Predicting Ego-Bicycle Trajectory: An LSTM-based Approach Using Camera and IMU

Jelte Koornstra, University of Twente, The Netherlands

In order to make bicycles with driver assistance systems a reality, a suitable trajectory prediction must be developed. This research will investigate sensor and trajectory prediction models that are suitable for the task, and develop such a model. In addition, two datasets consisting of sensor data collected with bicyles are created to train the model. Two different types of Long Short Term Memory (LSTM) trajectory prediction models are evaluated, one using a Convolutional Neural Network LSTM hybrid architecture, and one using only LSTM by itself. The performance of these different models is then evaluated and compared.

Additional Key Words and Phrases smart-bicycle, trajectory prediction, lstm, machine learning, sensors, internet of things

1 INTRODUCTION

Cars are getting smarter and smarter, many modern cars already use systems to assist the driver. These so-called Advanced Driver Assistance Systems are able to use sensors to detect driver mistakes and potential obstacles. While there has been a big push towards smarter cars, the same cannot really be said about bicycles. Modern e-bikes might have some safety features [8], but they are not equipped with sensors and driver assistance systems to the same extent modern cars are. These systems could still be beneficial to cyclists to assist them in avoiding collisions, and potentially correct some of their mistakes during cycling.

In order to make these so-called 'smart bicycles' possible, several subsystems would have to be developed. One of the subsystems that would be required for this is trajectory prediction. Trajectory prediction is a computational technique used to anticipate and forecast the future path or movement of an object or entity based on its past behavior and surrounding context. It involves analyzing patterns, historical data, and relevant factors such as velocity, acceleration, and environmental conditions to estimate the likely trajectory of an object over a specific period of time. Through applying mathematical models and/or machine learning algorithms, trajectory prediction aims to provide valuable insights for a wide range of applications, including autonomous navigation systems, object tracking, collision avoidance, and predictive analytics.
Current state of the art trajectory prediction methods used in cars often rely on data provided by expensive sensors, such as 3D Lidar. These advanced sensors use light pulses to accurately measure the distance of objects around the vehicle.

TScIT 37, July 7, 2023, Enschede, The Netherlands

Although Lidar sensors are impressive, they are also costly, with prices ranging from close to a thousand to several thousand euros. Moreover, many Lidar sensors are not suitable for smaller form factors. In light of these challenges, this research adopts an alternative approach by utilizing more affordable and compact sensors: a monocular camera and an IMU connected to a Raspberry Pi.

Currently there does not exist a publicly available dataset with sensor data collected by bicycles. Bicycle movement may be more unpredictable and different from that of automobiles or drones, which have been used to create the most popular datasets in the field. Therefore, two datasets will be created, consisting of sensor data collected by bicycles, to enhance the training of the trajectory prediction model.

1.1 Research Question

The objective of this research is to develop a bicycle trajectory prediction model using affordable sensors. We address this objective with the following research question:
How accurately can a smart-bicycle predict its trajectory using affordable sensors?

In order to answer this question, the following sub-questions should be answered first:

(1) Which sensors should be used?
(2) What is a suitable trajectory prediction method given the time and cost limitations?
(3) How accurately can this method predict the trajectory of the bicycle?

2 RELATED WORKS

In this section, we take a look at some of the commonly used trajectory prediction methods in the industry. There are many different approaches to do this, however they can generally be classified into four methods: physics-based, classic machine learning based, deep learning based and reinforcement learning based methods [3]. It is important to note that although there are many trajectory prediction methods, there is no definitive state of the art when it comes to bicycles specifically. The methods discussed in this section are generally tailored to automobiles.

Physics-based models generally require fewer computational resources to function than machine learning and deep learning based approaches. This characteristic makes them an attractive

choice for devices with limited computational resources, including smart bicycles. One of the physics-based methods is the family of Kalman Filtering methods. They are able to model uncertainty about the vehicle's state unlike several other physics-based approaches that assume the vehicle state is known. Lefkopoulous et al. [2021] propose an Interacting Multiple Model Kalman Filter for trajectory prediction. Their method not only considers the motion of the vehicle itself, but it also takes the intention of the driver into account. Furthermore, their model also considers other traffic participants. Their model was comparable to state-of-the-art models for predictions up to 5 seconds while being easier to train than deep learning-based models [5].

Although physics-based approaches do have an advantage in computational efficiency, they may not always perform as well in more complex scenarios as well as deep-learning or reinforcement-learning based approaches which can leverage large amounts of training data better.

For trajectory prediction in particular, being able to capture spatial and temporal dependencies is a valuable property. Deep learning-based models are especially adept at this. Compared to most physics based and classic machine learning based models, they function better in more complex scenes [3]. They possess the capability to consider elements of physics, road conditions, and interactions among various participants in traffic. Convolutional neural networks (CNN) are quite effective at handling spatial data, which may be valuable for trajectory predictions methods that take environmental factors into consideration. CNNs leverage convolutional layers with filters that pass over regions of the image, enabling them to recognize different groups of pixels. Though, most of the CNN based trajectory prediction methods rely on bird-eye view image data, which cannot be easily obtained on a smart-bicycle.

Another branch of deep learning-based models is the family of recurrent neural networks. Recurrent neural networks are sequential models that retain and utilize information from past time steps, incorporating hidden states along with the input to compute the output. However, one limitation of recurrent neural networks is that they are prone to the exploding and vanishing gradient problems due to the way the gradient is repeatedly multiplied by the weights of the recurrent connections. Long Short-Term Memory and Gated Recurrent Unit networks mitigate that problem. LSTM networks solve this problem by introducing a specialized memory cell that can selectively retain or forget information over multiple time steps, whereas GRU networks do this by using update and reset gates that dictate how much of the previous hidden state should be retained or forgotten, and how much new information should be added to the current hidden state. The LSTM model proposed by Hyeon Park et al. uses a decoder-encoder architecture that learns the pattern of the past trajectory using an LSTM based encoder, and then generates a future trajectory using a decoder based on the output of the encoder [5]. This model was able to outperform standard LSTM models and basic Kalman filter based models by a significant margin, achieving a mean absolute error of 0.93 on a prediction horizon of two seconds. Sun et al. [2020] were able to design an LSTM based trajectory prediction method that only uses data from a single monocular camera to make predictions [9].
The inputs to their network were a set of images obtained by the camera, and the output was a trajectory mask that could be overlaid onto the current image.

There also exist trajectory prediction models that use a combination of the aforementioned techniques. Xie et al. [2020] proposed a sequential model that fuses a CNN network and an LSTM network [10]. The vehicle data used in this research is from the NGSIM 101 dataset, which contains accurate vehicle positions collected on a highway. They use the box plot method to eliminate outliers from the vehicle data. The inputs to the model are the vehicle speed, left lane distance and right lane distance of the vehicles. The convolutional neural network performs a one-dimensional convolution on the input data to extract features, which are then passed to a max-pooling layer to reduce the data size. The resulting output is then passed to the LSTM network. Their CNN-LSTM model performed better than individual GRU, LSTM and CNN networks while also being faster.

Ma et al. [2020] propose a similar CNN-LSTM model for Aircraft 4D trajectory prediction. A one-dimensional convolution is leveraged to derive features from the spatial dimension of the trajectory, while Long Short-Term Memory (LSTM) is employed to capture features from the temporal dimension of the trajectory. This approach results in a 21.62% lower prediction error compared to a LSTM model by itself [6].

3 METHODOLOGY

3.1 Designing the Hardware Setup

When deciding which sensors to use for data collection on the bicycle, we need to consider the cost of the sensors. Most state-of-the-art prediction models rely on accurate 3D object maps obtained through Lidar sensors. However, for smart bicycles, we need to use more affordable sensors to keep costs in check. The core of our system is the Raspberry Pi 4B provided by the university. We opt for a monocular camera, such as the Raspberry Pi NoIR V2 camera, as some trajectory prediction models, like the one described by Sun et al. [7], can effectively work with it. In addition to being a rather inexpensive camera, this camera is also able to operate in low light conditions due to its ability to capture infrared light. However, relying solely on camera data might not be reliable in suboptimal weather or lighting conditions. To gather additional information, we consider accelerometer and gyroscope sensors. The accelerometer detects changes in velocity, while the gyroscope identifies changes in the bicycle's orientation. The university provides us with the Arduino Nano33 BLE Sense, equipped with a 6-axis Inertial Measurement Unit (IMU) that includes accelerometer and gyroscope data. The combined cost of these sensors is approximately €80, making the setup relatively affordable. Nonetheless, we still need another sensor to establish a reliable ground truth for the dataset. While GPS initially seems viable, it tends to be inaccurate, with deviations of several meters. For higher accuracy, GNSS-RTK sensors are ideal, achieving centimeter-level precision. Therefore, we use the Waveshare ZED-F9P GPS-RTK HAT, which offers the

desired accuracy and has the quickest delivery time of the eligible sensors. The camera is mounted on the front of the bicycle to have a clear view, the IMU is also mounted on the front next to the camera to ensure that its data does not deviate much from the camera data, and the GNSS sensor is mounted on the back of the bicycle. The Raspberry Pi is powered by a powerbank attached to the bicycle.

Fig. 1. The camera with the Raspberry Pi on the front of the bicycle

Fig. 2. The GNSS-RTK sensor on the back of the bicycle

3.2 Creating the Datasets

For the initial datasets, we choose two roads on the university campus. One is 'De Zul' behind Ravelijn, and the other one is a smaller bicycle lane parallel to this road, named 'Oude Drienerloweg'. These roads are chosen because they are relatively straight and do not have a high traffic density, which allows us to assess whether the model performs well with a limited number of external factors. We use Python code to obtain the data from the sensors, which is run by using TeamViewer on an external computer to access the Raspberry Pi. Several python libraries such as pyrtcm, opencv, picamera2 and pynmea are used to read and collect the data from the sensors, and for the GNSS-RTK sensor it is necessary to contact a valid NTRIP caster in order for it to yield data. The APEL00NLD0 NTRIP caster provided by Kadaster is chosen for this purpose, as it is relatively close to Enschede.

During the data collection process, the cyclist tries to maintain a constant speed of 13 km/h to further reduce the complexity of these first two datasets. The data is collected at around 5:00 PM in good lighting conditions in moderate traffic density. The grayscale images are collected at a resolution of 640x480 pixels and a frequency of 75 images per second, the IMU data at 119Hz. However, the GNSS data is only collected at 1Hz. This is not a technical limitation of the sensor, rather the NTRIP caster only sends a single measurement per second. In total, approximately 8000 image and IMU datapoints have been collected across the two datasets.

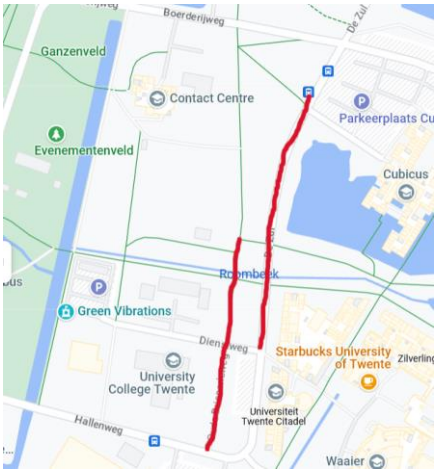

Fig. 3. The roads that are chosen for data collection

3.3 Preprocessing the Data

The data that is collected in the dataset must undergo several preprocessing steps before it is usable for the trajectory prediction model. Firstly, copies of the original images are downscaled to 224 by 224 pixels to reduce the complexity of the model later, and to increase training speed. Additionally, the IMU data values are normalized to be within a [-1,1] interval. The GNSS data is in decimal latitude and longitude degrees, however that is not representative of a 2D or 3D environment where trajectory prediction models are generally trained on. Therefore, the GNSS coordinates are converted to a cartesian coordinate system. This is done using pyproj which is a python library capable of performing cartographic transformations.

The WGS84 decimal degree latitude and longitude coordinates are accurately transformed to X and Y coordinates on the Mercator projection using pyproj's transform function. This pair of X and Y coordinates is used as the ground truth for the model. Height is not included in the dataset since it cannot not be determined with the sensors we have at our disposal, but the two roads the data is collected on are flat regardless, thus height should remain relatively constant. Furthermore, since the GNSS datapoints have been collected at a frequency of 1Hz, the missing datapoints need to be interpolated. Linear interpolation is performed on the dataset since the speed was kept constant during data collection. Moreover, since the IMU data was collected at a higher frequency than the images, only the IMU datapoints closest to the image captures are used.

As an additional preprocessing step, the images are fed to a pre-trained MobileNetV3Small convolutional neural network (CNN). This model is capable of recognizing objects and extracting features from images. Several features such as edges representing roadmarks and objects such as traffic signs may be important for the trajectory prediction model to know. This particular CNN is tuned for performance to be usable on smartphones, making it more suitable for real-time processing on low power systems such as the Raspberry Pi as well. The feature maps obtained by MobileNetV3 are stored on the disk to train the trajectory prediction model later on.

3.4 Choosing the Model

As discussed in the previous section, there are various approaches for a trajectory prediction model. Due to the temporal nature of the data, Recurrent Neural Networks are an intuitive approach. However, in traditional Recurrent Neural Networks, the exploding and vanishing gradient problems become more pronounced over long sequences. Long Short-Term Memory networks avoid this problem through the use of memory cells and gated units. LSTM still retains the strengths of recurrent neural networks, as it is able to capture long-term dependencies and patterns in sequential data. Trajectories often exhibit complex temporal dynamics, where past positions and velocities significantly influence future trajectory paths. LSTM's recurrent structure allows it to remember and utilize information from previous time steps, enabling it to model and predict these dependencies accurately.

Furthermore, LSTM excels at handling variable-length input sequences, which is crucial for trajectory prediction. Trajectories can have varying lengths depending on the observed time span, and traditional methods struggle to handle this variability. LSTM's inherent flexibility in processing sequences of different lengths allows it to handle such scenarios seamlessly, accommodating the diverse temporal nature of trajectory data.

Although LSTM models are well suited for capturing temporal dependencies, they are less adept at handling the spatial dependencies present in grid-like data structures such as images. However, these spatial dependencies are still important in the context of trajectory prediction since the trajectory of the bicycle could also be affected by the environment. The information about the environment needs to be extracted from images since we are not able to work with 3D models of the environment due to our sensor limitations. For an LSTM model this might be a difficult task.

Convolutional Neural Networks on the other hand are quite good at handling spatial data and are often used to process images. Therefore, we opt for a hybrid approach using both CNN and LSTM to try to combine the best of both by first passing the image to a CNN (MobileNetV3) and using its feature maps as input for the LSTM model, somewhat similar to what Xie et al. [2020] have done.

Additionally, we try an alternative approach without the CNN network, using only the LSTM network to find out whether the CNN actually improves the results.

4 EXPERIMENTAL SETUP

4.1 Dataset

We use the datasets that are created for the purpose of training the model. The two datasets contain data obtained from cycling on 'De Zul' and 'Oude Drienerloweg', consisting of approximately 8000 grayscale images and IMU readings. The interpolated and converted GNSS coordinates are used as a ground truth. The 'EuRoC MAV dataset' by Burri et al. [2016] will be used as an additional dataset to test the model on.

4.2 Memory Optimization

Due to the relatively large size of the image data, it becomes impractical to load this data into memory simultaneously. As a result, a preprocessing step is required to store the image features on the disk using a Hierarchical Data Format (HDF). This format allows for efficient storage and retrieval of large datasets. During the subsequent training phase of the model, the stored data is loaded from the disk as needed.

4.3 Validation Metrics

The dataset is split in three sets, one for training, one for validation and one for testing. 70% of the data was used for training, 15% for validation and the remaining 15% for testing. The metric used to evaluate the performance of the model is Mean Squared Error (MSE).

Mean Squared Error (MSE) is often used as a loss function for trajectory prediction models, including those that use Long Short-Term Memory (LSTM) networks, because it quantitatively measures the average squared deviations between the predicted and actual trajectory values. This squaring ensures that all errors are positive and that larger errors have more impact, which makes the model strive to reduce larger errors. This property aligns well with trajectory prediction tasks, where reducing larger deviations—i.e., differences between the predicted and actual paths—is typically more important than minimizing smaller ones. Furthermore, MSE is differentiable, which is a crucial property

for optimization algorithms such as gradient descent used during model training.

4.4 Implementation Details

The LSTM model is implemented with the Keras python library as a stateless LSTM sequential model. The LSTM layer consists of 64 neurons. Following this LSTM layer, there is a FC layer of 32 neurons, a dropout of 0.5 is applied to this layer meaning that there is a 50% chance that any given neuron will be dropped during training. This can help against overfitting by reducing co-adaptations of neurons, discouraging scenarios where a particular feature is only useful in the context of other features. Finally, the output layer is a fully connected layer with 2 neurons representing the X and Y coordinates of the position. Thus, the LSTM model has three layers in total: the LSTM layer, the FC layer and the output layer.

The sequence length of the LSTM model is 75 timesteps, representing one second of data. The output of the model is also a sequence of 75 X and Y coordinates, meaning the model uses the past 75 measurements to predict the future 75 positions. The batch size used for training the model is 8. The batch size is chosen to be relatively small as it has been shown that using large batches may lead to worse generalization [4].

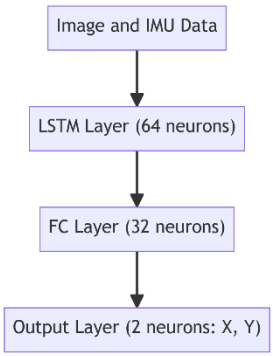

Fig. 4. The LSTM model architecture

We test two variations of the model, one is using the image data directly as the input for the LSTM, and the other one is a CNN-LSTM hybrid model which uses the feature maps obtained from MobileNetV3.

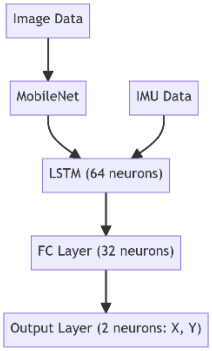

Fig. 5. The CNN-LSTM model architecture

For the first implementation, we take the grayscale image, convert it to a Numpy array and then normalize the values. This array is then flattened and concatenated with the normalized IMU readings before it is used as input for the LSTM. The second implementation uses the MobileNetV3 CNN. Center cropping is used to crop a 224x224 region of the center of the image, this image is then converted to a torch tensor, which is then used as input for the MobileNetV3Small model. MobileNet then outputs a feature map tensor, which is flattened and concatenated with the normalized IMU readings before it is used in the LSTM.

Hence, the input for the model at each timestep is the normalized IMU data and the processed image features. The model uses the X and Y coordinates obtained during the preprocessing as the ground truth. For compiling the model, the Adam optimizer is utilized using the Mean Squared Error loss function with a learning rate of 0.001.

4.5 MobileNetV3

MobileNetV3 is the CNN used throughout this research, it is designed to be used on mobile phone CPUs making it more suitable than some of the other CNNs in terms of hardware requirements. The MobileNetV3Small model was able to achieve state of the art performance at lower performance requirements than comparable models. It is 3.2% more accurate than its predecessor MobileNetV2 at image classification at 20% lower latency [2]. MobileNetV3Small was also able to achieve 63.38 mIOU (mean Intersection Over Union) on the Cityscapes dataset, indicating it is quite good at pixel-level segmentation tasks.

The pre-trained model of MobileNetV3Small that has been used in this research is trained on the ImageNet dataset. This dataset contains 1000 classes including some traffic related ones such as car mirrors, car wheels, bicycles and traffic lights. Though, these classes are mainly used for object recognition problems, which might not be useful when it comes to trajectory prediction.

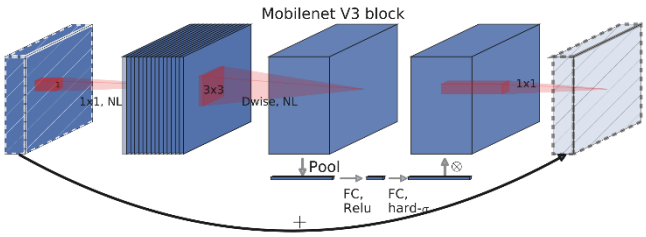

Fig. 7. MobileNetV3 Block [2]

MobileNetV3 is constructed using a stack of repeated blocks. These blocks are improved versions of the Inverted Residual and Linear Bottleneck structures introduced in MobileNetV2, they retain the same general structure but additionally Squeeze-and-Excite is applied in the residual layer. Squeeze and Excite modules perform 'channel-wise' attention, recalibrating the feature maps obtained from convolution operations by explicitly modelling the interdependencies between channels. This allows the model to emphasize meaningful features and suppress less useful ones, improving the overall accuracy.

The initial layers of the network perform low-level feature extraction, identifying simple patterns like edges and textures. Whereas the deeper layers of the network are able to identify higher level features and classes. Changes in the feature maps obtained by MobileNetV3 across different frames may be useful for the LSTM model to track the movement of the bicycle. However, the object recognition-oriented classes present in the pretrained model are not quite tuned for traffic-specific object recognition, so the model may not be able to detect the high-level features well in our case.

5 RESULTS

5.1 Dataset 1

This dataset is the obtained from 'Oude Drienerloweg'. In this section we first examine the metrics obtained after 100 epochs of training the CNN-LSTM hybrid model. Then we compare this to the alternative approach which only uses the LSTM. We also show the graphs that show the predicted and the actual trajectory of the test set. The X and Y units shown are in meters, the prediction horizon is one second.

CNN-LSTM Approach:

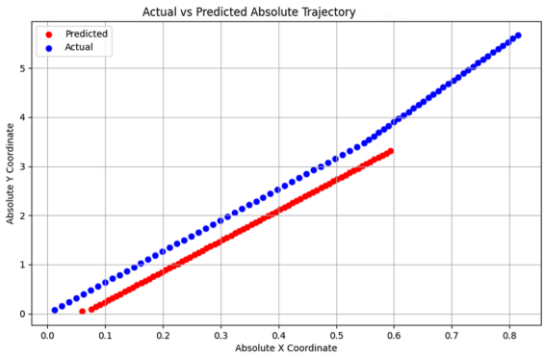

Fig. 8. The predicted trajectory on dataset 1 using the CNN-LSTM approach

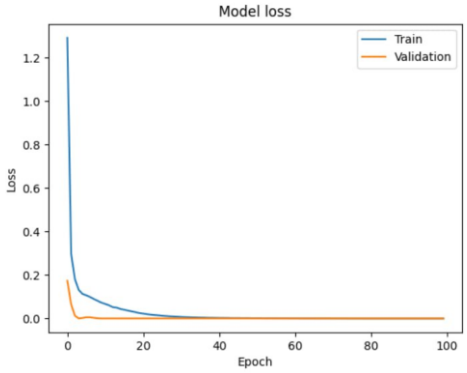

Fig 9. The MSE loss of the model using the CNN approach on the first dataset

The CNN-LSTM model achieves an MSE of 0.03 on the test set of this dataset.

LSTM Approach:

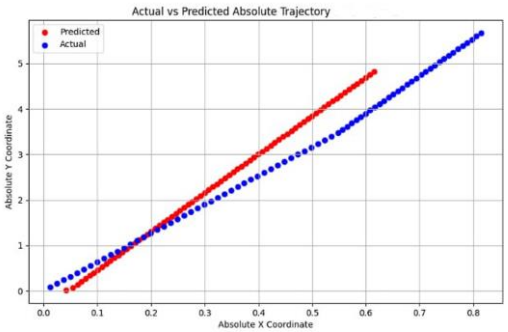

Fig. 10. The predicted trajectory on dataset 1 using the non-CNN approach

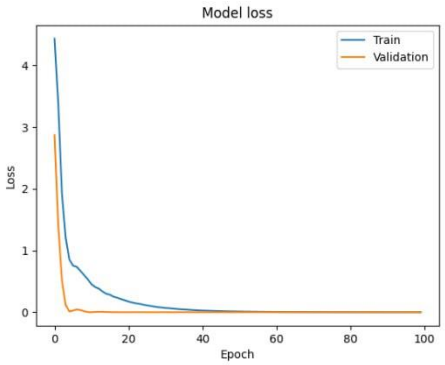

Fig. 11. The loss function of the model on the first dataset using the LSTM approach

The LSTM model achieves an MSE of 0.03 on the test set of this dataset.

5.2 Dataset 2

This dataset is the obtained from 'De Zul'. In this section we first examine the metrics obtained after 100 epochs of training the CNN-LSTM hybrid model. Then we compare this to the alternative approach which only uses the LSTM. We also show the graphs that show the predicted and the actual trajectory of the test set. The X and Y units shown are in meters, the prediction horizon is one second.

CNN-LSTM Approach:

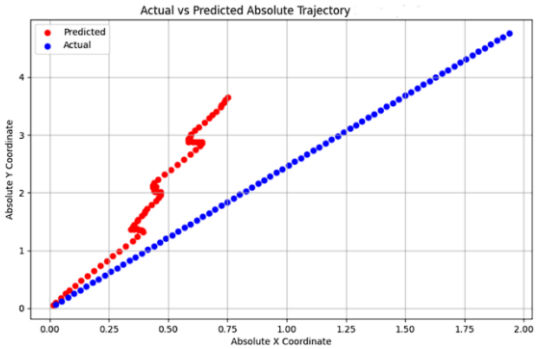

Fig. 12. The predicted trajectory on dataset 2 using the CNN approach

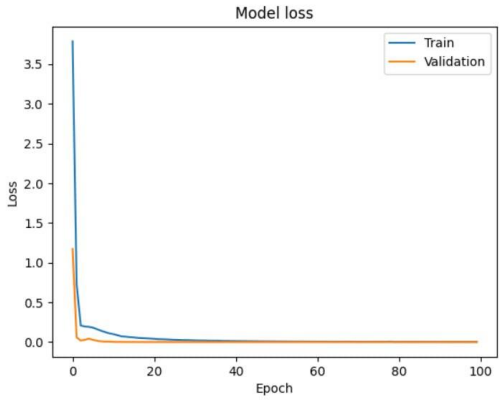

Fig. 13. The loss function of the model using the CNN approach on the second dataset

The CNN-LSTM model achieves an MSE of 0.04 on the test set of this dataset.

LSTM Approach:

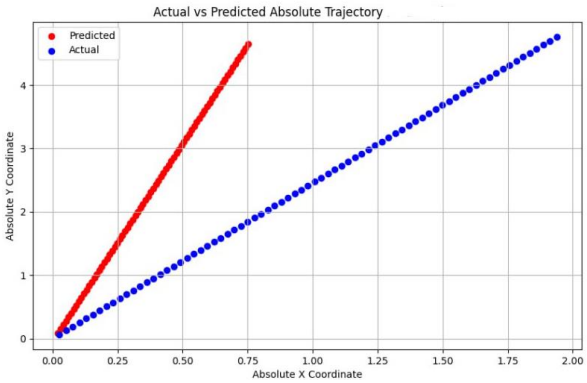

Fig. 14. The predicted trajectory on dataset 2 using the non-CNN approach

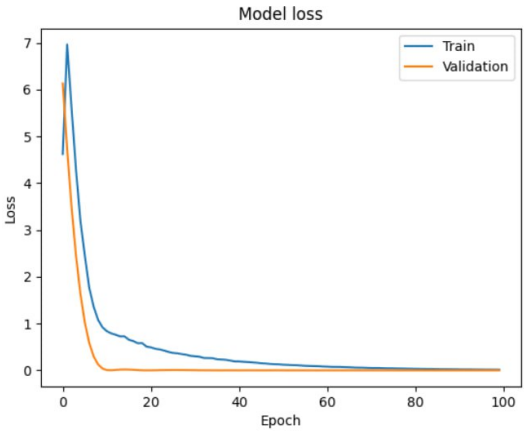

Fig. 15. The loss function of the model using the non-CNN approach on the second dataset

The LSTM model achieves an MSE of 0.05 on the test set of this dataset.

5.3 EuRoC MAV Dataset

This dataset is the 'Machine Hall 01' sequence from the EuRoC MAV dataset [1]. This dataset contains image and IMU data collected by a Micro Aerial Vehicle (MAV). It is used to test whether the model is able to perform better on a dataset with accurate ground truth. The LSTM model has been trained for 100 epochs here. The model has been slightly modified for this dataset since the positions are in 3D instead of 2D, meaning that the model now has to predict X, Y and Z coordinates.

CNN-LSTM Approach:

Fig. 16. The predicted trajectory on the EuRoC MAV dataset using the CNN-LSTM approach

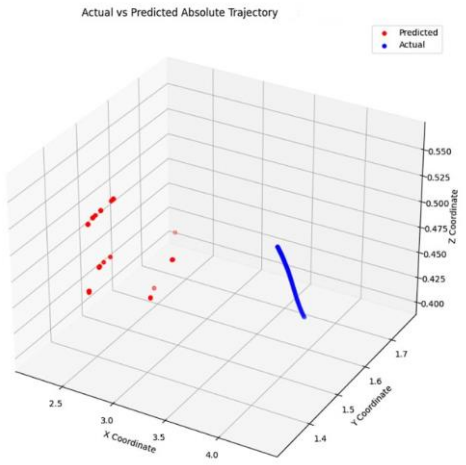

Fig. 16. The predicted trajectory on the EuRoC MAV dataset using the CNN-LSTM approach

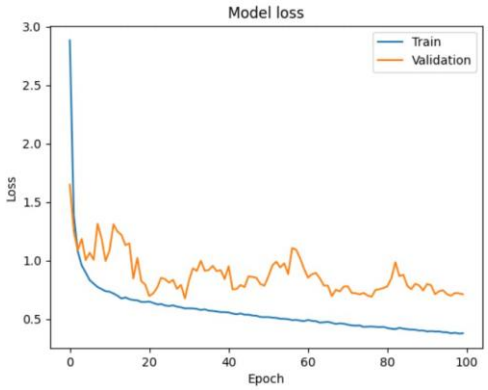

Fig. 17. The loss function of the model on the EuRoC MAV dataset using the CNN-LSTM approach

The CNN-LSTM model achieves an MSE of 3.4 on the test set of this dataset.

LSTM Approach:

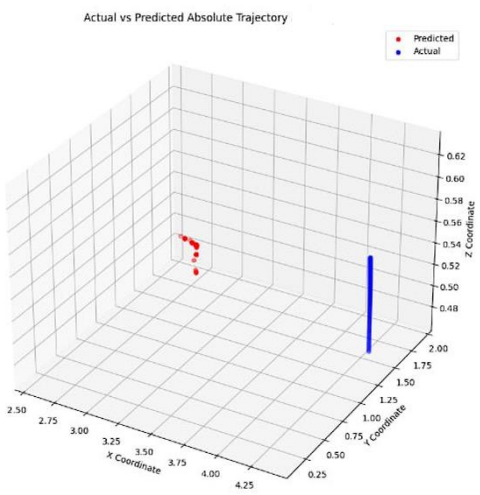

Fig. 18. The predicted trajectory on the EuRoC MAV dataset using the LSTM approach

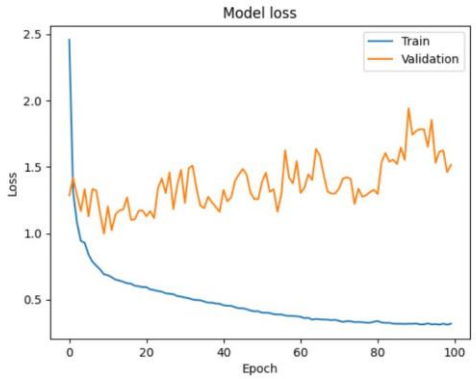

Fig. 19. The loss function of the model on the EuRoC MAV dataset using the LSTM approach

The LSTM model achieves an MSE of 5.6 on the test set of this dataset.

6 DISCUSSION

6.1 Research Questions

After this research, we have obtained the relevant results to answer our main research question:

(1) *Which sensors should be used?* The results show that it is possible to obtain sufficient results using just camera and IMU data.
(2) *What is a suitable trajectory prediction method given the time and cost limitations?* Due to the fact that we are dealing with temporal data, LSTM is a solid option as it is a recurrent neural network that does not suffer from the exploding and vanishing gradient problems
(3) *How accurately can this method predict the trajectory of the bicycle?* The previously shown results seem to indicate that the approach using the combination of CNN and LSTM is able to predict the trajectory to some degree, though this is not the case with the non-CNN approach.

To answer our main research question, a smart-bicycle can predict its trajectory approximately using affordable sensors, but more work is needed to achieve accurate results.

6.2 Datasets

Three datasets have been used in this research, two of which were custom made for this research and the other one being the EuRoC MAV dataset.

The datasets that were created for this research were lacking an accurate ground truth. This is due to the fact that the GNSS-RTK sensor was only limited to 1Hz, so linear interpolation had been used to fill in the missing data. This may have resulted in inaccuracies in the dataset, which may have affected the trajectory prediction model as well.

Furthermore, the datasets are relatively small compared to traditional datasets that are used in the field. Each dataset consists of approximately 4000 images, however due to the fact that the images were taken at 75Hz this only represents roughly a minute of data. Additionally, the images were of lower quality than expected, despite having a resolution of 640x480. This could be due to compression.

The datasets created for this research also consist of relatively simple straight-line trajectories, which may not be representative of trajectories in practice.

6.3 CNN-LSTM versus LSTM

The model using the CNN-LSTM approach seems to perform a bit better than the model using only the LSTM approach. On the first dataset both models perform relatively well, achieving a low MSE and a trajectory that is somewhat accurate. On the second dataset, both models perform worse. The MSE on the test set of the models is still relatively low but the trajectories are somewhat strange.
This may suggest potential inaccuracies in the model's performance evaluation. It is possible that the employed metrics might not represent the most optimal selection for assessing the model's effectiveness.

On the EuRoC MAV dataset both models performed significantly worse than on the other two datasets. The MSE is relatively high and the predicted trajectories appear to be far from the actual trajectories. The only difference with the other two datasets is that the model is now predicting the position in 3D rather than 2D. That could be one possible reason for the underwhelming performance. However, it is more probable that the model is not using the correct features to make accurate predictions.

From the results it appears that the CNN-LSTM model performed slightly better. It was able to achieve a lower MSE than the models which did not use the CNN and produced better graphs. Though, the difference is not significant enough to conclude that the CNN-LSTM approach is better.

6.4 CNN

Although a Convolutional Neural Network can improve the accuracy of LSTM trajectory prediction models as shown by Xie et al. [2020] and Ma et al. [2020], it is most likely not necessary to use a deep CNN like MobileNet. In the aforementioned research, only a single convolutional layer is used with a pooling layer, whereas a deep CNN like MobileNetV3 contains multiple convolutional layers and significantly more layers in total. This may not be necessary to achieve accurate results, and using a smaller CNN would improve the computational efficiency as well. In the context of smart bicycles it may therefore be better to opt for a smaller CNN than the one used in this research.

7 CONCLUSION

In conclusion, our findings show that an LSTM based approach for trajectory prediction using affordable sensors can predict the trajectory approximately. The CNN-LSTM hybrid architecture produced slightly better results, though these results are not convincing enough to conclude that the CNN-LSTM approach is superior. MobileNet, the CNN used in this research, may not have been the best choice. It is a deep network meaning that it may not be the best choice for performance constrained devices such as smart bicycles. Using a simpler CNN could improve computational efficiency and can still lead to good results as shown in other research.

The fact that relatively inexpensive sensors were used in this approach though does show that it is not necessary rely on expensive sensors such as Lidar for trajectory prediction. When it comes to accuracy though, the models shown in this research did not perform as well as they should, which indicates future research is necessary before these systems are usable in driver assistance systems for cyclists. The datasets created in this research were unfortunately lacking an accurate ground truth due to a sensor limitation. Though, even on the EuRoC MAV dataset with an accurate ground truth the models did not perform well, indicating a better trajectory prediction model architecture is needed. Potential improvements therefore include creating a larger and more accurate dataset for bicycles to train the model on, and an improved model architecture.

REFERENCES

[1] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M., Siegwart, R. 2016. The EuRoC micro aerial vehicle datasets. In International Journal of Robotic Research. DOI: 10.1177/0278364915620033.

[2] Howard, A. W., Pang, R., Adam, H., Le, Q. V., Sandler, M., Chen, B., Wang, W., Chen, L., Tan, M., Chu, G., Vasudevan, V. K., and Zhu, Y. 2019. Searching for MobileNetV3. DOI: https://doi.org/10.1109/iccv.2019.00140.

[3] Huang, Y., Du, J., Yang, Z., Zhou, Z., Zhang, L., & Chen, H. 2022. A Survey on Trajectory-Prediction Methods for Autonomous Driving, 652–674. https://doi.org/10.1109/tiv.2022.3167103.

[4] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. 2016. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In arXiv. Cornell University. DOI: https://doi.org/10.48550/arxiv.1609.04836.

[5] Lefkopoulos, V., Menner, M., Domahidi, A., & Zeilinger, M. N. 2021. Interaction-Aware Motion Prediction for Autonomous Driving: A Multiple Model Kalman Filtering Scheme. In IEEE Robotics and Automation Letters 6, 1 (2021), 80–87. https://doi.org/10.1109/lra.2020.3032079.

[6] Ma, L., Tian, S. 2020. A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction. In IEEE Access, vol. 8, pp. 134668-134680, 2020, doi: 10.1109/ACCESS.2020.3010963.

[7] Park, S. H., Kim, B., Kang, C. M., Chung, C. C., & Choi, J. W. 2018. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. In arXiv (Cornell University). https://doi.org/10.48550/arxiv.1802.06338.

[8] Stilo, L., Segura-Velandia, D. M., Lugo, H., Conway, P., & West, A. A. 2021. Electric bicycles, next generation low carbon transport systems: A survey. In Transportation Research Interdisciplinary Perspectives 10 (2021), 100347. https://doi.org/10.1016/j.trip.2021.100347.

[9] Sun, Y., Zuo, W., & Li, M. 2020. See the Future: A Semantic Segmentation Network Predicting Ego-Vehicle Trajectory With a Single Monocular Camera. In IEEE Robotics and Automation Letters 5, 2 (2020), 3066–3073. https://doi.org/10.1109/lra.2020.2975414.

[10] Xie, G., Shangguan, A., Fei, R. et al. 2020. Motion trajectory prediction based on a CNN-LSTM sequential model. In Sci. China Inf. Sci. 63, 212207. https://doi.org/10.1007/s11432-019-2761-y.