Better Pedestrian Intention Estimation for Autonomous Driving

Tianchen Ye
jackye@umich.edu

Tong Jin
tongjin@umich.edu

Xiang Shen
xsh@umich.edu

Xiang Lian
lianx@umich.edu

Yigao Fang
fgsepter@umich.edu

Abstract

Avoiding crossing pedestrians is one of the most important tasks for a secure autonomous driving application and estimating pedestrian's intention is the essence of any solution to this task. Aiming to tackle the intention estimation problem, this paper will utilize the Pedestrian Intention Estimation Dataset (PIE), which consists of high-resolution driving footage and meticulously annotated pedestrian information [2]. Temporal sequences of the pedestrian images will be extracted from the footage and used to train and test our deep neural network model which predicts the pedestrian's intention to cross. Our model offers an accuracy rate of 77.5%, outperforming Rasouli's baseline model with context image inputs. Our modified dataset interface and neural network model can be found at: `https://gitlab.eecs.umich.edu/xsh/eecs442_pj`

1. Introduction

High-level autonomous driving has been the focus of numerous computer vision research for years. One of the major challenges of ensuring the safety of highly autonomous vehicles is the difficulty of tracking and understanding pedestrian behaviors. In 2018, an Uber autonomous testing vehicle failed to react to a crossing pedestrian and fatally struck her at a speed of 43 mph. The incident is a clear indication of our current weakness in predicting pedestrian behavior, which has to be addressed before highly autonomous vehicles can be safely deployed.

The recent rapid development in Multiple Object Tracking (MOT) techniques, starting from the classical object tracking algorithms such as CSRT, KCF, and MOSSE [1], which generally provides lower MOTA, to the state of the art FairMOT model which achieves real-time tracking with MOTA up to 0.73, has significantly expanded our capacity to track the trajectories of pedestrians[5]. Therefore, it is high time that we shifted our focus to the more crucial and sophisticated problem of understanding pedestrian behaviors.

One of the leading researchers in studying pedestrian behavior prediction is Amir Rasouli at York University. In 2017, Rasouli published a paper on pedestrian crosswalk behavior together with a large-scale dataset called Joint Attention in Autonomous Driving (JAAD)[3][4]. In the paper, he focused on classifying pedestrians' behaviors and determining whether they are currently crossing. Then, Rasouli published another paper in 2019 with an even more comprehensive dataset called Pedestrian Intention Estimation (PIE). He asserted that it is crucial for autonomous vehicles to predict pedestrians' intention to cross before they actually do so, leaving more time for the vehicle to react. He proposed a deep learning model that can not only estimate pedestrians' intention but also predict the future trajectory of the pedestrians, using CNN and RNN networks.

Rasouli's model greatly inspired us when we make our own neural network design. However, unlike his model which also predicts the future trajectory of the pedestrian, our model will mainly focus on the intention estimation task. We will seek improvement by upgrading its obsolete CNN model, VGG16, to resNet18. We will replace its Tensorflow Convolutional LSTM pipeline with a PyTorch-compatible feature-vector LSTM pipeline. We will also use the cross-entropy loss instead of the mean squared error metrics due to our focus on the intention estimation task.

2. Approach

2.1. Extract the Pedestrian Images

PIE dataset provides us with numerous 10-minute video clips showing the scene in front of the vehicles, taken by onboard cameras. Predicting the intention of pedestrians to cross the road directly from such a large-scale video dataset can be difficult and time-consuming. Without access to powerful computational resources, we apply several preprocessing methods to make training

feasible and efficient.

First, video sets 1,2,5,6 are chosen and used for training, validation, and testing. Since the relative difference between adjacent frames is small for most images, we will extract one image from every two frames to reduce the memory and loading time. The extracted images with an original dimension of 1920×1080 are then resized 1600×900. This is process is illustrated in step one of Figure 1.

The second step is to crop images that focus on individual pedestrians. In the PIE dataset, each individual pedestrian is provided with the ground truth intention probability for the pedestrian to cross the road and the bounding box as well, which contains the location of each pedestrian. For each pedestrian, we crop a 128×128 image around the bounding box to contain the context of the pedestrians, and zero-padding is added when dealing with border cases as shown in Step 2 of Figure 1.

Since the duration that each pedestrian appears in the video varies, we will obtain different numbers of 128×128 images for each individual pedestrian from the previous. In order to batch different pedestrians together and make the training set consistent, we define a parameter *sequence size*, which denotes the number of images that we will sample uniformly from the original the sequences. For pedestrians with overly short clips which fall below sequence size threshold, we simply discard them. Now all the pedestrians will have the same number of image inputs.

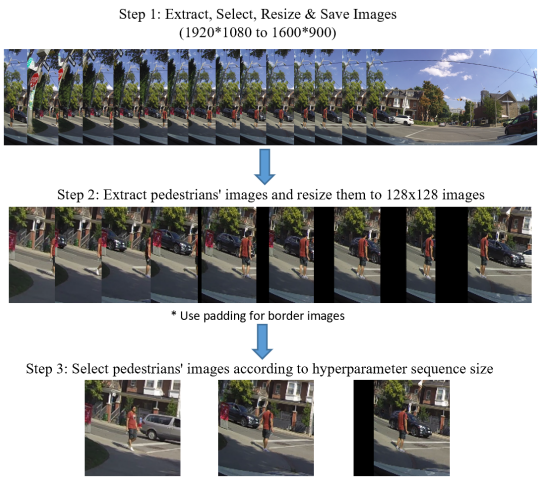

Figure 1. Flow Diagram of Pedestrian Extracting

The final step is loading the data. We created a list that stores the pedestrian image sequence and a Boolean intention variable derived from the ground-truth pedestrian intention probability, which will serve as the label of our classification task. Each image is normalized and stored as a PyTorch float tensor of shape $sequence\_size \times 3 \times 128 \times 128$, and intention probability will be a scalar tensor with a value of either 0 or 1. We shuffle the list in order to add additional stochasticity to our training process and use the Dataloader to load the train, validation, and test data under a ratio of 5:4:1.

To save work for reading all the annotations and attributes, we rewrite pedestrians' intention probability to a comma-separated value file with a key-value pair format of $(PID, intention\_probability)$, so that we can read it directly using Google Colab.

2.2. Recurrent Neural Networks

The PIE baseline model is a complex RNN network with the Tensorflow environment. On this basis, we designed our own network using the PyTorch environment.

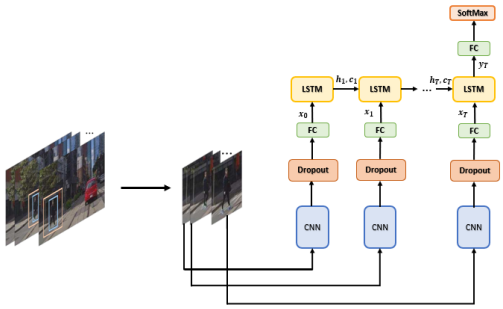

Figure 2. Recurrent Neural Network Architecture

As shown in Figure 2, we utilize pretrained res-Net18 network as our CNN to extract features from the input image sequence. Compared with the VGG16 network used in the PIE dataset, res-Net18 gives us faster and more accurate prediction results. Hence we decide to choose it as our CNN model. The total number of CNN components equals $sequence\_size$. The input of each CNN is a $sequence\_size \times 3 \times 128 \times 128$ image sequence for each pedestrian. We input the image sequence to several CNN components and add Drop Out layers followed by fully connected layers after each CNN component. This structure helps match the input size of the LSTM cell with the output size of our CNN as well as reducing the possibility of over-fitting. The RNN architecture receives the feature representation of the image area around the pedestrian and by adding the softmax layer on its output, we get our prediction on the probability of the pedestrian's intention. We utilize Cross-Entropy Loss as our loss function and ADAM algorithm as our optimizer to train the network. We also use Xavier distribution for the initialization of RNN architecture. The output of this RNN architecture is a binary classification for each pedestrian showing whether they have the inten-

tion to cross the road. Using this RNN architecture, we finally receive 77.5% accuracy of predicting the pedestrian intention.

3. Experiments

Pedestrian intention estimation for autonomous driving is a binary classification problem. We need to predict whether the given pedestrian has the intention to cross the road. Given the sets of pedestrian images extracted from the video and the ground truth intention probability, we are able to train the network and give intention probability predictions for practical application.

We use validation accuracy as our performance metrics. It is a quantitative metric, indicating the percentage of pedestrians that we gave a correct prediction for their intention. To experiment with the performance of our model, We split the dataset into a training set for training the model, a validation set for tuning hyperparameters, and a test set for measuring the final performance.

There are totally five hyperparameters for choosing: $sequence\_size$ influences the size of the training dataset, $learning\_rate$, $weight\_decay$, num_epoch, $batch\_size$ influence the performance of the RNN model. Larger $sequence\_size$ may result in much longer running time and memory usage, which may be inefficient. Hence we choose $sequence\_size = 15$ for better performance.

We then apply Grid Search to find the best values for $learning\_rate$ and $weight\_decay$. The validation accuracy for certain choices of hyperparameter values are recorded below in Table 1:

Table 1. Grid Search for Finding Best Hyperparameters

$learning\_rate$	$weight\_decay$	**Validation Accuracy**
6×10^{-6}	0.1	75.875%
9×10^{-6}	0.1	76.5%
12×10^{-6}	0.1	76.25%
6×10^{-6}	1	75.625%
9×10^{-6}	1	76.875%
12×10^{-6}	1	76.25%
6×10^{-6}	10	75.375%
9×10^{-6}	10	75.75%
12×10^{-6}	10	74.875%

We also test other choices of hyperparameters during the process, but due to the page limitation, we only record nine of them. According to Table 1, we finally choose:

$$learning\_rate = 9 \times 10^{-6}$$
$$weight\_decay = 1$$

$Batch\_size$ denotes the number of pedestrians we train at one time. We choose $batch\_size = 10$ so that our model gives the best efficiency.

A relatively small number of epochs may lead to under-fitting, while a larger number of epochs may easily result in over-fitting. We should pick the epoch with the lowest validation loss.

Figure 3 indicates the graph of training accuracy and validation accuracy concerning the number of epochs. After some testing, we finally choose the number of epochs equals 5. Training accuracy keeps increasing after epoch 5, while validation accuracy remains relatively the same. To avoid over-fitting, epoch 5 is the best point, and this model gives us a test accuracy of 77.5%

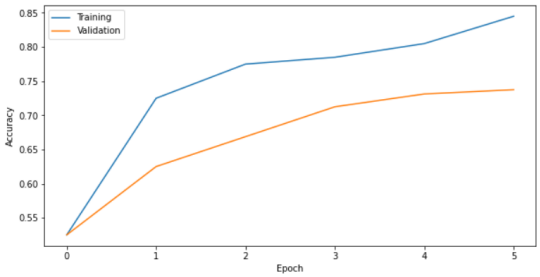

Figure 3. Training and Validation Accuracy vs. Epochs

We apply Cross-Entropy Loss as our Loss function. For a multi-class classification problem, we apply a softmax activation function after the output layer to obtain an n-dimension vector $\hat{y}$, where n is the number of classes to classify, with each entry of $\hat{y}$ be calculated as:

$$\hat{y_c} = \frac{exp(z_c)}{\sum_j exp(z_j)}$$

where z is the output logits from the neural network.

With the given ground truth distribution y, the Cross-Entropy Loss L can be calculated as:

$$L(y, \hat{y}) = -\sum_c y_c \log \hat{y_c}$$

PyTorch's `CrossEntropyLoss` class provides an implementation for Cross-Entropy Loss, and we can use it to calculate the Loss value. In our case, it is a binary classification question, so $n = 2$. The graph of training Loss and validation loss with respect to epochs is shown in Figure 4.

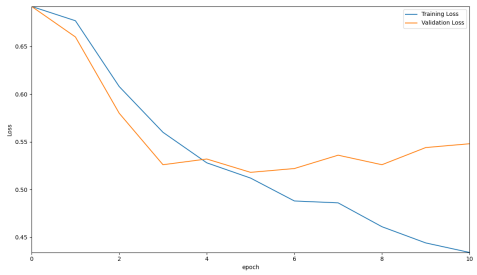

Figure 4. Cross Entropy Loss vs. Number of Epochs

According to Figure 4, we see that the training loss keeps decreasing with respect to the number of epochs, which fits our expectation. The minimal validation loss appears at epoch 5, and hence we choose num_epoch equals 5. More epochs may easily lead to over-fitting for the training data.

4. Implementation

We modified and used the PIE dataset interface to get our data ready. As is indicated in the approach section, we use the interface to extract images and intention probabilities while improving the interface by adding image size downsampling and sequence frequency selection options. All other preprocessing procedures, such as getting the final 128 x 128 linear sequence images, are our original work.

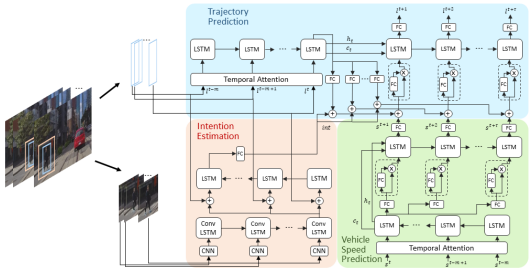

Figure 5. Network Architecture Implemented by Rasouli[2]

As shown in Figure 5, the original network architecture implemented by Rasouli[2] has some additional components to predict the future trajectory of the pedestrian. Our research focuses on the intention estimation task (the red box) of this architecture, with the main purpose of achieving higher accuracy for predicting the intention probability of the pedestrian, our model is shown in Figure 2. Taking inspiration from Rasouli's network, we implemented our own neural network architecture using PyTorch. There are three major differences between the two models. First, we used Resnet as our CNN model comparing to VGG16 in the baseline model. Second, our model is implemented using PyTorch while the baseline model is implemented using Tensorflow and Keras. Finally, we used the feature-vector LSTM pipeline instead of convolutional LSTM which is not built in the PyTorch library.

Of course, there are some further improvements that can be made to our model. For example, we can use data augmentation to further regularize our model and facilitate better feature extractions. We can also incorporate the bounding box coordinates and append them to our feature vector to achieve higher accuracy.

References

[1] Satya Mallic. Object tracking using opencv (c++/python). In *Learn OpenCV*, 2017.

[2] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K. Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *International Conference on Computer Vision (ICCV)*, 2019.

[3] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–213, 2017.

[4] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. It's not all about size: On the role of data properties in pedestrian detection. In *ECCVW*, 2018.

[5] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020.