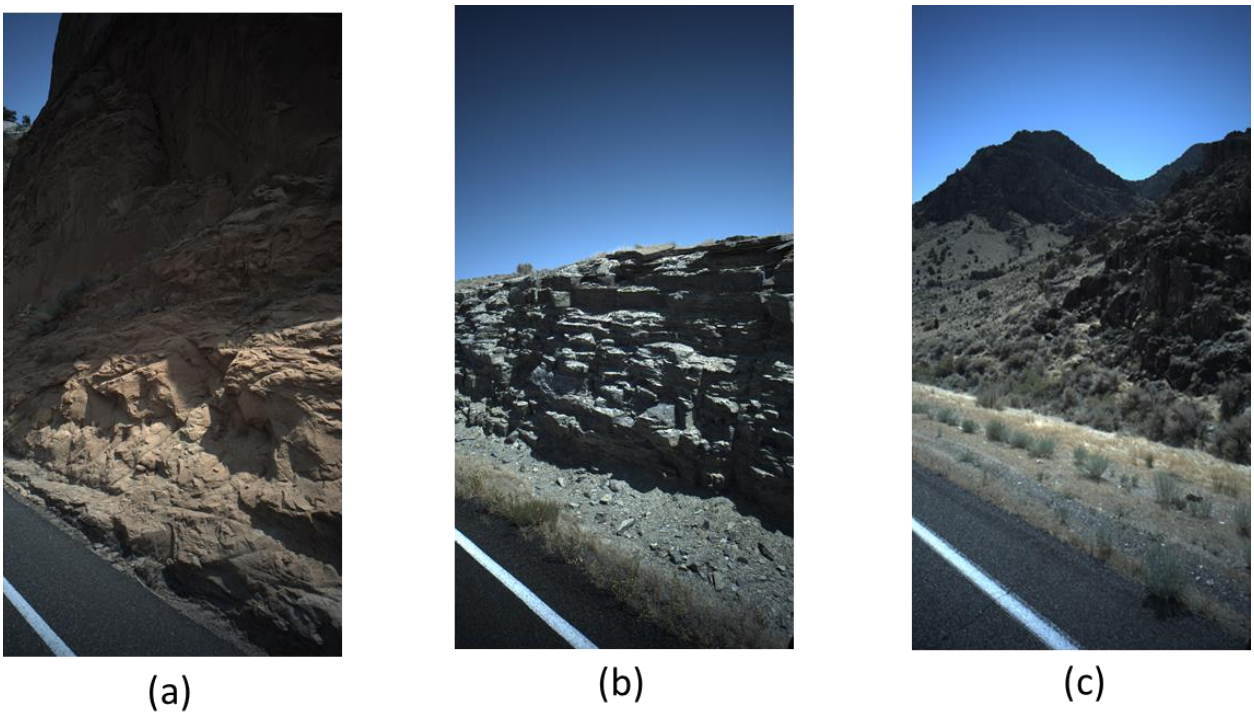

Figure 32. a) 0-6.5 ft, b) 6.5-10 ft, c) More than 10 ft

The developed model yielded an accuracy of 73% in detecting the distance of obstacles with the main road. In order to improve the accuracy of the model, more advanced algorithms and more labeled data are planned to be used.

4.2.4 Rigid Obstacle Detection (Model 3)

Due to the importance of rigid obstacles, the research team tried to improve the results by integrating the two previously developed models.

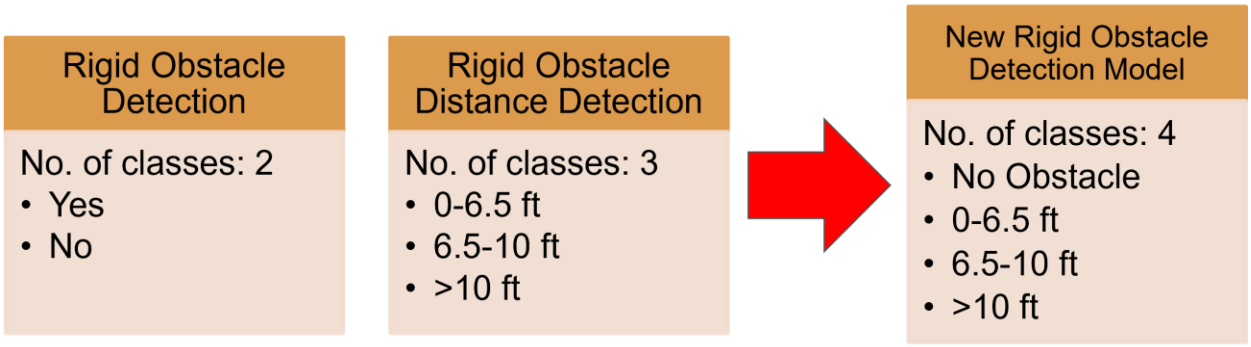

Figure 33. Rigid Obstacle Model Integration

The newly developed model yielded better results than previously developed models with 94% accuracy. Figure 34 shows the confusion matrix for rigid obstacle detection (model 3). A confusion matrix is also used for evaluating the performance of a multiclass classification model. In a multiclass confusion matrix, the rows and columns represent the predicted and true labels, respectively. Each cell in the matrix indicates the percent of instances predicted to belong to a certain class but belong to another class. The diagonal cells represent the percentage of correctly classified instances for each class.

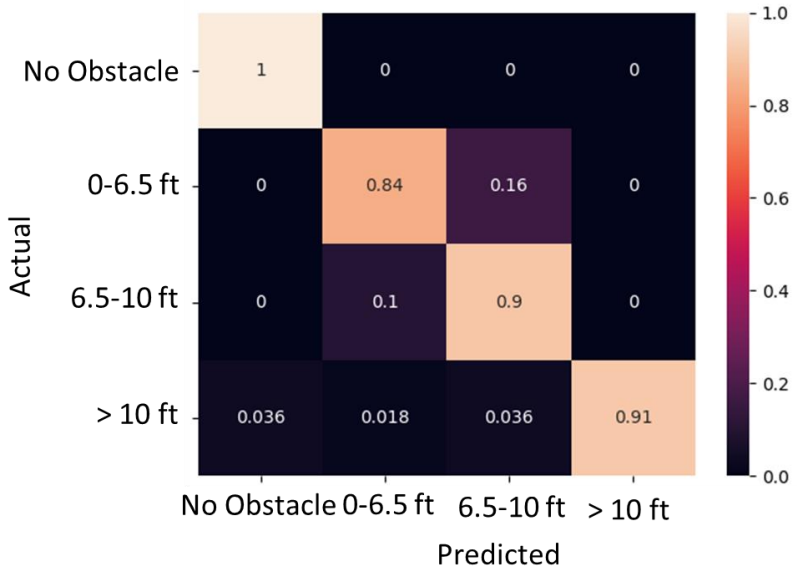

Figure 34. Confusion Matrix for Rigid Obstacle Detection

4.2.5 Clear Zone Detection

Figure 35 illustrates the definition of the clear zone, according to the Roadside Design Guide Manual (Roadside Design Guide):

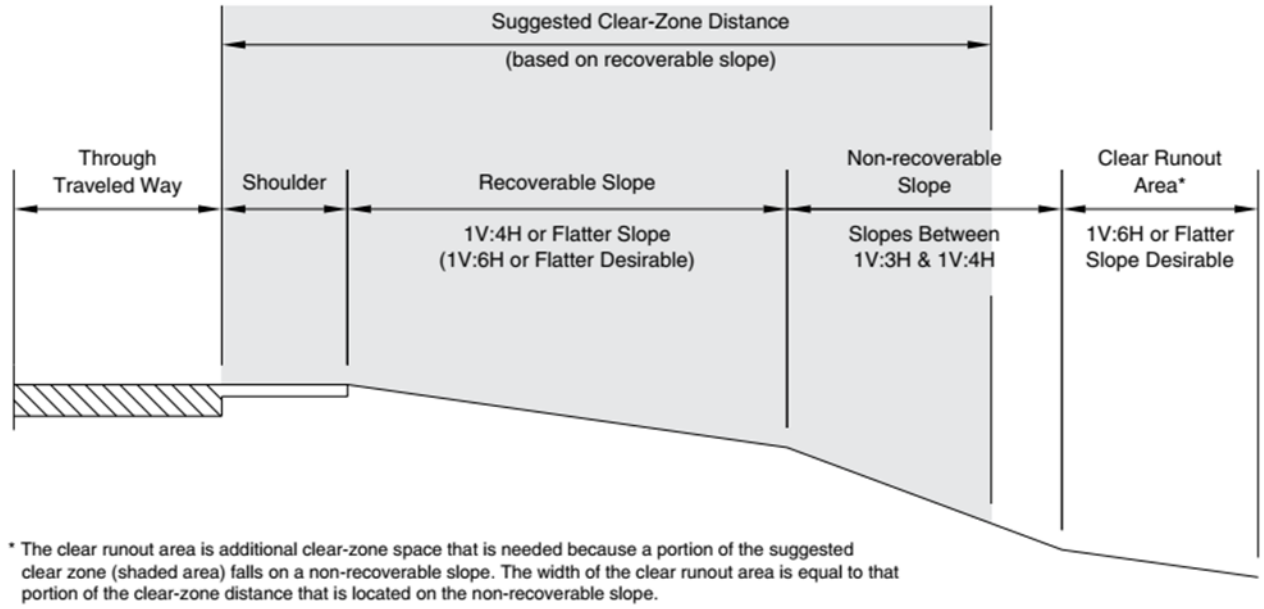

Figure 35. Clear Zone Definition Based on Roadside Design Guide (AASHTO Roadside Design Guide, n.d.)

Due to a couple of gaps in clear zone intervals, the FHWA intervals have been changed based on the approvals of UDOT engineers. Table 4 shows the FHWA and the suggested clear zone intervals.

Table 4. FHWA Intervals for Clear Zone and the Suggested Intervals

Class	FHWA Intervals	Suggested Intervals
1	Greater than 30 ft	Greater than 30 ft
2	20-25 ft	20-30 ft
3	About 10 ft	10-20 ft
4	5-10 ft	5-10 ft
5	Less than 5 ft	Less than 5 ft

Due to the importance of clear zones, multiple algorithms have been developed for detecting the available clear zone.

4.2.5.1 Logistic Regression

The model yielded 70% accuracy in detecting the available clear zone using multinomial logistic regression. Since 70% accuracy is unacceptable, more models have been developed using pre-trained deep learning algorithms and XGBoost as the classifier.

4.2.5.2 ResNet50 + XGBoost

Using ResNet50 and XGBoost, the model could outperform the logistic regression by 9 percentage points. In order to evaluate the performance of the model in each class and prevent overfitting, a confusion matrix is used to show and compare results. A confusion matrix or error matrix is a specific table layout that visualizes the performance of an algorithm by displaying the number of times the model correctly or incorrectly predicted each class. The confusion matrix of ResNet50 is shown in Figure 36.

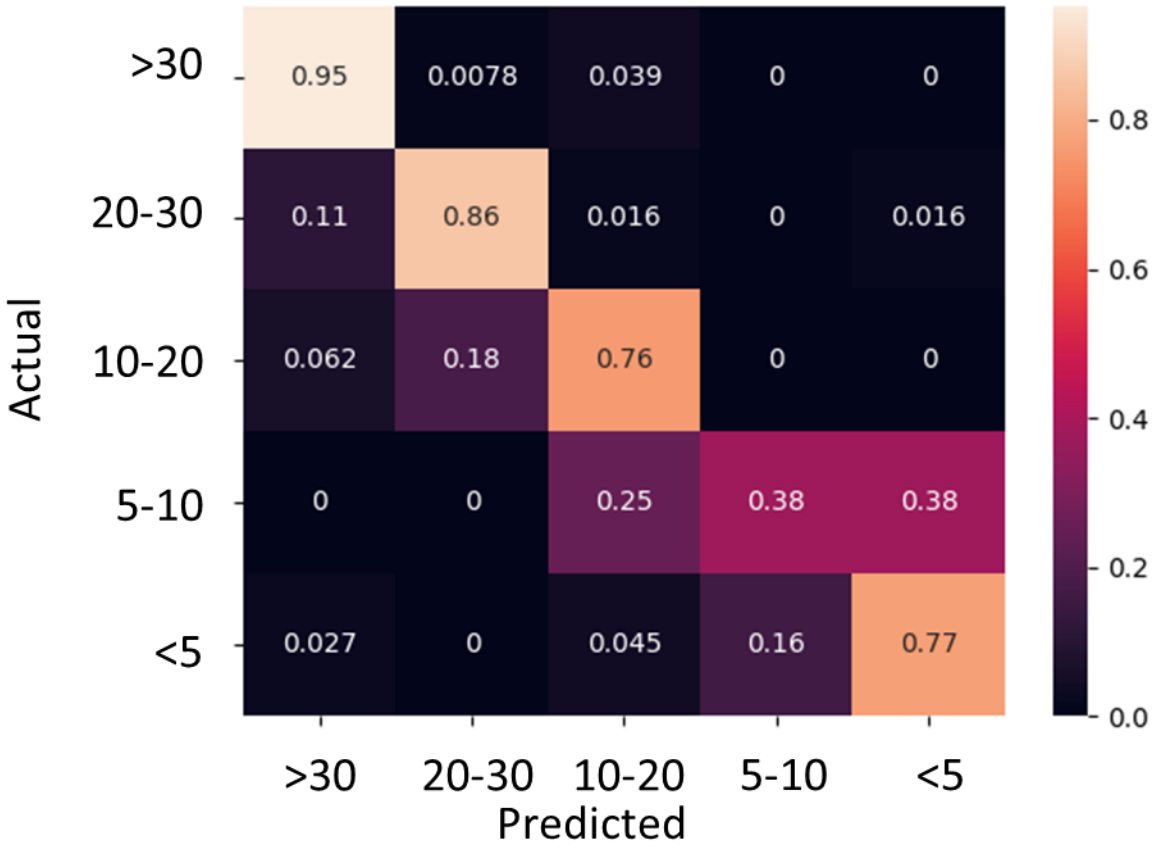

Figure 36. Confusion Matrix of ResNet50

As shown in Figure 36, the developed model was able to detect clear zone with high accuracy in most of the classes.

4.2.5.3 Inception V3 + XGBoost

Using Inception v3 and XGBoost, the model could outperform the logistic regression by 13 percentage points. The confusion matrix of Inception v3 is shown in Figure 37.

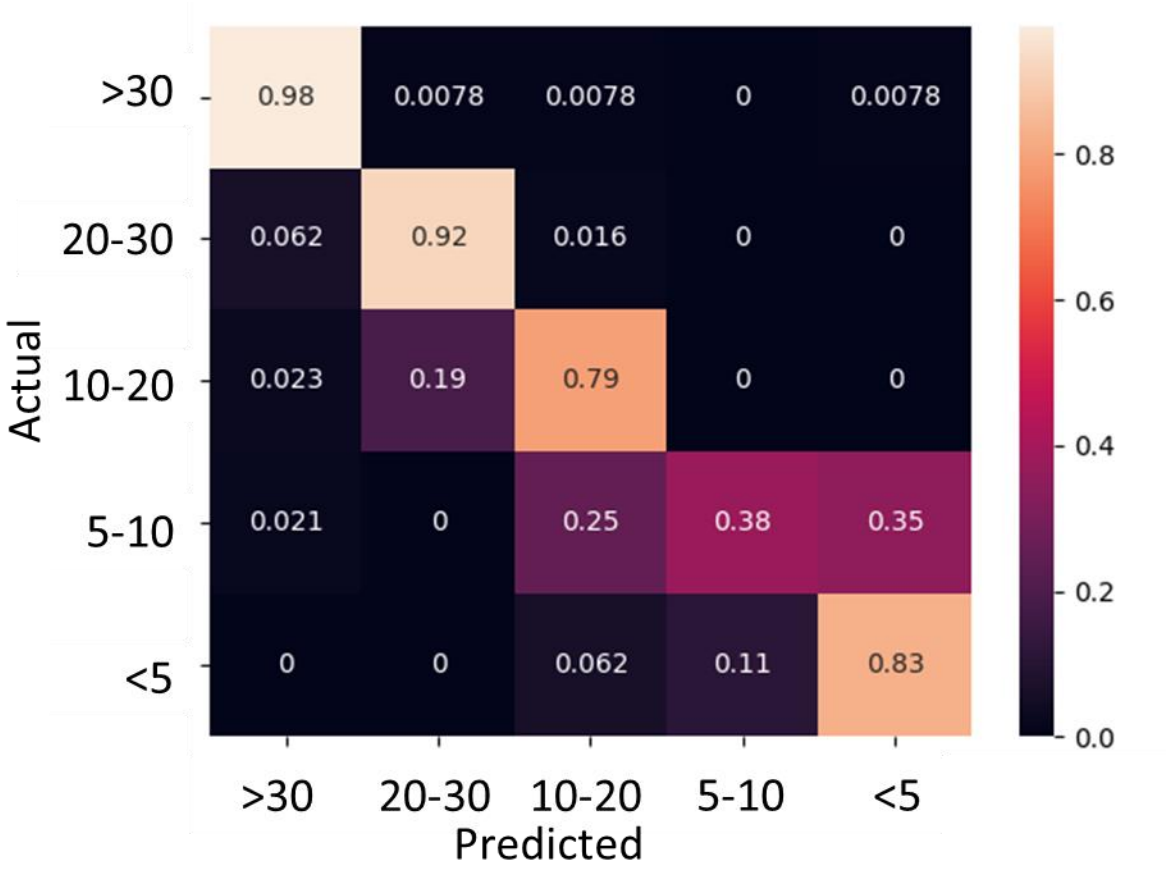

Figure 37. Confusion Matrix of Inception v3

As shown in Figure 37, the model performance in class "Greater than 30" is 98%, meaning the model could correctly detect 98% of images with a clear zone greater than 30.

4.2.5.4 VGG16 + XGBoost

Using VGG16 and XGBoost, the model could outperform the logistic regression by 13 percentage points. The confusion matrix of VGG16 is shown in Figure 38.

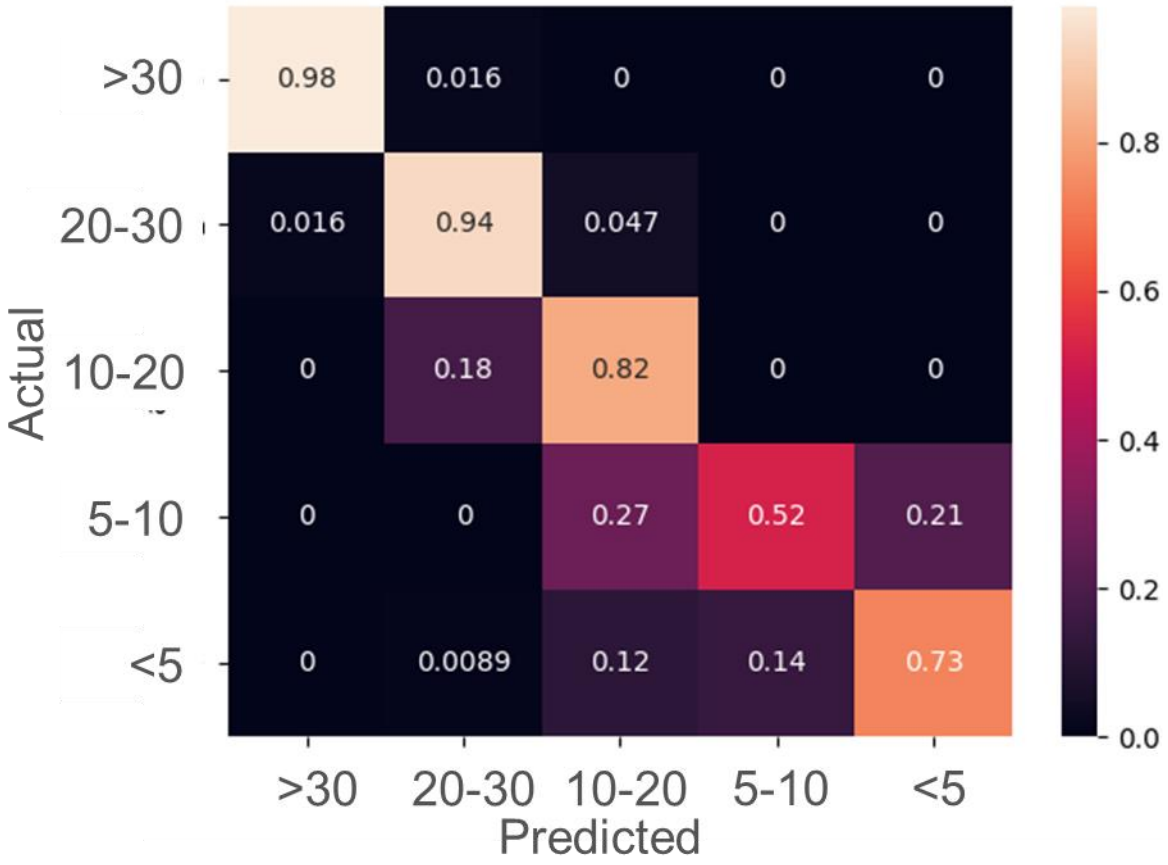

Figure 38. Confusion Matrix of VGG16

VGG16 has the best accuracy in detecting images with 5-10 ft clear zones compared to previous models. The accuracies of developed models are compared in Figure 39.

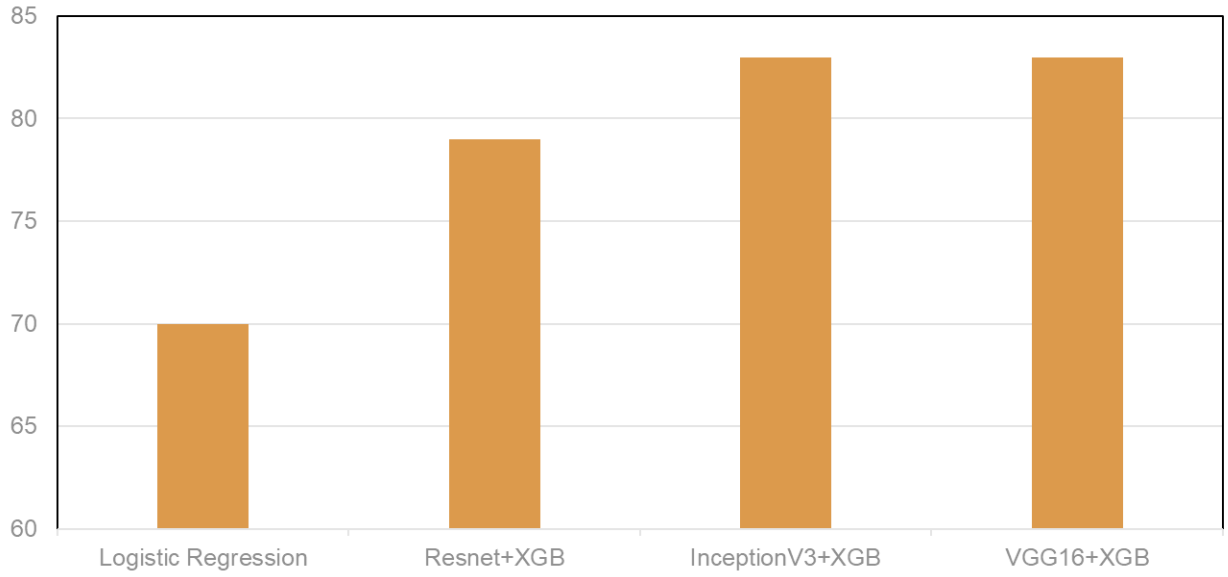

Figure 39. Accuracy Comparison of Different Developed Models for Clear Zone Detection

4.2.6 Side Slope Detection

According to the Roadside Design Guide, the roadside slope is a crucial consideration beyond the shoulder area (as illustrated in Figure 40). Additionally, the MIRE (Model Inventory of Roadway Elements) guidelines indicate that side slopes do not apply to roads when roadside barriers are present.

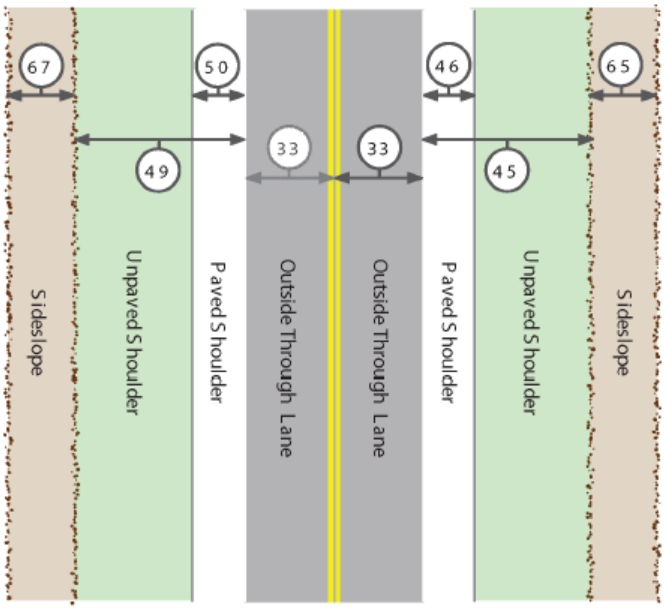

Figure 40. Illustration of Cross Section, Two-Lane Roadway

The FHWA rating system includes six distinct categories related to side slope, which are:

1. Side slope flatter than 1:4
2. Side slope about 1:4
3. Side slope about 1:3 or 1:4
4. Side slope about 1:3
5. Side slope about 1:2
6. Side slope 1:2 or steeper

However, detecting slopes solely through 2D images can be challenging. To simplify the classification process, we have grouped them into Low, Med, and High categories (Figure 41). Since the side slope range is nearly the same across different ratings, we can differentiate the rating by examining the available clear zone ranges.

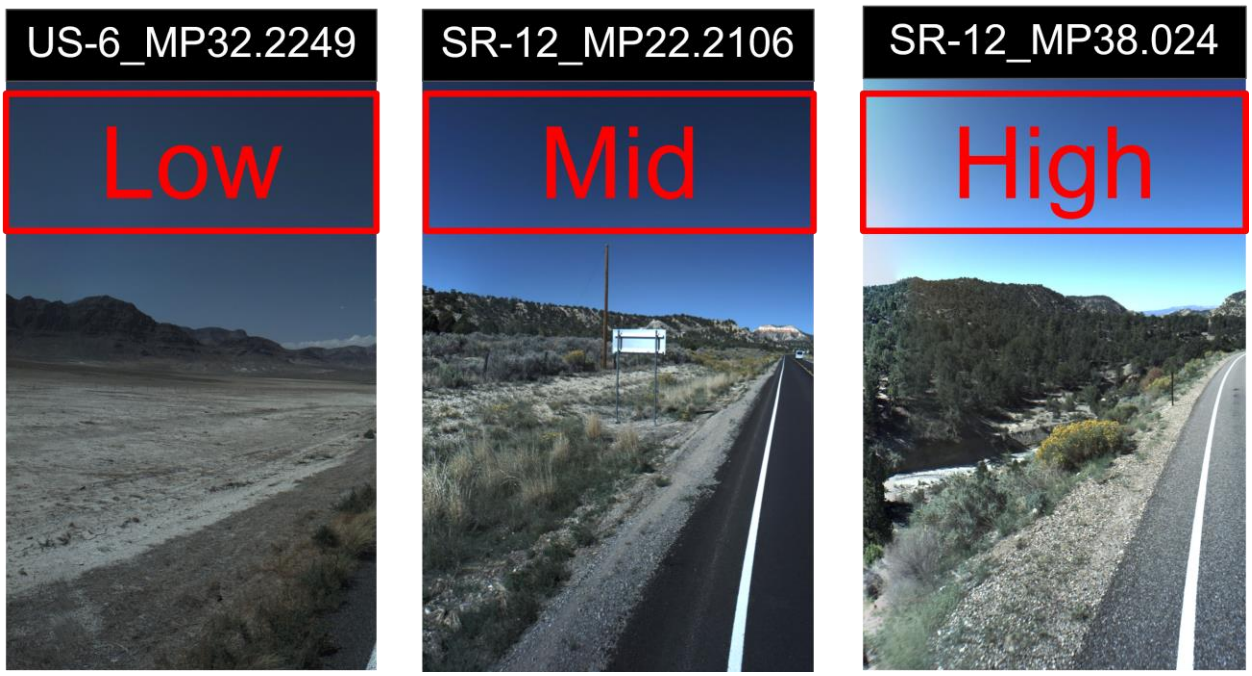

Figure 41. Sample Images for Side Slope Categorization

Based on the results, the developed model yielded significant results in categorizing images into three groups (Figure 42).

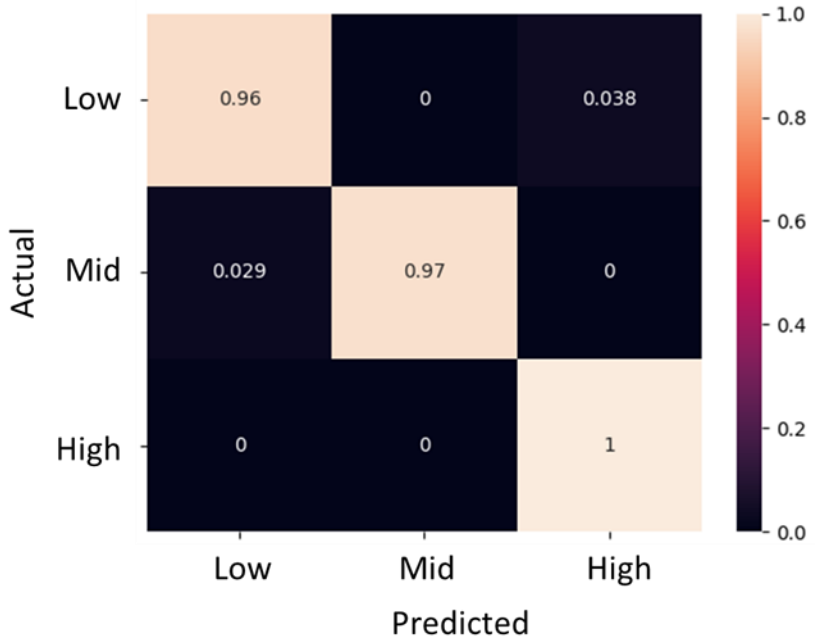

Figure 42. Confusion Matrix for Roadside Slope Detection

4.3 Safety Ranking

After extracting features from the roadside, the research team collaborated with UDOT specialists to develop an algorithm that rates roadside conditions. This algorithm is based on the four major features extracted using computer vision algorithms. These four features are guardrails, clear zone, rigid obstacles, and roadside slopes. Using these features, the following algorithm calculates a rating for each road segment. This rating system is useful for UDOT to monitor and assess the roadside's condition and prioritize maintenance and repair work. Using objective, data-driven measures to assess roadside conditions, UDOT can ensure that resources are allocated effectively and efficiently.

Algorithm 1 Roadside Safety Ranking

If Guardrail = yes **THEN**
    If 6.5 ft ≤ Rigid Obstacle ≤ 10 ft **THEN** Rating = 5
    ELSE Rating = 4
ELSEIF Clear Zone ≤ 5 ft **AND** Sideslope = High **THEN** Rating = 7
ELSEIF Clear Zone ≤ 5 ft **AND** Rigid Obstacle ≤ 6.5 ft **THEN** Rating = 7
ELSEIF Sideslope = High **AND** Rigid Obstacle ≤ 6.5 ft **THEN** Rating = 7
ELSEIF Clear Zone ≤ 5 ft **OR** Sideslope = High **OR** Rigid Obstacle ≤ 6.5 ft **THEN** Rating = 6
ELSEIF 6.5 ft ≤ Rigid Obstacle ≤ 10 ft **THEN** Rating = 5
ELSEIF Rigid Obstacle ≥ 10 ft **OR** 5 ft ≤ Clear Zone ≤ 10 ft **THEN** Rating = 4
ELSEIF 10 ft ≤ Clear zone ≤ 20 ft **OR** Sideslope = Mid **THEN** Rating = 3
ELSEIF 20 ft ≤ Clear zone ≤ 30 ft **THEN** Rating = 2
ELSEIF Clear zone ≥ 30 ft **THEN** Rating = 1

4.4 Final Product

The final product of this project is a shape file including image ID, road name, latitude, longitude, safety ranking, and roadside features at each given road segment for five state roads, US-6, SR-10, SR-12, US-40, and SR-150. Some sample images are shown in the following.

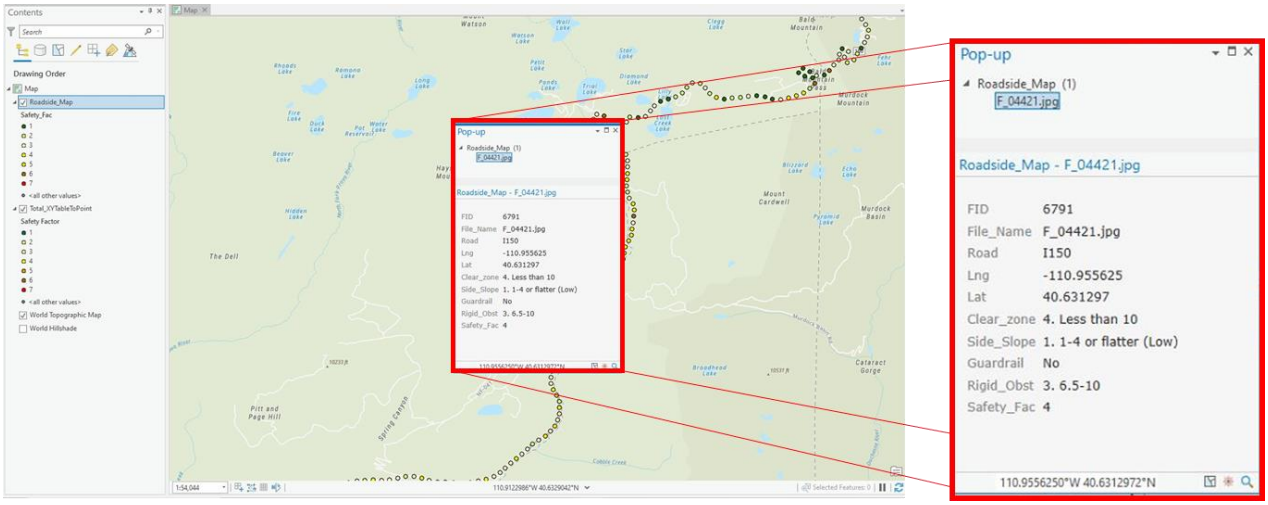

Figure 43. Example of Final Product

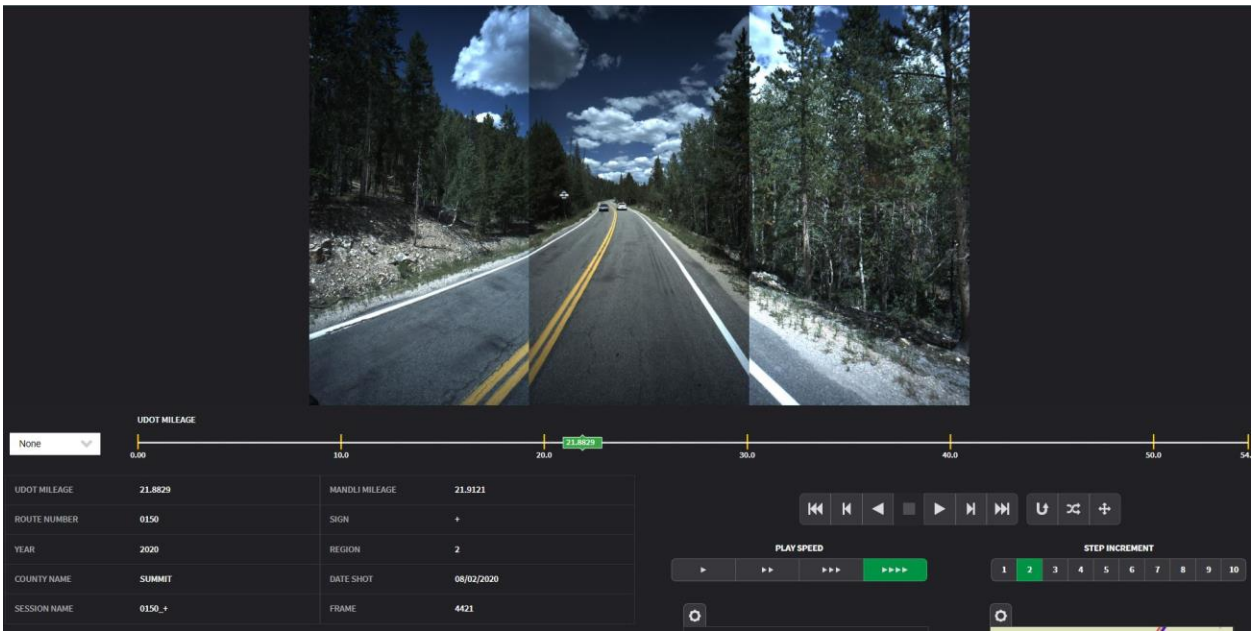

Figure 44. Actual Condition of the Point Listed in the Previous Figure

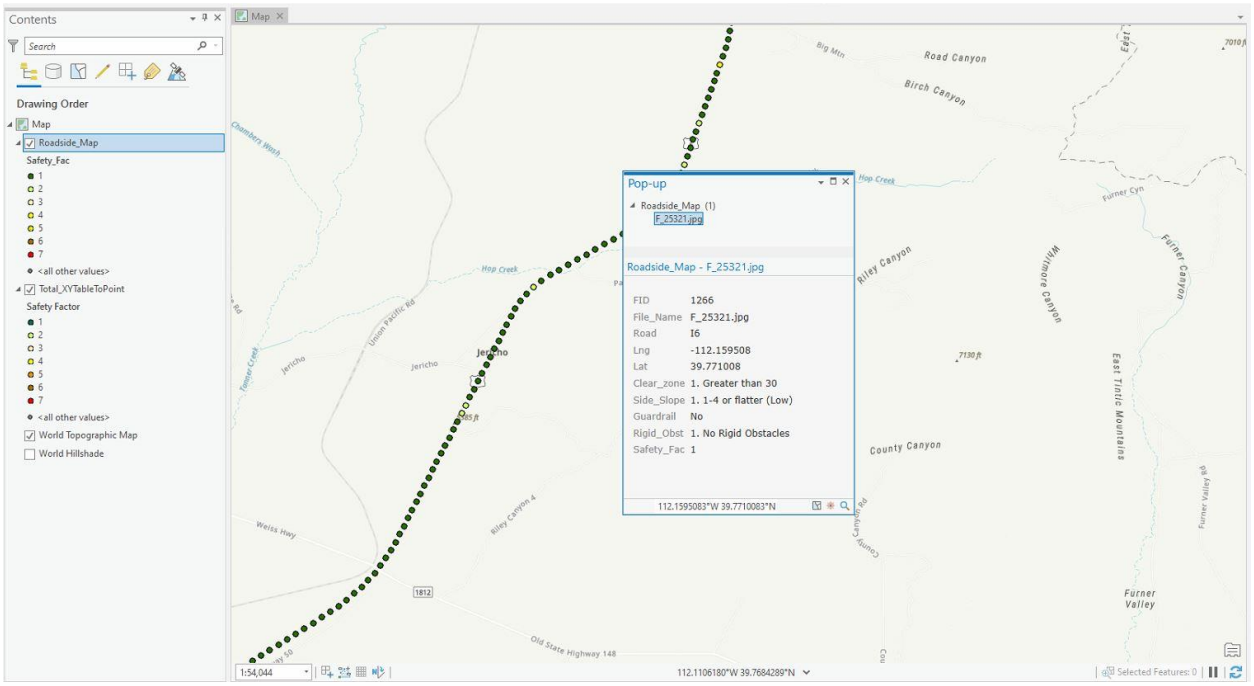

Figure 45. Example of Final Product

Figure 46. Actual Condition of the Point Listed in the Previous Figure

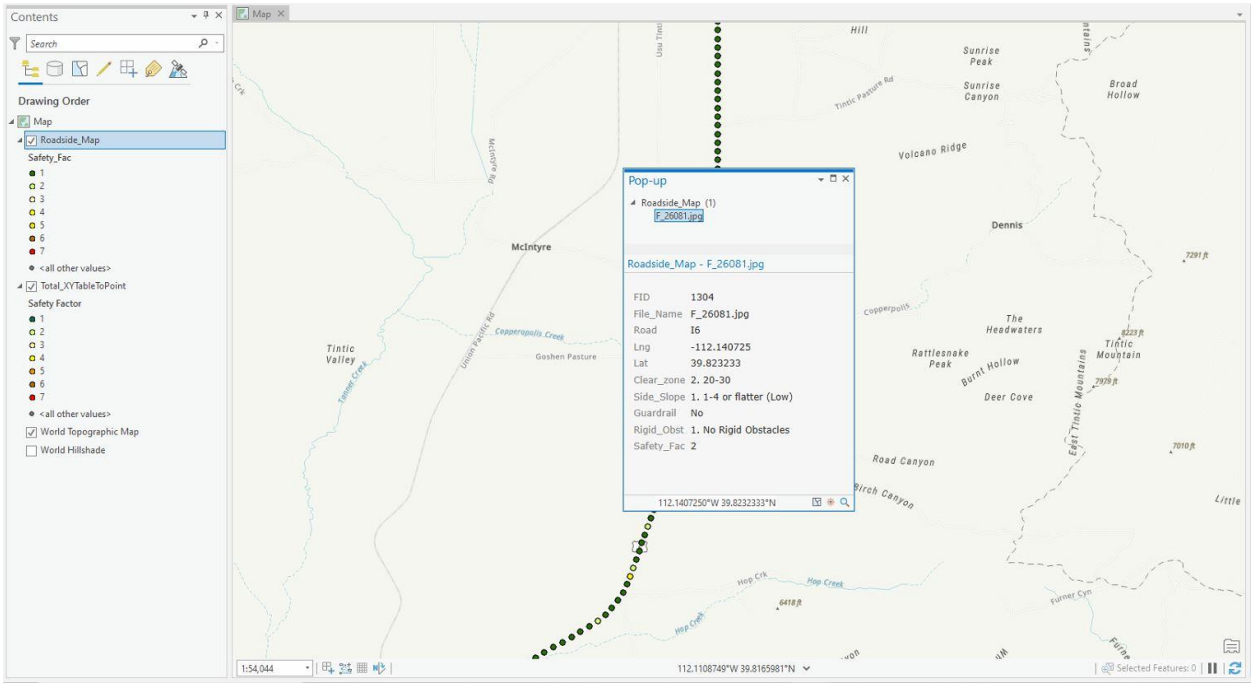

Figure 47. Example of Final Product

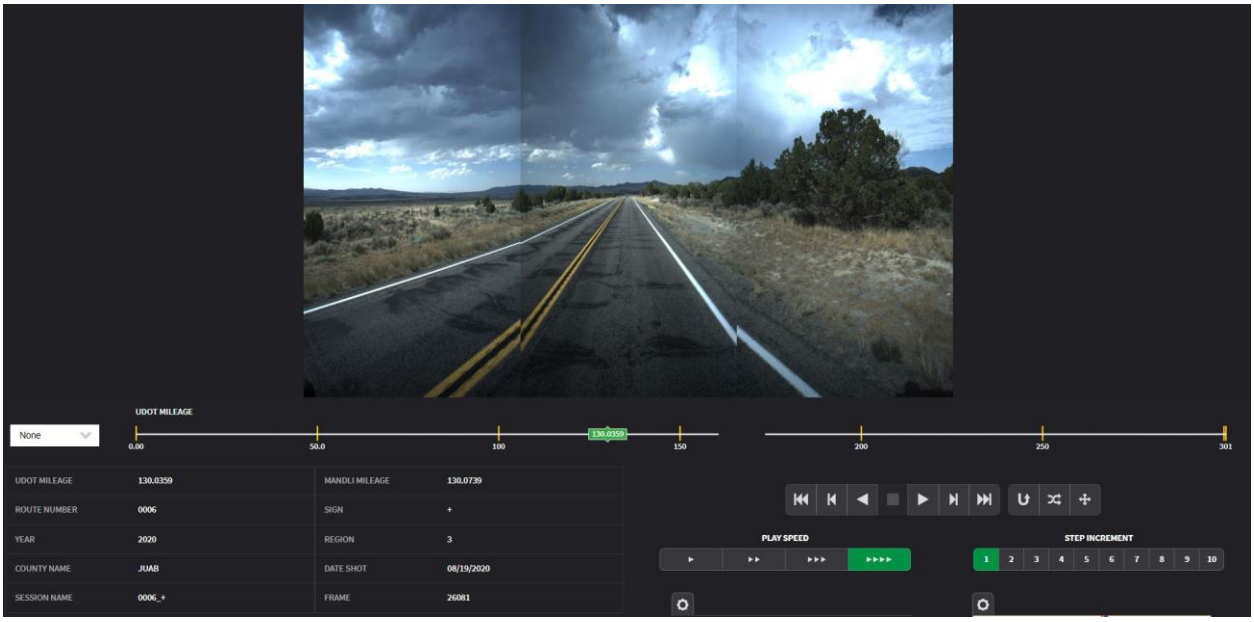

Figure 48. Actual Condition of the Point Listed in the Previous Figure

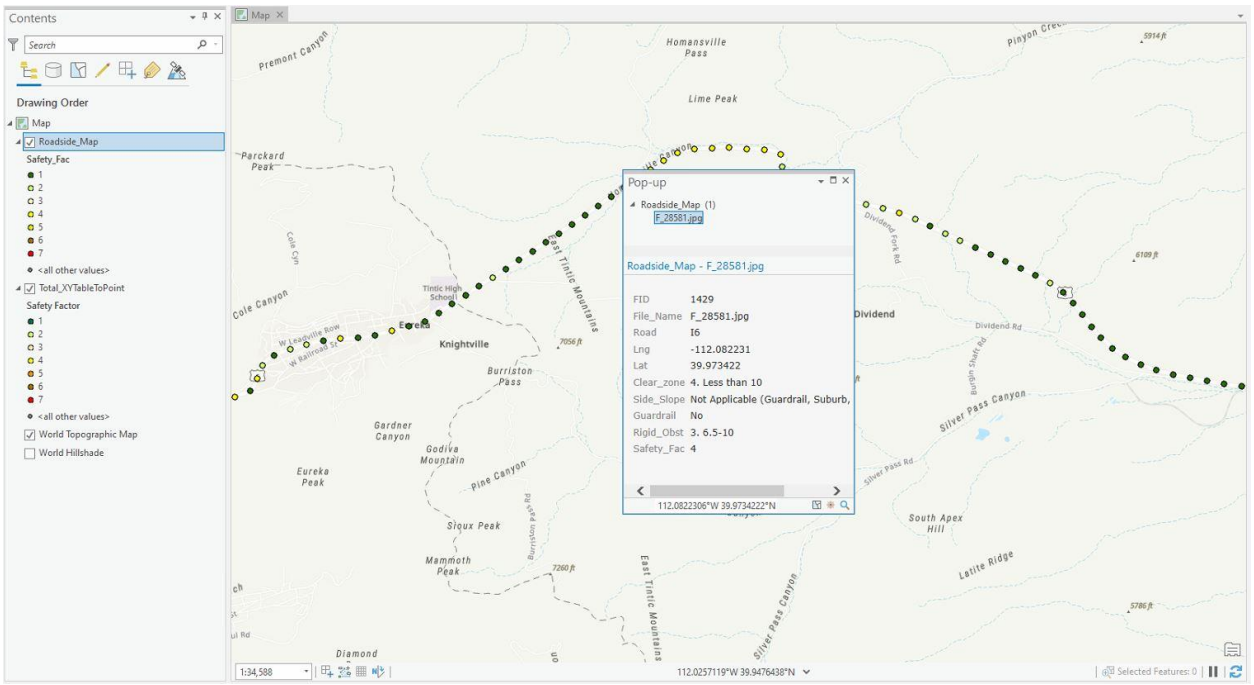

Figure 49. Example of Final Product

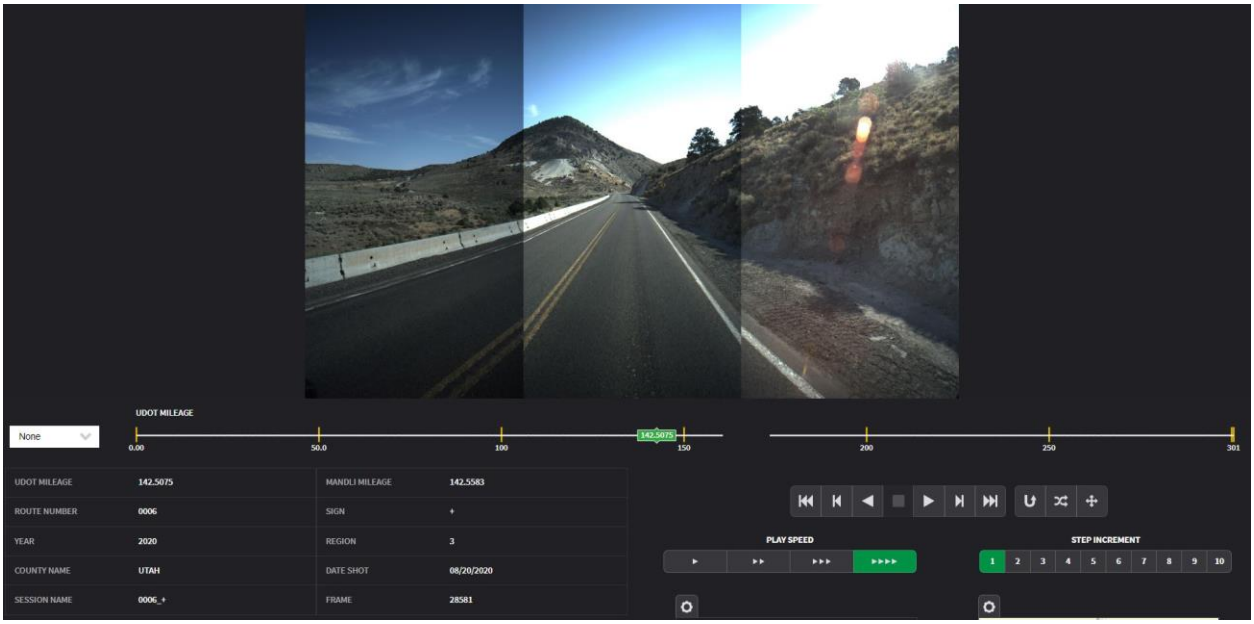

Figure 50. Actual Condition of the Point Listed in the Previous Figure

5.0 CONCLUSIONS

5.1 Summary

This project aims to assist the Utah Department of Transportation (UDOT) in identifying hazardous locations on Utah highways and accordingly prioritize projects aimed at improving safety levels (e.g., removing overgrown trees and adding guardrails). Machine-learning algorithms and Mandli images are utilized in this proposed method. The FHWA rating system is the primary standard used to evaluate roadside safety.

To automatically rank roadside safety, various computer vision algorithms have been developed for detecting roadside features, including guardrails, clear zone, rigid obstacles, and roadside slopes. The developed guardrail detection model has achieved an accuracy of 93%. The logistic regression model for rigid obstacle detection has shown 100% accuracy in identifying vegetation and embankments on the roadside. However, the multinomial logistic regression model for detecting the distance of rigid obstacles had a moderate accuracy of 73%. To enhance the results, a third model has been trained, combining the previous two models for detecting rigid roadside obstacles, yielding an accuracy of 94%. The pre-trained models have been developed to detect the available clear zone, achieving an accuracy of 83%. Furthermore, a pretrained model has been established for classifying images based on the side slope into three classes: low, mid, and high. The model demonstrated an accuracy of 94% in identifying the correct class for roadside slopes.

Using the extracted features and developed algorithm, a safety ranking has been assigned to road segments on five state roads, US-6, SR-10, SR-12, US-40, and SR-150. The final product is a GIS shapefile comprising safety ranking and roadside features at each given interval on these six roads. This final product can assist traffic engineers in the decision-making process for improving the safety level of each road segment. With this information, UDOT can prioritize projects that address problematic locations, such as removing overgrown trees and installing guardrails, thereby improving road safety and preventing crashes.

5.2 Limitations and Challenges

While using machine-learning algorithms and computer vision technologies to assess roadside safety is a promising approach, several limitations and challenges must be addressed.

First, one of the primary limitations of this approach is the requirement for high-quality roadside images, which may not always be available or accessible. Poor image quality can lead to inaccurate detection of roadside features, compromising the reliability of the results. Additionally, the models developed to detect specific roadside features may require periodic retraining to remain effective as roadside conditions change over time. The developed model is based on data collected from the year 2020. Considering the ever-changing conditions of the vegetation, the data collection and safety ranking process must be completed every couple of years.

Another limitation is the difficulty in obtaining the necessary images. The model was developed using data from only five routes, including no interstates. This limited dataset may not be representative of the entire state. In order to improve its accuracy, additional data would need to be collected and incorporated into the model.

Another challenge is the potential for false positives and false negatives. For instance, computer vision algorithms may mistake a non-hazardous roadside feature for a hazardous one or may miss an actual hazard altogether. This could result in costly and unnecessary road maintenance work or leave potentially hazardous conditions undetected. Human verification is needed before any decision to improve problematic road segments.

Finally, this approach may not address all factors contributing to roadside safety, such as driver behavior and weather conditions. As such, it should be viewed as a complementary tool to existing methods for assessing roadside safety, rather than a replacement. Moreover, the developed model is designed and trained on rural imagery data and cannot be applied to urban areas.

Overall, while using machine-learning algorithms and computer vision technologies shows great potential for improving roadside safety, addressing the limitations and challenges associated with this approach will be critical to ensuring its effectiveness and widespread adoption.

6.0 REFERENCES

AASHTO Roadside Design Guide, 4th Edition | National Association of City Transportation Officials. (n.d.). Retrieved March 29, 2023, from https://nacto.org/references/american-association-of-state-highway-and-transportation-officials-3/

Alayba, A. M., Palade, V., England, M., & Iqbal, R. (2018). A combined CNN and LSTM model for arabic sentiment analysis. *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 179–191.

Cheng, C.-F., Rashidi, A., Davenport, M. A., & Anderson, D. V. (2017). Activity analysis of construction equipment using audio signals and support vector machines. *Automation in Construction*, *81*, 240–253.

Daniello, A., & Gabler, H. C. (2011). Fatality risk in motorcycle collisions with roadside objects in the United States. *Accident Analysis & Prevention*, *43*(3), 1167–1170.

Deep Learning for Computer Vision. (n.d.). Retrieved April 28, 2022, from https://www.run.ai/guides/deep-learning-for-computer-vision

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Ewan, L., Al-Kaisy, A., & Hossain, F. (2016). Safety effects of road geometry and roadside features on low-volume roads in Oregon. *Transportation Research Record*, *2580*(1), 47–55.

Farhadmanesh, M., Cross, C., Mashhadi, A. H., Rashidi, A., & Wempen, J. (2021a). Highway Asset and Pavement Condition Management using Mobile Photogrammetry. *Transportation Research Record*, 03611981211001855.

Farhadmanesh, M., Cross, C., Mashhadi, A. H., Rashidi, A., & Wempen, J. (2021b). Use of Mobile Photogrammetry Method for Highway Asset Management. *Transportation Research Board 100th Annual MeetingTransportation Research Board*, *TRBAM-21-01864*.

Farhadmanesh, M., Marković, N., & Rashidi, A. (2022). Automated Video-Based Air Traffic Surveillance System for Counting General Aviation Aircraft Operations at Non-Towered Airports. *Transportation Research Record*, 03611981221115087.

Farhadmanesh, M., Rashidi, A., & Marković, N. (2022). General Aviation Aircraft Identification at Non-Towered Airports Using a Two-Step Computer Vision-Based Approach. *IEEE Access*, *10*, 48778–48791.

Gao, J., Chen, Y., Junior, J. M., Wang, C., & Li, J. (2020). Rapid extraction of urban road guardrails from mobile liDAR point clouds. *IEEE Transactions on Intelligent Transportation Systems*.

Gouda, M., Arantes de Achilles Mello, B., & El-Basyouny, K. (2021). Automated object detection, mapping, and assessment of roadside clear zones using LiDAR data. *Transportation Research Record*, *2675*(12), 432–448.

Gross, F., Jovanis, P. P., & Eccles, K. (2009). Safety effectiveness of lane and shoulder width combinations on rural, two-lane, undivided roads. *Transportation Research Record*, *2103*(1), 42–49.

Haghighi, N., Liu, X. C., Zhang, G., & Porter, R. J. (2018). Impact of roadway geometric features on crash severity on rural two-lane highways. *Accident Analysis & Prevention*, *111*, 34–42.

Harbaš, I., Prentašić, P., & Subašić, M. (2018). Detection of roadside vegetation using Fully Convolutional Networks. *Image and Vision Computing*, *74*, 1–9.

Lau, M. M., Lim, K. H., & Gopalai, A. A. (2015). Malaysia traffic sign recognition with convolutional neural network. *2015 IEEE International Conference on Digital Signal Processing (DSP)*, 1006–1010.

Lee, J., & Mannering, F. (2002). Impact of roadside features on the frequency and severity of run-off-roadway accidents: an empirical analysis. *Accident Analysis & Prevention*, *34*(2), 149–161.

Lord, D., Brewer, M. A., Fitzpatrick, K., Geedipally, S. R., & Peng, Y. (2011). *Analysis of roadway departure crashes on two lane rural roads in Texas.* Texas Transportation Institute.

Mandli X-35 - Mandli Communications. (n.d.). Retrieved April 28, 2022, from https://www.mandli.com/solutions/mandli-x-35/

Manuel, A., El-Basyouny, K., & Islam, M. T. (2014). Investigating the safety effects of road width on urban collector roadways. *Safety Science*, *62*, 305–311.

Mashhadi, A. H., Farhadmanesh, M., Rashidi, A., & Marković, N. (2021a). Review of methods for estimating construction work zone capacity. *Transportation Research Record*, *2675*(9), 382–397.

Mashhadi, A. H., Farhadmanesh, M., Rashidi, A., & Marković, N. (2021b). State-of-the-Art Methods in Estimating Freeway Work zones Capacity: A Literature Review. *Transportation Research Board 100th Annual MeetingTransportation Research Board, TRBAM-21-01863*.

Matsumoto, H., Mori, Y., & Masuda, H. (2021). Extraction of Guardrails from MMS Data Using Convolutional Neural Network. *International Journal of Automation Technology*, *15*(3), 258–267.

MobileNet vs ResNet50 - Two CNN Transfer Learning Light Frameworks. (n.d.). Retrieved April 28, 2022, from https://analyticsindiamag.com/mobilenet-vs-resnet50-two-cnn-transfer-learning-light-frameworks/

Mohammadi, P., Rashidi, A., Malekzadeh, M., & Tiwari, S. (2023). Evaluating various machine learning algorithms for automated inspection of culverts. *Engineering Analysis with Boundary Elements*, *148*, 366–375.

Pan, J., Shapiro, J., Wohlwend, J., Han, K. J., Lei, T., & Ma, T. (2020). ASAPP-ASR: Multistream CNN and self-attentive SRU for SOTA speech recognition. *ArXiv Preprint ArXiv:2005.10469*.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. v. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *ArXiv Preprint ArXiv:1904.08779*.

Park, J., & Abdel-Aty, M. (2015). Assessing the safety effects of multiple roadside treatments using parametric and nonparametric approaches. *Accident Analysis & Prevention*, *83*, 203–213.

Reducing Rural Roadway Departures: Where and Why Departures Occur - TAPCO - Traffic and Parking Control Co., Inc. (n.d.). Retrieved April 28, 2022, from https://www.tapconet.com/blog/reducing-rural-roadway-departures-why-rural-roadway-departures-happen-/

Rezapour, M., & Ksaibati, K. (2021). Convolutional Neural Network for Roadside Barriers Detection: Transfer Learning versus Non-Transfer Learning. *Signals*, *2*(1), 72–86.

Roadway Departure Safety - Safety | Federal Highway Administration. (n.d.). Retrieved April 28, 2022, from https://safety.fhwa.dot.gov/roadway_dept/

Roque, C., Moura, F., & Cardoso, J. L. (2015). Detecting unforgiving roadside contributors through the severity analysis of ran-off-road crashes. *Accident Analysis & Prevention*, *80*, 262–273.

Satterfield, C., Meeks, M., Albin, D., & Scott Davis, W. (n.d.). *Reducing Rural Roadway Departures Every Day Counts Round 5 Center for Accelerating Innovation*. Retrieved April 28, 2022, from https://www.fhwa.dot.gov/innovation/

Schneider, S., Taylor, G. W., Linquist, S., & Kremer, S. C. (2019). Past, present and future approaches using computer vision for animal re-identification from camera trap data. *Methods in Ecology and Evolution*, *10*(4), 461–470.

Sherafat, B., Rashidi, A., & Asgari, S. (2022). Sound-based multiple-equipment activity recognition using convolutional neural networks. *Automation in Construction*, *135*, 104104.

Stewart, T. (2022). *Overview of Motor Vehicle Crashes in 2020*. (Report No. DOT HS 813 266). National Highway Traffic Safety Administration.

Tang, T., Zhu, S., Guo, Y., Zhou, X., & Cao, Y. (2019). Evaluating the safety risk of rural roadsides using a Bayesian network method. *International Journal of Environmental Research and Public Health*, *16*(7), 1166.

Top 4 Pre-Trained Models for Image Classification | With Python Code. (n.d.). Retrieved April 28, 2022, from https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

Utah SHSP. (n.d.). Retrieved March 28, 2023, from https://www.udot.utah.gov/shsp/roadwaydepartureCrashes.html

Widiastuti, N. I. (2019). Convolution neural network for text mining and natural language processing. *IOP Conference Series: Materials Science and Engineering*, *662*(5), 052010.

Xia, C., Fu, L., Liu, Z., Liu, H., Chen, L., & Liu, Y. (2018). Aquatic toxic analysis by monitoring fish behavior using computer vision: a recent progress. *Journal of Toxicology*, *2018*.

Yuan-Fu, Y. (2019). A deep learning model for identification of defect patterns in semiconductor wafer map. *2019 30th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 1–6.

Zhong, M., Verma, B., & Affirm, J. (2019). Point cloud classification for detecting roadside safety attributes and distances. *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1078–1084.

Zou, Y., Tarko, A. P., Chen, E., & Romero, M. A. (2014). Effectiveness of cable barriers, guardrails, and concrete barrier walls in reducing the risk of injury. *Accident Analysis & Prevention*, *72*, 55–65.

7.0 APPENDIX

7.1 Preprocessing

Preprocessing is an essential step in image classification, as it allows us to transform raw image data into a more suitable format for analysis. Image cropping is one of the most common preprocessing techniques used in image classification. Image cropping involves removing parts of an image irrelevant to the classification task, such as borders or background clutter. Cropping the image helps to isolate the object of interest, making it easier to classify.

Cropping can also help to reduce the amount of noise or variability in an image. This is particularly important when working with images that have a lot of background or foreground clutter, which can make it difficult for a classification algorithm to distinguish between different objects. Removing irrelevant parts of the image can increase the signal-to-noise ratio and improve classification accuracy.

For this project, we chose to crop all Mandli-taken images to their right-down section. This section contains the most relevant information for identifying the type of roadside object in the image. This is because roads generally have a consistent layout with signs and other objects placed in specific locations relative to the road.

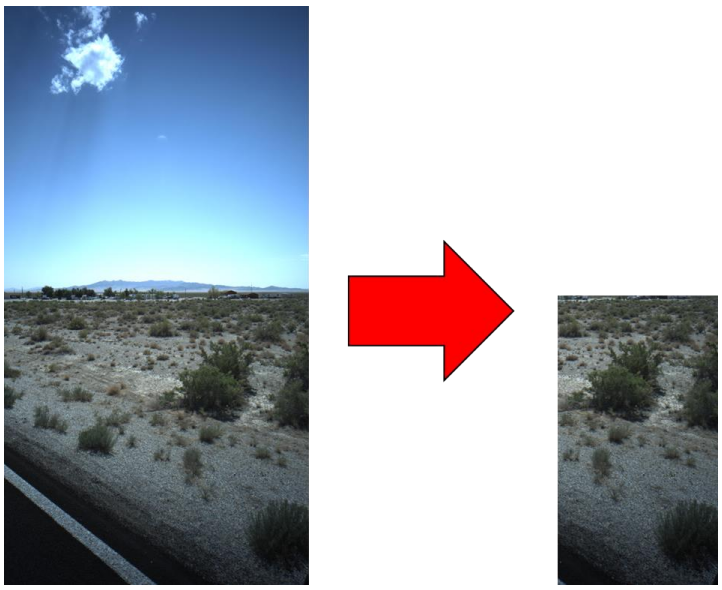

Figure 51. Image Cropping

Here is the python implementation for this purpose:

```python
from PIL import Image
import os.path, sys
from tqdm import tqdm

roads = ["0006_+_20"]

for road in roads:
    path = f"E:\\Mike Dataset\\2020\\{road}"
    dirs = os.listdir(path)
    for item in tqdm(dirs):
        fullpath = os.path.join(path,item)
        if os.path.isfile(fullpath):
            im = Image.open(fullpath)
            w, h = im.size
            f, e = os.path.splitext(fullpath)
            # imCrop = im.crop((0, 2644, 2160, 4096)) #2160x4096          (left, upper, right, lower)
            #imCrop = im.crop((int(w/2), int(h/2), w, h)) #5250x3000          (left, upper, right, lower)
            imCrop = im.crop((0, int(h/2), w, h))
            imCrop.save(f + '_Cropped.jpg', "JPEG", quality=100)
    print(f"Finished {road}")
```

Figure 52. Python Implementation for Image Cropping

This Python code crops images in a specified directory path and saves the cropped images in the same directory. Here's a breakdown of what the code does:

1. The first line of the code imports the necessary modules: PIL (Python Imaging Library), os.path, sys, and tqdm.
2. The second line of the code initializes a list of "roads" that represent different directories where images are stored.
3. The third line of the code starts a loop that iterates over each road in the list of roads.
4. The fourth line of the code constructs the directory path for the current road using an f-string, which substitutes the name of the road into the path.
5. The fifth line of the code reads the list of files in the directory path using the os.listdir() function.
6. The sixth line of the code starts a loop that iterates over each file in the directory.
7. The seventh line of the code constructs the full path of each file by joining the directory path and file name using the os.path.join() function.
8. The eighth line of the code checks if the current item in the loop is a file using the os.path.isfile() function.

9. The ninth line of the code opens the file using the Image.open() function from the Python Imaging Library (PIL) and gets its width and height using the size attribute of the image object.
10. The tenth line of the code splits the file path into its base name and extension using the os.path.splitext() function.
11. The next three lines of the code crop the image using different parameters that are commented out. The first parameter of the crop() function represents the left, upper corner of the crop box, and the last two parameters represent the right, lower corner of the crop box. The values of these parameters are expressed as pixel coordinates.
12. The thirteenth line of the code saves the cropped image in the same directory as the original file with "_Cropped" appended to the base name and ".jpg" appended to the extension using the save() method of the image object.
13. The fourteenth line of the code prints a message indicating that the processing of the current road is complete.

This is a Python code that crops images located in a specific directory. If you are not familiar with Python, you can still use this code by following these steps:

1. Install Python: If you do not have Python installed on your computer, you can download it from the official website: https://www.python.org/downloads/
2. Install Required Packages: You need to install the following packages to run this code:

- Pillow (PIL): This package is used for image processing. You can install it by running the following command in your command prompt/terminal: pip install pillow
- tqdm: This package is used to display a progress bar. You can install it by running the following command in your command prompt/terminal: pip install tqdm

3. Modify the code: You need to modify the following line in the code to specify the directory where your images are located:
    - roads = ["0006_+_20"]
    - path = f"E:\\Mike Dataset\\2020\\{road}"

    Change E:\\Mike Dataset\\2020 to the directory path where your images are located.

4. Run the code: Open your command prompt/terminal, navigate to the directory where the code is saved using the cd command, and then run the code using the following command: python 'your_file_name.py'

Replace 'your_file_name.py' with thePython file name where you saved the code.

7.2 Application

Here is a snippet of the application:

```
SIZE = 256  #Resize images

#Capture training data and labels into respective lists
train_images = []
train_labels = []

for directory_path in glob.glob("images/classification/train/*"):
    label = directory_path.split("\\")[-1]
    for img_path in tqdm(glob.glob(os.path.join(directory_path, "*.jpg"))):
        #print(img_path)
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (SIZE, SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        train_images.append(img)
        train_labels.append(label)

#Convert lists to arrays
train_images = np.array(train_images)
train_labels = np.array(train_labels)

```

Figure 53. Snippet of Application

This code performs roadside image classification using a pre-trained VGG16 model as a feature extractor and an XGBoost model as a classifier. The dataset used is stored in the "images/classification" directory, which contains subdirectories "train" and "test", each containing images belonging to different classes.

The code first reads the images from the "train" directory and resizes them to 256x256 pixels. It then reads the labels from the subdirectory names and encodes them as integers using the LabelEncoder class from sci-kit-learn. The images and their corresponding labels are stored in

separate NumPy arrays, and the data is split into training and validation sets using the train_test_split function from sci-kit-learn.

The pixel values of the images are normalized to be between 0 and 1. The VGG16 model is then loaded without its classifier/fully connected layers, and the loaded layers are made non-trainable. The convolutional layers of the VGG16 model are then used to extract features from the training set, and these features are reshaped and used as input to the XGBoost model for training.

The XGBoost model is then used to predict the labels of the validation set, and the inverse transform is applied to convert the encoded labels back to their original text form. The predicted labels for the test set are also saved to a CSV file.

To use this code for roadside image classification, a non-programmer can follow the steps below:

1. Prepare the test images: crop the roadside images that you want to classify and save them in a new folder, for example, "images/classification/test".
2. Download the code and required libraries: download the code and install the necessary libraries, including numpy, matplotlib, OpenCV, pandas, keras, scikit-learn, seaborn, xgboost, and tqdm.
3. Open the code: open the code in a text editor, such as Notepad or TextEdit.
4. Run the code: run the code by executing it in a Python environment, such as Jupyter Notebook or Spyder.
5. Wait for the results: the program will take some time to train the model and make predictions on the test images. Once it is finished, it will save the results to a CSV file in the same folder as the code.
6. Check the results: open the CSV file to view the predictions for each test image. The file name and predicted class will be listed for each image.

Note: the code assumes that the test images are in JPEG format and have the file extension ".jpg". If your images are in a different format, you may need to modify the code to read them correctly.

7.3 Mapping

To display the results on a map, we need to extract the GPS information associated with Mandli images after running the model and obtaining labels for any number of images. Below is a code snippet for extracting GPS information:

```
for index, image_member_list in enumerate(image_members):
    print(f"Image {index} contains {len(image_member_list)} members:")
    print(f"{image_member_list}\n")

for index, image in enumerate(images):
    print(f"Device information - Image {index}")
    print("----------------------------")
    print(f"Altitude: {image.gps_altitude}")
    print(f"Longitude: {image.gps_longitude}")
    print(f"Latitude: {image.gps_latitude}\n")

i=0
for index, image in enumerate(images):
    df.loc[i,0] = index
    df.loc[i,1] = image.gps_altitude
    df.loc[i,2] = str(image.gps_longitude)
    df.loc[i,3] = str(image.gps_latitude)
    i+=1
df.to_csv("coordinates_2020_r6.csv", index_label=os.listdir(r6))
# df.set_index(os.listdir(r6))
```

Figure 54. Code Snippet for GPS Extraction

This code is a Python script that extracts GPS (Global Positioning System) coordinates from images. The images are stored in five folders located at "E:\Mike Dataset". The script uses a Python library called "exif" to extract the GPS coordinates from the image files. It also uses another library called "tqdm" to show a progress bar for processing the images.

The script creates an empty Pandas dataframe (a type of data structure) to store the GPS coordinates. It then loops over the images in each folder using the "os" library to get a list of files in each directory. The script then creates an Image object for each image using the "Image" class from the "exif" library.

The script then loops over the Image objects to extract the member variables in each object. It prints out the number of members and the names of the members for each Image object.

The script then loops over the Image objects again to extract the GPS coordinates for each image. It prints out the GPS coordinates for each image.

Finally, the script saves the GPS coordinates in a CSV file named "coordinates_2020_r6.csv" with the help of Pandas DataFrame. Each row of the CSV file contains the index of the image and the GPS coordinates.

This code extracts GPS information from Mandli images and saves the results in a CSV file. To use this code as a non-programmer, follow these steps:

1. Install Python on your computer.
2. Install the necessary Python packages by running the following commands in your command prompt or terminal:
    a. pip install exif
    b. pip install tqdm
    c. pip install opencv-python
    d. pip install pandas
3. Create an " images " folder in the same directory as the Python script.
4. Put all of your Mandli images from which you want to extract GPS information into the "images" folder.
5. Open the Python script and replace the path path_to_images = "./images" with the path to your "images" folder.
6. Run the Python script.
7. The script will extract the GPS information from the images and save it in a CSV file named "coordinates.csv" in the same directory as the Python script.