Projet Ingénieur

Mesurer les Infrastructures Routières

Equipe:
BALAKRISHNAN Sylvain
BONNAIL Julie

Table des matières

Liste des figures

1 Introduction

Les infrastructures routières sont essentielles lorsqu'il s'agit de voyager. Qu'il s'agisse de déplacements quotidiens ou de voyages ponctuels, des millions de personnes conduisent leur véhicule sur la route pour aller d'un point A à un point B. Il est indéniable que l'état de détérioration des routes a un impact considérable sur la sécurité des conducteurs et des passagers. Un trou inattendu sur une route peut amener le conducteur à changer brusquement de direction ou à perdre le contrôle de son véhicule. L'effet d'une route mal entretenue sur un véhicule est généralement négligé, mais il semble logique que les trous et les bosses sur une route soient susceptibles d'endommager les véhicules, réduisant ainsi la sécurité et augmentant les coûts d'entretien du véhicule. À plus grande échelle, les transports peuvent être ralentis par des routes détériorées, ce qui signifie que l'ensemble du processus d'échanges économiques se déroule à un rythme plus lent, ce qui nuit à l'économie des villes ou même des pays. Enfin, la force militaire d'un pays peut être évaluée en fonction de l'état des réseaux routiers. En cas d'urgence, les forces militaires doivent se déplacer rapidement. Là encore, l'état des routes est un facteur clé.

L'objectif de ce projet est de développer une solution basée sur l'IA afin de faciliter l'entretien des routes. En formant une IA à reconnaître les dégradations sur une route, les cantonniers pourraient plus facilement entretenir les routes et ainsi améliorer la sécurité et l'expérience des usagers.
Dans cette étude, l'IA sera principalement entraînée sur les données d'accélération mesurées sur les véhicules. Pour fonctionner correctement, le modèle doit être capable de détecter diverses dégradations (bosses et obstacles, trous et fissures ainsi que graviers), quel que soit le type de véhicule d'où proviennent les données. Pour les besoins de l'étude, deux méthodes seront utilisées pour collecter les données d'accélération. Tout d'abord, en utilisant une carte Arduino et une *unité de mesure inertielle* (IMU). Dans un second temps, en utilisant l'accéléromètre contenu dans les smartphones.
Comme preuve de concept, une application pour smartphone sera créée et démontrera l'efficacité de cette méthode d'évaluation de la qualité des routes.

2 Rappels R1, R2 et R3

Les premières étapes de notre projet ingénierieur ont été consacrées à la compréhension de la problématique, des enjeux et des défis associés. Des recherches approfondies ont été menées pour explorer les méthodes d'évaluation de la qualité des routes, afin de recueillir des informations sur les solutions existantes et les méthodologies pertinentes.

L'organisation du projet s'est ensuite concrétisée grâce à la définition des objectifs des sprints, l'utilisation de "user stories" et l'évaluation de la complexité des tâches. Ces sprints ont servi à planifier l'avancement global du projet. Le projet a été divisé en trois parties clés : la collecte de données, la construction et l'entraînement d'une intelligence artificielle, et le développement d'une application Android.

Nous avons commencé par la collecte de données à l'aide d'un dispositif basé sur Arduino, enregistrant les premières données et les comparant avec d'autres sources telles que les données de M. Helbert et un jeu de données en ligne. Ces enregistrements ont jeté les bases de notre jeu de données initial.

En parallèle, le développement d'une application Android a été entamé pour permettre une collecte de données plus précise et efficace.

Dans le rapport 3, nous avons noté que grâce à l'expérience acquise lors de la première collecte de données et aux progrès réalisés dans le développement de l'application Android, nous avons pu construire un jeu de données plus robuste et de meilleure qualité. Cependant, pour entraîner efficacement un modèle d'intelligence artificielle, nous avons dû augmenter artificiellement la taille du jeu de données en utilisant différentes techniques.

Une phase importante de pré-traitement des données a été réalisée pour préparer les données à l'entraînement de l'IA, notamment l'automatisation de la création de séquences numériques et l'étiquetage manuel des données en fonction des informations recueillies lors des enregistrements. De plus, une phase ultérieure de pré-traitement a été identifiée pour uniformiser la longueur des séquences.

Pour la suite du projet, plusieurs approches de classification des séquences seront explorées, notamment des méthodes classiques telles que les approches d'apprentissage profond comme les réseaux LSTM, ou CNN ainsi que des techniques de détection d'anomalies. Une évaluation des modèles sera réalisée pour sélectionner la meilleure approche pour résoudre la problématique du projet. Le modèle sélectionné sera déployé, et des efforts d'amélioration continue seront entrepris pour affiner ses performances au fil du temps.

3 Réorganisation

Lors de la planification des différentes étapes du projet, incluant la spécification des "user stories" et la segmentation en sprints pour définir les objectifs intermédiaires entre les rapports successifs, notre équipe initiale était composée de trois membres. Toutefois, en raison de circonstances imprévues liées au parcours académique de Fabien ALLEMAND, qui a décidé de suivre un cheminement différent de ses études en troisième année à Telecom Physique Strasbourg, une réorganisation de notre programme initial s'est avérée nécessaire. Cette réorganisation a été entreprise afin d'adapter notre planification à deux personnes, tout en préservant au mieux la continuité et l'efficacité du projet.

Voici le nouveau diagramme de Gantt correspondant à nos besoins:

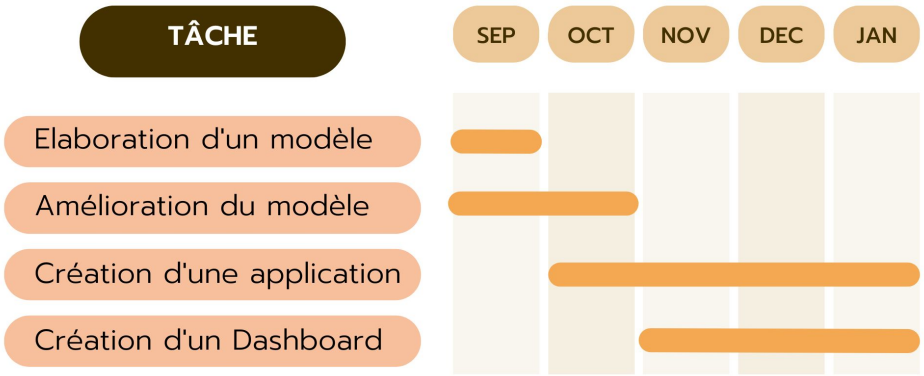

Figure 1: Diagramme de Gantt

Cette fois, le planning s'étend de septembre à janvier. Dans le but d'optimiser notre efficacité, nous avons structuré notre plan de manière à ce que deux tâches puissent être exécutées en parallèle à tout moment. Cette approche nous permet de répartir les responsabilités au sein de l'équipe, tout en maintenant la possibilité d'apporter un soutien mutuel si nécessaire pour la réalisation des tâches individuelles.

Plus spécifiquement, notre plan actuel prévoit la phase d'élaboration d'un modèle d'intelligence artificielle au cours du mois de septembre, suivie d'une période d'amélioration continue tout au long du mois d'octobre. Parallèlement à cette activité, à partir du mois d'octobre, nous nous engageons à poursuivre l'amélioration de l'application, en la rendant de plus en plus fonctionnelle et performante. Enfin de novembre à janvier nous créerons un dashboard, qui fournira une vue synthétique et visuelle des données permettant de faciliter la prise de décision.

4 Analyse de données

Avant d'aborder en détail la création du jeu de données et l'implémentation des modèles, il est essentiel de comprendre les données que nous avons recueillies à l'aide d'une application mobile. Ces données comprennent à la fois des informations d'accéléromètre et des données GPS, capturées lors du déplacement d'un robot

véhicule. Cette analyse préliminaire nous permettra d'obtenir un aperçu de la nature de nos données et de leur qualité.

Voici une image permettant de visualiser chaque axe individuellement sur un téléphone portable:

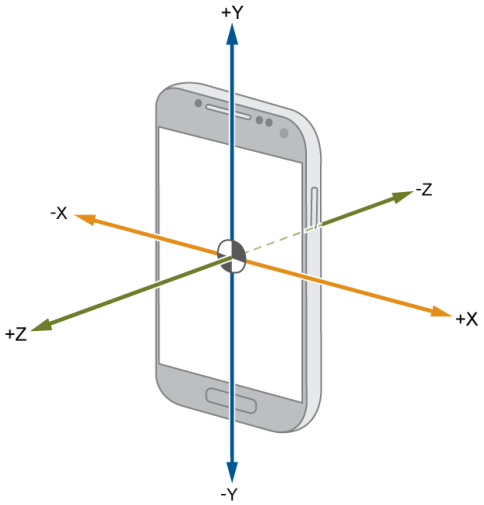

Figure 2: Orientation des axes sur un téléphone portable

4.1 Accéléromètre

Commençons par examiner les données d'accéléromètre, qui mesurent l'accélération dans les trois axes (X, Y, Z). Voici un aperçu de ces données :

```
acc_data.describe()
```

	ax	ay	az
count	347928.000000	347928.000000	347928.000000
mean	0.147932	-0.083627	9.915687
std	0.283331	0.263920	0.288943
min	-2.279280	-1.857900	6.694188
25%	0.028730	-0.162806	9.777920
50%	0.181959	-0.047884	9.902418
75%	0.277727	0.067038	10.036493
max	3.581726	1.331176	12.928689

Figure 3: Statistiques sommaires des données accélérométriques

Nous pouvons observer les statistiques descriptives telles que la moyenne, l'écart-type, les valeurs minimales et maximales pour chaque axe. Cela nous donne une idée de la variabilité des données d'accélération.

4.1.1 Graphiques d'Accélération

Pour mieux visualiser les données d'accélération, nous avons créé des graphiques. Voici un exemple de graphique montrant l'accélération dans les trois axes au fil du temps :

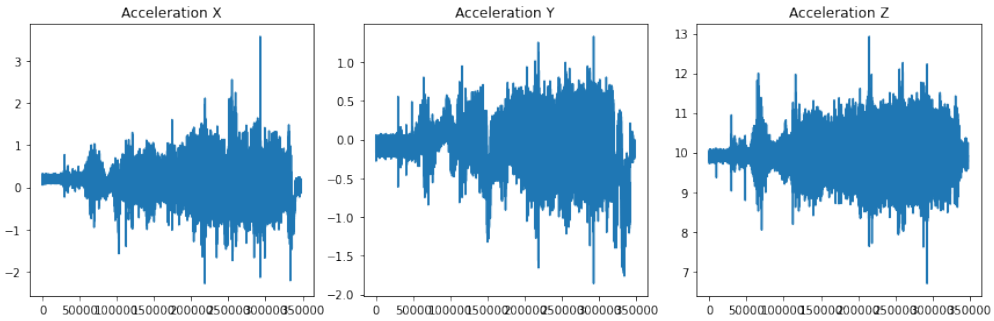

Figure 4: Graphique d'accélération au fil du temps

4.2 Données GPS

En parallèle, nous avons également collecté des données GPS pour suivre la position du robot véhicule. Voici un aperçu de ces données :

```
gps_data.describe()
```

	lat	long
count	127.000000	127.000000
mean	48.403793	7.446247
std	0.003757	0.003521
min	48.397830	7.438549
25%	48.400252	7.444659
50%	48.404109	7.447500
75%	48.407728	7.448993
max	48.408625	7.450468

Figure 5: Statistiques sommaires des données GPS

Nous pouvons voir des statistiques sommaires pour les données GPS, nous donnant une idée des valeurs minimales et maximales des coordonnées GPS.

4.3 Tests et Nettoyage des Données

Enfin, nous avons effectué des tests pour évaluer la qualité des données collectées. Nous avons détecté et identifié des fichiers de données potentiellement invalides en fonction du nombre de lignes. Tout fichier contenant moins de 1500 lignes a été signalé pour un examen ultérieur.

Cette analyse préliminaire des données est cruciale pour comprendre la nature de nos enregistrements et pour préparer le terrain en vue de la création du jeu de données et de l'implémentation des modèles.

5 Jeu de données

La création d'un jeu de données labellisé est une étape fondamentale dans le développement de modèles d'apprentissage automatique. Dans cette section, nous décrirons en détail comment nous avons constitué notre jeu de données, en nous inspirant d'une méthodologie issue d'un tutoriel de prétraitement de données séquentielles pour la classification.

5.1 Création du Jeu de Données

La création de notre jeu de données s'est articulée autour de deux aspects essentiels :

1. Enregistrement des Données : Tout d'abord, nous avons stocké les données brutes dans des fichiers au format texte. Chacun de ces fichiers représente une session distincte de collecte de données.

2. Attribution des Labels : Simultanément, nous avons créé un fichier de labels avec l'extension ".labels". Chaque fichier de données a été associé à un label spécifique, indiquant la nature de l'environnement ou de l'obstacle au moment de la collecte des données.

5.2 Consolidation des Données

Pour rassembler l'ensemble de nos données en un seul jeu de données cohérent, nous avons fusionné tous les enregistrements dans un fichier CSV unique. Ce fichier contient toutes les données d'entrée requises pour l'entraînement et l'évaluation de nos modèles.

Le jeu de données que nous utilisons dans le cadre de cette étude a été généré à l'aide d'une application Android spécialement conçue pour collecter des données d'accéléromètre. Les données les plus récentes ont été collectées jusqu'à la date du 14 avril 2023.

5.3 Création du Dataframe d'Accélération

La création du dataframe à partir des données brutes d'accélération s'est déroulée en trois étapes :

1. Lecture des Fichiers : Nous avons identifié tous les fichiers de données au format texte (.txt) présents dans le dossier de données.

2. Conversion en DataFrame : Chaque fichier a été lu et converti en un DataFrame Pandas. Les données d'accélération ont été enregistrées dans les colonnes "ax", "ay", et "az". Les colonnes "file" et "index" ont été ajoutées pour enregistrer respectivement le nom du fichier source et l'indice de l'échantillon.

3. Concaténation des DataFrames : Enfin, nous avons concaténé tous les DataFrames individuels pour obtenir un seul DataFrame global contenant l'ensemble des données d'accélération.

Le DataFrame résultant fournit une vue d'ensemble complète des données, accompagnée de statistiques descriptives telles que la moyenne, l'écart-type, les valeurs minimales et maximales pour chaque axe d'accélération (ax, ay, az).

5.4 Exemple de Données d'Accélération

Voici un extrait des données d'accélération présentes dans le dataset :

```
In [7]: df.head()
```

Out[7]:

	file	index	ax	ay	az
0	DATA43.txt	0	0.220267	0.172383	9.959879
1	DATA43.txt	1	0.181959	0.105345	10.046070
2	DATA43.txt	2	0.325611	0.220267	9.768343
3	DATA43.txt	3	0.277727	0.105345	9.787497
4	DATA43.txt	4	0.287304	0.220267	9.950302

Figure 6: Extrait des données d'accélérations dans le dataset

5.5 Création du Dataframe des Étiquettes

Pour chaque fichier de données d'accélération, nous avons généré un label correspondant. Les labels ont été créés selon une convention simple, où chaque fichier de données est associé à un label de la forme "DATAx.txt". Par exemple, le premier fichier est associé au label "DATA1.txt", le deuxième au label "DATA2.txt", et ainsi de suite. Ces labels ont été enregistrés dans un fichier séparé nommé "DATE.labels", où chaque ligne contient le nom du fichier et son label correspondant.

5.6 Structure du Jeu de Données

Notre jeu de données revêt une importance capitale dans nos expérimentations. Il renferme des données d'accélération couplées à des labels, formant ainsi une base solide pour l'apprentissage automatique supervisé. Ce jeu de données joue un rôle essentiel dans l'évaluation des performances de nos modèles dans la classification des obstacles et des types de surfaces.

6 Augmentation des Données

Conformément à ce qui avait été discuté lors de la troisième revue de notre projet, nous avons entamé le processus d'augmentation de données. Plus spécifiquement, nous avons identifié cinq types d'augmentation de données qui préservent les informations pertinentes pour nos futurs modèles, à savoir : le renversement de la séquence dans le temps, l'inversion de l'axe correspondant à la rotation du robot, l'ajout de bruit, l'ajout d'un signal porteur, et le découpage des séquences.

À ce stade, nous avons combiné les opérations de renversement dans le temps (R), d'inversion de l'axe de rotation du robot (I), d'ajout de bruit (N) et d'ajout d'un signal porteur (M) pour générer un ensemble de seize signaux distincts, tous dérivés d'un signal initial. Grâce à cette combinaison d'opérations, nous avons considérablement enrichi notre jeu de données accélèrométriques.

Il convient de noter que l'ordre dans lequel ces modifications ont été appliquées n'a pas d'incidence sur le résultat final, ce qui nous a permis de créer un total de 1312 données accélèrométriques uniques à partir des 82 enregistrements initiaux collectés par l'application Android. Ces données augmentées seront utilisées pour l'entraînement de notre modèle, renforçant ainsi sa robustesse et sa capacité à généraliser à partir d'un ensemble de données plus diversifié.

7 Intelligence Artificielle

7.1 CNN

Nous avons décidé d'implémenter en premier lieu un modèle CNN (Convolutional Neural Network). Ce type de modèle est adapté aux données séquentilles comme celles que nous possedons. Les CNN peuvent apprendre automatiquement des caractéristiques intéressante pour classifier les données. Ici ils peuvent identifier des motifs spatiaux et temporels. Pour nous aider à implémenter ce modèle, nous nous sommes aidé de keras et en particulier d'un article sur la documentation "Timeseries classification from scratch" [1] (expliquer ce qu'est keras) Grâce à cet exemple de code nous avons pu élaborer un programme qui entraine un modèle cnn. Tout d'abord il charge les données et les labels a utiliser. Ici nous avons 1312 fichiers à utiliser qui représentent 7 classes (speed_bump,crack,hole,tree_bumps,bumpy_tiles,hole+crack+bumps,speed_bump+hole). Il sépare les données en entrainement et test avec un pourcentage de 70% pour l'entrainement et 30% pour les test. Ici 918 fichiers seront utilisés pour l'entrainement. Enfin le modele cnn a comme architecture l'architecture suivante:

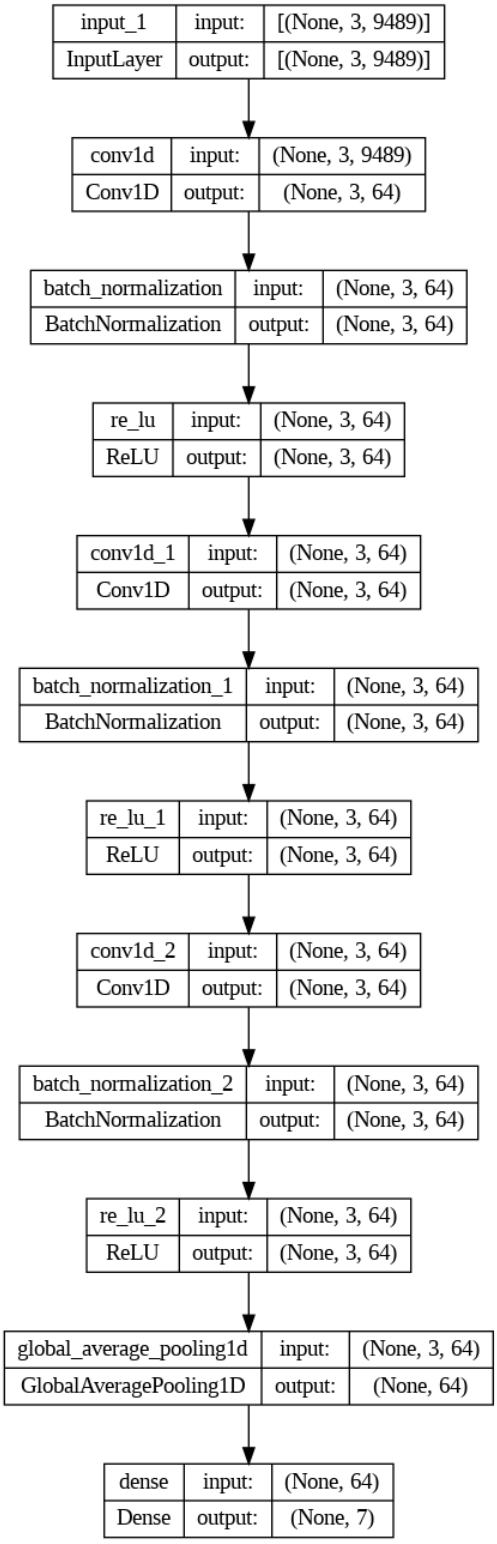

Figure 7: Architecture du modèle CNN

En utilisant trois couches de convolution avec des noyaux de taille 3 et des filtres de 64, nous pouvons capturer des motifs locaux significatifs dans les données. La normalisation par lots et la fonction d'activation ReLU sont appliquées après chaque couche de convolution pour stabiliser l'entraînement et introduire de la non-linéarité. Le Global Average Pooling est utilisé pour réduire la dimensionnalité des caractéristiques extraites tout en préservant leur pertinence. Enfin, la couche de sortie avec une activation softmax nous permet de classifier les dégradations en différentes catégories.

Pour l'entraînement de notre modèle, nous avons configuré plusieurs paramètres. Nous avons décidé d'effectuer 500 époques d'entraînement, chacune avec un lot de 32 échantillons. Pour garantir un bon apprentissage, nous avons utilisé trois callbacks. Le premier, "ModelCheckpoint," sauvegarde le modèle qui obtient la meilleure performance sur la perte de validation. Le deuxième, "ReduceLROnPlateau," ajuste automatiquement le taux d'apprentissage si la perte de validation ne s'améliore pas depuis 20 époques. Enfin, le troisième, "EarlyStopping," arrête l'entraînement si la perte de validation ne s'améliore pas pendant 50 époques consécutives. Notre modèle a été compilé avec l'optimiseur "adam" et la perte "sparse_categorical_crossentropy" pour la classification, avec une métrique de suivi de la précision "sparse_categorical_accuracy". L'entraînement a été effectué sur les données d'entraînement, en utilisant le batch size spécifié, avec 20% des données réservées à la validation. L'historique de l'entraînement est stocké dans la variable "history" pour une analyse ultérieure. Ces paramètres et callbacks contribuent à un entraînement efficace tout en évitant le surapprentissage grâce à une régularisation adaptée.

Grâce à cet entrainement, nous obtenons un modèle enregistré dans "best_model.h5" qui atteint une accuracy de 93% et une loss de 17% ce qui est très encourageant pour la suite.

```
Test accuracy 0.9263959527015686
Test loss 0.16705915331840515
```

Figure 8: Résultats de l'évaluation du modèle CNN

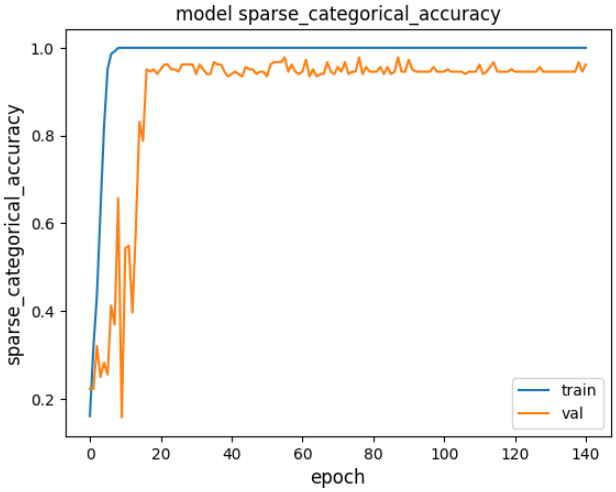

Figure 9: Evolution de l'accuracy en fonction des epochs

Voici un exemple d'un résultat donné par le modèle à partir d'une entrée donnée aléatoirement:

```
1/1 [==============================] - 0s 28ms/step
Nom de la donnée choisie: acc_data_469
Classe prédite: 2
Classe réelle: ['DATA29_ri_n.txt' 2]
```

Figure 10: Résultat d'une classification du modèle CNN

Ici la donnée en entrée est un fichier contenant les données accélérométriques correspondant à un trou (classe 2), qui a été renversé dans le temps et dans l'espace (ri) et auquel on a rajouté du bruit (n). Le modèle classifie bien cette entrée comme un trou (classe prédite 2).

7.2 Utilisation du modèle

Ce modèle apporte des résultats satisfaisant. Cependant il ne détecte qu'une classe à la fois, or une route peut contenir plusieurs déformations à la suite. De plus, ce modèle ne nous donne pas où la déformation apparait. Par exemple, pour tester les limites de ce modèle, nous avons concaténé 5 séquences de classe différentes.

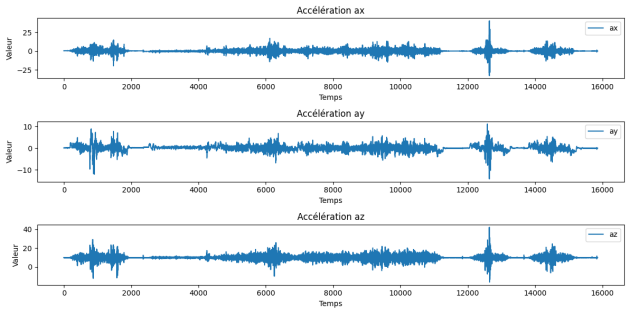

Figure 11: Graphique de la séquence créée à partir de 5 classes différentes

Voici ce que le programme renvoit si on lui donne cette nouvelle séquence :

```
1/1 [==============================] - 0s 152ms/step
Nom de la donnée choisie: concat
Classe prédite: 0
Classe réelle: ['concat.txt' [0,1,2,3,5]]
```

Figure 12: Classification du modèle CNN avec la séquence provenant de la concaténation

La classification définitive est actuellement associée à la première catégorie. Dans l'objectif d'améliorer les performances de notre classifieur, nous entreprendrons une série de démarches visant à identifier la catégorie au sein d'une fenêtre donnée, à itérer cette fenêtre de manière incrémentielle, et à déterminer l'emplacement spécifique où une déformation particulière se manifeste. Nous pourrons ensuite déterminer dans le fichier gps où a eu lieu la déformation en comparant avec nos résultats.

7.3 Implémentation du Modèle LSTM

Après avoir exploré l'architecture du modèle Convolutional Neural Network (CNN) pour la classification des obstacles, nous avons également développé un modèle basé sur Long Short-Term Memory (LSTM) pour cette tâche. Les réseaux LSTM sont particulièrement adaptés à la modélisation de séquences, ce qui les rend appropriés pour notre problème de classification basée sur des données séquentielles provenant de capteurs embarqués.

7.3.1 Prétraitement des Données

Avant de plonger dans les détails du modèle LSTM, nous avons commencé par prétraiter les données d'entrée et de sortie. Les données brutes se composent de séquences de mesures provenant de capteurs inertiels, accompagnées d'étiquettes de classe correspondantes. Les étapes de prétraitement comprenaient :

- Conversion des noms de fichiers en valeurs numériques pour un traitement plus facile.
- Encodage des étiquettes de classe en utilisant LabelEncoder pour les convertir en valeurs numériques.
- Création de séquences de données de longueur uniforme en ajoutant un rembourrage (padding) aux séquences plus courtes.

7.3.2 Création du Jeu de Données

Nous avons divisé nos données en ensembles d'entraînement et de test en utilisant la fonction `train_test_split` de scikit-learn. Cela nous a permis de disposer d'un ensemble d'entraînement pour former notre modèle LSTM et d'un ensemble de test pour évaluer ses performances.

7.3.3 Architecture du Modèle LSTM

Le modèle LSTM est construit en utilisant PyTorch. Nous nous sommes inspirés de deux vidéos trouvées sur internet pour réaliser ce modèle. [2] [3] Il comporte plusieurs couches, notamment une couche LSTM avec des paramètres tels que le nombre de fonctionnalités en entrée, le nombre de couches cachées, et un taux de dropout pour régulariser le modèle. La sortie de la couche LSTM est ensuite envoyée à une couche entièrement connectée (Linear) pour la classification finale.

Le modèle est défini dans la classe `SequenceModel`, qui est encapsulée dans la classe `ObstaclePredictor`. Cette dernière est responsable de la formation, de la validation et des prédictions du modèle LSTM.

7.3.4 Fonction de Perte et Métriques d'Évaluation

La fonction de perte utilisée pour l'apprentissage est la CrossEntropyLoss, qui est adaptée à la classification multiclasse. Pour évaluer les performances du modèle, nous avons utilisé la métrique de précision pondérée, calculée à l'aide de la bibliothèque TorchMetrics.

7.3.5 Entraînement et Évaluation du Modèle

Nous avons essayé de lancer l'entraînement du modèle LSTM sur plusieurs époques à l'aide du module PyTorch Lightning. Cependant une erreur technique nous a empêcher d'évaluer les performances du modèle.

L'implémentation du modèle LSTM pour la classification des obstacles basée sur des données séquentielles s'est avérée être un défi complexe et a rencontré des difficultés lors de l'exécution. Malheureusement, le modèle n'a pas pu être correctement formé en raison de certaines erreurs techniques.

L'une des principales raisons de l'échec de l'entraînement du modèle LSTM réside dans des problèmes liés à la configuration de l'environnement d'exécution, notamment des incompatibilités potentielles entre les bibliothèques utilisées et les dépendances matérielles. Des erreurs liées à TensorFlow et à l'absence de certaines bibliothèques nécessaires ont été rencontrées, ce qui a empêché l'utilisation efficace du GPU.

En conséquence, les performances du modèle LSTM n'ont pas pu être évaluées sur l'ensemble de test, et aucune conclusion ne peut être tirée quant à son efficacité dans la classification des obstacles.

Ces problèmes techniques sont actuellement en cours de résolution, et des efforts supplémentaires seront déployés pour résoudre ces problèmes et évaluer correctement le modèle LSTM à l'avenir. Néanmoins, ces obstacles techniques soulignent l'importance de la préparation de l'environnement de développement et de la gestion des dépendances pour garantir le bon fonctionnement des modèles d'apprentissage automatique.

Dans la prochaine section, nous analyserons en détail les résultats de nos modèles CNN et LSTM, ce qui nous permettra de tirer des conclusions importantes pour notre application de détection d'obstacles.

8 Conclusion

Dans le cadre de ce projet de mesure des infrastructures routières, nous avons exploré différentes approches, y compris l'utilisation de modèles d'apprentissage automatique tels que les CNN et les LSTM. Les résultats obtenus avec notre modèle CNN sont très encourageants, avec une précision de 93% dans la classification des obstacles. Cependant, des obstacles techniques ont entravé l'entraînement complet du modèle LSTM. Ces problèmes techniques feront l'objet d'une résolution ultérieure afin de permettre une évaluation complète de ce modèle.

En fin de compte, ce projet représente une étape cruciale vers le développement d'un système de détection d'obstacles sur les infrastructures routières, ce qui pourrait avoir un impact significatif sur la sécurité routière et l'entretien des routes. Les découvertes et les modèles développés ici serviront de base pour des travaux futurs visant à améliorer encore la précision et l'efficacité de notre système.

Bibliographie

[1] Hassan ISMAIL FAWAZ. Timeseries classification from scratch. https://keras.io/examples/timeseries/timeseries_classification_from_scratch/. Accessed: 2023.

[2] Venelin Valkov. Multivariate time series classification tutorial with lstm in pytorch, pytorch lightning and python. https://www.youtube.com/watch?v=PCgrgHgy26c. Accessed: 2023.

[3] Venelin Valkov. Multivariate time series data preprocessing with pandas in python. https://www.youtube.com/watch?v=jR0phoeXjrc. Accessed: 2023.