## **Project Report:**

**Title:** Damaged Road detection with IOT using computer vision.

### **Introduction:**

Damaged roads and water filled potholes are major concerns for citizens during every season. Though the wear and tear would damage roads with time, a computer aided analysis would help in measuring the damage caused by rain and other reasons.

No specific technology to detect and determine the after effects of rains on roads gives a niche space for an IoT device to monitor the state of roads. This can also be used to either compare or detect potholes at any site of the camera.

**Summary:**

This project delivers two designs:

Version 1: Compares the current road condition with the existing previously held image of the road to find the potholes.

(ii) A fresh image of the road can be taken now and compared with the road everytime after a regular interval of time.

**Problem:**

Road is always with traffic, debris and other random material that carries away with wind and water. This can result in a false alarm as a pothole even if one does not exist.

**Solution:**

Using machine learning neural networks algorithm to train the program so as to identify only the potholes and ignore other things on road. Eg: people, animals and traffic.

We have chosen YOLO-v4 for achieving this.

YOLO (You Only Look Once) is a real-time object detection algorithm that is faster and more efficient than many other object detection algorithms, such as R-CNN (Regional Convolutional Neural Network) and its variants.

One of the main reasons why YOLO is faster than R-CNN and its variants is that it performs object detection in a single pass through the network. In contrast, R-CNN and its variants require multiple passes through the network, which can be slower and more computationally intensive.

Another reason why YOLO is more efficient than R-CNN and its variants is that it uses a fully convolutional neural network (CNN) to perform object detection. This means that it does not require region proposals, which are used in R-CNN and its variants to identify potential object locations. Instead, YOLO uses a single neural network to predict the bounding boxes and class probabilities for all objects in an image.

Overall, YOLO has become a popular choice for real-time object detection tasks because it is fast, efficient, and easy to implement. However, it is worth noting that different object detection algorithms may be better suited for different tasks, depending on the specific requirements and constraints of the problem at hand.

**Methodology:**.

Training a YOLO (You Only Look Once) model involves several steps, which include:

* Gathering and preprocessing a dataset of images: You will need a dataset of images that contains a variety of objects, including the objects you want to detect. The images should be labeled with bounding boxes around the objects of interest and their corresponding class labels. You may need to preprocess the images by resizing or cropping them to a standard size.
* Splitting the dataset into training and validation sets: You should split your dataset into a training set and a validation set. The training set will be used to train the YOLO model, while the validation set will be used to evaluate the model during training.
* Configuring the YOLO model: You will need to choose the architecture of the YOLO model, such as the number of layers and the size of the convolutional filters. You will also need to specify the hyperparameters of the model, such as the learning rate and the batch size.
* Training the YOLO model: Once the model has been configured, you can start training it using the training set. You will need to specify the number of epochs (iterations over the training set) and the optimization algorithm to use. During training, the model will learn to recognize the features that are characteristic of the objects in the training set.
* Evaluating the YOLO model: After training the model, you should evaluate it on the validation set to see how well it performs on unseen data. You can use a metric such as mean average precision (mAP) to measure the model's performance.
* Fine-tuning the YOLO model: If the model's performance is not satisfactory, you may need to fine-tune the model by adjusting the hyperparameters or by adding or removing layers from the architecture. You can then repeat the training and evaluation process until you achieve the desired performance.
* Training a YOLO model can be a complex and time-consuming process, especially if you are working with a large dataset. It requires a good understanding of machine learning and computer vision techniques, as well as access to appropriate tools and resources. However, with the right expertise and resources, it is definitely possible to train a YOLO model for object detection tasks.

**Alternate technique:**

An alternate technique would be to use a existing pre trained model and using it to detect the potholes. The weights and configuration file should be correctly referred to give the desired behavior when running the software to initiate the detection.

**Project Schedule/Milestones:**

Milestone 1:

Installations: Raspberry pi set up, camera setup.

Milestone 2:

Version 1:

* Image Processing: To compare the existing image with the image taken later and find the difference between the images
* Libraries imutils(from opencv) or skimmage are used.

**Steps**:

* Resize the image
* Find the absolute differences - function: absdiff
* Threshold the resultant image - makes it easier to analyse
* Finding contours : shape analysis and object detection.

Sample outputs:



Milestone 3:

Able to use twitter api to tweet the road conditions so it can reach to large audience.



The entire documentation of usage and implementation of twitter api has been shared in skillplot.

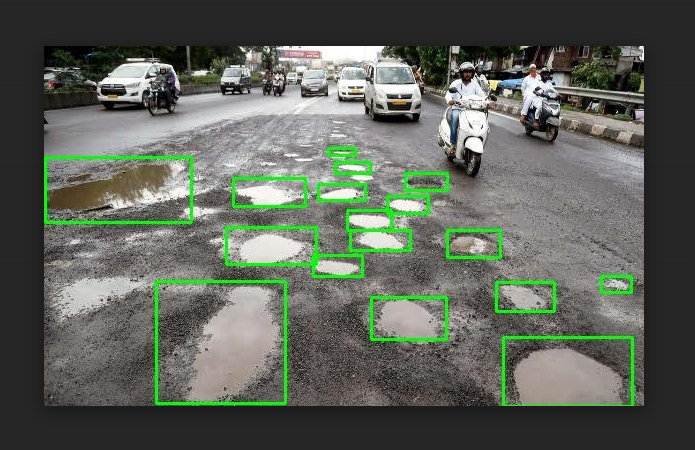
Please note that elevated access is required to post images on twitter. Essential access only allows to read a tweet.

**Milestone 4**: Version 2:

Using yolo to detect potholes on road. We have used random images to test and see the efficiency of the algorithm. The weights and config file are most important and have to be rightly utilized to detect the potholes.

1. In YOLO (You Only Look Once), weights refer to the parameters of the neural network that are learned during training. These weights are stored in a file and are used to make predictions on new data.
2. A YOLO configuration file is a text file that specifies the architecture and hyperparameters of the YOLO model. The cfg file is used to specify the number of layers in the model, the size of the convolutional filters, the number of classes, and other important details about the model.
3. During training, the YOLO model learns the weights that are optimal for the task at hand by minimizing a loss function. The weights and the cfg file are used together to make predictions on new data.
4. For example, suppose you have trained a YOLO model to detect objects in images. You can use the cfg file to specify the architecture of the model, and you can use the weights file to store the learned parameters of the model. To make predictions on a new image, you can use the cfg file to set up the model and the weights file to load the learned parameters into the model. The model can then use these parameters to make predictions about the objects in the new image.

Sample output:



Milestone 5:

Used an API to email this information to gmail account. This would help in notifying higher authorities of the road conditions so the initiate measures to fix the road. The SMTP usage, methods and other details are well explained in the [Raspberry pi Set up email API.docx](https://github.com/skillplot/cs-g518-iot/blob/main/projects/sem-1-2022-23/Project_KSKumar/Version2/Docs/Raspberry pi Set up email API.docx" \o "Raspberry pi Set up email API.docx)

**Conclusions:**

Overall, our project demonstrates the use of OpenCV and machine learning algorithms to find the road conditions and update the required authorities when a pothole is found.

Advance libraries like skimage and yolo are used in both version to show efficient results in minimum time(faster processing). The algorithm works faster in yolo as well thought the weights file(trained model to identify) is large. This is because yolo outperforms all other algorithms like SCNN. Please note thonny does support skimage due to installation issues, visual studio code is used to run version 1. Version 2 is the major component and can be executed straight from the command prompt/terminal. It should be powered with external power source like powerbank, the execution should be initiated by connecting to laptop and once the execution starts, disconnect from the laptop/pc and raspberry pi can work as a standalone device now.

**Potential Improvement:**

Additional training can be given with the raspberry pi images. As the random images of potholes on the internet are of better cameras, it might be more efficient if raspberry pi images are used for training as they are lower pixel when compared to the ones over the internet. This would enhance the results of the raspberry pi detection and allows to notify the right team about the road damages at early stages.