Engineering Projet

Road Quality Assessment

Team:
BALAKRISHNAN Sylvain
BONNAIL Julie

Alcatel·Lucent
Enterprise

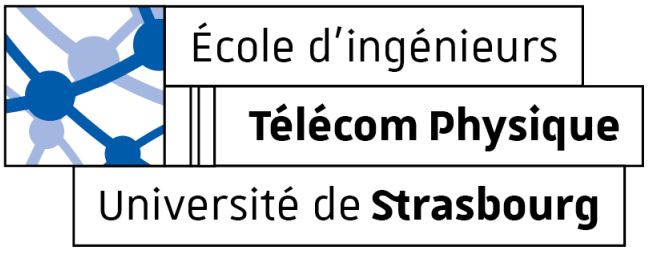

Table of contents

List of figures

1 Introduction

The road infrastructures are crucial when it comes to traveling. Whether it's daily commutes or occasional trips, millions of people drive their vehicles from point A to point B on the road. It's undeniable that the deteriorating state of roads has a considerable impact on the safety of drivers and passengers. An unexpected hole on a road can cause the driver to suddenly change direction or lose control of their vehicle. The effect of a poorly maintained road on a vehicle is generally overlooked, but it seems logical that potholes and bumps on a road are likely to damage vehicles, thereby reducing safety and increasing vehicle maintenance costs. On a larger scale, transportation can be slowed down by deteriorating roads, meaning the entire process of economic exchange operates at a slower pace, adversely affecting the economy of cities or even countries.

In this study, the AI will primarily be trained on acceleration data measured on vehicles. For the model to function correctly, it needs to be capable of detecting various deteriorations (bumps and obstacles, holes and cracks, as well as gravel), regardless of the type of vehicle from which the data originates. For the purposes of the study, two methods will be used to collect acceleration data. Firstly, by using an Arduino board and an Inertial Measurement Unit (IMU). Secondly, by utilizing the accelerometer contained within smartphones.

As a proof of concept, a smartphone application will be created and will demonstrate the effectiveness of this method for assessing road quality."

Certainly, here is a summarized version:

"Our engineering project began with in-depth research into road quality assessment methods. It was organized around data collection, AI model creation, and Android application development. Initial data collection involved Arduino-based recordings, while simultaneous Android app development improved precise data gathering. Refining the dataset through manual labeling was crucial for effective AI model training.

The project then progressed to exploring sequence classification approaches like LSTM, CNN, and anomaly detection. Following a team reorganization, the project's timeline was adjusted. A preliminary analysis of the data collected via a mobile application provided valuable insights.

This dataset creation involved recording raw data in text files with specific labels, and a data augmentation process was initiated to enrich the dataset. The CNN model implementation showed promising performance with 93% accuracy and a 17% loss, marking a significant milestone in our classification goal."

2 Android Application

In the pursuit of developing an Android application for the detection of road degradation, we encountered several challenges with the existing application. This section outlines the development process and improvements made to the application to meet our project requirements.

2.1 Previous Application

The original application was developed by Fabien, who is no longer part of the project team. This application was written in Kotlin, a language with which the team was not fully comfortable. One significant limitation was the difficulty in accessing the recorded data, as it required using Android Studio for data retrieval. Furthermore, the application lacked GPS data recording, which is crucial for our project's objectives. These limitations prompted the decision to start from scratch, redeveloping the application in Java and investing time in mastering Android Studio.

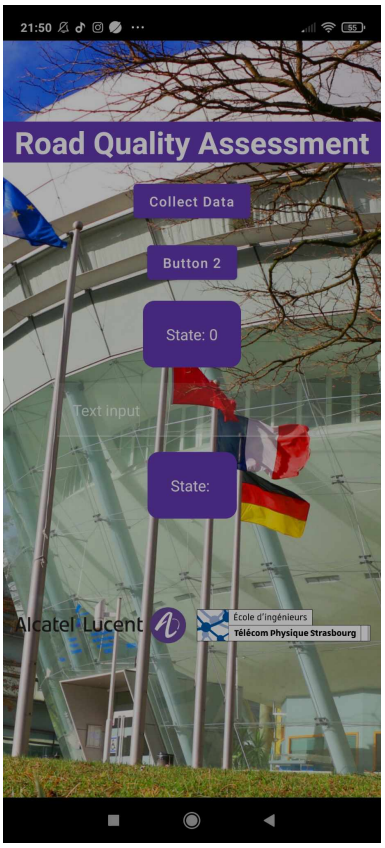

Figure 1: Previous Application

2.2 New Application Development

With extensive research and the aid of tutorials, we successfully developed a new Android application tailored to our project requirements. The application now records both accelerometer data and GPS coordinates, ensuring that the data is readily accessible within the device's memory.

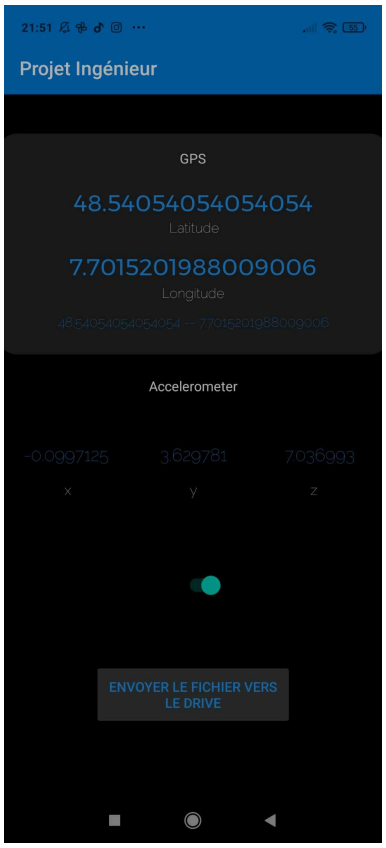

Figure 2: New Application

2.3 Integration with Google Drive

To facilitate the use of the recorded data with classification algorithms, we devised a solution to directly send the data file to Google Drive. This enables us to utilize the data within a Colab notebook or any other analysis tool that is compatible with Google Drive. The process involves integrating the application with the Google Drive API by using a Google API key.

2.4 How It Works

To use the application, follow these steps:

1. Log in with your Google account credentials.
2. Initiate data acquisition (the application allows pausing, and data is continuously displayed on the screen).
3. Once data acquisition is complete, click on the "Send File to Drive" button to easily retrieve the data in CSV format.

2.5 Data Format

The output file is named "MonFichierCSV.csv" and contains the following columns:

- `ax`: Accelerometer x-axis data
- `ay`: Accelerometer y-axis data

- `az`: Accelerometer z-axis data
- `GPS_lat`: Latitude data from GPS
- `GPS_long`: Longitude data from GPS

	A	B	C	D	E
1	AX	AY	AZ	GPS_Lat	GPS_Long
2	0.3698055	1.7565	8.467107		
3	0.626124	1.871484	8.3904505		
4	48.52252252	7.72598433	0.429693	1.598397	8.850387
5	48.52252252	7.72598433	-1.6759515	-0.006588	8.546158
6	48.52252252	7.72598433	1.593906	0.8869335	9.604969
7	1.9220895	1.320519	8.634791		
8	48.52252252	7.72598433	0.123069	0.8342325	8.383264
9	48.52252252	7.72598433	0.913584	0.4317885	8.50783
10	48.52252252	7.72598433	1.5627645	0.7719495	8.471897
11	48.52252252	7.72598433	3.4695826	0.865374	8.203602
12	2.4011896	0.6761295	8.3377495		
13	48.52252252	7.72598433	1.3998705	0.5276085	8.253907
14	48.52252252	7.72598433	0.812973	0.137142	8.847991
15	48.52252252	7.72598433	1.3232145	0.17547	9.015676
16	48.52252252	7.72598433	1.339983	0.2401485	9.070772
17	1.08606	0.3934605	8.812058		
18	48.52252252	7.72598433	0.142233	1.4043615	8.572509
19	48.52252252	7.72598433	0.9590985	1.5337185	8.713843
20	48.52252252	7.72598433	1.0501275	1.5433005	8.644374

Figure 3: Few lines of the CSV file

Limitations

At present, one limitation of the application is that, since it is not validated by Google, only the holder of the API key can save the data file to Google Drive.

3 Validity of Augmented Data

In the process of assessing the augmented data obtained from frequency sequences, it was imperative to ensure the integrity and reliability of the generated data. Several validation steps were conducted to verify the fidelity and accuracy of the augmented dataset.

Initially, the integrity of the augmented data was confirmed by comparing the original dataset with the manipulated dataset. This involved a series of normalization procedures applied to the sequences. Notably, the normalization involved raising the values to the power of 5 and scaling the data with respect to the absolute maximum value in the sequence. Subsequently, a comparative analysis was executed between the original and normalized datasets to ensure consistency and validity.

Moreover, the symmetry of the augmented data was evaluated by determining the barycenter of the normalized sequences. This process involved identifying the index representing the median value within the sequence. Additionally, to confirm symmetry, the dataset was divided into two halves, and their correspondence was examined to check if the sequence was symmetrical with respect to its barycenter. The assessment of the symmetry involved summing the absolute values of the first and second halves of the dataset. Comparing these sums was crucial in confirming the symmetry of the sequence. The comparison provided valuable insights into the symmetrical properties of the augmented data, which further reinforced its validity and reliability.

Here is the graphical representation of the normalization for the ax axis of one recording :

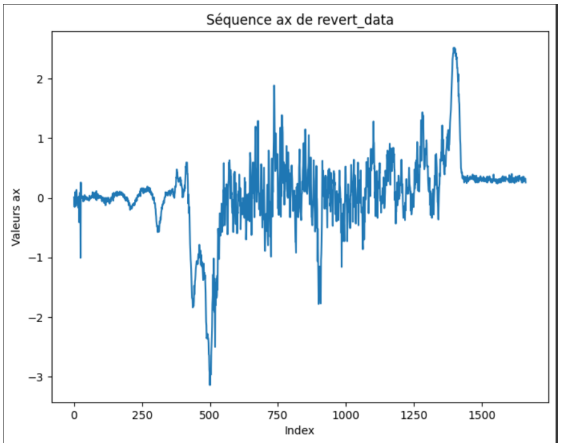

Figure 4: Initial reversed sequence of ax revert_data

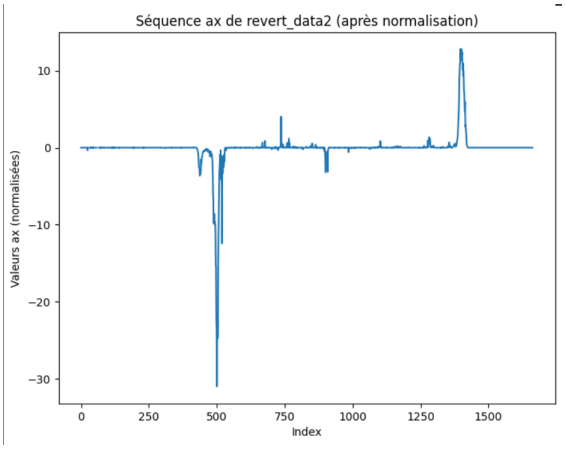

Figure 5: Normalized reversed sequence of ax revert_data

The verification process conducted on the augmented frequency sequences indicated that the generated data preserves essential characteristics of the original dataset. This procedure significantly enhances the confidence in the augmented data's applicability and utility for subsequent analyses and machine learning applications.

4 Artificial Intelligence: Multi-class Classification

In the previous project review, we introduced a Convolutional Neural Network (CNN) classification algorithm that demonstrated promising results for single-class classification. However, our project's goal extended beyond single-class classification; we aimed to implement a multi-class classification system.

In our case of road damage detection, the goal is to accurately identify various types of road deteriorations within a continuous sequence of accelerometric records. Each segment in the sequence represents a unique class corresponding to different types of road damage, such as potholes, cracks, bumps, or other forms of degradation.

Assigning multiple classes to one instance requires a sophisticated approach in the model architecture. It necessitates a deep learning model capable of understanding and classifying distinct segments within a continuous stream of data, allowing for the simultaneous identification of multiple classes.

4.1 Sliding Window: CNN - Sliding Window Approach

We aimed to implement this system, with the idea of utilizing a sliding window approach. Although discussions with our project jury emphasized that segmentation might be a more effective method, we were faced with considerable challenges in implementing a segmentation algorithm. Given that we had already initiated the development of a sliding window algorithm, we were determined to see it through. In this section, we will elucidate the concept of the sliding window algorithm, its implementation, and the results obtained.

4.1.1 Sliding Window Algorithm Concept

The sliding window algorithm operates by applying a classification model to a long sequence. Unlike traditional classification methods that analyze data as a whole, the sliding window approach segments the

data by moving a window of predefined size and step length across the sequence. As the window slides, the model performs classification within each window, effectively breaking the data into smaller sections.

4.1.2 Algorithm Implementation

Our journey with the sliding window approach began by concatenating five random basic data sequences and manually labeling the different sections within the concatenated data sequence.

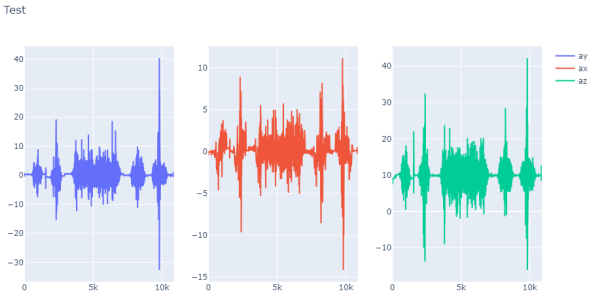

Figure 6: Concatenated Data Sequence

We assigned labels such as 1, 2, 1, 4, and 4 to these segments based on their characteristics.

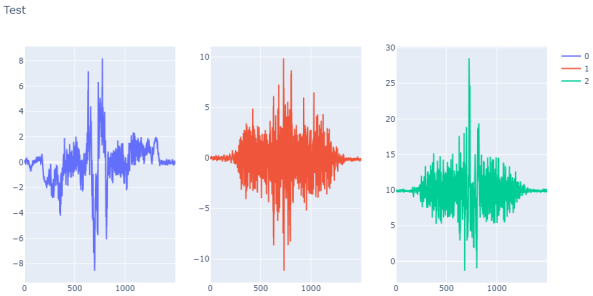

Figure 7: Cut of the concatenated data from data 7500 to 9000, here the predicted class is 4

To implement the sliding window algorithm, we utilized the CNN model, fine-tuning it to handle this segmentation task. Significant effort was invested in finding the optimal window size and step length to ensure accurate classification.

We also introduced a threshold in the predicted probabilities to determine whether a label should be assigned or not. This addition allowed us to avoid forcing the algorithm to assign a label in situations where one might not be present.

4.1.3 Results and Visualization

To evaluate the algorithm's performance, we generated both graphical and label-based results. The graphical representation showed the progression of the sliding window across the data sequence, while the labels assigned by the model were displayed as colored squares at specific locations along the sequence.

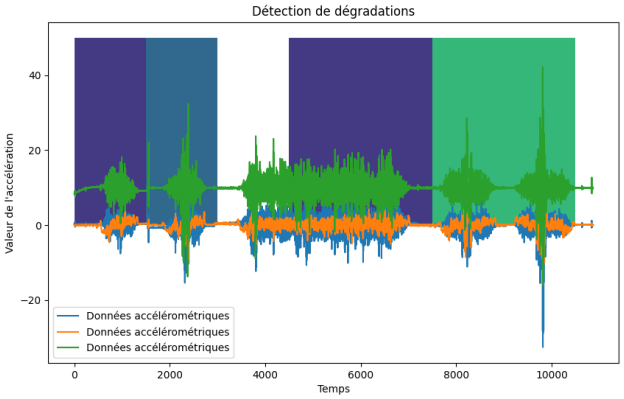

Figure 8: Visualisation of the Sliding Window Result

While the detected labels may not precisely align with the actual regions of road damage due to the size of the windows, the results were encouraging. The algorithm found the same sequence as the one found manually. It's worth noting that issues can arise with very long sequences, where a label may span multiple windows and, as a result, go undetected.

Although our initial results are promising, they are not without limitations, especially in the context of long sequences. As a result, we acknowledge that while the sliding window approach has merits, it may not be sufficient for our project's comprehensive objectives. Therefore, we remain committed to exploring other methodologies, particularly segmentation, to enhance the accuracy and efficiency of road damage classification in our project.

4.2 Segmentation: CNN

4.2.1 Algorithm Concept: Segmentation in Deep Learning

Segmentation in the context of deep learning involves the division of an input sequence into distinct segments, with each segment belonging to a specific class or category. Unlike traditional classification methods where an entire sequence is labeled as a single class, segmentation assigns multiple classes to different sections within the sequence.

The CNN employed in this context is designed to recognize patterns and features within segments of the data sequence, enabling the model to distinguish between different types of road damage. This multi-class classification capability is crucial in accurately identifying and labeling various segments in the sequence, contributing to more comprehensive road damage detection.

By segmenting the input sequence and assigning multiple classes to each segment, the deep learning model gains the ability to capture and classify diverse features present in the data, resulting in a more nuanced and detailed understanding of different types of road damage within the given sequence.

4.2.2 Algorithm Implementation

The model employed a sequence of convolutional layers, batch normalization, and ReLU activation functions for feature extraction and dimensionality reduction. The model's final output layer was structured for multi-class classification, enabling the identification of various types of road deteriorations.

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 input_9 (InputLayer)        [(None, 3, 9489)]

 conv1d_30 (Conv1D)          (None, 3, 64)

 batch_normalization_24 (Ba  (None, 3, 64)
 tchNormalization)

 re_lu_24 (ReLU)             (None, 3, 64)

 conv1d_31 (Conv1D)          (None, 3, 64)

 batch_normalization_25 (Ba  (None, 3, 64)
 tchNormalization)

 re_lu_25 (ReLU)             (None, 3, 64)

 conv1d_32 (Conv1D)          (None, 3, 64)

 batch_normalization_26 (Ba  (None, 3, 64)
 tchNormalization)

 re_lu_26 (ReLU)             (None, 3, 64)

 global_average_pooling1d_1  (None, 64)
  (GlobalAveragePooling1D)

 dense (Dense)               (None, 7)

=================================================================
Total params: 1847879 (7.05 MB)
Trainable params: 1847495 (7.05 MB)
Non-trainable params: 384 (1.50 KB)
```

Figure 9: Structure of the CNN Segmentation Model

The training phase involved compiling, fitting, and optimizing the model using encoded multi-class labels and categorical cross-entropy as the loss function. Training and validation data were properly formatted to comply with the input structure expected by the model.

The model training process encompassed several epochs, with continuous validation to ensure model accuracy and prevent overfitting. The trained model was then evaluated using a separate test dataset, providing insight into its accuracy and loss metrics.

4.2.3 Results

Overfitting Analysis

The model training process for the CNN-based segmentation encountered signs of overfitting. At the initial epochs, the training phase reflected a substantial improvement in accuracy, while the validation accuracy remained stagnant or even declined. This discrepancy in the accuracies between the training and validation data suggested the model had memorized the training data excessively and was not generalizing well to new, unseen data.

```
1/1 [==============================] - 0s 411ms/step - loss: 2.0202 - accuracy: 0.2400
Test accuracy: 0.23999999463558197
Test loss: 2.020185947418213
```

Figure 10: Evaluation of the CNN Segmentation Model

During the training period of 500 epochs, the model displayed a notable rise in accuracy, reaching a perfect accuracy score on the training data. However, this sharp improvement did not align with the validation set,

where accuracy remained relatively unchanged.

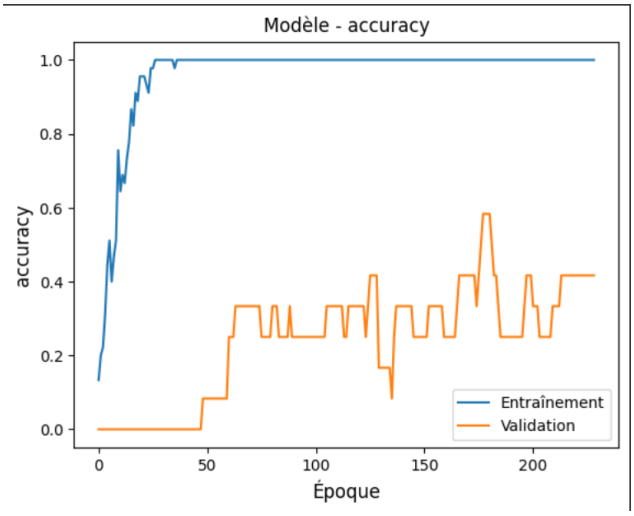

Figure 11: Graphical Visualization of CNN Segmentation Model Learning and Evaluation

The model's evaluation stage further confirmed this trend, where the accuracy on the test data remained considerably lower compared to the accuracy seen on the training data. Therefore, it can be concluded that the model experienced overfitting, implying the need for regularization techniques or model adjustments to prevent the learning process from focusing too heavily on the training data's specific details.

5 Conclusion

Our endeavor to develop AI solutions for road damage detection has led to valuable insights and achievements. We explored two primary AI methodologies: the sliding window approach and segmentation-based CNNs.

While the sliding window approach showed promise for smaller sequences, it posed limitations in processing longer data sequences.

The segmentation-based CNN model, designed to handle multi-class classification, encountered challenges related to overfitting. However, the model showcased potential for segmenting sequences and assigning multiple classes.

Our future focus involves refining the segmentation model, addressing overfitting challenges, and optimizing our approach to handle longer sequences. This pursuit aims to enhance the accuracy of road damage detection and streamline road quality assessment.

Simultaneously, our mobile application proved to be a milestone, facilitating direct data storage in Google Drive, streamlining data collection and integration with analysis tools.

In conclusion, our commitment to ongoing research and model refinement remains steadfast. Our aim is to contribute to safer road infrastructures and more efficient road quality assessment through innovative AI solutions and an effective mobile application.