REDUCING THE RATE OF ACCIDENTS INVOLVING PEDESTRIANS IN TRAFFIC USING DL

Fatih Ay, Bayram Kaya & Fatma Akçakır
Department of Computer Science
Hacettepe University
Ankara, Türkiye
b21945815@cs.hacettepe.edu.tr

ABSTRACT

Pedestrian intention to cross is a topic that has long been included in active research. But since there is no common training and evaluation procedures, it is difficult to measure the progress towards the target. In this study, we will analyze the results using 2 different procedures.

1 INTRODUCTION

The theme for our assignment is “Computer vision for emergencies”. We decided to work on ”Emergencies caused by humans”, one of the sub-branches of this theme.
When pedestrians do not pay attention to cars while crossing the street, there is a high risk of pedestrian accident. Our aim is to predict the time when pedestrians want to cross and to warn drivers in advance and by doing this we are aiming to reduce the number of accidents.
For this purpose, we need a model that understands the pedestrian’s intention to cross. But it is very difficult to prepare such a data set. To find out people’s intentions, you have to either do a live poll or guess yourself. The difficulties do not end after finding the data set. There is a wide variety of data to take into account, making it difficult to prepare the model and process the data set accordingly.

2 RELATED WORKS

There are a lot of researches for preventing accidents caused by drivers (Civik & Yuzgec (2023), Iftikhar et al. (2022) etc.). But our aim is to use intention of pedestrians.
There is various approaches for detect the intention of pedestrians, such as:

2.1 PEDESTRIAN DETECTION AND TRACKING

This is one of the basic approaches. With this approach they aim to predict pedestrian intent based on intrinsic pedestrian features such as poses. (Rehder et al. (2017) , Varytimidis et al. (2018))

2.2 TRAJECTORY PREDICTION

In this approach, they aim to predict where the pedestrian will be in the future and use this information to infer the pedestrian’s intention. (Habibi et al. (2018))

2.3 ACTION PREDICTION

This approach predicts pedestrian intent by using the relationship between the past, current and potential future information(Farha et al. (2018))

2.4 Scene graph parsing and visual reasoning

This approach predicts pedestrian intent by using the Spatio-temporal relationship between the various objects in the scene (Zellers et al. (2017))

3 Method

Recurrent neural networks (RNNs) are a particular kind of neural network where the results of one step are fed into the next stage's input. And our models based on RNN.The SF-RNN(Rasouli et al. (2020)) model and the MultiRNN(Bhattacharyya et al. (2017)) model. RNNs can learn the temporal relationships in sequence data because they have recurrent hidden states. By stacking several layers of RNN units on top of one another, RNNs may also improve their spatial depth in addition to their temporal depth. This strategy is efficient for enhancing sequential data modeling in difficult jobs.
The models were made using the Gated recurrent unit. A Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN). GRU can process sequential data such as text and speech. The fundamental principle of GRU is to selectively update the network's hidden state at each time step via gating techniques. The reset gate and the update gate are two of the GRU's two gating systems. The update gate determines how much of the new input should be used to update the hidden state, while the reset gate determines how much of the prior hidden state should be forgotten. (Gupta (2023)). If you want to know more about the GRU you can check out the source (Kostadinov (2017))

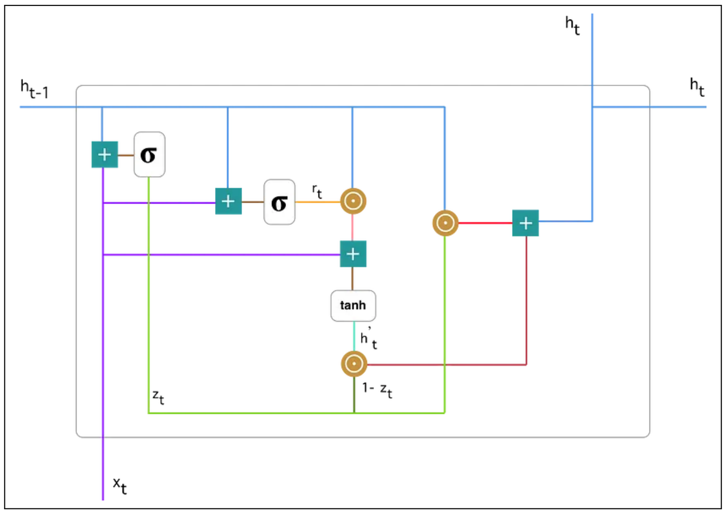


Figure 1: A gated recurrent unit

To train these models, we first prepared the data. We processed the PIE data set (Rasouli et al. (2019)) to fit our models using the VGG16 model(Simonyan & Zisserman (2014)). VGG16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 for ImageNet Large Scale Visual Recognition Challenge.They won 1st place in object localization and 2nd place in image classification
The pose data is from OpenPose (18 body joint coordinates concatenated into a 36D feature vector). This data set, which was prepared in accordance with the PIE data set, was also processed.
Each pedestrian's observation data is sampled so that the last frame of observation was between 1 and 2 seconds (or 30–60 frames) before the crossing event began.This hyperparameter is called TTE(time to event). We fixed the minimum observation length to 16 frame(Kotseruba et al. (2021)). We want to use our model in vehicles, to warn the car or driver if pedestrians want to cross dangerously. Therefore, the TTE(time to event) and minimum observation length had to be neither too long nor too short(because of reflex time).The numbers are the result of taking these into account.

This image visualizes the TTE and minimum observation length

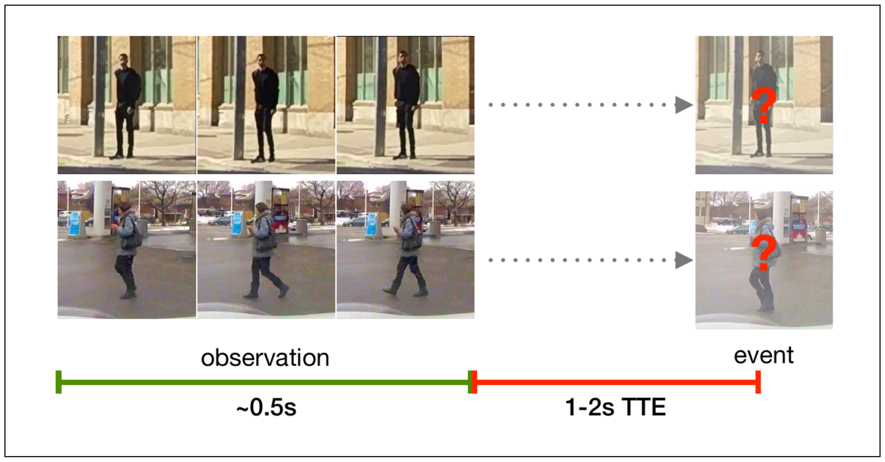


Figure 2: Length of observations

Multiple RNN are build of from different RNN streams that process each feature type individually.The RNNs' hidden states are combined and fed into an FC layer for prediction at the very end(Özgür Türkoğlu (2020)).
For this model local box, pose, local surround and box features are used in the order they are written. We tried to sort it from more important data to less important data.

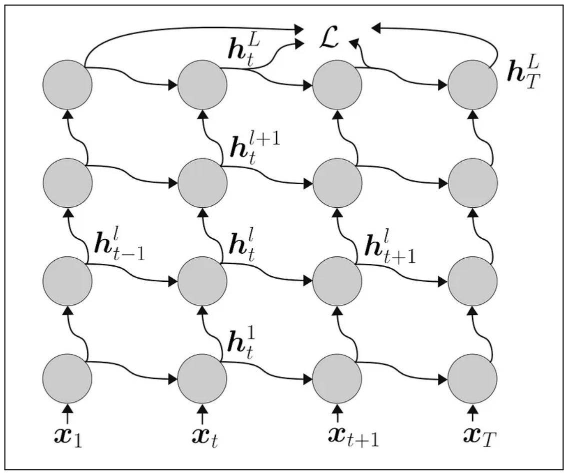


Figure 3: Multiple RNNs

SF-RNN model is based on this research (Rasouli et al. (2020)). The five GRUs that make up the proposed algorithm SF-RNN's design each process a concatenation of features from various modalities as well as the hidden states of the GRU from the level before. Depending on how complicated the characteristics are, the information is gradually integrated into the network. In order to forecast whether a pedestrian would cross the street given the observed environment up to a certain time m, they describe pedestrian crossing prediction as a binary classification issue.
The following characteristics are fused from bottom to top layers: Local box, Local surround, Pose, Speed, Box

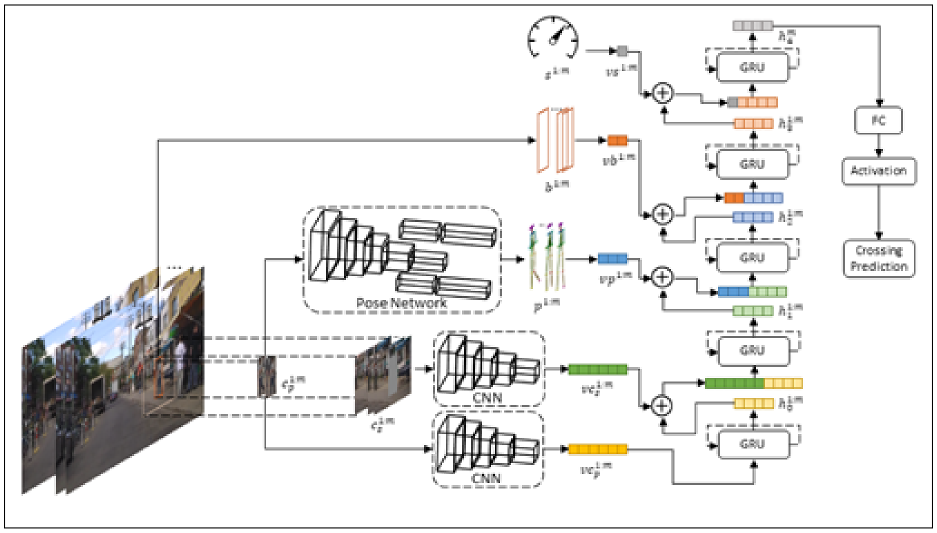


Figure 4: SF-RNN

4 Experimental settings

The PIE data set was made to examine pedestrian behavior in traffic. They recorded 6 hours of HD video with on-board camera at 30 FPS and split into approximately 10 minute chunks. The data set contains accurate ego-vehicle information from OBD sensor such as speed, spatial annotations such as traffic lights, behavioral annotations such as walking and intention annotations. In the next picture you can see some statistics about the data set

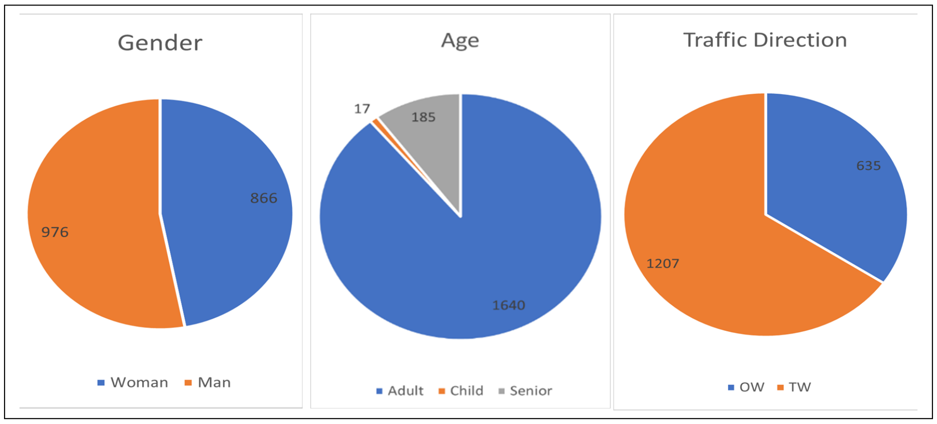


Figure 5: Gender, Age, and Traffic Direction

PIE data set is one of the largest publicly accessible data sets for analyzing pedestrian traffic, the collection includes over 290K annotated video frames with 1842 pedestrian samples.
The PIE data set also includes 429 pedestrians who do not intend to cross, 519 pedestrians who do mean to cross but do so in front of the vehicle, and 894 pedestrians who intend to but do not cross.
There is 6 different sets in the PIE data set. We used set 02, set 03, and set 06 to train, set 04 and set 05 for validation, and set 03 to test.
For train there are 3576 negative 1194 positive sample, for validation there are 1062 negative 270 positive sample and for test there are 2742 negative 1074 positive sample. We see that the negative rate naturally outweighs, but we decided not to balance the data set

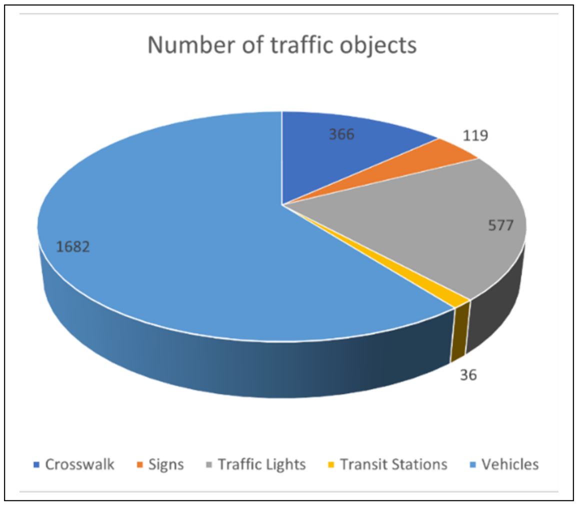
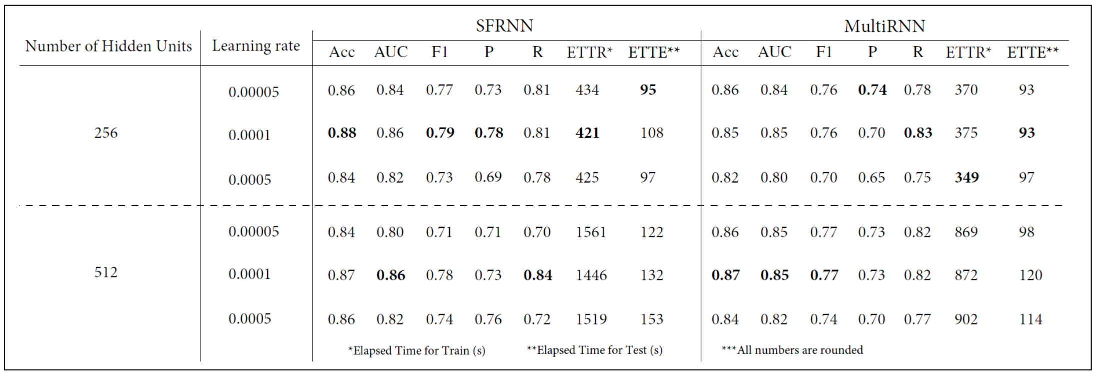


Figure 6: Number of traffic objects

I didn't mention what we are going to test. In order to anticipate whether a pedestrian would begin crossing the street at some time t given an observation of length m, we construct the pedestrian action prediction as a binary classification issue. Our goal is %75 f1 score. We tested our models with 3 different learning rates and 2 different hidden numbers of units. The epoch number is locked to 100 and the batch size is locked to 10. There is an early stop observing the validation loss with patience equal to 5

5 Experimental Results

It took me about 2 days to process the data set using VGG16 but sadly, I didn't keep the time. As for the train and test you can see the results in this table:

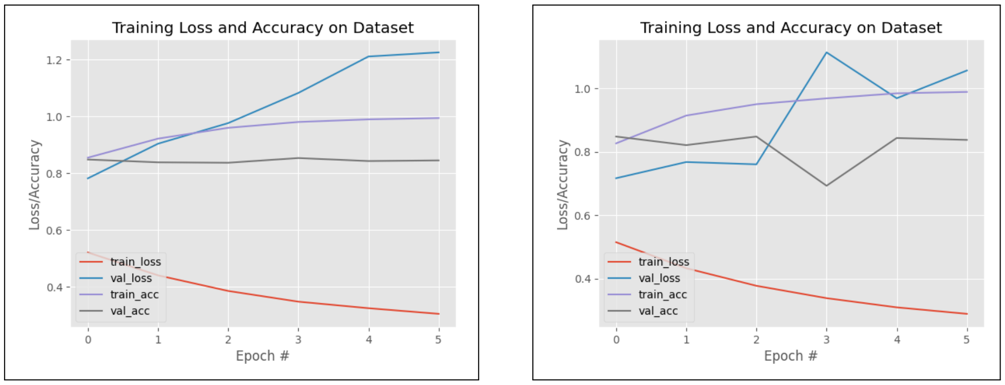


Number of Hidden Units	Learning rate	SFRNN							MultiRNN						
		Acc	AUC	F1	P	R	ETTR*	ETTE**	Acc	AUC	F1	P	R	ETTR*	ETTE**
256	0.00005	0.86	0.84	0.77	0.73	0.81	434	**95**	0.86	0.84	0.76	**0.74**	0.78	370	93
	0.0001	**0.88**	0.86	**0.79**	**0.78**	0.81	**421**	108	0.85	0.85	0.76	0.70	**0.83**	375	**93**
	0.0005	0.84	0.82	0.73	0.69	0.78	425	97	0.82	0.80	0.70	0.65	0.75	**349**	97
512	0.00005	0.84	0.80	0.71	0.71	0.70	1561	122	0.86	0.85	0.77	0.73	0.82	869	98
	0.0001	0.87	**0.86**	0.78	0.73	**0.84**	1446	132	**0.87**	**0.85**	**0.77**	0.73	0.82	872	120
	0.0005	0.86	0.82	0.74	0.76	0.72	1519	153	0.84	0.82	0.74	0.70	0.77	902	114

*Elapsed Time for Train (s) **Elapsed Time for Test (s) ***All numbers are rounded

Figure 7: Results

We saw that the model was very fastly overfitting when we used larger learning rates. I think this is because of the size of the data set. We used the values from the projects we got help for other hyperparameters. Although the epoch number was 100, there was no model that saw 20 due to Early Stopping. We also use ReduceLROnPlateau from tf.keras. It is reducing learning rate when a metric(in our case it is value loss) has stopped improving
We found that using the PIE dataset is quite easy once you get used to it. Those who prepared the data set also prepared many functions for using the data set. The most difficult part was to process the data in accordance with the model.



(a) Our best SFRNN model (b) Our best multiRNN model

Figure 8: Best models' learning curves

I think it is difficult to find easy-to-understand lessons and community support for this topic. Most of the resources we could find were scientific papers. In terms of hardware, although it was very difficult in terms of storage at first, I solved it later.
Looking at the results, we see that SFRNN clearly gives better results and we have seen that increasing the number of hidden units didn't help SFRNN at all. Increasing the number of hidden units increased the accuracy of MultiRNN and it did not extend the training period as much as SFRNN and MultiRNN is faster than SFRNN according to the our tests. In addition to these both models mostly gave higher recall values versus precision. This shows that both models have high false positive rates. In terms of learning rate, 0.001 gave better results in both models.

6 Discussions/Conclusion

We consider the result of the project as a moderately success. We reached the target score, even surpassed it, but we kept the content of the project low. It is clearly seen in the results that SFRNN is a better architecture (at least for this topic). Being slower may be a problem, but SFRNN gave the best results with 256 hidden units and MultiRNN with 512. Comparing the speed of MultiRNN with 512 hidden units to SFRNN with 256 hidden units, SFRNN is by far faster. First of all, I would like to talk about the aspects that we found successful;
Of course, we consider the value we receive as a result successful, and we also consider our preprocessing successful. If you remember, we said that the size of the data set is 3TB and our hardware is not suitable for it which we talked about in the first report we sent. I found a solution to this and we were able to use the entire data set. We did the processing of the photos in VGG16 one by one for each video before train. as a result the data set is about 14GB. The data set took 2 days to process and we couldn't change some hyperparameters because of this, but we still think the gains far outweigh the losses. If we had more time and better hardware I would like to try preprocessing with Resnet50 model
One more thing we consider successful is that we adapted quickly. If you remember, after you accepted the project proposal we sent you at the beginning of the semester, Bayram asked if we could use the trained models towards the middle of the semester. As a result of your refusal, we decided to do it ourselves, but approximately 1 week before sending the first report, we realized a huge mistake. The model we took as an example also predicted that the person waiting at the red light was intending to cross. Semantically this may be true, but that's not what we wanted. We don't want the driver to be warned or the car working with deep learning to stop when there was a pedestrian waiting at the red light. Therefore, we quickly looked at other projects made with the PIE data set and wrote that report for this project.
As for the aspects we consider unsuccessful;
The support we got from projects done by others exceeded my expectations and as the subject of the project changed a week before the first report and this school period was generally unproductive,

we stayed with 2 models when we could try a lot more different things. And of course there was the dispiritedness of us considering using only a few videos of the data set. I found the solution that allows us to use the whole data set about 3 weeks before submitting the project.
We also could not find the reason why value loss often never drops. Because of this the trainings were very short. Being short can actually be considered a good thing, but the reason for this is the low number of epochs, which reverses this situation.
If we want to improve our models, we must first investigate why the value loss is increasing almost continuously. Because of this, none of our models could even reach 20 epochs.Of course, I could turn off early stopping but that would just be to allow overfitting.
Apart from this, we can also try preprocessing the data set with different models such as ResNet-50. Also, we can balance the data or play with some hyperparameters that I did not mention such as regularization value, overlap, enlarge ratio.. And lastly, we can try to give the data in a different order and even add more various data such as optical flow, scene context, local context etc. to the input of the models.
But what I would like to do most (except for the value loss issue) is to use other models besides these two models such as Convolutional3D and Inflated3D. After all, this project is a assignment for a lesson and our main goal is to learn.

References

Esra Civik and Ugur Yuzgec. Real-time driver fatigue detection system with deep learning on a low-cost embedded system. 2023.

Sundas Iftikhar, Zuping Zhang, Muhammad Asim, Ammar Muthanna, Andrey Koucheryavy, and Ahmed A. Abd El-Latif. Deep learning-based pedestrian detection in autonomous vehicles: Substantial issues and challenges. 2022.

Eike Rehder, Florian Wirth, Martin Lauer, and Christoph Stiller. Pedestrian prediction by planning using deep neural networks. 2017.

Dimitrios Varytimidis, Fernando Alonso-Fernandez, Boris Duran, and Cristofer Englund. Action and intention recognition of pedestrians in urban traffic. 2018.

Golnaz Habibi, Nikita Jaipura, and Jonathan P. How. Context-aware pedestrian motion prediction in urban intersections. 2018.

Abu Farha, Alexander Richard, and Juergen Gall. When will you do what? - anticipating temporal occurrences of activities. 2018.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. 2017.

Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. Pedestrian action anticipation using contextual feature fusion in stacked rnns. 2020.

Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. 2017.

Alind Gupta. Gated recurrent unit networks. 2023.

Simeon Kostadinov. Understanding gru networks. 2017.

Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K. Tsotsos. Pedestrian intention estimation (pie) dataset. 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. Benchmark for evaluating pedestrian action prediction. 2021.

Mehmet Özgür Türkoğlu. Deep multi-layer rnns that can be trained. 2020.