Pothole Detection on Android

Ojas Mehta
Computer Science Department
University of Massachusetts, Lowell
Lowell, Massachusetts, USA
Ojas_Mehta@student.uml.edu

Gregory LeMasurier
Computer Science Department
University of Massachusetts, Lowell
Lowell, Massachusetts, USA
Gregory_LeMasurier@student.uml.edu

***Abstract*—This paper aims to demonstrate the use of an Android smartphone to detect potholes, speed bumps, and generate a heat map on a Google Map interface for a visual overview. The app processes the live sensor data from a smartphone, extracts various statistical features, and applies them to a machine learning model, which classifies a pothole or speed bump upon detection. While there have been many approaches to solving the problem of pothole detection, our app does so just based solely on sensor data, and doesn't require any image recognition system. Our core idea is basic. (1) Detect potholes by inputting the features of raw sensor values from a smartphone IMU to a machine learning model. (2) Provide a visual interface in form of a MapView fitted with a heatmap of speedbumps, and the 3 levels of potholes detected (small, large, deep).**

Smartphones are a common accessory that everyone carries around with them. This pothole detection system is a simple use of the sensors on these smartphones to improve the safety of drivers on the road. The system is very naive in its current state, but can be expanded to work with a network of other such systems, to provide a universal database of detected potholes. Such a database can be implemented across various navigation apps, to provide for a safer driving experience for everyone.

I. Introduction

Potholes are a huge hassle when driving on the road; even more so in colder regions. The freezing and thawing cycle during the winter is what causes the cracks on the road. Furthermore, during winter, salt is applied on these roads to reduce ice accumulation. This further accelerates the freezing-thawing process, causing damage to the road, and its condition. Eventually, the water infiltrates these cracks, causing even more deterioration. That is how a pothole is formed. These potholes can be filled, but this process keeps recurring.

Driving over a pothole is no doubt uncomfortable, but it can potentially cause damage to the vehicle, putting the driver's life at risk. "Pothole damage costs U.S. drivers $3 billion per year, according to a new study from the AAA auto club. If you are driving through a pothole every day on your daily commute. Some of the more common damage is a flat tire or damage to your tires, bent or damaged rims, suspension damage, steering damage, and even damage to the body of the car. Potholes can even knock your car out of alignment so it will affect the way the tires wear and can lead to replacing tires before earlier than expected." [1] "Hitting a pothole can cause vehicle component damage, particularly with the shocks and struts. Shocks and struts control ride and handling, and serve as a cushion to dampen the bouncing action of the vehicle's springs. They also regulate spring and suspension movement, keeping the car's tires in contact with the road to facilitate proper steering, stability, and braking. However, "on heavier traveled roads, the constant traffic kicks up pavement and existing potholes can grow larger and cause even more serious damage in just a few hours." [2].

Our only solutions against these situations are to avoid potholes entirely, by looking out for them while driving and identifying their locations for repair. But in doing so, we are also forced to take our attention away from our surrounding traffic, increasing the likelihood of a potential accident. That is the problem we are aiming to solve.

It is very common for people to use a smartphone for navigation. With our pothole detection app, we aim to enhance the traditional use of smartphone-based GPS navigation, by developing a framework for detecting potholes as a driver runs over them. The app looks out for instances when a car drives over a pothole, or a speed bump then gathers the location and the intensity of the bump and displays it on a heat map for visualization. This framework can be implemented in navigation apps such as Google Maps, or Waze. Since our app maintains a database of known pothole locations, it can also be used by the cities to fix them. As of now, the app is still in its early stages, but many other features can be added, thus improving the overall end-user experience of the app.

II. Related Work

There are several different approaches to pothole detection. Several groups have investigated using external sensors to identify potholes. Kang and Choi proposed a pothole detection algorithm using two 2D lidar sensors and a camera [3]. Li et al. created a pothole detection system using stereo vision, where depth is calculated from the differences between two different camera streams [4]. Jo and Ryu created a pothole detection system using a blackbox camera [5]. The major drawback of these approaches is that they are computationally expensive and require additional computers and sensors to be added to the car.

Another approach to pothole detection is through the use of sensors which are commonly found in smartphones. Hoffman et al. used accelerometer data to identify bumps in the road using a Naive Bayes model [6]. They found that their model could achieve a true positive rate of 82%. Additionally, Wu et al. proposed using a Random Forest classification model to

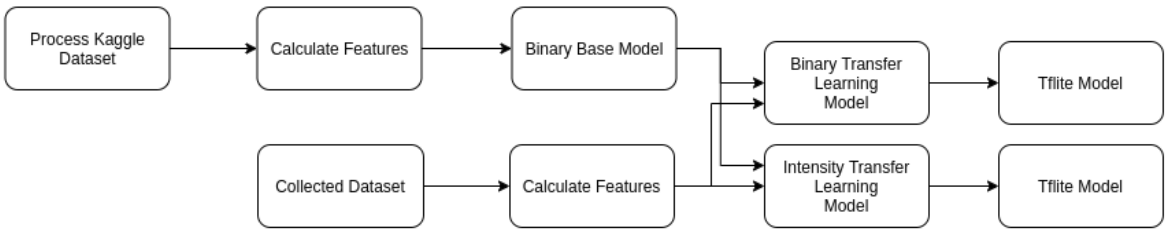

Fig. 1. A high level overview of our base and transfer learning classification models

classify accelerometer data [7]. They achieved 88.5% precision and 75% recall in their approach. Bansal et al. analyzed many different classification models for pothole detection using features that were calculated from accelerometer and gyroscope data [8]. Ultimately, they found that their best performing model was a Random Forest model with a 90% true positive rate with a low 80% precision, recall, and f1-score. Our approach uses similar features to those used by Bansal et al. The proposed approach to pothole classification is novel as we not only enable binary classification of potholes, but we also attempt to classify potholes by their intensity.

III. Overall System Architecture

A. Machine Learning

To enable the classification of potholes and their intensity, we have implemented several machine learning models. These models take in an input of 12 features including the mean, min, max, variance, standard deviation, and zero-crossing rate of the accelerometer and gyroscope data. Each set of features has a corresponding label which is used for validation in this supervised learning model. Tflite models were generated for all three of our models. This tflite model was then used in our Android application to enable real-time classification. A high-level overview of our machine learning models can be seen in Figure 1.

1) Base Model: First, we created a simple sequential Keras model for the binary classification of potholes. This model uses the 12 features as input and learns relationships between these features which correspond with each potential label. This model consists of four initial dense layers with a total of 7,580 trainable parameters. These layers use relu activation. The final layer of this model is a sigmoid activation layer which outputs a binary label. When a pothole is detected, this model will output 1, otherwise, it will output 0 to indicate that the input features are not a result of a pothole. This model uses a binary cross-entropy loss function and an adam optimizer. To prevent overfitting, we use a learning rate decay callback function. A summary of this base model can be seen in Figure 2.

2) Transfer Learning for Binary Classification: To enable the classification of an additional source of data, we have expanded upon our base model using transfer learning. First, we freeze the base model's layers. Then we pop off the last classification layer and add a dropout layer with a dropout

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               1300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 7,591
Trainable params: 7,591
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. A summary of our base model

value of 0.5. This layer was included to prevent overfitting to the base model. Then we add a new sigmoid classification layer for the binary classification of potholes. This model also uses a binary cross-entropy loss function and an adam optimizer. We also use the same learning rate decay callback function as the base model. A summary of this transfer learning binary classification model can be seen in Figure 3.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_input (InputLayer)     [(None, 12)]              0
_________________________________________________________________
dense (Dense)                (None, 100)               1300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
_________________________________________________________________
dropout (Dropout)            (None, 10)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 11
=================================================================
Total params: 7,591
Trainable params: 11
Non-trainable params: 7,580
_________________________________________________________________
```

Fig. 3. A summary of our binary transfer learning model

3) Transfer Learning for Intensity Classification: Additionally, we investigated the classification of potholes based on the intensity level. To enable this classification we expand upon

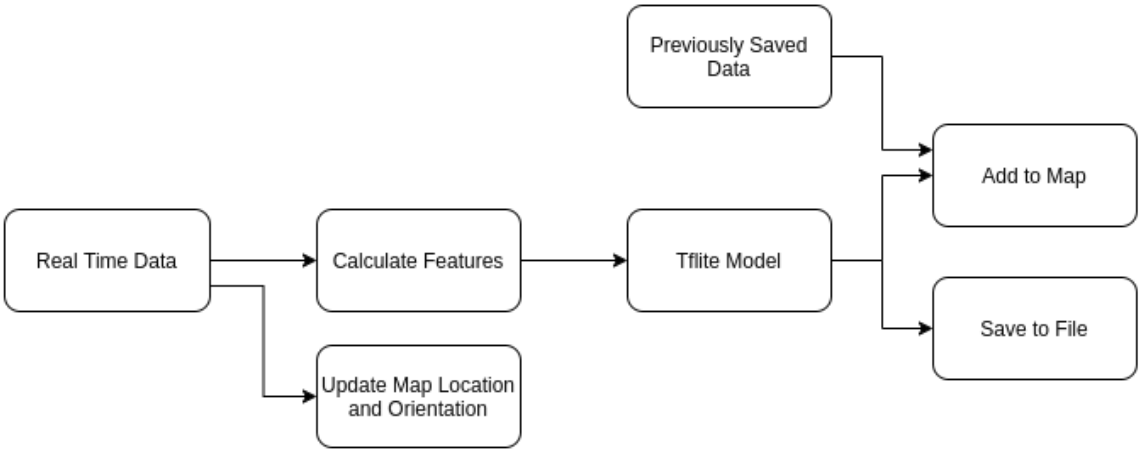

Fig. 4. A high level overview of our app.

our base model using transfer learning. First, we freeze the base model's layers and then pop off the last classification layer. Then we add a dropout layer with a dropout of 0.5 to prevent overfitting to the base model. Then we add a new dense layer. Finally, we add a new softmax classification layer for the classification of pothole intensity. The output of this model is a label which can be: 0 (no pothole), 1 (small pothole), 2 (large pothole), 3 (deep pothole), or 4 (speed bump). This model uses a categorical cross-entropy loss function and an adam optimizer. We also use the same learning rate decay callback function as the base model. A summary of this transfer learning intensity classification model can be seen in Figure 5.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_input (InputLayer)     [(None, 12)]              0
_________________________________________________________________
dense (Dense)                (None, 100)               1300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
_________________________________________________________________
dropout (Dropout)            (None, 10)                0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                110
_________________________________________________________________
dense_6 (Dense)              (None, 5)                 55
=================================================================
Total params: 7,745
Trainable params: 165
Non-trainable params: 7,580
```

Fig. 5. A summary of our intensity transfer learning model

B. Pothole Detection App

Modern-day smartphones are embedded with various technologies, and sensors, that allow them to integrate with our day-to-day activities. As a result, they are capable of doing many other complex tasks such as navigation.

1) Realtime Data Processing: Our devices are equipped with a GPS chip, that communicates with the GPS satellites using a process called triangulation. The process involves the GPS chip determining its distance away from at least 3 GPS satellites. The user's approximate location is then calculated in latitude, and longitude, based on the distances. These location coordinates are provided to GPS-based navigation apps such as Google Maps, or Waze. While the pothole detection app is running, it constantly requests updated location coordinates. Based on the location changes, the user's current location pin is updated. The camera of the map is also updated to keep in sync with the moving current location pin.

While the GPS is updated, the sensors, including the accelerometer, and gyroscope are also being updated every second. The accelerometer, and gyroscope, both provide values in the XYZ coordinate system. The data from these sensors are the most crucial components for detecting potholes. When a car runs over a pothole, an intrinsic pattern is recorded. The machine learning model identifies these patterns in the sensor data to classify potholes. However, to maximize the predictability, and prevent false classification due to different device orientations, we process each of the raw sensor data to extract its maximum, minimum, standard deviation, zero-crossing rate, mean, and variance. These features, along with the epoch time in Millis, form the input for our tflite model. The model returns an integer value between the range 0-4. An output of 0 implies no pothole was detected. While the output of 1, 2, or 3 infer that a small, large, or deep pothole was detected, respectively. The output of 4 is reserved for speed bumps.

When a pothole is detected, the location coordinates of that pothole are stored in a CSV file. This CSV file acts as a log for every instance of a pothole that is detected. Every time that the app is launched, it iterates through the list of previously-stored

location coordinates from that CSV file and marks them on a heatmap based on their intensity. The small and large potholes are marked in green, while the deep potholes and speed bumps are assigned the color red. While a user is driving, these colors serve as visual cues alerting them of an approaching pothole.

IV. Experimental Evaluation

To evaluate our application we used a publicly available dataset as well as our own collected data for the training and validation of our models. Then we applied our models using a real-time pothole classification app.

A. Kaggle Dataset

To train our models for pothole classification, we began by using a dataset found on Kaggle[1]. This dataset consisted of timestamp, location, gyroscope, accelerometer, and pothole timestamp data. This data was collected over five different trips. First, we calculated the appropriate features and labels for this data. Then we saved the features into a CSV file for later use by our machine learning models. These CSV files were concatenated into a single dataframe in our implementation.

B. Data Collection

In order to classify potholes based on their intensity levels, we needed to collect our own data that recorded the appropriate labels. To collect all this data, we made our own data collection app. On there, we had four buttons, each representing a type of pothole. We also had a textview widget display our sensor data as it is being recorded. A screenshot of our data collection app can be seen in Figure 6. We collected the x, y, z for the accelerometer, gyroscope, and the time in milliseconds. Additionally, we collected the location data in latitude and longitude. Then we had a pothole counter, to allow us to keep track of the number of potholes that had currently been encountered in that particular trip. Finally, we recorded the type of pothole we are classifying as a label for this data.

To collect this data, we mounted a phone on a car mount and launched the app. As soon as the app launches, it starts logging the sensor data. Then, as we approached a pothole, we held onto the button, which increments our pothole counter and changes the pothole type to its relevant type.

C. Dataframe Generation

When collecting data for pothole classification, most of the routes in the public dataset as well as our own collected data were on road surfaces without potholes. Thus, the input data was heavily unbalanced which led to a very low accuracy across every label. To prevent this issue of imbalance between the different label types we resampled our data so that every label had equal representation in our dataframe. By doing so we noticed a large improvement in our overall classification accuracy across labels. These dataframes were then split into 64% training, 20% test, and 16% validation data.

[1]Kaggle Dataset: https://www.kaggle.com/dextergoes/pothole-sensor-data

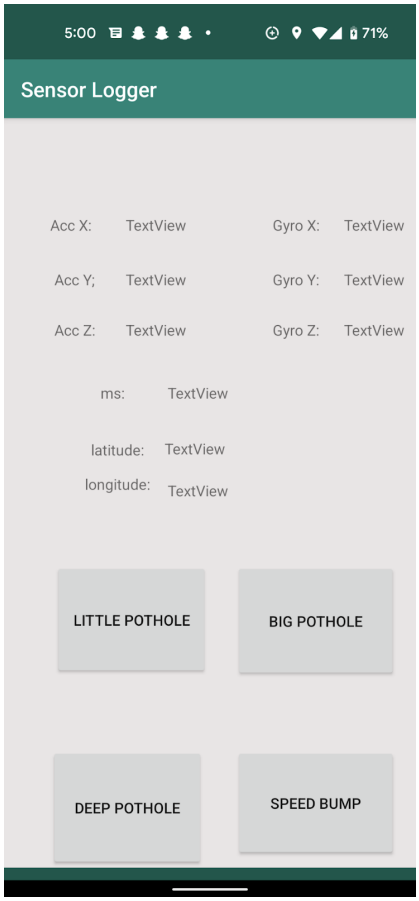

Fig. 6. A screenshot of our pothole data collection app.

D. Base Model

Our base model was trained on using only the Kaggle dataset. A breakdown of this model's performance can be found in Table I. Ultimately, this model had an overall accuracy of 85%. Our results were comparable to, and outperformed several of the reported results of Bansal et al. [8]. Figure 7 shows the confusion matrix for this model.

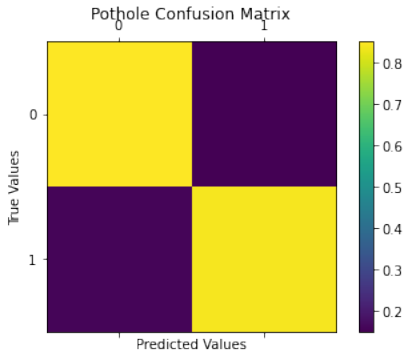

Fig. 7. The confusion matrix for our base model

TABLE I
PRECISION, RECALL, AND F1-SCORES FOR EACH POSSIBLE LABEL IN THE BASE MODEL. THE OVERALL ACCURACY OF THIS MODEL WAS 85%

Label	Precision	Recall	F1-Score
No Pothole	84%	85%	84%
Pothole	85%	84%	85%

TABLE II
PRECISION, RECALL, AND F1-SCORES FOR EACH POSSIBLE LABEL IN THE TRANSFER LEARNING BINARY CLASSIFICATION MODEL. THE OVERALL ACCURACY OF THIS MODEL WAS 69%

Label	Precision	Recall	F1-Score
No Pothole	77%	56%	65%
Pothole	64%	82%	72%

E. Transfer Learning for Binary Classification

Our transfer learning binary classification model was trained on by expanding the base model using our collected data as training data. A breakdown of this model's performance can be found in Table II. Ultimately, this model had an overall accuracy of 69%. Figure 8 shows the confusion matrix for this model.

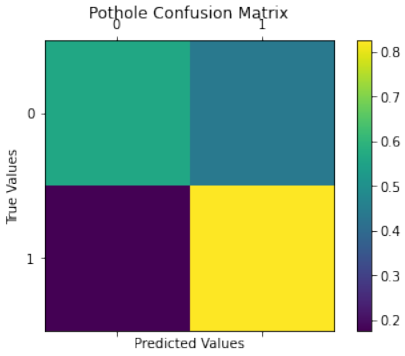

Fig. 8. The confusion matrix for our binary transfer learning model

F. Transfer Learning for Intensity Classification

Our transfer learning intensity classification model was built off of the base model using our collected data as training data. A breakdown of this model's performance can be found in Table III. This model had an overall accuracy of 32%. A confusion matrix for this model can be seen in Figure 9.

TABLE III
PRECISION, RECALL, AND F1-SCORES FOR EACH POSSIBLE LABEL IN THE TRANSFER LEARNING BINARY CLASSIFICATION MODEL. THE OVERALL ACCURACY OF THIS MODEL WAS 32%

Label	Precision	Recall	F1-Score
No Pothole	42%	76%	54%
Small Pothole	23%	47%	31%
Large Pothole	27%	18%	22%
Deep Pothole	0%	0%	0%
Speed bump	43%	19%	26%

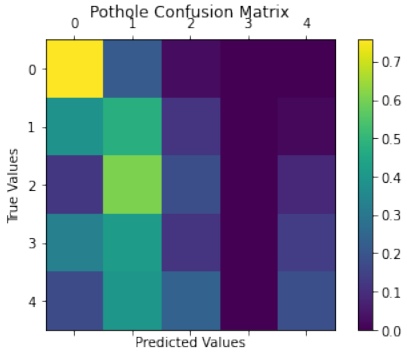

Fig. 9. The confusion matrix for our intensity transfer learning model

G. Realtime Pothole Detection

To evaluate the performance of our app and the machine learning models, we drove around Lowell while running the app. In this experiment, we used our best performing model, which was the binary base model. Our app was unable to detect potholes in real-time with high accuracy. If we were to shake the phones, a new pothole was plotted on our map, however, the app was unable to recognize potholes on the road very well. This is likely because our models are not very complex and did not learn the appropriate features that result in a pothole. In the following sections, we describe potential limitations and solutions to improve our models' accuracy.

Ultimately, we were able to view new pothole classifications on our heat map. We were also able to load in previous pothole intensity data from the phone into the current heatmap. An example of a heatmap in part of Lowell can be seen in Figure 10.

V. LIMITATIONS

One of the major limitations of pothole intensity classification is that the data collection is very subjective. Having better-defined metrics or automatic labeling would help reduce any effects of this subjective metric.

Additionally, our models are simple Keras models. If we replace these models with more complex models we should see an increase in our overall accuracy.

Another limitation of our work is that our sampling method is rather simple as well. Better sampling methods exist, such as by creating artificial data that is based on the original data. This would give us more data to train on, which could potentially increase our models' overall performance.

VI. FUTURE WORK

Future work can investigate improve the performance of the classification models through the use of more complex models. Additionally, we can improve our sampling method to improve the performance of our models.

Additionally, in the future, we can improve our app to provide notifications when the car is approaching a known pothole. These could be through text to speech or through other audio tones.

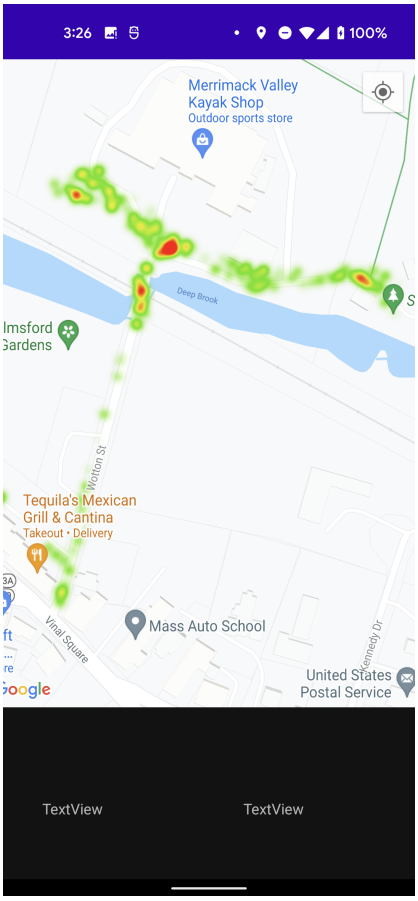

Fig. 10. A heatmap of pothole intensity in Lowell

Finally, in our future work, we should investigate better-defined metrics for intensity classification. In our current implementation, this is very subjective, through better-defined metrics we would enable more consistent pothole labeling for training data. Having consistent labels is very important for training accurate models.

VII. Conclusion

We have developed an app for pothole detection which was evaluated using two datasets and in a real-time scenario. This work has investigated the classification of potholes based on their intensity levels, this is a novel contribution of our work. While our real-world testing accuracy was surprisingly much lower than our testing accuracy we can improve our accuracy through more complex models, better sampling methods, and better-defined metrics for pothole intensity.

References

[1] L. Fix. (2018) Pothole dangers - drivers beware. [Online]. Available: https://www.forbes.com/sites/laurenfix/2018/04/16/pothole-dangers-drivers-beware/?sh=42cc1abb3f1a

[2] Why are potholes so bad every spring? [Online]. Available: https://www.sstire.com/tracy/why-are-potholes-so-bad-every-spring/

[3] B.-h. Kang and S.-i. Choi, "Pothole detection system using 2d lidar and camera," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2017, pp. 744–746.

[4] Y. Li, C. Papachristou, and D. Weyer, "Road pothole detection system based on stereo vision," in *NAECON 2018-IEEE National Aerospace and Electronics Conference*. IEEE, 2018, pp. 292–297.

[5] Y. Jo and S. Ryu, "Pothole detection system using a black-box camera," *Sensors*, vol. 15, no. 11, pp. 29 316–29 331, 2015.

[6] M. Hoffmann, M. Mock, and M. May, "Road-quality classification and bump detection with bicycle-mounted smartphones." in *UDM@ IJCAI*. Citeseer, 2013, p. 39.

[7] C. Wu, Z. Wang, S. Hu, J. Lepine, X. Na, D. Ainalis, and M. Stettler, "An automated machine-learning approach for road pothole detection using smartphone sensor data," *Sensors*, vol. 20, no. 19, p. 5564, 2020.

[8] K. Bansal, K. Mittal, G. Ahuja, A. Singh, and S. S. Gill, "Deepbus: Machine learning based real time pothole detection system for smart transportation using iot," *Internet Technology Letters*, vol. 3, no. 3, p. e156, 2020.