MODELING SPATIOTEMPORAL PEDESTRIAN-ENVIRONMENT INTERACTIONS FOR PREDICTING PEDESTRIAN CROSSING INTENTION FROM THE EGO-VIEW

by

Chen (Tina) Chen

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Indianapolis, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Lingxi Li, Co-Chair

Department of Electrical and Computer Engineering

Dr. Renran Tian, Co-Chair

Department of Computer Information and Graphics Technology

Dr. Lauren Christopher

Department of Electrical and Computer Engineering

Dr. Zhengming Ding

Department of Computer Science

Approved by:

Dr. Brian King

ACKNOWLEDGMENTS

I want to thank Renran Tian and Lingxi Li whom I had the pleasure of knowing as coworkers first before starting on my journey in graduate school. They have supported and mentored me since I was an undergrad intern in the lab to when I was a research associate to now at the end of my Master's degree. It was especially under Renran's guidance that I focused my study on computer vision applications for pedestrian safety. I want to thank Dr. Zhengming Ding for sharing his expertise in computer vision, without him, I would have spent many more nights scratching my head. I also want to thank Dr. Lauren Christopher for her time and valuable feedback on my thesis.

Additionally, I want to thank Sherrie Tucker of the ECE department for everything that she is, and everything that she has done to assist me my entire time here. There isn't a question that she doesn't know the answer to. Special thanks to Rob Meagher and the CNC Help team for their assistance with our computers in our many labs, especially during the work from home era.

I want to give special thanks to my family and friends who encouraged me, and always showed an interest in my research even though they had no idea what I was talking about. Most importantly, I want to give thanks to my partner, Ryan, who always knew when I needed a little push or a snack, and never hesitated to stay up into the wee hours of the morning with me to keep me company.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	artificial intelligence
AV	autonomous vehicle
CNN	convolutional neural network
ConvLSTM	convolutional LSTM
GAE	graph autoencoder (generic)
GCN	graph convolutional network
GAN	generative adversarial network
GNN	graph neural network
JAAD	Joint Attention in Autonomous Driving (dataset)
LSTM	long short-term memory
NHTSA	National Highway Traffic Safety Administration
PIE	Pedestrian Intention Estimation (dataset)
RNN	recurrent neural network
STIP	Standford-TRI Intent Prediction (dataset)
TITAN	Trajectory Inference using Targeted Action Priors Network (dataset)
Y_C	pedestrian crossing action class
Y_I	pedestrian crossing intention class

ABSTRACT

For pedestrians and autonomous vehicles (AVs) to co-exist harmoniously and safely in the real-world, AVs will need to not only react to pedestrian actions, but also anticipate their intentions. In this thesis, we propose to use rich visual and pedestrian-environment interaction features to improve pedestrian crossing intention prediction from the ego-view. We do so by combining visual feature extraction, graph modeling of scene objects and their relationships, and feature encoding as comprehensive inputs for an LSTM encoder-decoder network.

Pedestrians react and make decisions based on their surrounding environment, and the behaviors of other road users around them. The human-human social relationship has already been explored for pedestrian trajectory prediction from the bird's eye view in stationary cameras. However, context and pedestrian-environment relationships are often missing in current research into pedestrian trajectory, and intention prediction from the ego-view. To map the pedestrian's relationship to its surrounding objects we use a star graph with the pedestrian in the center connected to all other road objects/agents in the scene. The pedestrian and road objects/agents are represented in the graph through visual features extracted using state of the art deep learning algorithms. We use graph convolutional networks, and graph autoencoders to encode the star graphs in a lower dimension. Using the graph encodings, pedestrian bounding boxes, and human pose estimation, we propose a novel model that predicts pedestrian crossing intention using not only the pedestrian's action behaviors (bounding box and pose estimation), but also their relationship to their environment.

Through tuning hyperparameters, and experimenting with different graph convolutions for our graph autoencoder, we are able to improve on the state of the art results. Our context-driven method is able to outperform current state of the art results on benchmark dataset Pedestrian Intention Estimation (PIE). The state of the art is able to predict pedestrian crossing intention with a balanced accuracy (to account for dataset imbalance) score of 0.61, while our best performing model has a balanced accuracy score of 0.79. Our model especially outperforms in no crossing intention scenarios with an F1 score of 0.56 compared to the state of the art's score of 0.36. Additionally, we also experiment with training the state of the

art model and our model to predict pedestrian crossing action, and intention jointly. While jointly predicting crossing action does not help improve crossing intention prediction, it is an important distinction to make between predicting crossing action versus intention.

1. INTRODUCTION

1.1 Overview

The National Highway Traffic Safety Administration (NHTSA) reported 36,096 fatalities due to motor vehicle traffic crashes for 2019 [1]. Of those fatalities, 6,205 of them were pedestrians. Additionally, another 76,000 pedestrians were injured in traffic crashes. Vulnerable road user fatalities continue to rise every year, while vehicle occupant fatalities steadily decline. NHTSA reports that 9 out of 10 serious roadway crashes are caused by human error. The develop of autonomous vehicle (AV) technology can reduce that number, and save thousands of lives [2].

One autonomous driving technology that has increased safety for pedestrians on the road is pedestrian automatic emergency braking. Pedestrian automatic emergency braking combines pedestrian detection [3], and emergency braking to intervene on the driver's behalf if the vehicle detects a pedestrian in the vehicle's path that could result in a collision. This type of emergency braking is a reaction to an event that has already happened. To further protect pedestrians from harm, simply reacting to events that have already happened is not enough. We must develop AV technology that can predict pedestrian behavior such that emergency braking is not the only solution for collision avoidance. Furthermore, accurately predicting pedestrian behavior would facilitate meaningful communication between AVs and pedestrians, and assist in the harmonious co-existence of both road users.

Artificial intelligence (AI) aims to simulate human intelligence in machines. The increasing popularity of using neural networks to tackle this task was made possible through the introduction of back-propagation [4] in 1986. Additionally, advancements in convolutional neural networks (CNNs), and recurrent neural networks (RNNs) have been instrumental in creating and validating AV solutions [5]–[7]. Predicting human behavior is one such area that has benefited from neural network advancements.

In this thesis, we are primarily concerned with predicting pedestrian crossing intention to avoid potential conflicts between the ego-vehicle and pedestrian. A recurrent neural network solution that leverages visual and spatial reasoning across the temporal space is developed to improve crossing intention prediction accuracy from the ego-view.

1.2 Thesis Motivation

In the current research space, most of the state of the art in pedestrian trajectory or intention prediction is centered around bird's-eye view camera positions. This may not translate well to AV applications due to two major discrepancies: (1) the camera angle change, and (2) continuous changes to the relative distance, and scale between the camera and our object of interest (i.e. pedestrians). Both of these discrepancies need to be addressed to properly design a system for AV use.

In research that does address these discrepancies, the research discards valuable visual contextual information obtained through the on-board cameras. As human drivers, we continuously absorb visual information from the road, and adjust our driving behavior from that information. For example, we may slow down or anticipate pedestrian crossings at designated crosswalks. All the rich visual information around the ego-vehicle not only dictates how the driver will behave, but also other road user behavior. In the same example, a pedestrian at a designated crosswalk may be more confident to cross even as vehicles approach compared to in the middle of the road where there is no crosswalk.

Along the same vein, traffic objects and agents are all interconnected on the road, affecting one another's behavior. To accurately predict a pedestrian's intention, we need to consider how all the other road objects and agents could be affecting said pedestrian's behavior. Modeling the pedestrian's spatio-temporal relationship to their surroundings could offer that insight.

1.3 Thesis Objective

In this thesis, we train a neural network on a pedestrian-centric dataset that was collected from the ego-view through naturalistic driving to more accurately predict pedestrian crossing intention. Our key insight is to leverage the rich visual information, and spatio-temporal pedestrian-environment relationships readily available in the dataset to improve intention prediction accuracy by providing context around the pedestrian's behavior.

Our proposal outperforms state of the art crossing intention prediction algorithms on benchmark dataset Pedstrian Intention Estimation (PIE) . Our model especially excels at

predicting no crossing intention cases, and is not as negatively impacted by imbalanced training data.

1.4 Organization

The remainder of this thesis is structured as follows. In Chapter 2, we broadly review the literature in human behavior estimation along with the various feature extraction, and graph embedding methods we use to build our model. Additionally, we summarize datasets that are focused on autonomous vehicle research. Chapter 3, we present an overview of our proposed model for predicting pedestrian crossing intention, and introduce PIE, the dataset which we train and evaluate our model on. We also define the evaluation metrics we use to evaluate our model's performance. In Chapter 4, we elaborate on the algorithms we use to extract human pose estimation, and appearance features from the pedestrians and traffic objects in PIE. Chapter 5 introduces graph modeling, and embedding of the scene. We compare different graph convolutional networks (GCNs) for scene modeling, and graph autoencoders (GAEs) for graph embedding. Chapter 6 is dedicated to predicting pedestrian crossing intention. In this chapter, we compare our model with the state of the art. Finally, in Chapter 7, we summarize our results, and recommend future research directions.

2. BACKGROUND AND RELATED WORK

In this chapter, we review prior work on human behavior estimation, and the various machine learning methods we applied to our proposed method to create our own pedestrian intention model. Section 2.1 summarizes the various types of human behavior estimation that are studied today across many fields, such as surveillance, robotics, and autonomous driving. We will primarily focus on camera-based approaches. Section 2.2 reviews two areas of computer vision that are critical for our model: human pose estimation, and visual feature extraction. We then review Graph Neural Networks (GNNs) as a means for applying neural networks on unstructured data in Section 2.3. In Section 2.4, we summarize, and compare publicly available datasets used for researching autonomous driving. Finally, we conclude the chapter in Section 2.5.

2.1 Human Behavior Estimation

The harmonious co-existence of intelligent machines and humans is dependent on correctly modeling human behavior for machines to interpret. Being able to track and predict human behaviors plays an important role in achieving this goal. We break human behavior estimation into three subset categories:

1. action anticipation and prediction
2. trajectory prediction
3. intention prediction

Human behavior is complicated and nuanced, often influenced by not only external stimulants, but also internal stimulants. An example of this are the factors that influence aggressive driving. Aggressive driving is often contributed to the driver's personality, age or gender (internal stimuli), but driving conditions are an equally important factor to be considered [8]. As driving conditions worsen, e.g. road congestion, the rate of aggressive actions (i.e. honking, cursing, and cutting other cars off) increases [9]. This is all to say comprehensive human behavior modeling is not an easy feat, but there are many real-world applications we work towards:

- Autonomous vehicles: The safe deployment of AVs in urban environments depend on their ability to anticipate, and react to other road users [10].
- Surveillance: Long-term video surveillance through stationary or aerial cameras [11] using detection, and tracking in heavily populated or sensitive areas, such as airports, and government buildings, can raise awareness of suspicious activity without constant human monitoring [12].
- Robotics: Self service robots are automating the hospitality industry [13]. In robotics, plan, activity, and intention recognition (PAIR) [14], are the guiding principles for motion planning [15].

2.1.1 Action Anticipation and Prediction

Action anticipation is predicting an action *before* it happens. There is a wide range of domains that use action anticipation so actions are defined by the application. In [16], Carl et al. train deep neural networks to predict the visual representations of future frames in unlabeled data. Then, because visual representations offer more semantic value than pixels, they use action recognition algorithms to classify the visual representations. Aliakbarian et al. in [17] use multi-stage long short-term memory (LSTM) cells to leverage context and action features, and a novel loss function that encourages correct classification at the earliest time possible. The authors in [18] use Dynamic Images [19] to represent human motion in a generative model that uses dynamic, and classification losses. In [20], Gammulle et al. use a recurrent generative adversarial network (GAN) for visual, and temporal feature synthesis. The synthesis learning is jointly performed with early action anticipation to ensure the future features are representative of future actions.

2.1.2 Trajectory Prediction

Trajectory prediction can be modeled as a sequence of observations $X_{obs} = \{x_1, x_2, \ldots, x_t\}$, and the network predicts a sequence, $Y_{pred} = \{y_{t+1}, y_{t+2}, \ldots, y_{tpred}\}$, where t_{pred} is the number of time steps to predict, and Y_{pred} is the future location of the pedestrian.

Studies use different ways to measure pedestrian trajectory, but the two most common methods are listed here [21]:

1. Single point xy-coordinate[22].
   - A single point to mark the head, torso or center between both feet.
2. Bounding box coordinates [23], [24].
   - Typically in the form of (x_c, y_c, h, w) or $(x_{tl}, y_{tl}, x_{br}, y_{br})$ where c, tl, and br are for center, top left, and bottom right points of the bounding box, and h and w are the height and width of the bounding box.

By encasing the non-occluded parts of a pedestrian in a box, bounding boxes offer more context clues than single location markers, because bounding boxes not only show the change in the relative distance from the moving ego-vehicle, but also show the relative size the pedestrian is compared to other objects in the scene. Bounding box prediction is less accurate than predicting a single point simply because there are more features to predict, but a single point to represent a pedestrian in the ego-view will not suffice when the perspective of the camera is constantly changing in a moving vehicle.

One of the earliest pedestrian motion prediction models, termed *Social Force* model [25], theorized attractive and repulsive forces guide pedestrians to their desired goal. Since then, many others have modified or extended the use of the *Social Force* model not only for path prediction [26], but also to robotics [27], anomaly detection [28], multiple person tracking [29], and activity recognition [30]. Similar approaches have been used that model human-human interactions using cultural norms and behavior. This is prevalent in crowded scenes [31]–[33] where the majority of external stimuli are from other humans. Social-LSTM [34] introduces a "Social" pooling layer that shares trajectory information between pedestrians in close proximity. Social-GAN [35] builds on top of Social-LSTM by adding generative adversarial training, and a "global" pooling vector that shares LSTM hidden states among all pedestrians in the scene. Algorithms based on social or crowd behavior are better suited for prediction from the bird's-eye view.

Context-driven methods derive behavior affected by the scene to make trajectory predictions. Scene-LSTM [36] imposes a two-layer grid on the scene to learn the scene data through coupled LSTM networks for the scene and the pedestrian. Social-Scene-LSTM [37] uses three different LSTMs for person, social, and scene scale in a hierarchical LSTM network. SoPhie [38] leverages both social and physical attention in GANs that injects scene features into the physical attention. Liang et al. in [39] use rich visual features in LSTMs to model person-scene and person-object relations for joint activity and trajectory prediction. Others focus on the pedestrian themselves to gather cues for prediction. Hasan et al. in [40] use head pose estimation to highlight the region of interest for object avoidance and short-term destination prediction.

For autonomous vehicle applications, it is better to look at studies that were designed with that application in mind as the angle of the ego-vehicle camera is completely different than bird's-eye view. Additionally, vehicle information, such as speed, and acceleration, and road specific pedestrian behavior should be considered. Bhattacharyya et al. use a two-stream architecture with Bayesian RNNs to jointly predict pedestrian trajectory from bounding boxes, and ego-motion through the vehicle odometry. In [41], Mangalam et al. use global, and local streams on human pose estimation sequences in a Quasi RNN to account for depth and ego-motion. Along with the proposal of the PIE dataset [42], Rasouli et al. use a network of LSTMs to use odometer and crossing intention prediction to improve trajectory prediction. The same research group later in [43] jointly use multi-modal data, and interaction modeling for multi-task prediction. TITAN [44] uses hierarchically ranked action priors to provide environment-driven context to understand motion behavior.

2.1.3 Crossing Intention Prediction

In the context of AVs, intention prediction is centered around pedestrian crossing intention. Unlike trajectory prediction, where the pedestrian can be moving anywhere, intention prediction is only concerned with the pedestrian crossing in front of the ego-vehicle.

The earliest intention prediction started with Joint Attention in Autonomous Driving (JAAD) [45]. Rasouli et al. proposed a richly annotated dataset to study the effects of traffic

scenes on pedestrian crossing behavior. Since then, many others have expanded upon their work. Varytimidis et al. added environmental tags to JAAD in [46] to provide additional context for their prediction model. Their prediction model uses CNNs to extract features of the pedestrian head, and legs to estimate head orientation, and motion for crossing behavior classification using support vector machines (SVMs). In [47], Gujjar and Vaughan directly use images as inputs in an encoder-decoder network where the encoder is a spatio-temporal network composed of convolutional layers, and the decoder convolutional LSTM layers. The output is predicted future frame images, which are used for classifying crossing behavior.

Additionally, Fussi-Net [24] fuse bounding boxes and pose estimation in early, late, and combined fusion mechanisms to reduce intention classification false positive errors. SPI-Net [48] uses only pose estimation for an efficient and context-invariant model. SPI-Net is made up of two branches: (1) the first focuses on the evolution of relative distance between posture points over time, (2) the second focuses on the evolution of spatial coordinates of posture points in the global Cartesian coordinate system. In [49], Rasouli et al. uses four modalities: (1) semantic map of the scene, (2) image context around the pedestrian, (3) pedestrian bounding boxes, and (4) ego-motion to generate visual, and dynamic encodings for their hybrid architecture. In [50], Liu et al. introduce their own dataset, Stanford-TRI Intent Prediction (STIP), and propose using graph convolution to construct a graph scene for crossing prediction.

Rasouli et al. predicts pedestrian trajectory as its end goal in [42], but as a secondary task predicts crossing intention using local context around the pedestrian to assist trajectory prediction. Cao and Fu use a similar network architecture as [42] in their own work in [51], but add a spatio-temporal graph module, HT-STGCN, to map pedestrian posture for additional feature learning. Alvarez et al. take their experiments [52] into the real-world, testing their pose, and GRU-based framework on real pedestrians. Similarly, [53] predicts crossing using CNNs to estimate pose. SF-GRU [54] takes features of different modalities, and stacks them with five GRUs. Lastly, Hoy et al. in [55] use variational recurrent neural networks (VRNN) to learn how to track pedestrians for crossing prediction.

2.2 Computer Vision Tasks for Visual Information

In the previous section, we reviewed many human behavior prediction methods. Many of them use visual features that are not part of the original ground-truth annotations of the dataset. To obtain these visual features, we must add computer vision tasks, such as object detection, human pose estimation, and feature extraction, on top of human behavior prediction. In this section, we will review recent works in human pose estimation, and visual feature extraction as we use these features for our model.

2.2.1 Human Pose Estimation

Classical approaches [56]–[59] to human pose estimation phased out once deep neural networks became more mainstream. DeepPose [60], one of the earliest neural network methods, uses cascading deep neural networks to regress the (x, y) coordinates of joints in a holistic fashion. In [61], Tompson et al. joins a cascading convolutional network with a "position refinement" network to improve localization results. Convolutional Pose Machines (CPMs) [62] incorporate pose machines [63] into convolutional networks in a sequential model. The authors introduce intermediate supervision to address the vanishing gradient problem in multi-stage networks. "Stacked hourglass" is an convolutional network architecture proposed in [64] where repeated bottom-up and top-down processing create an architecture shaped like stacked hourglasses. The successive pooling and upsampling captures features at every scale, which is supportive of joint localization and joint relationships. OpenPose [65] is the first real-time, multi-person pose detection system that is able to detect not only body, but also hand, facial, and foot keypoints. It uses features extracted from the VGG19 [66] network to create confidence maps and Part Affinity Fields (PAFs) to learn to associate body parts with persons in an image. HR-NET [67] uses parallel multi-resolution subnetworks to maintain a high resolution representation throughout the network. The same research group proposes HigherHR-NET [68] using HR-NET as the backbone to solve bottom-up multi-person pose estimation with scale variation.

2.2.2 Visual Feature Extraction

Feature extraction is used to obtain relevant information in an image, and represent that information in a reduced dimension. If the features are carefully and correctly chosen, the reduced representation can be used for object classification. Traditional techniques for feature extraction started with corner detection [69], [70]. Those techniques were replaced once CNNs were more commonplace. SuperPoint [71] is a fully convolutional network that jointly computes interest point locations and descriptors in a single forward. D2-Net [72] uses a single convolutional network to be both a dense feature descriptor and a feature detector. Zhang and Lee in [73] propose a novel method for feature matching by using GNNs to transform feature points to local features. More commonly now, features are extracted using trained object classification networks. A classification network has feature extraction layers before classification layers. The classification and detection convolutional network, VGG16 [66], is used to extract features by taking the output of the last pooling layer of the network.

2.3 Graph Neural Networks

The bulk of advancements in neural networks have been centered around tasks that use data represented in the Euclidean space. This poses a challenge for unstructured data that are in the non-Euclidean domain. There are an increasing number of applications where the data needs to be represented in graph form to map the interdependency and complicated relationships between nodes. Some examples of applications that use graph representation in neural networks:

1. Recommendation systems [74]: The relationship the user has with the products (e.g., online shopping or movies) that they click on or purchase can be used to make recommendations on other products that could be of interest.

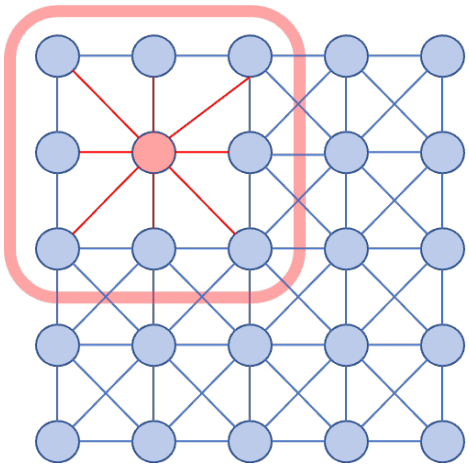
2. Optimization: Many optimization tasks are already in graph form, and can be easily applied on GNNs, such as the traveling salesman problem [75].

3. Knowledge graphs: Completing missing information in a knowledge database [76].

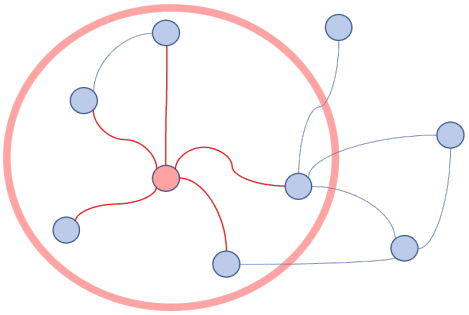
4. Chemistry: Molecules are very naturally represented in graph form, which has aided in researching existing molecular structure and discovering new ones [77] for uses in drug design [78].

2.3.1 Graph Convolutional Networks

Graph convolutional networks (GCNs) generalize the convolution operation from structured, grid-like data, to unstructured graph data. Figure 2.1a transforms an image into a grid with the nodes representing image pixels. The red filter is the convolution operation taking the average values of the red node and its neighbors, where the neighbors are determined by filter size. In comparison, Figure 2.1b performs a similar operation on unstructured data with the neighbors of the red node being determined by node edges.



(a) Convolution on 2-D data. The filter, red rectangle, decides which neighbors are aggregated onto the current (red) node. The number of neighboring nodes is a fixed number determined by the filter size.



(b) Generalized convolution operation on unstructured data. Neighboring nodes, determined by edge connectivity, are aggregated on the current (red) node. The number of neighboring nodes vary from node to node.

Figure 2.1. Convolution operation on structured, grid-like data, and unstructured, graph data. Image adapted from [79] © 2020 IEEE.

Figure 2.2 shows a basic GCN architecture with multiple, stacked graph convolutional layers. Each convolutional layer aggregates the graph nodes' feature information from their neighbors to extract high-level node representations. A non-linear activation, typically

ReLU, is applied to each hidden layer. The final output is the hidden representation of each node, which can be used for node classification. By adding pooling, and softmax layers to the architecture in Figure 2.2, we can transform the GCN from a node classifier into a graph classifier.

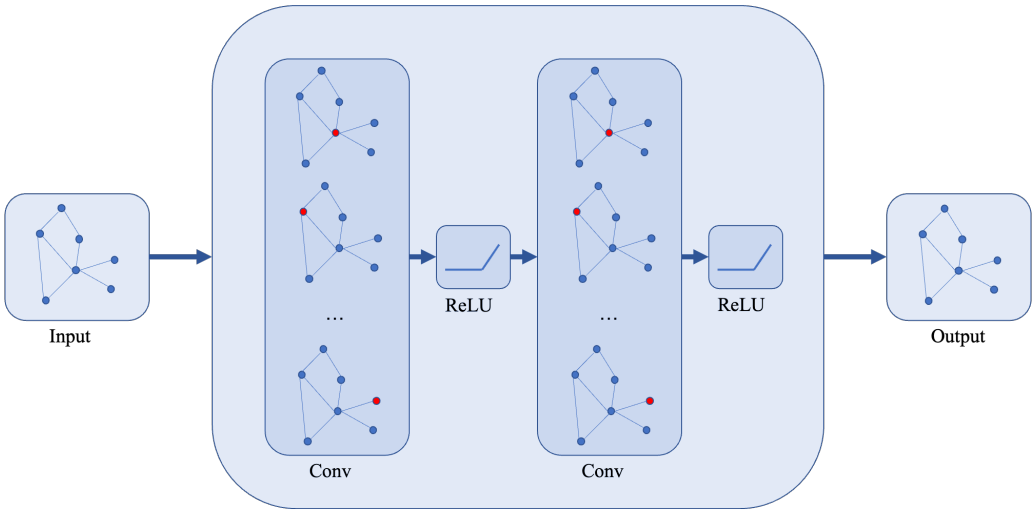


Figure 2.2. A two-layer graph convolutional network with ReLU activation functions. The inputs are the features and adjacency matrix of the graph. The output is the hidden representation of each node in the graph. Pooling and softmax layers can be added after the convolution layers to get the hidden representation of the graph.

Kipf and Welling in [80] remove assumptions on graph node similarity based on connectivity by conditioning their model on both the data and adjacency matrix (A), and introducing a renormalization trick to solve the the exploding/vanishing gradient problem. The Adaptive Graph Convolution Network (AGCN) [81] learn the underlying relationships between different nodes to learn a "residual" graph in addition to the original. The learnable graph convolutional network (LGCN) [82] aggregates neighbor node features by performing max pooling, and then uses CNNs to on the results to compute hidden representations. GraphSAGE [83] is an inductive framework that generates embeddings by sampling and aggregating local neighborhood nodes. Graph wavelet neural network (GWNN) [84] works by replacing the graph Fourier transform with a graph wavelet transform.

2.3.2 Graph Autoencoders

In the previous section, we introduced convolution operations on graphs to create GCNs. In this section, we show how GCNs can be used for building graph autoencoders (GAEs). GAEs embed nodes into the latent space, and decode graph information from the embeddings. GAEs can be used for two goals, network embedding, and graph generation.

Kipf's GAE proposed in [85] leverages GCNs to encode node structure, and feature information simultaneously. The decoder in the model reconstructs the adjacency matrix from the embedding matrix, but this could lead to overfitting. To solve this problem, Kipf and Welling propose Variational Graph Autoencoder (VGAE) in the same paper, which learns the distribution of data to limit overfitting. Adversarially Regularized Varational Graph Autoencoder (ARVGA) [86] uses GANs to train generative models to learn an encoder that produces a distribution of data similar to the prior distribution. Graph Variational Autoencoder (GraphVAE) [87] models nodes, and edges as independent variables with a convolutional encoder, and multi-layer perception as the decoder. Molecular Generative Adversarial Network (MolGAN) employs convolutional layers, GANs, and reinforcement learning to train the network to distinguish fake graphs from empirical data.

2.4 Datasets

Collecting and annotating datasets is a time and labor-intensive task, but annotated datasets are vital to advancements in deep learning. Supervised learning methods rely heavily on the annotation quality to produce good results. As AVs and deep learning methods gain more traction in research and the public eye, more publicly available datasets are being released to spur innovation. In this section, we will focus on AV datasets used in the previously reviewed papers. Table 2.1 offers a comprehensive summary of the datasets discussed here [21].

JAAD [45] is one of the first naturalistic datasets from the ego-view that contains rich annotations not only for the pedestrian, but also for the video context. The annotations for pedestrians include pedestrian bounding boxes, actions (e.g., walking, crossing, looking), appearance (e.g., clothing, number of pedestrians in the group), and attributes (e.g., age

group, gender, crossing location). Environment tags list infrastructure elements such as traffic lights, crosswalks, and stop lights. On top of the previously mentioned annotations, JAAD also has ego-vehicle action (e.g., slowing down, speeding up) annotated. JAAD's focus is the wide range of factors that contribute to crossing decisions. JAAD's annotations allow these factors to be considered when creating prediction models. However, JAAD's crossing annotations are crossing action labels that do not equate intention. In the simplest example, a pedestrian wants to cross the street, but does not by the time the ego-vehicle passes. The pedestrian action would be labeled as "not crossing," and it would be incorrect to infer that to intention.

The same team that created JAAD recognized the need for another dataset that measures crossing/not crossing as a measured intention rather than just action recognition. To measure intention, they conducted a study that aggregated human responses to pre-cut video clips of pedestrian activity that was recorded in a similar method to JAAD. The subjects were asked to view these clips up to a certain point, and then answer, "Does the pedestrian **want** to cross?" The aggregated responses are used as a measurement of the probability the pedestrian has crossing intention. The intention probability is only part of the annotations in their dataset Pedestrian Intention Estimation (PIE). PIE also includes many of the same types of annotations as in JAAD, and with more descriptors (e.g., stop light color). Additionally, instead of ego-vehicle action labels, PIE has ground-truth ego-vehicle speed, gps coordinates, and heading direction collected from the vehicle. PIE is the first of its like to propose a method of measuring intention that is not annotating action labels. PIE's proposal opens up the discussion on how intention can be measured when collecting the ground-truth is not a feasible task for naturalistic datasets. In Section 3.2, we will further describe how crossing intention is measured, and justify our choice to use this dataset for our model.

BDD-100K [88] is an enormous dataset with over 1,000 hours of diverse video, and annotations that are not commonly seen in other datasets. While the dataset was not specifically collected for pedestrian-vehicle interactions, the length and diversity of video collected guarantees all driving scenarios and environments are captured. Using BDD-100K would require more pre-processing than either JAAD or PIE, because each video clip only has one frame annotated. However, BDD-100K offers a different set of annotations compared to JAAD or

PIE. In addition to pedestrian and road object bounding boxes, BDD-100K has lane marking, drivable area, and full frame segmentation. The segmentation annotations can be used to guide preventative measures the ego-vehicle take to avoid dangerous vehicle-pedestrian interactions.

Another massive dataset is STIP [50] with 900 hours of video footage. STIP is collected for predicting pedestrian intention, so the videos are pedestrian dense. Being such a massive dataset, pedestrian bounding boxes are annotated at 2 fps, and interpolated using an off-the-shelf algorithm to get annotations for every frame. Like JAAD, the crossing/not crossing actions are annotated for each frame. Unique to STIP, the video footage was collected using three cameras covering the front, left, and right of the ego-vehicle. Having a wider range of vision can increase observation lengths, which can assist in getting earlier predictions.

TITAN [44] focuses on pedestrian and vehicle annotations using 50 labels to classify their actions and attributes. The pedestrian action labels are grouped in a five-tier complexity hierarchy that provides context to the motivation of pedestrians. The first tier describes posture actions such as sitting, standing, and bending. The second tier includes actions such as bicycling, exiting, and crossing that add context to the first tier labels. The third tier are complex contextual actions that involve a sequence of basic actions. Lastly, the fourth, and fifth tiers are transportive and communicative actions respectively. By knowing the context of the pedestrian's actions, there is more information to infer the motivation on the pedestrian's trajectory.

Daimler [89] is a smaller dataset that is collected in a controlled environment with designed vehicle-pedestrian interactions. Actors are recorded portraying pedestrians performing a series of scripted actions in front of the ego-vehicle. Naturalistic interactions are preferable for training networks that could be used on the road, but by directing the pedestrian's actions, the Daimler dataset can unequivocally state the ground-truth intentions of the pedestrian.

Table 2.1. Summary of publicly available datasets that are collected and annotated towards computer vision advancements in AV applications. This table is reproduced from [21] © 2021 IEEE.

Dataset	Scene Description	Duration & Tracks	Annotations & Sampling Rate
JAAD (2017)	daytime streets of downtown centers in North America and Eastern Europe	346 clips (5-10 seconds)	pedestrian bounding boxes, pedestrian actions, pedestrian attributes, pedestrian appearance, environment tags, ego-vehicle vehicle action (30 fps)
PIE (2019)	daytime urban streets of Toronto, Canada	6 hours split into 10 minute chunks	pedestrian bounding boxes, pedestrian behavior tags, traffic agent bounding boxes, intention annotations, ego-vehicle data (30 fps)
BDD-100K (2018)	coastal cities in the United States under diverse weather and time of day conditions	100,000 clips (40 seconds long)	pedestrian bounding boxes, road object bounding boxes, image tagging, lane markings, drivable areas, instance segmentation (1 keyframe at the 10th second)
Daimler (2013)	designed vehicle-pedestrian interactions with one pedestrian, and no occlusions	19,612 stereo image pairs (68 pedestrian sequences)	pedestrian bounding boxes, pedestrian detector measurements, vehicle data, event tags, time-to-event labels (16 fps)
STIP (2020)	dense traffic cities in the United States	900 hours	pedestrian bounding boxes, crossing/not crossing labels (2 fps ground truth to 20 fps tracking)
TITAN (2020)	urban scenes in Tokyo, Japan	700 clips (10-20 seconds)	pedestrian bounding boxes, traffic agent bounding boxes, vehicle states, pedestrian actions, pedestrian age groups (10 fps)

2.5 Summary

In this chapter, we summarize the literature on human behavior estimation, human pose estimation, image feature extraction, GNNs, and datasets used for predicting human behavior from the ego-view. In the remaining chapters, we apply the methodology reviewed here for our pedestrian crossing intention model, and compare our results with the state of the art.

3. BASIC FRAMEWORK OF CROSSING INTENTION PREDICTION MODEL

In this chapter, we introduce our proposed algorithm model and methodology for predicting pedestrian crossing intention from the ego-view. In Section 3.1, we present a high-level description of our model and its components. Then in Section 3.2, we review the benchmark dataset, PIE, that we used to train, and evaluate our model. This is followed by Section 3.3 which goes over the evaluation metrics used to evaluate the performance of our model. Lastly, we conclude this chapter with Section 3.4.

3.1 Network Architecture

As shown in Figure 3.1, our model is broken up into these four major components:

- Pedestrian pose estimation embedding
- Traffic object/agent appearance embedding
- Graph autoencoder module
- Pedestrian crossing intention prediction module

Both the pose estimation, and traffic object/agent appearance embedding modules take cropped images obtained through bounding box coordinates as input. For pose estimation, only pedestrian bounding boxes are considered, whereas all traffic objects/agents are processed for appearance embedding. A full list of annotated objects can be found in Section 4.2.2. Since there have been significant performance advances in algorithms that complete computer vision tasks, we use pre-trained models to help us pre-process our data. To estimate pose, we use High Resolution Net (HR-NET) [67] pre-trained on the COCO dataset [90]. It is open source, well documented on Github, used as the backbone of other architectures, and has competitive results on pose estimation challenges. To obtain appearance embeddings, we use VGG16 [66] pre-trained on ImageNet [91]. Keras, a Python deep learning API, has a VGG16 deep learning model with pre-trained weights that makes applying VGG16 onto PIE images a simple task. VGG16 is trained to be an image classifier, but we do not need

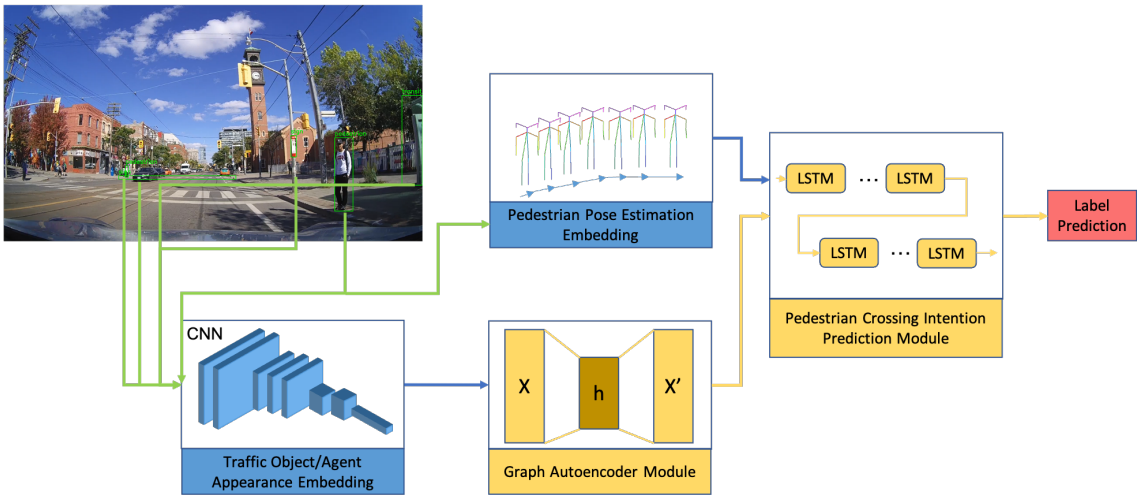


Figure 3.1. Our proposed model for pedestrian crossing intention prediction broken up into four modules: traffic object/agent appearance embedding, pedestrian pose estimation embedding, graph autoencoder module, and pedestrian crossing intention prediction module. Cropped images of traffic objects/agents are inputs to the two embedding modules for visual feature extraction. Label prediction can either be for only Y_I or Y_I and Y_C jointly.

to classify our cropped images, because we already have the ground truth labels. Instead, we take the results of a dense layer before the classification layer as the object appearance embedding.

Each annotated object in an image has its own appearance embedding based on the cropped image of the object. Figure 3.2 shows a sample frame from PIE with the annotated bounding boxes visualized. The objects are cropped according to the bounding boxes. The appearance embeddings of each object in the image are translated into a star graph centered around the pedestrian of interest. Figure 3.3 visualizes the pedestrian-centric star graph with the cropped images in place of the embeddings. The star graph models the spatial relationship the pedestrian has to its surrounding objects. Each frame has its own, unique star graph so to increase the speed of our crossing intention prediction model, we reduce the dimensionality of the star graphs. To do so, we use a GAE module to learn to reconstruct the star graphs. The latent space embedding of the trained GAE module is then an accurate, lower dimension representation of the star graphs.

Figure 3.2. Sample frame from PIE with traffic agents' and objects' bounding boxes annotated.

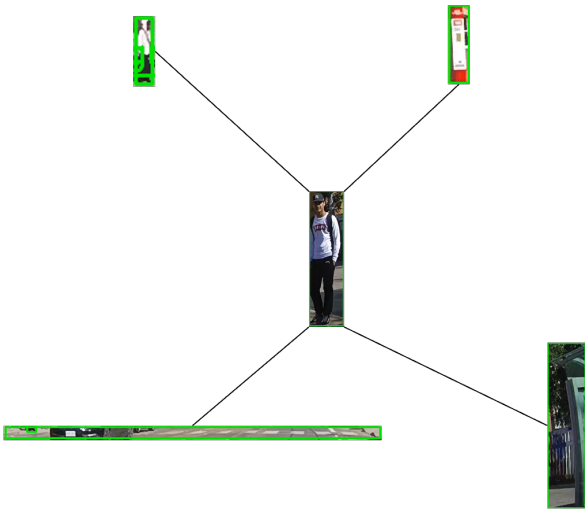


Figure 3.3. A pedestrian-centric star graph with the cropped images representing the features extracted using VGG16. Every object/agent in the image is connected to the pedestrian.

To predict crossing intention, we use an LSTM encoder-decoder configuration in the crossing intention prediction module. The latent space embeddings from the GAE combined with the LSTMs will model the spatiotemporal relationship of the traffic objects/agents in the scene. Additionally, the pose estimation of the pedestrian will provide further context to the pedestrian's behavior and action. The LSTMs will observe 15 frames, and predict the pedestrian's crossing intention for the remaining of the sample length, which is 45 frames.

3.2 PIE Dataset

We choose to use the PIE dataset, because it is the first of its kind to provide a method of quantifying, and labeling crossing intention as opposed to crossing action. As previously mentioned in Section 2.4, other existing benchmark datasets either annotate crossing action or bounding box trajectory, neither of which indicate the pedestrian's intention.

To measure crossing intention, Rasouli et al. conducted a subject survey framed around the question "Does this pedestrian **want** to cross the street?" Subjects were asked to view a pre-sliced video clip that pauses at a pre-determined critical crossing point, and answer the previously stated question. Subjects could answer the question on a 5-point scale with 5 and 1 being definitive 'yes' and 'no', respectively. The dataset contains 1,842 videos, which were viewed in its entirely by 5 subjects. To gather more responses, Rasouli et al. crowdsourced responses on Amazon Mechanical Turk (AMT) to gather another 10 responses per video. In total, each video was viewed by 15 subjects. The responses were aggregated to produce a crossing intention probability for each pedestrian. The closer the probability is to 1, the more likely the pedestrian has crossing intention. This is the intention label our model will predict on.

In addition to pedestrian crossing intention data, PIE also has rich pedestrian behavior, and demographic annotations. Pedestrian behavior includes movement, gaze, hand gesture, and crossing action. Demographics include age group, and gender. For the ego-view scene, various traffic agents/objects that may be critical to the pedestrian's crossing behavior are annotated with bounding boxes. Figure 3.2 shows a sample frame from PIE with the annotated bounding boxes. Any attributes of the traffic agent/object other than location are also

noted. For example, the color of a traffic light since that will dictate right-of-way. While not used in our algorithm, Rasouli et al. also calibrated the ego-vehicle's sensor data (e.g., heading angle, latitude, longitude, speed) to include in PIE.

3.3 Evaluation Metrics

To evaluate our proposed model, we have to evaluate the performance of both the GAE, and crossing intention prediction module. HR-NET and VGG16 are pre-trained, and state of the art models so we do not need to evaluate them.

The goal of the GAE module is to find a lower dimension representation, H, of the visual appearance star graphs, X. The closer the reconstruction results X' from H is to X, the better H represents X. To measure the difference between X and X', we use mean squared error (MSE) to evaluate n data points

$$MSE = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}(X_{\mathrm{i}} - X'_{\mathrm{i}})^2 \tag{3.1}$$

The pedestrian crossing intention prediction module is trained to perform two types of classifications. The first is a binary classifier that predicts pedestrian crossing intention, Y_I, as defined by PIE. Y_I has two labels: 0 for no crossing intention, and 1 for having crossing intention. The second type of classification is a two-class multi-label prediction. In addition to Y_I, this classifier jointly predicts crossing action, Y_C. Y_C is defined as the pedestrian's current crossing action, and also has two labels: 0 is not crossing, and 1 is crossing.

We evaluate the performance of both classifiers. The results of the binary classifier on Y_I can be directly compared with the state of the art [42] using accuracy and F1 score. The multi-class multi-label classifier is evaluated using the same metrics, but for its performance on predicting Y_I and Y_C together, and separately for each class. The testing set is highly imbalanced ($1 \gg 0$), so we will additionally compare balanced accuracy, recall, and precision scores of each label for each class. These evaluation metrics can be expressed in terms of true positive (TP), false positives (FP), true negatives (TN), and false negatives (FN).

$$precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$recall/sensitivity = \frac{TP}{TP + FN} \tag{3.3}$$

$$specificity = \frac{TN}{TN + FP} \tag{3.4}$$

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.5}$$

$$balanced\ accuracy = \frac{sensitivity + specificity}{2} \tag{3.6}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.7}$$

3.4 Summary

This chapter outlines our proposed method, and lays the foundation for the next four chapters. We summarize the different components of our model, and how they feed into one another. This includes data preprocessing, and prediction tasks. We also describe the benchmark dataset which we train and test our model on. Finally, we identify the evaluation metrics we use to evaluate our proposed model's performance.

4. EXTRACTING RICH VISUAL FEATURES OF THE TRAFFIC SCENE

This chapter outlines the steps we take to preprocess the images from PIE to extract meaningful visual features from the scene. Human drivers follow traffic laws, and take cues from the environment to make their driving decisions. Without such context, drivers would not know how to behave on the road or be able to anticipate other road users' actions. Based on how human drivers make decisions, we theorize a context-driven model can improve pedestrian intention prediction. In Section 4.1 we describe the process of extracting 2-D pose estimation from pedestrian images. Then in Section 4.2 we begin our description of the feature extraction process for traffic objects relevant to decision making. Finally, we conclude this chapter in Section 4.3.

4.1 2-D Human Pose Estimation of Pedestrians

There are two approaches to human pose estimation, bottom-up or top-down. In the bottom-up approach, keypoints are found first, and then mapped to different human bodies in the image. As opposed to the top-down approach where humans are first detected through bounding boxes, and then the keypoints are estimated within the bounding box. In our experiments, we use HR-NET for human pose estimation, which is a top-down method that uses Faster R-CNN [92] for object detection.

Currently, the top-down method is the preferred approach for tackling human pose estimation tasks. However, due to top-down methods needing a separate object detection network on top of the pose estimation network, top-down methods are computationally intensive. This is opposed to a faster bottom-up method that localizes identity-free keypoints through anatomical heatmaps, and then groups the points together to form a pose estimation. While top-down may be slower than bottom-up, top-down generally performs better due to its ability to normalize bodies to the same scale from the results of the object detection step. When there are scale variations between persons in the same image, bottom-up methods cannot account for the differences.

4.1.1 HR-NET Architecture

Sun et al. proposed HR-NET in IEEE's 2019 Computer Vision and Pattern Recognition (CVPR) conference. It immediately broke records in three COCO benchmarks: keypoint detection, multi-person pose estimation, and pose estimation. While HR-NET was primarily proposed for pose estimation, it can also be applied to a wide range of computer vision tasks, such as image classification, scene segmentation, object detection, and facial landmark.

The novelty of HR-NET lies in that fact that it is a parallel structure that can maintain high resolution representations throughout the network. At the time of HR-NET's proposal, most methods were connected in series, and high resolution representations were obtained from the low resolution representations in a high-to-low resolution network. The redrawn HR-NET architecture in Figure 4.1 shows the high resolution representation branch (1×) calculated in parallel as the lower resolution representation branches (2×, and 4×). For the parallel branches to receive information from one another, Sun et al. introduce "exchange units" that fuse multi-resolution representations repeatedly, thus maintaining a high resolution representation.

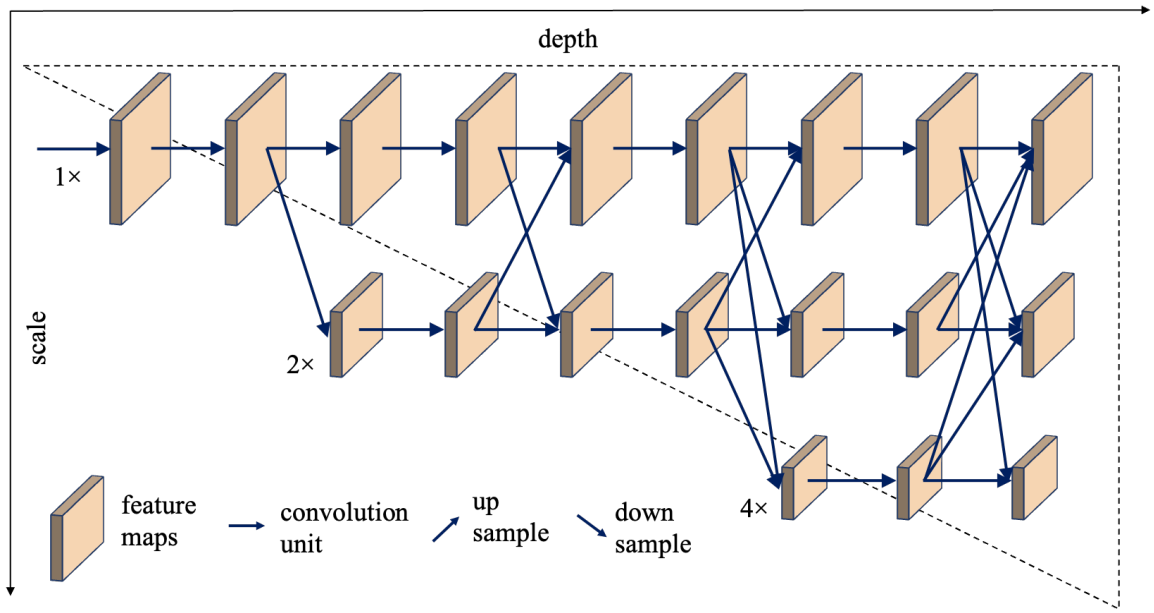


Figure 4.1. Network architecture of HR-NET. The high resolution branches are in parallel with the lower resolution branches to maintain a high resolution representation throughout the network. Redrawn from [67] © 2019 IEEE.

HR-NET is trained on the COCO keypoint detection dataset, and the MPII Human Pose dataset. For our purposes, we used the open-source implementation of HR-NET trained on the COCO keypoint detection dataset. PIE annotations included pedestrian bounding boxes so we directly used the ground-truth data instead of implementing an object detection algorithm. The 17 keypoints predicted for each pedestrian bounding box is in Table 4.1. Figure 4.2 shows some of the images in the COCO keypoint detection dataset with the ground-truth human pose estimation on top.

Table 4.1. Definition of COCO human keypoint estimation joints. There are 17 joints that are explicitly defined in COCO.

Index	Keypoint
0	nose
1	left eye
2	right eye
3	left ear
4	right ear
5	left shoulder
6	right shoulder
7	left elbow
8	right elbow
9	left wrist
10	right wrist
11	left hip
12	right hip
13	left knee
14	right knee
15	left ankle
16	right ankle

4.1.2 Results

In PIE, we processed 740,901 unique poses for about 1,800 pedestrians. Figure 4.3 shows some examples of the human pose prediction results. We can see from the results that the poses are descriptive of what the pedestrians are doing. The poses not only provide context for movement behavior (standing or walking), but also further context for higher-level actions such as Figure 4.3(b) where the pedestrian is looking down at their phone.

Figure 4.2. Human keypoint illustrations from COCO keypoint estimation dataset. Images are copied from © 2015 COCO Consortium.

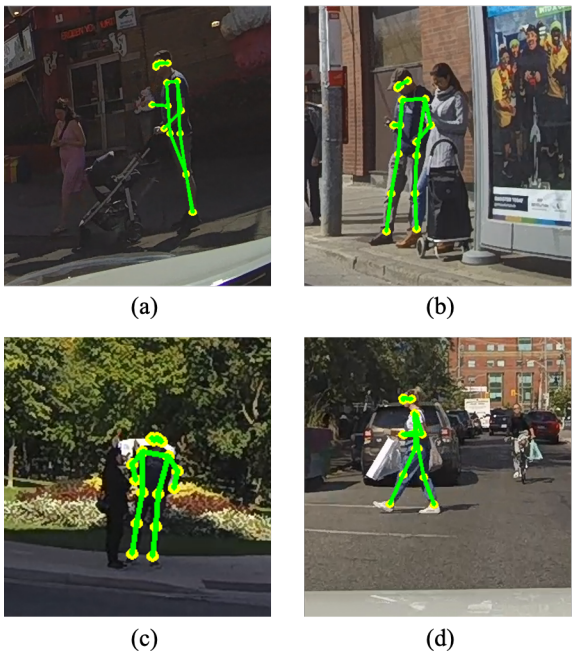


Figure 4.3. Samples of keypoint estimation results from PIE using HR-NET to estimate human pose. (a) Side view of a pedestrian standing, and pushing a stroller. (b) Angled front view of a pedestrian standing at a transit station while looking down at their phone. (c) Front view of a pedestrian standing. (d) Side view of a pedestrian crossing the street.

4.2 Feature Extraction of Traffic Objects

At its core, feature extraction of an image is a dimension reduction technique that preserves the relevant information of the image. On the computer, images are stored as 3-D arrays defining the red, green, and blue (RGB) components of each pixel. This is how human eyes view images, but it is not the most efficient way for machines to store, and "see" images in neural networks. Feature extraction removes redundant information, and keeps the important features to be used for computer vision tasks such as object recognition.

In our experiment, we use a VGG16 network pretrained on ImageNet [91] for image classification to extract visual features. We do not need to classify the traffic objects so instead of getting the results from the last layer (softmax layer) of VGG16, where it will be classified into 1 of the 1,000 classes in Imagenet, we retrieve the results from an earlier layer of the network. Figure 4.4 shows the various convolution, and fully connected layers of VGG16.

4.2.1 VGG16 Architecture

Simonyan and Zisserman submitted VGG16 to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014. While VGG16 was only the runner-up model in that year's image classification challenge, it was the first model to get an error rate under 10% in the history of the competition. It is still a widely used model that is made available through Keras Applications with pre-trained weights.

As seen in Figure 4.4, VGG16 takes images that are resized to size (224, 224, 3) as network inputs. Each convolution layer has filters of size 3×3 with a stride of 1, and each max pooling layer has filters of size 2×2 with strides of 2. This pattern holds until the fully connected layers followed by the softmax layer. In total, VGG16 has 16 layers with weights, and the network has approximately 138 million parameters.

To extract the visual features of traffic objects/agents in PIE, we use the ground-truth bounding box annotations to crop, and resize the images to the required dimensions for VGG16. We directly took the 1×4096 vector from layer "fc6" to be the feature representation. Each traffic object/agent has its own feature representation for every frame it is in.

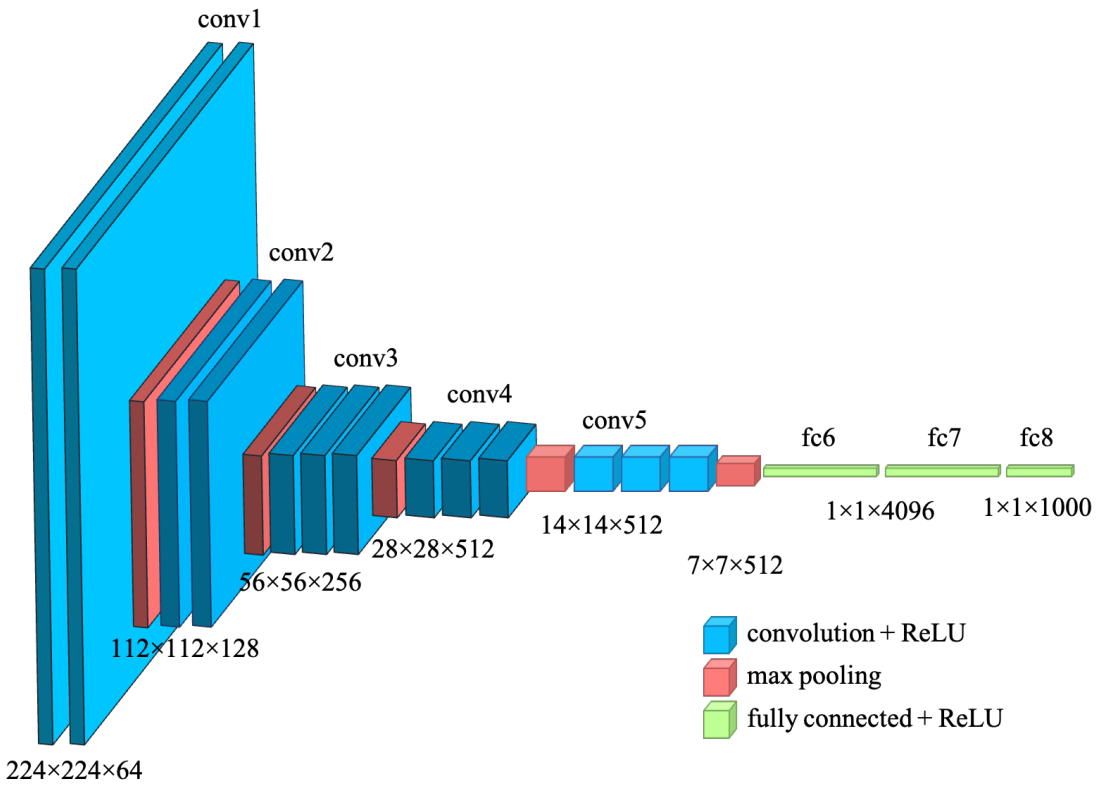


Figure 4.4. Network architecture of VGG16.

4.2.2 Traffic Objects of Interest and Results

The traffic objects and agents that we are interested in for feature extraction are listed in Table 4.2. These are annotated in PIE with bounding boxes, and additional attributes that provide more context. Figure 4.5 shows image samples of traffic objects/agents with their bounding boxes annotated around them. Table 4.3 shows the number of traffic objects and agents in PIE that are processed for visual feature extraction using VGG16.

Table 4.2. Summary of traffic objects/agents annotated with bounding boxes in PIE.

Label	Additional Attributes
pedestrian	action: standing, walking gesture: hand_ack, hand_yield, hand_rightofway, node, other look: looking, not_looking cross: cross, not_crossing, crossing_irrelevant
vehicle	type: car, truck, bus, train, bicycle, bike
traffic_light	type: regular, transit, pedestrian state: red, yellow, green
sign	type: ped_blue, ped_yellow, ped_white, ped_text, stop_sign, bus_stop, train_stop, construction, other
crosswalk	None
transit_station	type: bus, streetcar

Table 4.3. Number of traffic object/agent instances with features extracted using VGG16.

Label	Number of Instances
pedestrian	725,763
vehicle	526,439
traffic_light	310,143
sign	49,621
crosswalk	135,717
transit_station	13,207
Total	1,760,890

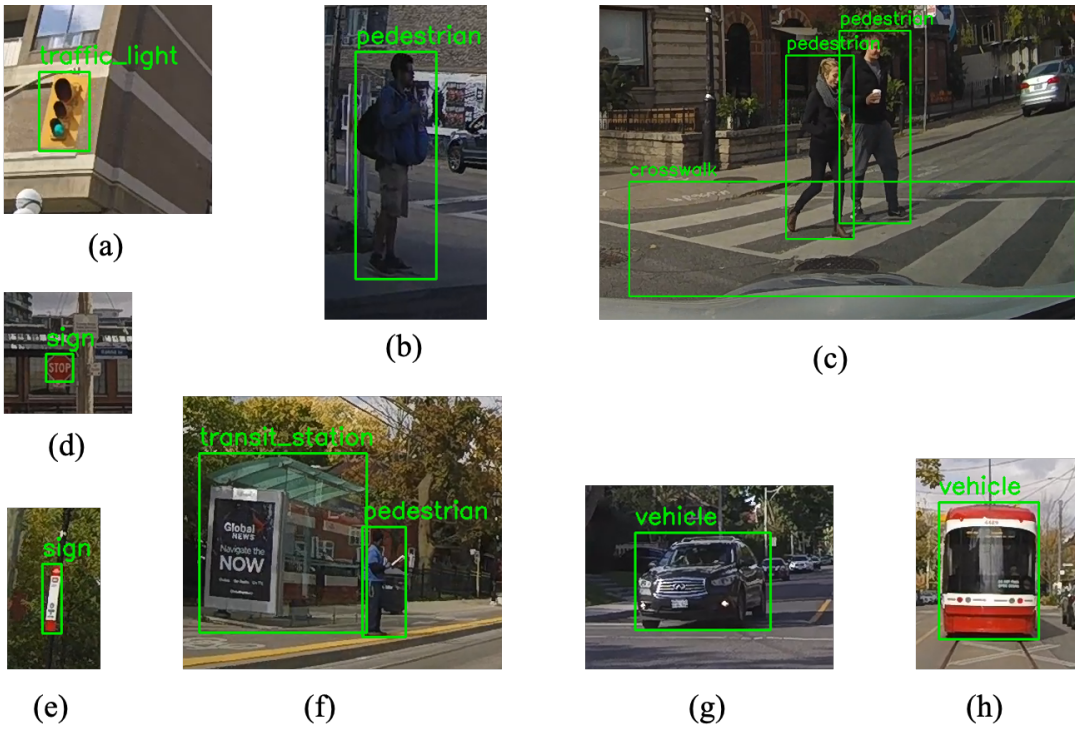


Figure 4.5. Samples of bounding box annotations in PIE.(a) traffic light (green). (b) pedestrian (standing). (c) two pedestrians (walking) at a crosswalk. (d) sign (stop). (e) sign (bus stop). (f) pedestrian (standing) at a transit station. (g) vehicle (car). (h) vehicle (train).

4.3 Summary

In this chapter, we walk through our two main processes for obtaining rich visual features for the ego-view scene. For pose estimation, we use HR-NET trained on the COCO keypoint detection dataset, because it is a top-down method that performs well on PIE. To extract visual appearance features for traffic objects, we use a pre-trained VGG16 network, and retrieve feature vectors from one of the fully connected layers of the model. Pose estimation provides context to the behavior of the pedestrian, while appearance features provide context to the scene. Both of these visual features will be used to predict on pedestrian crossing intention in later chapters.

5. GRAPH MODELING OF THE TRAFFIC SCENE

In the previous chapter, we introduced VGG16, and described the process for extracting rich visual features of traffic objects/agents that impact driving decisions, and pedestrian behavior. The appearance features by themselves are not sufficient to represent the scene as a whole. In this chapter, we use the appearance features on GCNs [80] to model the spatial relationships between the traffic objects. Our goal is to find an effective, and efficient method for modeling spatial relationships in a lower dimension embedding than the feature vectors extracted from VGG16. To get a lower dimension embedding, we use an autoencoder variant of GCNs that can learn a lower dimension representation of the graph embedding.

This chapter is organized as follows. We start by introducing GNNs, and how to transform them into Graph Convolutional Networks (GCNs) in Section 5.1. We then introduce two GCN models: GCNConv [80] and GraphConv [93] for the basis of our autoencoder in Sections 5.2 and 5.3, respectively. We evaluate the autoencoders' performance in Section 5.4. Finally, we conclude the chapter with Section 5.5

5.1 Graph Neural Networks

Typical neural networks such as CNNs or RNNs are not well-suited to work on arbitrary data structures, such as graphs and networks. As a result, GNNs were developed to generalize neural network methods on irregular structures. GCNs being one of the variants of GNNs. GCNs have a similar convolution operation as CNNs do, hence the word "convolutional" in GCNs. The convolution operation in CNNs take a filter and slides it across the entire image to learn the features. This is similarly done in GCNs to learn the features of neighbor nodes, the only difference being GCNs are generalized for unstructured data.

In a GCN model, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ takes as input:

- Feature matrix X in shape $N \times D$ (N: number of nodes, D number of input features) such that x_{i} is the feature description for node i
- Adjacency matrix A to structure the graph in matrix form

and produce feature matrix Z in shape $N \times F$ (F: number of output features for each node) at the node level. Then we can generalize each neural network layer as

$$H^{(l+1)} = f(H^{(l)}, A), \tag{5.1}$$

where L is the number of layers, and $H^{(0)} = X$ and $H^{(L)} = Z$.

GCNs can be categorized into two major categories: spectral GCNs, and spatial GCNs. Spectral GCNs are based on spectral graph theory [94], which is theorized on graph signal processing. This means we perform an Eigen decomposition on the Laplacian matrix in the Fourier space. This is a costly, and inefficient method compared to spatial GCNs [95] that look to its neighbor nodes to understand the node's properties. Typically spatial GCNs are preffered due to its flexibility and low computational costs.

5.2 GCNConv

While typically spatial GCNs are preferred due to its flexibility, and low computation cost, we will focus on the spectral-based GCN implementation proposed in [80]. Kipf and Welling propose GCNConv, a simplified approach that achieves faster training time, and higher prediction accuracy through adjusting the convolution operation.

Using the same notations as Equation 5.1, we define layer propagation as

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{5.2}$$

where $W^{(l)}$ is the weight matrix for layer l, and σ is a non-linear activation function. In [80], the authors slightly adjust Equation 5.2 to

$$f(H^{(l)}), A) = \sigma(\hat{D}^{\frac{-1}{2}}\hat{A}\hat{D}^{\frac{-1}{2}}H^{(l)}W^{(l)}) \tag{5.3}$$

where $\hat{A} = A + I$ (I is the identity matrix), and $\hat{D}$ is the diagonal degree matrix of $\hat{A}$. The adjustments are to account for the limitations in Equation 5.2:

- Add the identity matrix to A so that the sum of the neighboring feature vectors include the node itself
- Symmetric normalization of A to scale all multiplication operations

5.2.1 Autoencoder Architecture

In Chapter 4.2, we summarized how we extract size (1×4096) feature vectors from traffic objects using VGG16. To construct the feature matrix X, we concatenated the feature vectors x_{i} along the y-axis to create an $(N \times 4096)$ matrix. The maximum number of objects in a frame in PIE is 32, so N can be any value between 1-32. Later on, Xs that are smaller than (32×4096) are padded with zeros to ensure all input tensors are the same shape. Matrix A is constructed by connecting all objects in the scene to the pedestrian we are predicting intention on.

For our GCNConv autoencoder, we propose a 2-layer GCN, which performs two propagations (Equation 5.3) in the forward pass to embed our Xs from $(N \times 4096) \rightarrow (N \times 512) \rightarrow (N \times 256)$ encodings. Figure 5.1 shows our proposed GAE with GCNConv layers, and ReLU activations.

5.3 GraphConv

In [93], Morris et al. propose a generalization of GNNs (k dimension-GNNs), based on k-Weisfeiler-Leman (WL), that takes higher-order graph structures at multiple scales into account so we can exploit the hierarchical organization of most real-world graphs. We name their GNN network as GraphConv. Since our graphs are in relatively simple star shapes, we utilize 1-k-GNNs for neighborhood aggregation with a simple skip connection.

5.3.1 Autoencoder Architecture

For our autoencoder using GraphConv as our convolutional layers, we propose a 3-layer GCN with mean aggregation of neighborhood nodes at each layer, and ReLU activation functions after the first two layers. The embedding of the input data, X, in the encoder will

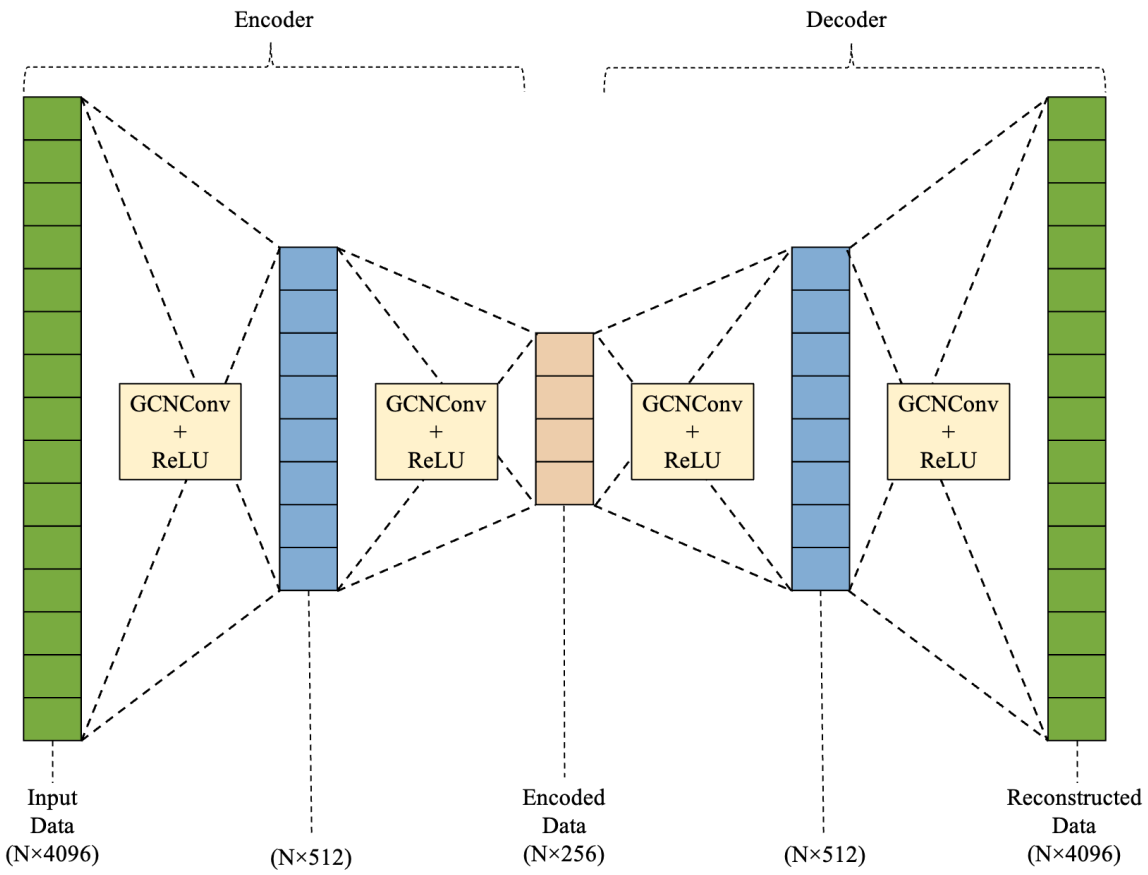


Figure 5.1. Network architecture of GAE with GCNConv layers. The encoder portion of the GAE embeds the input data into a lower dimensionality using GCNConv convolutional layers. The decoder attempts to reconstruct the encoded data back into the original input. Through convolution and learning, the encoded data becomes a lower dimension representation of the original input data.

transform from $(N \times 4096) \rightarrow (N \times 512) \rightarrow (N \times 512) \rightarrow (N \times 256)$. A symmetrical decoder is used to calculate the reconstruction error for X'.

5.4 Experimental Results

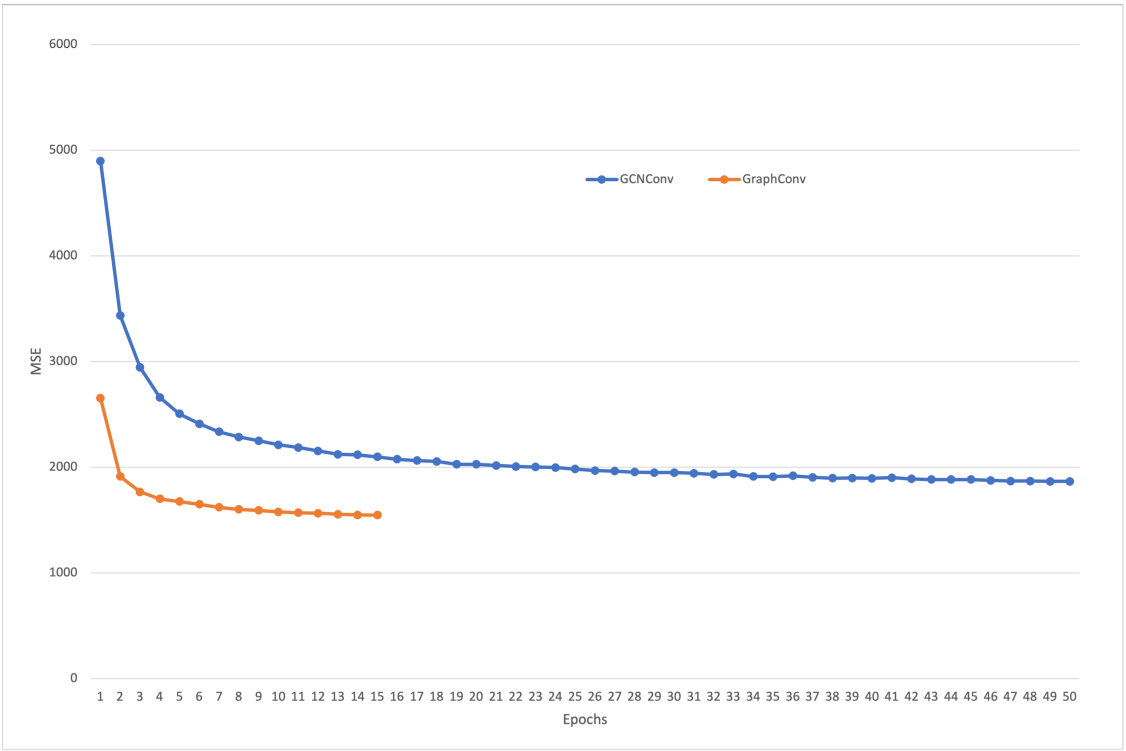
In this section, we will only compare the results of the autoencoder reconstruction of X. We will compare the crossing intention prediction results using the autoencoder encoding later in Section 6.3. Here, we want to minimize the MSE between the input data, and

reconstructed data so that the autoencoder encoding is an accurate representation of the input data.

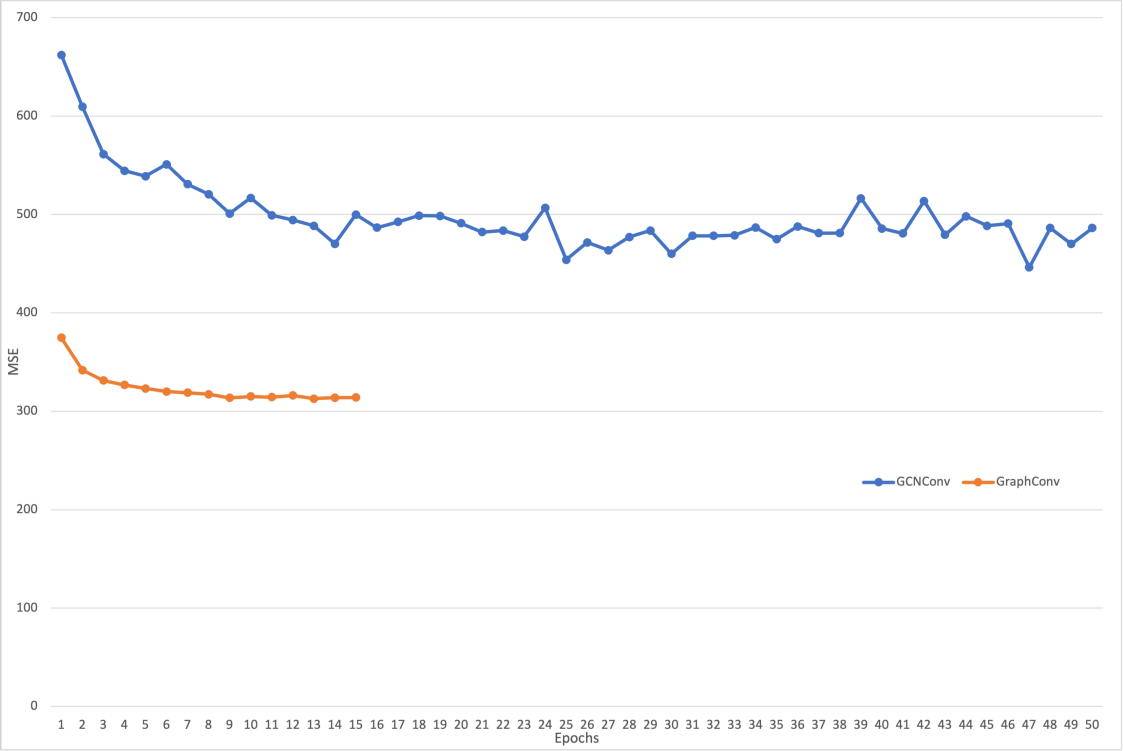
The only parameter that needs to be tuned for the autoencoders is the learning rate. Both GCNConv, and GraphConv autoencoders use a learning rate of 0.001. Figure 5.2 compares the training and testing results for our best GCNConv, and GraphConv autoencoder models. GCNConv was trained for 50 epochs, while GraphConv was only trained for 15 epochs, because GraphConv plauteaus smoothly starting at epoch 15, but GCNConv has a sporadic testing curve. GCNConv was trained longer to see how the testing curve would behave in the longer training environment. For both the training, and testing sets, GraphConv beats out GCNConv in data reconstruction accuracy. The rate as which MSE plateaus is similar for GraphConv, and GCNConv, but GraphConv starts at a much lower MSE value than GCNConv. Computation time-wise, to train 15 epochs, GraphConv took 80 minutes whereas GCNConv took 68 minutes. The difference in training time is not impactful enough to negate better results GraphConv achieves.

5.5 Summary

In this chapter, we present convolutional layers GCNConv and GraphConv for building GCNs. We use GCNConv and GraphConv separately in two GAEs to encode our scene features X. Through the comparison of MSE for reconstruction error of X from X', we know our GraphConv GAE is a better model for graph encoding. Later in Chapter 6, we will use results from both GAEs to predict pedestrian crossing intention.



(a) Training set MSE for GCNConv, and GraphConv GAE.



(b) Testing set MSE for GCNConv, and GraphConv GAE.

Figure 5.2. Training and testing error for GCNConv, and GraphConv GAEs.

6. PREDICTING PEDESTRIAN CROSSING INTENTION

In the previous chapters, we walked through the steps we took to process the raw images of PIE to prepare the visual features that we will use for predicting pedestrian crossing intention. In this chapter, we will use those visual features as inputs to an LSTM prediction model configured in an encoder-decoder configuration.

The organization of this chapter is as follows. In Section 6.1, we will review the overall network of our prediction model, integrating the modules discussed in previous chapters with the prediction module presented here. This will be followed by Section 6.2, which will define the prediction targets for the LSTM network. Then, in Section 6.3 we will present our prediction results. Finally, we conclude the chapter with Section 6.4.

6.1 Spatiotemporal Model For Prediction

Our proposed intention prediction module, shown in Figure 6.1, consists of an encoder LSTM layer connected to a fully connected layer followed by the decoder LSTM connected to another fully connected layer. The last fully connected layer makes the classification. The encoder LSTMs only return the last cell state, and its hidden states, c_t and h_t, which are used to initialize the LSTM decoder cells.

Three visual features are used as inputs in our prediction model. The pedestrian-centric graph embeddings that model pedestrian-environment relationships from Chapter 5 are the inputs to the encoder LSTM cells. For each observed frame (15 for each sample), there is an array of size (32×256) that represents the graph embedding for that frame. 32 is the maximum number of objects/agents in an image in PIE, and for images that have less than 32 objects/agents, we pad the extra nodes with zeros. 256 is the size of the feature vector from the GAE's latent space. The graph embedding needs to be flattened into shape (1×8192) before we can use them in the LSTM cells. For 15 observed frames, we then have an array of shape (15×8192) that represents the temporally changing pedestrian-environment relationship.

The other two visual features are pedestrian bounding boxes, and human pose estimation. Each observed frame has one pedestrian bounding box, and one human pose estimation for

the pedestrian that we are predicting crossing intention on. The pedestrian bounding box is represented with $(x_{tl}, y_{tl}, x_{br}, y_{br})$, so a bounding box is shape (1×4). Human pose estimation is represented with 17 keypoint joints with xy-corrdinates each, so a pose is shape (1×34). The pedestrian's bounding box and pose estimation are concatenated to form an array of shape (1×38). Just like graph embeddings, for 15 observed frames, we then have an array of shape (15×38). This array is concatenated with the output from the fully connected layer of the encoder LSTMs to become the input for the decoder LSTMs. The decoder LSTMs are initialized with the hidden states of the last encoder LSTM cell. The LSTM cells have 128 hidden units, $softsign$ activation, 0.4 dropout, and 0.2 recurrent dropout. These parameter values are determined later in Section 6.3.3 when we tune parameters on our model. After the LSTM decoder is a fully connected layer that is used to classify pedestrian crossing intention.

Figure 6.2 combines the traffic object/agent appearance embedding module, pedestrian pose estimation embedding module, graph autoencoder module, and pedestrian crossing intention prediction module in a full model architecture illustration.

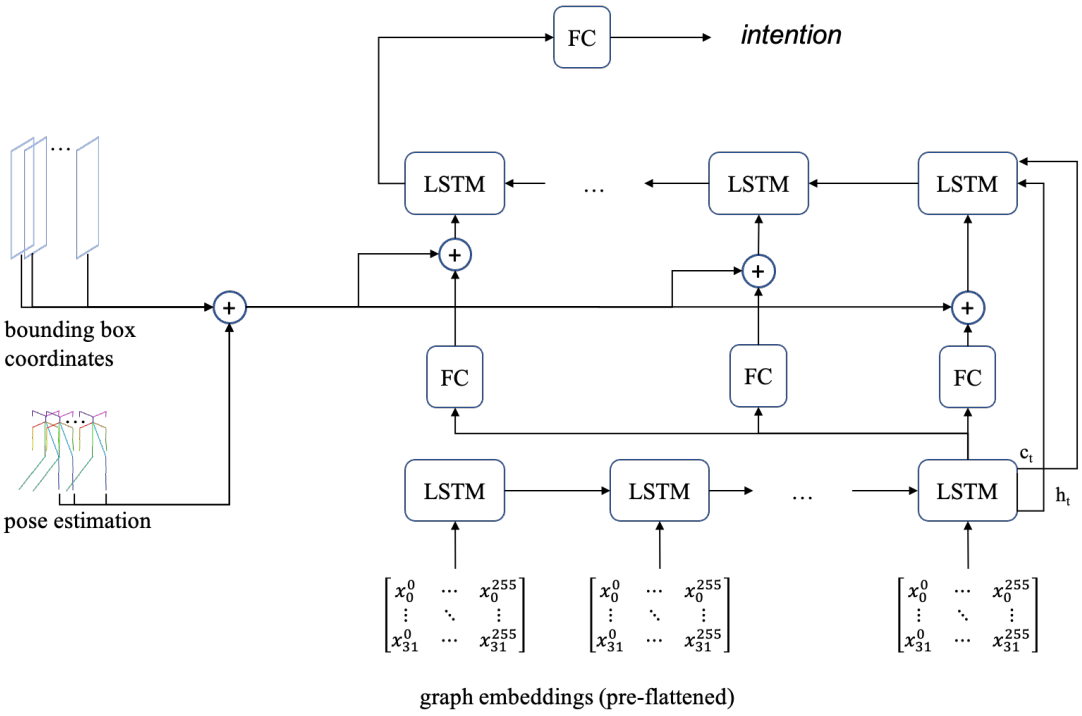


Figure 6.1. Network architecture of our proposed crossing intention prediction module. The graph embeddings from our graph autoencoder module is the input for the LSTM encoder cells. Pedestrian bounding box coordinates, pose estimation, and the output of the LSTM encoder concatenated together are the inputs to the LSTM decoder cells. 15 frames are observed for each sample. $\bigoplus$ denotes concatenation of features, c_t is the LSTM cell state, and h_t is the LSTM hidden states.

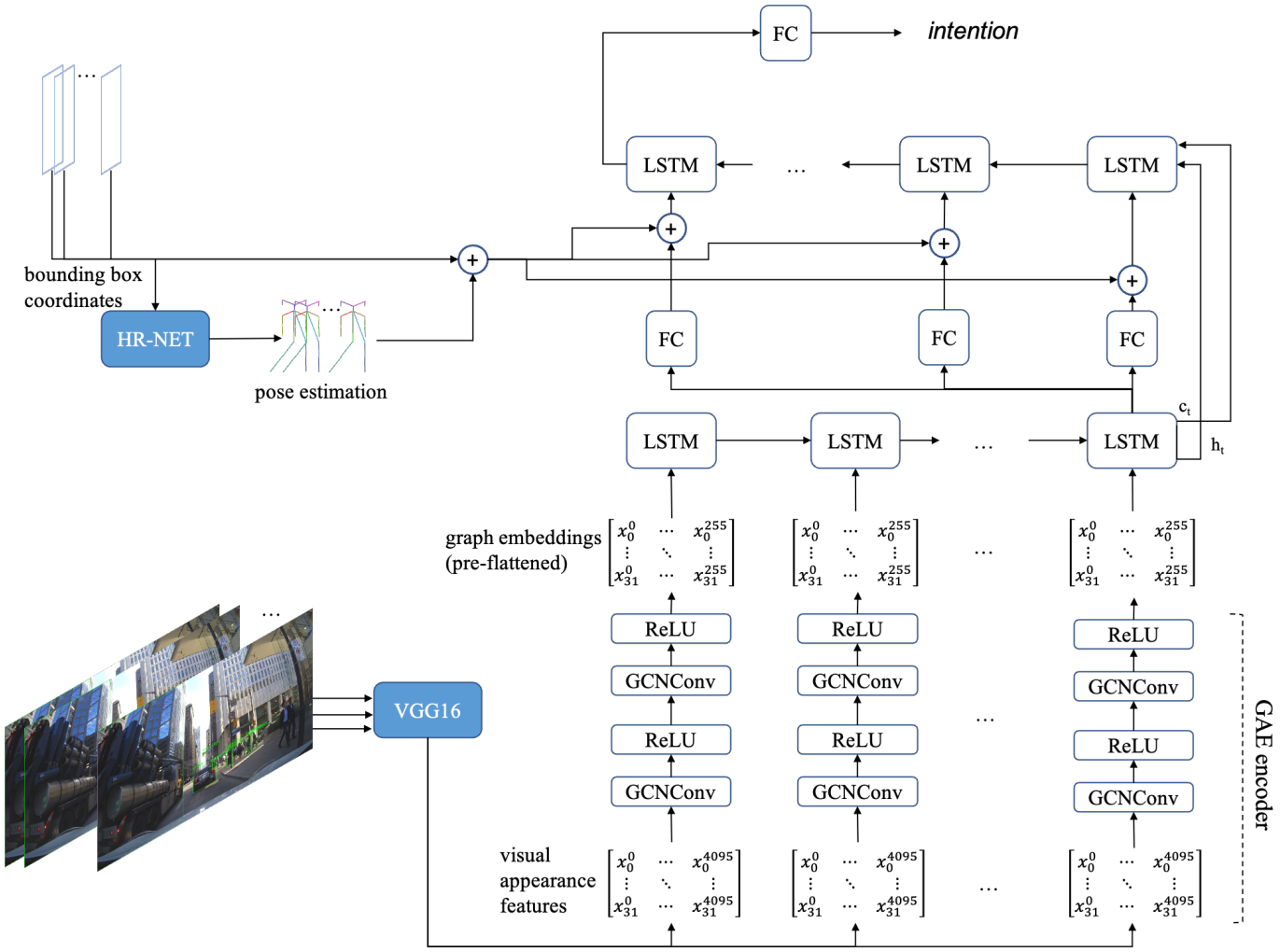


Figure 6.2. Network architecture of our overall proposed model for crossing intention prediction. This architecture combines VGG16 for the traffic object/agent appearance embedding module, HR-NET for the pedestrian pose estimation embedding module, the GAE with GCNConv convolutional layers for the graph autoencoder module, and the LSTM encoder-decoder pedestrian crossing intention prediction module. This model takes 15 observed frames, and predicts the crossing intention of the pedestrian for the next 45 frames (length of a sample). $\bigoplus$ denotes concatenation of features, c_t is the LSTM cell state, and h_t is the LSTM hidden states.

6.2 Binary vs. Multi-Label Classification

For our pedestrian crossing intention prediction task, we attempt prediction using our model trained for two types of classification. The first is binary classification of pedestrian crossing intention, Y_I, as defined by PIE. This is the same classification as [42], which we will compare with as a baseline method. The second is jointly predicting pedestrian crossing action, Y_C, with Y_I using multi-class multi-label classification. As defined by PIE, we will use 15 frames of observed data, one LSTM cell for each frame, to predict the pedestrian's crossing intention for the next 45 frames.

6.2.1 Binary Classification With Crossing Intention

Crossing intention is defined by PIE through the subject study their research team conducted. Subjects were asked to view video clips up to a critical point, and answer the question "Does this pedestrian **want** to cross the street?" with a 5-point scale. All responses for the same video clip were aggregated, and normalized between $[0, 1]$. The normalized score is the crossing intention probability for the pedestrian in the video. The probability is converted into binary values for classification purposes where

- 0 - no crossing intention
- 1 - has crossing intention

6.2.2 Multi-Label Classification With Crossing Action And Crossing Intention

To use labels that provide more meaning to the pedestrian's behavior, and intent, we combine the action, and intent labels to turn this into a multi-class classification. We change the crossing actions defined by PIE:

- 0 - not crossing in the path of the ego-vehicle
- 1 - crossing in the path of the ego-vehicle
- -1 - crossing irrelevant, because crossing is not in front of ego-vehicle

to convert all -1 labels to 0. Combining the crossing action, and intention labels into $[Y_C Y_I]$ gives us four new labels:

- [00]: no crossing action + no crossing intention
- [01]: no crossing action + crossing intention
- [10]: crossing action + no crossing intention
- [11]: crossing action + crossing intention

The existence of scenario [10] would mean the crossing intention label was incorrectly assigned. Fortunately, out of the 1,842 videos clips in PIE, only two of them have been labeled as [10]. That means the subjects were very good at determining crossing intention when there was crossing action. However, this particular disagreement scenario between the observers, and the pedestrian creates a dangerous situation. In an AV application, this disagreement could cause the vehicle to crash into the pedestrian.

6.3 Experimental Results

In this section, we present the results of our binary, and multi-class multi-label classification experiments. Additionally, we analyze the dataset, and take into consideration any class imbalances the dataset may have. The results are compared between the baseline model, PIE, and our model using evaluation metrics discussed in Section 3.3.

6.3.1 Dataset Imbalance

PIE has 1,842 video clips (or 1,842 pedestrians) with crossing intention annotated. Table 6.1 breaks down the number of video clips belonging to each label, and shows that there are many more pedestrians with crossing intention than without in this dataset.

The videos clips are split into train, val, and test sets with ratio 50%, 10%, and 40%, respectively. We use the same data split for our experiments. A sliding window is applied to each video clip so that samples are 60 frames in length. Table 6.2 breaks down the number

Table 6.1. Distribution of video clips in PIE using multi-class multi-label classification.

Label	Number of Video Clips
no crossing behavior + no crossing intention	430
no crossing behavior + crossing intention	898
crossing behavior + no crossing intention	2
crossing behavior + crossing intention	512

of samples that are in the train, val, and test sets by label. Again, the dataset is imbalanced with $1 \gg 0$ for crossing intention.

Table 6.2. Number of samples in train, validation, and test set of PIE using multi-label classification.

Label	Train Set	Val Set	Test Set
no crossing behavior + no crossing intention	2,623	715	8,772
no crossing behavior + crossing intention	4,000	1,278	25,971
crossing behavior + no crossing intention	0	32	132
crossing behavior + crossing intention	2,607	568	14,917

To account for the imbalance in the dataset, we will not only compare accuracy, and F1-scores, but also balanced accuracy, recall, and precision. Additionally, we will perform stratified random sampling on the prediction results to compare accuracy, and F1-scores for a balanced dataset.

6.3.2 Baseline Model

The baseline model we compare our network prediction results with is the PIE Intention Prediction module in [42]. It is the state of the art method for intention prediction on the PIE dataset. The PIE Intention Prediction module, shown in Figure 6.3, has two inputs for their prediction model. The first are the bounding boxes around the pedestrian. The second are the visual features of the image cropped around the pedestrian to $2\times$ the size of the bounding box. VGG16 trained on ImageNet is used to extract the features of the cropped image. Rather than cropping the image around the bounding box, by cropping the image to

2× the size of the bounding box there is more context to the area immediately surrounding the pedestrian.

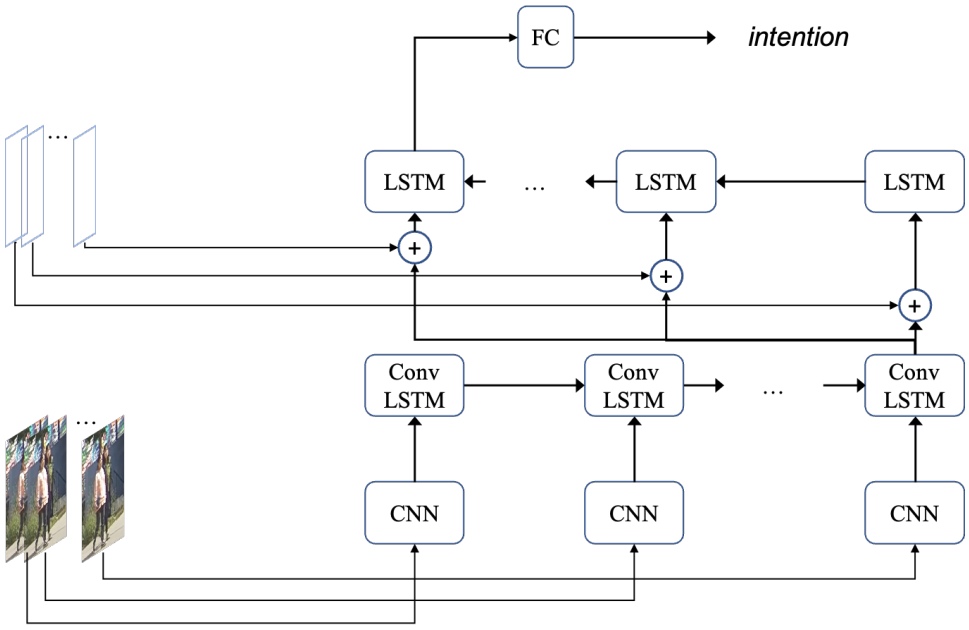


Figure 6.3. Network architecture of PIE Intention Prediction module. The encoder extracts features using VGG16 from a cropped image of the pedestrian. The decoder concatenates the encoder output with the pedestrian's bounding box coordinates to predict their crossing intention. Redrawn from [42] © 2019 IEEE.

The encoder in this configuration are the Convolutional LSTM (ConvLSTM) cells that use 64 filters, and kernel size 2 × 2 with stride 1. The decoders are LSTM cells with 128 hidden units, *tanh* activiation, 0.4 dropout, and 0.2 recurrent dropout. This baseline model predicts crossing intention with accuracy of 0.79 and F1-score of 0.87.

6.3.3 Parameter Tuning on Binary Crossing Intention Prediction

To train our LSTM network for binary crossing intention prediction, we start by tuning three parameters to compare with the baseline model: (1) regularizer value, (2) number of hidden units, and (3) activation function.

Tables 6.3 and 6.4 compare the results of PIE, and our model with three different regularizer values. The highest value for each evaluation metric is in bold for easy comparison.

Comparing just our models, **Ours-1,2,3**, we see that our model greatly benefits from having a regularizer value of 0.0001 or smaller. Across all evaluation metrics, **Ours-2,3** beat **Ours-1**, which has a regularization value of 0.001. Between **Ours-2** and **Ours-3**, the only noticeable difference is that **Ours-2** performs five percentage points better in recall on the negative label. **PIE** outperforms our best model **Ours-2** in both accuracy and F1 score. However, to properly evaluate the results, we look at evaluation metrics in Table 6.4 that account for imbalanced datasets. By comparing those metrics, **Ours-2** beats state of the art **PIE** in balanced accuracy, 0.79 to 0.61. **Ours-2** outperforms **PIE** by almost 20 percentage points when we account for the dataset imbalance. The F1 scores for the positive, and negative labels explain why **Ours-2** is able to outperform **PIE** by so much. **PIE** has a slightly higher F1 score for the positive label, but **Ours-2** increases the F1 score on the negative label from 0.36 to 0.55. So while **PIE** can classify cases with crossing intention slightly better, **Ours-2** improves the classification of cases with no crossing intention significantly, even with less data samples for the negative label.

In Tables 6.5 and 6.6, we tune the number of hidden units in the LSTM cells. Since we previously concluded a regularizer value of 0.0001 gives us our best performing models, all of our models compared here will use 0.0001. Comparing just our models **Ours-2,4,5**, we see that changing the number of hidden units in the LSTM cells does not change the performance of our model much. We can say **Ours-2** with 128 hidden units slightly outperforms **Ours-4,5** with 256, and 512 hidden units, respectively, but only very slightly. Since **Ours-2** is still our best performing model, the comparisons with **PIE's** performance still holds.

Our last parameter to tune, activation function, is reported in Tables 6.7 and 6.8. **Ours-2** uses $tanh$, and **Ours-6** uses $softsign$. Comparing accuracy and balanced accuracy, they both are the same at 0.76 and 0.79. If we compare the F1 score for each label, **Ours-6** outperforms **Ours-2** by one percentage point in both, thus **Ours-6** is now our best performing model. Considering the closeness in performance between **Ours-2** and **Ours-6**, **Ours-6** compares the same to **PIE** as **Ours-2** does. The difference between **Ours-2** and **Ours-7** is **Ours-2** uses GCNConv convolutional layers for its GAE, while **Ours-7** uses GraphConv convolutional layers in its GAE. In Section 5.4, when we compared the reconstruction error for the two types of convolutional layers used in our GAE, GraphConv had lower MSE.

However, here in Tables 6.7 and 6.8, **Ours-2** using GCNConv layers outperforms **Ours-7** in every metric by significant amounts. GCNConv is better suited for our GAE to get better intention prediction results.

In additional to using evaluation metrics that account for the dataset imbalance in PIE, we also performed stratified sampling of the test set to create a balanced dataset. We included all 8,904 negative samples with no crossing intention in the balanced dataset. Then we randomly sampled 8,904 positive samples that have crossing intention to add to the negative samples. This creates a balanced dataset with 17,808 samples. We randomly sampled 10 times to create 10 balanced datasets, and then predicted on those datasets using **PIE** and **Ours-6**. The average accuracy and F1 score from those predictions are reported in Table 6.9. **Ours-6** ourperforms **PIE** in both accuracy and F1 score. **Ours-6** has an average accuracy of 0.79, and average F1 score of 0.78. **PIE** has an average accuracy of 0.62, and average F1 score of 0.70. The average accuracy calculated for prediction on the balanced datasets for both these models are similar to the balanced accuracy reported in Table 6.7 where the prediction was on the imbalanced dataset. So either using balanced accuracy on the imbalanced dataset or accuracy on a balanced dataset is sufficient to account for the dataset imbalance.

Through tuning parameters, we conclude **Ours-6** with LSTM regularizer value 0.0001, 128 hidden units, and $softsign$ activation function is our best performing model. **Ours-6** is able to outperform the state of the art **PIE**, because **Ours-6** is stronger at predicting scenarios where there is no crossing intention. In an imbalanced dataset such as PIE, it's important to use evaluation metrics that can account for the imbalance or use a sampling method to create a balanced dataset to ensure fair comparisons. In the next section, we will use **Ours-6's** parameters for our multi-class multi-label classifier to predict Y_I and Y_C jointly.

Table 6.3. Overall evaluation results of our model trained on crossing intention, Y_I, tuning LSTM regularizer value.

Model Name	Autoencoder	Regularizer Value	No. of Hidden Units	Activation Function	Accuracy	F1 Score
PIE	-	0.001	128	*tanh*	**0.79**	**0.87**
Ours-1	GCNConv	0.001	128	*tanh*	0.72	0.81
Ours-2	GCNConv	0.0001	128	*tanh*	0.76	0.83
Ours-3	GCNConv	0.00001	128	*tanh*	0.76	0.84

Table 6.4. Continued evaluation results of Table 6.3 with metrics to account for class imbalance.

Model Name	Balanced Accuracy	Recall (Y_I=1)	Precision (Y_I=1)	Recall (Y_I=0)	Precision (Y_I=0)	F1 Score (Y_I=0)
PIE	0.61	**0.89**	0.86	0.33	0.41	0.36
Ours-1	0.74	0.71	0.94	0.77	0.37	0.50
Ours-2	**0.79**	0.74	**0.96**	**0.85**	0.41	**0.55**
Ours-3	0.78	0.76	0.95	0.8	**0.42**	**0.55**

Table 6.5. Overall evaluation results of our model trained on crossing intention, Y_I, tuning LSTM number of hidden units.

Model Name	Autoencoder	Regularizer Value	No. of Hidden Units	Activation Function	Accuracy	F1 Score
PIE	-	0.001	128	*tanh*	**0.79**	**0.87**
Ours-2	GCNConv	0.0001	128	*tanh*	0.76	0.83
Ours-4	GCNConv	0.0001	256	*tanh*	0.76	0.83
Ours-5	GCNConv	0.0001	512	*tanh*	0.76	0.84

Table 6.6. Continued evaluation results of Table 6.5 with metrics to account for class imbalance.

Model Name	Balanced Accuracy	Recall (Y_I=1)	Precision (Y_I=1)	Recall (Y_I=0)	Precision (Y_I=0)	F1 Score (Y_I=0)
PIE	0.61	**0.89**	0.86	0.33	0.41	0.36
Ours-2	**0.79**	0.74	**0.96**	0.**85**	0.41	**0.55**
Ours-4	0.78	0.75	0.94	0.80	**0.42**	**0.55**
Ours-5	0.78	0.75	0.95	0.82	0.41	0.54

Table 6.7. Overall evaluation results of our model trained on crossing intention, Y_I, tuning LSTM activation function, and using different graph convolution layers for the GAE.

Model Name	Autoencoder	Regularizer Value	No. of Hidden Units	Activation Function	Accuracy	F1 Score
PIE	-	0.001	128	*tanh*	**0.79**	**0.87**
Ours-2	GCNConv	0.0001	128	*tanh*	0.76	0.83
Ours-6	GCNConv	0.0001	128	*softsign*	0.76	0.84
Ours-7	GraphConv	0.0001	128	*tanh*	0.69	0.78

Table 6.8. Continued evaluation results of Table 6.7 with metrics to account for class imbalance.

Model Name	Balanced Accuracy	Recall (Y_I=1)	Precision (Y_I=1)	Recall (Y_I=0)	Precision (Y_I=0)	F1 Score (Y_I=0)
PIE	0.61	**0.89**	0.86	0.33	0.41	0.36
Ours-2	**0.79**	0.74	**0.96**	**0.85**	0.41	0.55
Ours-6	**0.79**	0.75	0.95	0.84	**0.42**	**0.56**
Ours-7	0.74	0.66	0.95	0.81	0.34	0.47

Table 6.9. Evaluation results from stratified sampling to create a balanced dataset. The dataset was randomly sampled 10 times, and the average accuracy and F1 score are reported here.

Model	Accuracy	F1 Score
PIE	0.62	0.70
Ours-6	**0.79**	**0.78**

6.3.4 Multi-Class Multi-Label Classification

In Section 6.2.2, we defined our multi-class multi-label classification task. In this section, we train new models using the best parameters found in the previous section to predict Y_I and Y_C jointly. We train both our model and **PIE** for this. Additionally, we compare three optimizers, RMSprop, Adam, and SGD to find the best optimizer for our model.

Table 6.10 compares the results of our models with **PIE-2**, which has the same architecture and parameters as **PIE** only trained to predict both Y_I and Y_C for multi-label classification. Of our three models listed in Table 6.10, only two of them are valid classifiers. **Ours-10**, which uses an SGD optimizer, predicts every sample to be label [0 1]. Of course, **Ours-10** is a terrible classifier, but we show its results to further demonstrate how class imbalances in this dataset can skew evaluation metrics. For predicting pedestrian crossing action, and intention jointly, our model **Ours-9** with an Adam optimizer outperforms **PIE-2** in every evaluation metric in Table 6.10. **Ours-9** has an accuracy score of 0.59, while **PIE-2's** is 0.54. Additionally, **Ours-9** has a weighted F1 score of 0.79, while **PIE-2** is only 0.68. Our model is much better suited for more complicated predictions that involve pedestrian actions, and intentions.

In Tables 6.11 and 6.12, we separate the predictions on $Y - I$ and Y_C, and compare them individually. Table 6.11 evaluates just the crossing action prediction, Y_C. With F1 scores close to 0, and balanced accuracy at 0.5, we can see both **PIE-2** and **Ours-10** are poor action prediction models. They both predict all or the majority of scenarios as not crossing actions. Comparing recall and precision for positive labels for all the models in Table 6.11, it is clear that predicting positive crossing action is a more difficult task.

In Table 6.12, we can compare prediction results for crossing intention, Y_I. Comparing just accuracy and F1 score, **PIE-2** outperforms all our models, and **PIE**. However, we have previously proven high accuracy and F1 score on an imbalanced dataset does not necessarily make a better classifier. Additionally, given that we know **Ours-10** is a bad classifier, high performance in accuracy and F1 score is all the more proven to be bad evaluation metrics for this dataset. Comparing balanced accuracy, **Ours-8** has the best performance for intention prediction with 0.65, while **PIE** and **PIE-2** both are 0.61. Predicting Y_I and Y_C jointly did not improve the prediction results on Y_I compared to our best binary classifier model **Ours-6**. However, our models are able to outperform the state of the art re-trained to do the same. Since crossing action and intent are different, it is important that we distinguish them in our classification tasks. While the main goal of this thesis is to predict crossing intention, it is imperative that we start thinking about this joint classification task.

Table 6.10. Multi-class multi-label classifier evaluation results. Comparisons between different optimizers.

Model Name	Optimizer	Accuracy	F1 Score (weighted)	Balanced Accuracy	Recall (weighted)	Precision (weighted)
PIE-2	RMSprop	0.54	0.68	-	0.70	0.72
Ours-8	RMSprop	0.55	0.75	-	0.73	0.78
Ours-9	Adam	**0.59**	**0.79**	-	**0.78**	**0.79**
Ours-10	SGD	0.52	0.65	-	0.73	0.6

Table 6.11. Evaluation of only *crossing action* class in our multi-class multi-label prediction network.

Model Name	Accuracy	F1 Score	Balanced Accuracy	Recall (Y_C=1)	Precision (Y_C=1)	Recall (Y_C=0)	Precision (Y_C=0)
PIE	-	-	-	-	-	-	-
PIE-2	0.69	0.05	0.5	0.03	0.35	0.97	0.70
Ours-8	0.71	0.42	0.61	0.36	0.53	0.86	0.75
Ours-9	**0.74**	**0.53**	**0.67**	**0.49**	**0.58**	0.84	**0.79**
Ours-10	0.70	0	0.5	0	0	**1**	0.69

Table 6.12. Evaluation of only *crossing intention* class in our multi-class multi-label prediction network.

Model Name	Accuracy	F1 Score	Balanced Accuracy	Recall (Y_I=1)	Precision (Y_I=1)	Recall (Y_I=0)	Precision (Y_I=0)
PIE	0.79	0.87	0.61	0.89	0.86	0.33	0.41
PIE-2	**0.83**	**0.9**	0.61	0.95	0.85	0.27	**0.59**
Ours-8	0.79	0.87	**0.65**	0.86	**0.88**	**0.44**	0.41
Ours-9	0.8	0.88	0.64	0.88	0.87	0.41	0.44
Ours-10	0.82	**0.9**	0.5	**1**	0.82	0	0

6.4 Summary

In this chapter we evaluate our proposed model for pedestrian crossing intention prediction against the state of the art. Our key insights are:

- Our best performing model, **Ours-6**, is trained to use 0.0001 regularizer value, 128 hidden units, and $softsign$ activation on the LSTM cells.
- **Ours-6** outperforms the state of the art **PIE** on predicting pedestrian crossing intention with balanced accuracy 0.79 compared to **PIE's** 0.61.
- Our models greatly outperform the state of the art in predicting no crossing intention cases. **Ours-6** has an F1 score of 0.56 on the negative intention label, while **PIE** has an F1 score of 0.36 on the same labels.
- When we account for dataset imbalance, both our binary intention classifier, and multi-class multi-label classifier outperform the state of the art. Our best multi-class multi-label classifier **Ours-9** predicts with an accuracy of 0.59 and F1 score of 0.79, while **PIE-2** has an accuracy of 0.54 and F1 score of 0.68.
- Accuracy and F1 score are not adequate evaluate metrics for imbalanced datasets, as evidenced by **Ours-10**. Balanced accuracy is a better metric to use.
- Sampling the imbalanced dataset to create a balanced dataset is also a useful method for evaluating performance with accuracy correctly.
- The embeddings from our GraphConv GAE has lower reconstruction error, but performs worse when used for intention prediction. **Ours-6** with GCNConv layers has balanced accuracy of 0.79, while **Ours-7** with GraphConv layers has balanced accuracy of 0.74.

7. CONCLUSIONS AND FUTURE WORK

In this thesis, we present our deep learning based approach to predicting pedestrian crossing intention from the ego-view. Our main belief is pedestrian behavior on the road is heavily impacted by other road objects and agents. Deep learning solutions that are trained on videos and images from a static, bird's eye view camera are not appropriate for use on an onboard camera. The angle change from top-down to perpendicular affects how trajectory is predicted. The constantly changing perspective, and distance between road objects and the ego-vehicle are also not considered in top-down camera approaches. To that end, we train our network on naturalistic driving dataset, PIE, which is collected from an onboard camera from the ego-view perspective.

We model pedestrian-environment relationships using GCNs, and then utilize GAEs to encode those relationships in a lower dimension. Additionally, we use state of the art feature extraction techniques to embed visual representations of road objects and agents to encourage relationship learning from rich visual features. With our LSTM encoder-decoder crossing intention prediction framework, we use bounding box coordinates, pose estimation, and graphical pedestrian-environment visual feature encodings to improve on state of the art pedestrian intention prediction.

Using our best performing binary pedestrian crossing intention prediction model, **Ours-6**, we are able to predict crossing intention with balanced accuracy of 0.79, an F1 score on the positive label of 0.84, and an F1 score on the negative label of 0.56. This is compared with **PIE** which predicts crossing intention with balanced accuracy of 0.61, an F1 score on the positive label of 0.87, and an F1 score on the negative label of 0.36. We use balanced accuracy and F1 scores on both labels to compare the performance, because PIE is an imbalanced dataset heavily skewed towards the positive labels. While **PIE** slightly outperforms **Ours-6** in positive intention prediction, **Ours-6** is much stronger in predicting negative intention labels.

We also train the state of the art model and our model to predict pedestrian crossing action, Y_C, and pedestrian crossing intention, Y_I jointly. The best overall model is **Ours-9** with an accuracy of 0.59, and weighted F1 score 0.79. Our model beats the state of the art

trained on this classification. **PIE-2** predicts with an accuracy score of 0.54, and F1 score 0.68. While this multi-class multi-label classification isn't able to predict crossing intention with better performance than our binary classifier, it is helpful for expanding on future work that can predict crossing action, and intention jointly with a high degree of accuracy.

Further future work include data collection for more scenarios where the pedestrian has no crossing intention will be necessary for training algorithms that can perform equally well on both crossing and no crossing intention interactions. The capability to predict accurately on either case will not only improve pedestrian safety on the road, but also communication between the pedestrian and the ego-vehicle to ensure fluid traffic movement.

Finally, to further improve the quality of feature extraction, and expand visual feature representation, scene segmentation can be used to extract visual features from the entire image. Scene segmentation is a costly process, and with the rapidly changing scenes of an ego-vehicle camera, it might prove difficult to implement this process for real-time processing.

REFERENCES

[1] National Highway Traffic Safety Administration, *Overview of motor vehicle crashes in 2019*, Dec. 2019. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813060.

[2] National Highway Transportation Safety Administration, *Automated driving systems 2.0.: A vision for safety*, Sep. 2017. [Online]. Available: https://bit.ly/3gKW5FR.

[3] P. J. Navarro, C. Fernandez, R. Borraz, and D. Alonso, "A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data," *Sensors*, vol. 17, no. 1, p. 18, 2017.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[5] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Systematic testing of convolutional neural networks for autonomous driving," *arXiv preprint arXiv:1708.03309*, 2017.

[6] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[7] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd International Conference on Automated Software Engineering (ASE)*, IEEE, 2018, pp. 132–142.

[8] D. Shinar, "Aggressive driving: The contribution of the drivers and the situation," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 1, no. 2, pp. 137–160, 1998.

[9] D. Shinar and R. Compton, "Aggressive driving: An observational study of driver, vehicle, and situational variables," *Accident Analysis & Prevention*, vol. 36, no. 3, pp. 429–437, 2004.

[10] D. Ridel, E. Rehder, M. Lauer, C. Stiller, and D. Wolf, "A literature review on the prediction of pedestrian behavior in urban scenarios," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3105–3112.

[11] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[12] M. Szczodrak, J. Kotus, K. Kopaczewski, K. Lopatka, A. Czyzewski, and H. Krawczyk, "Behavior analysis and dynamic crowd management in video surveillance system," in *2011 22nd International Workshop on Database and Expert Systems Applications*, IEEE, 2011, pp. 371–375.

[13] A. Tuomi, I. P. Tussyadiah, and J. Stienmetz, "Applications and implications of service robots in hospitality," *Cornell Hospitality Quarterly*, pp. 232–247, 2020.

[14] G. Sukthankar, C. Geib, H. H. Bui, D. Pynadath, and R. P. Goldman, *Plan, activity, and intent recognition: Theory and practice*. Newnes, 2014.

[15] S. Tadokoro, M. Hayashi, Y. Manabe, Y. Nakami, and T. Takamori, "On motion planning of mobile robots which coexist and cooperate with human," in *International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, IEEE, vol. 2, 1995, pp. 518–523.

[16] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating visual representations from unlabeled video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 98–106.

[17] M. Sadegh Aliakbarian, F. Sadat Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson, "Encouraging lstms to anticipate actions very early," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 280–289.

[18] C. Rodriguez, B. Fernando, and H. Li, "Action anticipation by predicting future dynamic images," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Sep. 2018.

[19] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3034–3042.

[20] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, "Predicting the future: A jointly learnt model for action anticipation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5562–5571.

[21] T. Chen and R. Tian, "A survey on deep-learning methods for pedestrian behavior prediction from the egocentric view," in *International Conference on Intelligent Transportation Systems*, IEEE, 2021, to appear.

[22] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.

[23] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1463–1470, 2021.

[24] F. Piccoli, R. Balakrishnan, M. J. Perez, M. Sachdeo, C. Nunez, M. Tang, K. Andreasson, K. Bjurek, R. D. Raj, E. Davidsson, *et al.*, "Fussi-net: Fusion of spatio-temporal skeletons for intention prediction network," *arXiv preprint arXiv:2005.07796*, 2020.

[25] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[26] F. Zanlungo, T. Ikeda, and T. Kanda, "Social force model with explicit collision prediction," *EPL (Europhysics Letters)*, vol. 93, no. 6, p. 68 005, 2011.

[27] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 464–469.

[28] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 935–942.

[29] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker," in *International Conference on Computer Vision (ICCV) Workshops*, IEEE, 2011, pp. 120–127.

[30] W. Choi and S. Savarese, "Understanding collective activities of people from videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1242–1257, 2013.

[31] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5275–5284.

[32] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *European Conference on Computer Vision*, Springer, 2016, pp. 263–279.

[33] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2203–2210.

[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[35] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[36] H. Manh and G. Alaghband, "Scene-lstm: A model for human trajectory prediction," *arXiv preprint arXiv:1808.04018*, 2018.

[37] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1186–1194.

[38] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[39] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5725–5734.

[40] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, M. Cristani, and F. Galasso, "" seeing is believing": Pedestrian trajectory forecasting using visual frustum of attention," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1178–1185.

[41] K. Mangalam, E. Adeli, K.-H. Lee, A. Gaidon, and J. C. Niebles, "Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2784–2793.

[42] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos, "Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6262–6271.

[43] A. Rasouli, M. Rohani, and J. Luo, "Pedestrian behavior prediction via multitask learning and categorical interaction modeling," *arXiv preprint arXiv:2012.03298*, 2020.

[44] S. Malla, B. Dariush, and C. Choi, "Titan: Future forecast using action priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 186–11 196.

[45] A. Rasouli, I. Kotseruba, and J. 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*, 2017, pp. 206–213.

[46] D. Varytimidis, F. Alonso-Fernandez, B. Duran, and C. Englund, "Action and intention recognition of pedestrians in urban traffic," in *International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, IEEE, 2018, pp. 676–682.

[47] P. Gujjar and R. Vaughan, "Classifying pedestrian actions in advance using predicted video of urban driving scenes," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2097–2103.

[48] J. Gesnouin, S. Pechberti, G. Bresson, B. Stanciulescu, and F. Moutarde, "Predicting intentions of pedestrians from 2d skeletal pose sequences with a representation-focused multi-branch deep learning network," *Algorithms*, vol. 13, no. 12, p. 331, 2020.

[49] A. Rasouli, T. Yau, M. Rohani, and J. Luo, "Multi-modal hybrid architecture for pedestrian action prediction," *arXiv preprint arXiv:2012.00514*, 2020.

[50] B. Liu, E. Adeli, Z. Cao, K.-H. Lee, A. Shenoi, A. Gaidon, and J. C. Niebles, "Spatiotemporal relationship reasoning for pedestrian intent prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3485–3492, 2020.

[51] D. Cao and Y. Fu, "Using graph convolutional networks skeleton-based pedestrian intention estimation models for trajectory prediction," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1621, 2020, p. 012 047.

[52] W. M. Alvarez, F. M. Moreno, O. Sipele, N. Smirnov, and C. Olaverri-Monreal, "Autonomous driving: Framework for pedestrian intention estimation in a real world scenario," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 39–44.

[53] O. Ghori, R. Mackowiak, M. Bautista, N. Beuter, L. Drumond, F. Diego, and B. Ommer, "Learning to forecast pedestrian intention from pose dynamics," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1277–1284.

[54] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Pedestrian action anticipation using contextual feature fusion in stacked rnns," *arXiv preprint arXiv:2005.06582*, 2020.

[55] M. Hoy, Z. Tu, K. Dang, and J. Dauwels, "Learning to predict pedestrian intention via variational tracking networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3132–3137.

[56] L. Ladicky, P. H. Torr, and A. Zisserman, "Human pose estimation using a joint pixel-wise and part-wise formulation," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3578–3585.

[57] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[58] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[59] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3674–3681.

[60] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660.

[61] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 648–656.

[62] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[63] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh, "Pose machines: Articulated pose estimation via inference machines," in *European Conference on Computer Vision*, Springer, 2014, pp. 33–47.

[64] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*, Springer, 2016, pp. 483–499.

[65] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[66] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[67] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.

[68] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5386–5395.

[69] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.

[70] J. Shi and C. Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1994, pp. 593–600.

[71] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.

[72] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint description and detection of local features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8092–8101.

[73] Z. Zhang and W. S. Lee, "Deep graphical feature learning for the feature matching problem," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5087–5096.

[74] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[75] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem," *arXiv preprint arXiv:1906.01227*, 2019.

[76] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," *arXiv preprint arXiv:1706.05674*, 2017.

[77] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[78] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *arXiv preprint arXiv:1806.02473*, 2018.

[79] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[80] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[81] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[82] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416–1424.

[83] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[84] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," *arXiv preprint arXiv:1904.07785*, 2019.

[85] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[86] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

[87] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International Conference on Artificial Neural Networks*, Springer, 2018, pp. 412–422.

[88] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2636–2645.

[89] F. Flohr and D. Gavrila, "Pedcut: An iterative framework for pedestrian segmentation combining shape models and multiple data cues," in *Proceedings of the British Machine Vision Conference*, 2013.

[90] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, Springer, 2014, pp. 740–755.

[91] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, 2009, pp. 248–255.

[92] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[93] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4602–4609.

[94] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[95] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.