Synthetic Data Generation Framework, Dataset, and Efficient Deep Model for Pedestrian Intention Prediction

Muhammad Naveed Riaz[1], Maciej Wielgosz[2], Abel García Romera[1], and Antonio M. López[1]

Abstract

Pedestrian intention prediction is crucial for autonomous driving. In particular, knowing if pedestrians are going to cross in front of the ego-vehicle is core to performing safe and comfortable maneuvers. Creating accurate and fast models that predict such intentions from sequential images is challenging. A factor contributing to this is the lack of datasets with diverse crossing and non-crossing (C/NC) scenarios. We address this scarceness by introducing a framework, named ARCANE, which allows programmatically generating synthetic datasets consisting of C/NC video clip samples. As an example, we use ARCANE to generate a large and diverse dataset named PedSynth. We will show how PedSynth complements widely used real-world datasets such as JAAD and PIE, so enabling more accurate models for C/NC prediction. Considering the onboard deployment of C/NC prediction models, we also propose a deep model named PedGNN, which is fast and has a very low memory footprint. PedGNN is based on a GNN-GRU architecture that takes a sequence of pedestrian skeletons as input to predict crossing intentions. ARCANE, PedSynth, and PedGNN will be publicly released[1].

I. Introduction

As evidenced in an early Google self-driving car report [24], the 10% of their self-driving malfunctions on streets were due to incorrect behavior predictions of other road users, including pedestrians. While there have been significant efforts to improve the accuracy of pedestrian intention prediction [15], [3], [16], [23], [18], [36], [5], there is still ample room for improvement. Currently, two datasets, JAAD [28] and PIE [27], are being used to benchmark such prediction models. In these datasets, the core ground truth (GT) consists of labeling if pedestrians are crossing or are going to cross in front of the ego vehicle. As for other onboard perception tasks (*e.g.*, object detection and tracking [4], semantic segmentation [39], monocular depth estimation [17]), synthetic datasets have been proposed to train C/NC prediction models [1], [2]. We propose to go beyond these datasets by introducing a framework, named ARCANE[2], where traffic scenarios of pedestrian behavior can be programmatically defined. This opens the possibility of introducing underrepresented vehicle-to-pedestrian traffic situations. For being aligned with the research community, ARCANE has been developed on top of the CARLA simulator [11]. As an example, we have used ARCANE to generate PedSynth which is a large and diverse synthetic dataset with pedestrian C/NC labels. Note that this type of labeling is not provided by the CARLA simulator, but it is generated by ARCANE. PedSynth consists of 947 video clips of pedestrian C/NC situations. Each video clip runs $\sim$ 20s at 30fps, so resulting in approximately 5 H and 26 min of labeled videos. Figure 1 shows several frames of two video clips from PedSynth. On the other hand, users can generate their own datasets by working with ARCANE.

Focusing on the demanding hardware requirements for onboard perception, we also propose a lightweight model for C/NC prediction, named PedGNN. This model has a 27KB GPU memory footprint and runs on $\sim$ 0.6ms on an NVIDIA GTX 1080 GPU. Compared to a state-of-the-art C/NC prediction model, here named PedGraph+ [5], PedGNN is one order of magnitude smaller and one order of magnitude faster. Even though, PedGNN outperforms PedGraph+ in terms of the class-balanced F1-score classification metric. PedGNN is based on a GNN-GRU architecture that takes a sequence of pedestrian skeletons as input to predict crossing intentions (see Fig. 2). Note that, so far, the spatiotemporal analysis of pedestrian skeletons has been shown as one of the most relevant sources of information to predict pedestrian crossing intentions [15], [16], [2], [5].

Using PedGNN and PedSynth to complement the training sets of both JAAD and PIE, allows us to outperform pedestrian C/NC prediction in the respective testing sets.

II. Related Work

A. Pedestrian intention prediction

Pioneering approaches cast C/NC prediction as a trajectory prediction problem, which requires the explicit estimation of the future location, speed, and acceleration of the observed pedestrians [33], [20]. In practice, the corresponding dynamic models were difficult to adjust and require to extract the silhouette of the pedestrians, dense depth, and dense optical flow

Corresponding author: nriaz@cvc.uab.cat

(1) Naveed, Abel, and Antonio are with the Dpt. Ciències de la Computació and the CVC, at Univ. Autònoma de Barcelona (UAB).

Maciej acknowledges the funding from the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No. 801342 (Tecniospring INDUSTRY) and the Government of Catalonia's Agency for Business Competitiveness (ACCIÓ). This allowed to develop ARCANE. This work only expresses the opinion of the author and neither the European Union nor ACCIÓ are liable for the use made of the information provided.

[1] <URL address to be provided with the camera-ready version of the paper>

[2] ARCANE stands for ***a**dver**s**arial **c**ases* for ***a**uto**n**omous **v**ehicles*, the generic project supporting the development of the framework.

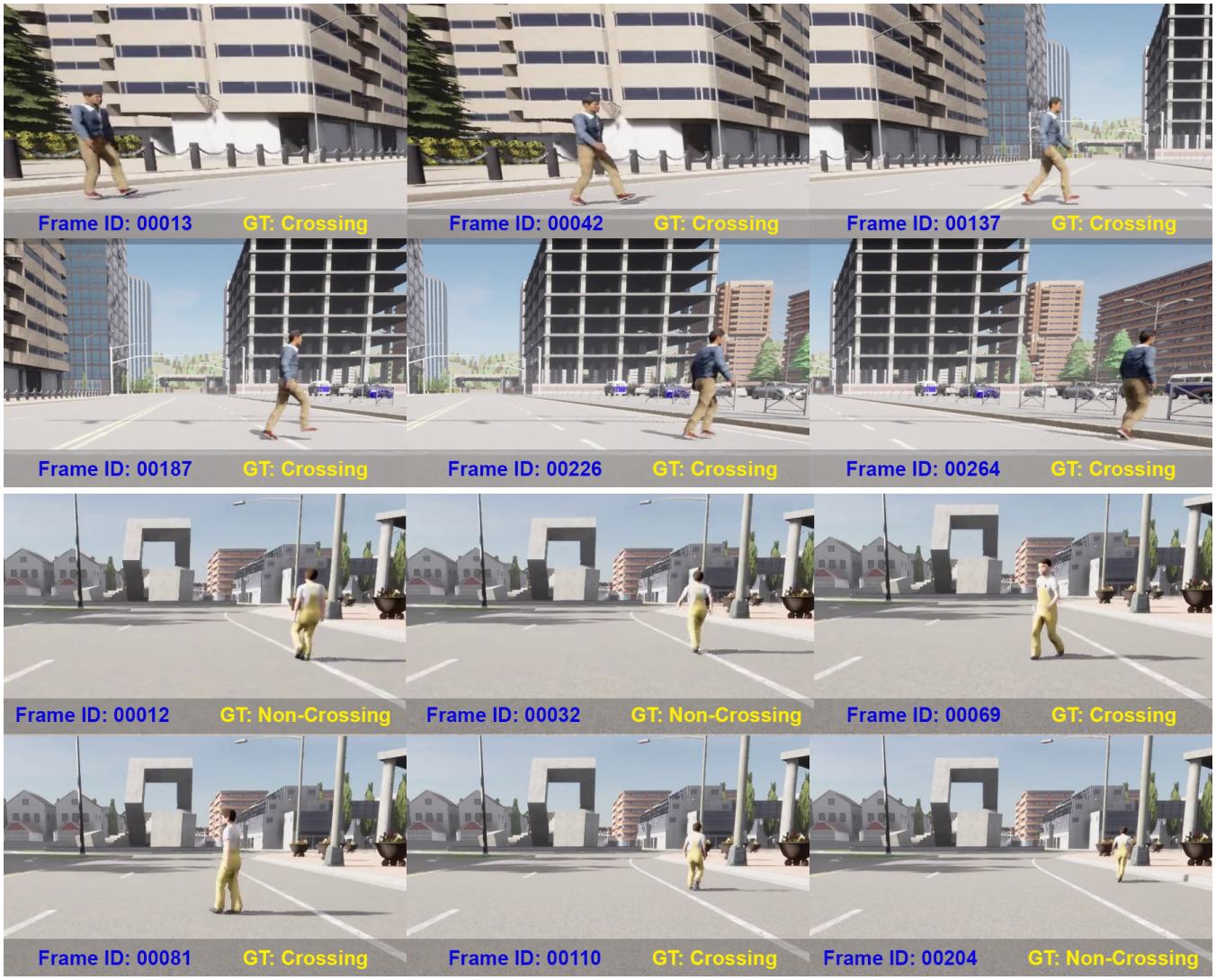


Fig. 1: Summary of two video clips from PedSynth. Top rows: a pedestrian crosses the road perpendicularly to the ego-vehicle moving direction. Bottom rows: a pedestrian change the intention of crossing the road at mid-lane. In both examples, the pedestrians enter the road at locations not enabled for crossing.

with ego-motion compensation. On the other hand, Schneemann and Heinemann [32] concluded that pedestrians' posture and body movement are essential to take faster C/NC predictions. Accordingly, methods relying on the temporal evolution of pedestrian skeletons gained popularity, especially due to the increasing accuracy of the deep models adjusting them in 2D images, *e.g.*, see [8] (becoming OpenPose [7]) and [14] (becoming AlphaPose [13]). For instance, a sequence of pedestrian skeletons was used as input to classical lightweight and fast machine learning models such as a random forest C/NC classifier [15] and as input to more resources-consuming but accurate deep models such as a Graph Convolutional Network (GNN) [6]. Skeleton extraction and C/NC prediction have been also tackled as a joint multi-task problem [29].

Semantic and contextual information is also considered in different works. In [23], it is proposed a deep model based on recurrent neural networks (RNNs) and attention modules, which takes as input the ego-vehicle speed, the bounding box (BB), skeleton, and local context (RGB crops) of pedestrians. In [40], RNNs and attention modules are also used to train two intermediate deep architectures whose output is fused (mid/late fusion) to provide C/NC predictions. One architecture considers the ego-vehicle speed, the BB, and the skeleton of pedestrians. The other considers local and global (scene semantic segmentation) contexts. Finally, [6] became a state-of-the-art model [5], here named PedGraph+, by incorporating the ego-vehicle speed and pedestrian local context to the initial skeleton-based GNN architecture.

In this paper: As these C/NC prediction approaches, we also rely on an off-the-self model to obtain the pedestrian skeletons that PedGNN requires as input. Since these skeletons are structured as graphs, we think that GNNs are natural architectures to work with them, *i.e.*, as done by PedGraph+. However, we want PedGNN to be more lightweight and faster. Thus, we use a different GNN architecture than PedGraph+.

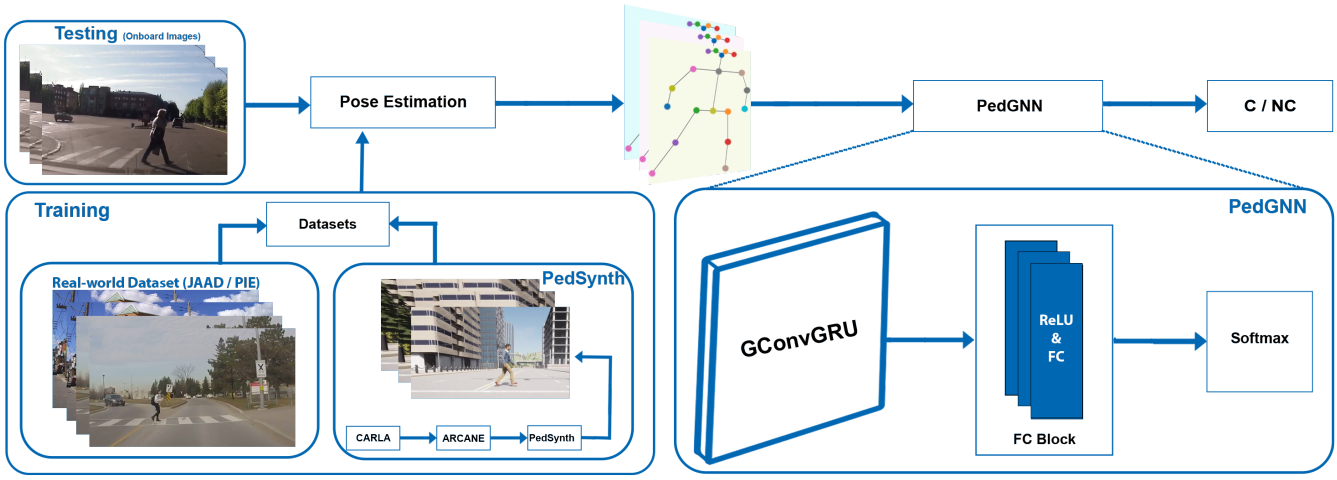


Fig. 2: To perform C/NC predictions PedGNN processes sequences of pedestrian skeletons. To process onboard sequences while driving, we use a temporal sliding window of a 1-frame step. PedGNN consists of a graph convolutional gated recurrent unit (GConvGRU), followed by a block of three (ReLU + Fully connected) layers, and a final Softmax. Synthetic datasets with C/NC examples can be used for training PedGNN. For instance, in this paper, we use PedSynth, a synthetic dataset that we have generated using ARCANE, a framework that we introduce in this paper too (see Fig. 3).

B. *Synthetic datasets focusing on C/NC prediction*

As with other vision-based tasks, C/NC prediction research started with relatively small and non-naturalistic datasets [33], [15]. Fortunately, larger and naturalistic datasets such as JAAD [28] and PIE [27] appeared progressively, so helping to accelerate this research. It was also a matter of time to use synthetic data to support C/NC prediction research. In fact, onboard pedestrian detection was one of the first tasks for which a model was trained on synthetic images to perform later in real-world images, this was done more than a decade ago [25]. Since then, there have been many works leveraging synthetic data to support the training of perception models or performing simulations [31], [30], [39], [4], [11], [34]; being synth-to-real domain adaptation a core ingredient to encourage the use of synthetic data [37], [10].

Focusing on pedestrians, synthetic data has been mainly generated and used for the tasks of detection and tracking either onboard or from static infrastructure locations [19], [12], [9], [21], [35]; where the required GT for each pedestrian consists of a 2D/3D BB, pixel-level segmentation and depth, an ID, and, eventually, a body skeleton. In addition to this kind of GT, for collecting samples to develop C/NC prediction models we must control pedestrian behavior in the simulator, *e.g.*, to force C/NC situations as we wish, and we must label each frame accordingly as in Figure 1. Recent attempts to do so [1], [2] rely on the CARLA simulator [11]. In [1], the CP2A dataset was introduced with 220K video clips with per-frame C/NC labels, where 25% of the clips contain crossing (c) examples. In [2], the Virtual-Pedcross-4667 dataset was introduced with 4,667 video clips specially prepared to cover a variety of weather and lighting conditions and per-frame C/NC labels, where 61% of the clips contain crossing (c) examples. On the other hand, these datasets lack some corner cases like the one shown as the bottom example in Figure 1. Overall, the experiments provided in [1], [2] encourage the use of synthetic data to train C/NC models.

In this paper: We contribute to the use of synthetic data to develop C/NC prediction models. As [1], [2] we rely on the CARLA simulator. However, rather than only providing a specific synthetic dataset, we introduce ARCANE, a framework prepared to programmatically generate synthetic datasets of pedestrian C/NC videos. As an example, we have used ARCANE to generate PedSynth, which consists of 947 video clips recorded under different weather and lighting conditions over 400 locations in CARLA cities, with $\sim$ 398K frames with C/NC labels. Moreover, using PedGNN, we show that PedSynth is a good complement for the training sets of both JAAD and PIE, so boosting C/NC prediction performance in the respective testing sets.

III. Methods

Figure 2 summarizes the role of the main contributions of this paper: ARCANE, PedSynth, and PedGNN, which we present in the following subsections.

A. *The ARCANE framework*

ARCANE framework is built on the top of CARLA simulator. It enables the generation of different types of video clips through the parameterization of the distribution of pedestrian models, pedestrian velocity, and onboard camera settings. It is

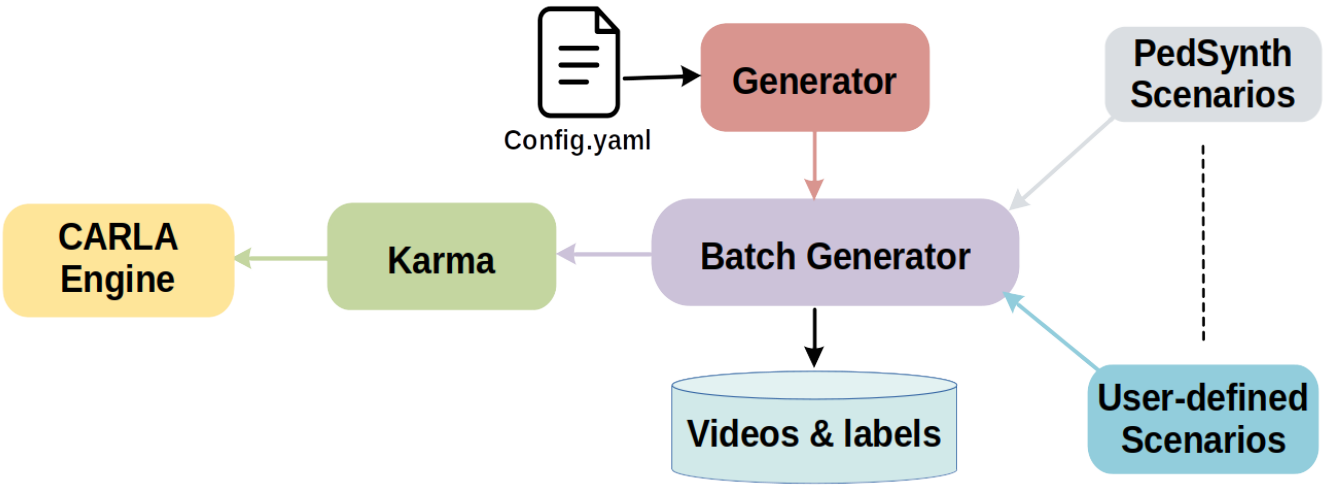


Fig. 3: Block diagram of ARCANE dataset generator.

Feature	JAAD	PIE	PedSynth
Video clips with C/NC labels	323	55	947
Video clips length (s)	$\sim 5-15$	~ 600	~ 20
Frames per second (fps)	30	30	30
Frame resolution (pix)	1920×1080	1920×1080	1600×600
Frames with C/NC labels	$\sim$ 75K	$\sim$ 293K	$\sim$ 398K
Semantic segmentation	no	no	yes
Pedestrian skeleton	no	no	yes
Weather variability	yes	no	yes

TABLE I: Features of the datasets used in this paper.

possible to create scenarios in a single Python file, which can establish the trajectories of pedestrians in the scene. ARCANE also contains a series of mechanisms that allow for the filtering of not useful videos (*e.g.*, when a pedestrian is not visible). ARCANE is structured around five primary modules, as shown in Fig. 3.

Generator: This module manages the simultaneous execution of multiple batch generator objects. It oversees the randomization of the data generation process and finalizes the process once completed. The module maintains a count of the number of generated videos, ensuring the target quantity is achieved. In the event of crashes, the module has a predefined number of retry attempts to prevent endless generations.

Batch Generator: This module spawns pedestrians in a city, positions cameras, and regulates the data generation process. An integral part of its role involves verifying the presence of pedestrians in the generated videos. This is done via semantic segmentation checks and skeleton existence verification. It supervises the C/NC labeling process too, *e.g.*, marking pedestrians as crossing (C) if they are entering a driving area. These tasks are fulfilled by interacting with Karma.

Karma: This module acts as a facade for the CARLA API. It automates the process of creating the virtual environment in CARLA, pedestrian spawning, and other tasks, by relying on CARLA-based scenario runner functionalities.

CARLA Engine: This module serves as an instance of the CARLA simulator, which is supplemented with extra routines to ensure its operation within a Docker container during the data generation phase. The module is equipped to restart the container in case of any simulation crashes.

Based on these modules, ARCANE allows for both *simple* and *advanced usage*. Advanced usage refers to the possibility of programmatically defining dynamic traffic scenarios. For instance, leveraging CARLA cities it is possible to write a Python code to choreograph the behavior of pedestrians in these towns, so forcing situations interesting for C/NC prediction. This is what we have done for generating the PedSynth dataset, as illustrated in Fig. 3. Given one of such user-defined Python files to generate traffic scenarios, it is possible to generate variations by generic parameters (*i.e.*, scenario agnostic) which allow controlling the types of pedestrians to be included, their speed, the pedestrian density, *etc.* These parameters are included in a configuration file, named `config.yaml` in Fig. 3. Therefore, this configuration file enables simple usage provided we are satisfied with the traffic scenarios in place.

Overall, the development of ARCANE took approximately half a year. The code repository includes 3,267 lines of Python code in 47 files, supplemented by a multitude of additional files of other formats.

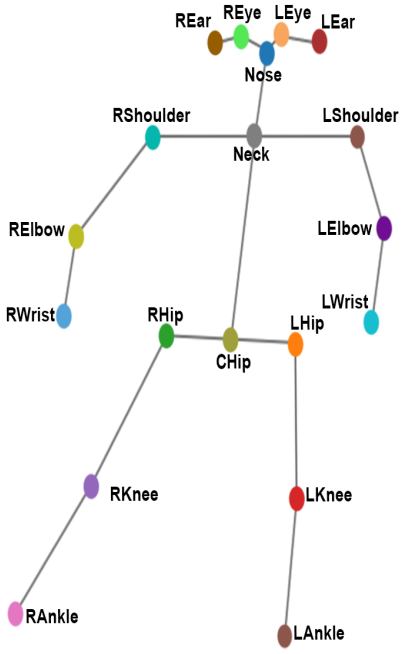


Fig. 4: Pedestrian skeleton as expected by PedGNN. We consider 19 joints connected as an undirected graph.

B. The PedSynth dataset

We have written a Python code, named `PedSynth Scenarios` in Fig. 3, which is consumed in ARCANE to generate video clips with C/NC labels according to the settings provided through the `config.yaml` file. With this information, ARCANE has generated the PedSynth dataset. It covers $\sim$ 400 locations from different CARLA cities, thus, including different city styles and road lanes. Varying pedestrians and environmental conditions, we have generated 947 video clips with C/NC labels, resulting in a total of $\sim$ 398K frames with C/NC labels. Table I summarizes the main features of PedSynth compared to the real-world datasets JAAD and PIE. We can see how PedSynth contains $\sim$ 100K more frames with C/NC labels than PIE and more than $\sim$ 300K compared to JAAD. Note that, as in real-world datasets, PedSynth's videos include frames with no pedestrians, where C/NC prediction models should not rise false warnings. Beyond C/NC labels, we can also leverage GT already present in the CARLA simulator itself, such as pixel-level class semantics (semantic segmentation), pedestrian skeletons, *etc.* We provide more detailed information about ARCANE and PedSynth in the technical report [38].

C. PedGNN model

As we have mentioned in Section II, the temporal evolution of pedestrian pose is considered core information to determine C/NC intentions. Today, there are robust deep models able to provide human-body skeletons from 2D images [7], [13]. Even by using hand-crafted features and traditional machine learning models, the temporal evolution of 2D-fitted pedestrian skeletons was shown to be effective to determine C/NC intentions [15]. Therefore, as per the state-of-the-art literature, for our C/NC prediction model, we also assume that a sequence of 2D-fitted pedestrian skeletons is used as input to determine C/NC intentions. Figure 4 shows the joints we consider and their connections. They cover the head, arms, trunk, and legs. The poses of hands and feet are not considered since they cannot be clearly perceived by an onboard camera, and most likely they are irrelevant for determining C/NC intentions. To process sequences of images in a continuous manner, we use a temporal sliding window approach with a frame step to be adjusted experimentally according to the frame rate of the onboard camera (*e.g.*, we use a 1-frame step for cameras working at 30fps).

Since pedestrian skeletons can be represented as undirected graphs, natural deep architectures to process them are GNNs (graph neural networks). In fact, since, for each pedestrian, we work with a sequence of skeletons, a graph convolutional gated recurrent unit (GConvGRU) is a very convenient model for C/NC prediction. Therefore, we adopt it by using the implementation in the PyTorch Geometric (PyG) library[3]. The output of the GConvGRU is flattened and processed by three consecutive blocks of (ReLU + FC) layers, which feed Softmax to obtain the C/NC prediction (see Fig. 2).

As input information at a graph node, we use the (x_j, y_j) coordinates of the joint j associated with the node and its fitting confidence c_j as provided by the skeleton fitting model in use. As is recommended for GNNs [22] and for skeleton-based

[3] `https://github.com/pyg-team/pytorch_geometric`

Int.	JAAD			PIE			PedSynth		
Label	Train	Val.	Test	Train	Val.	Test	Train	Val.	Test
#C	39.7K	6.3K	32.8K	116.2K	18.1K	130.9K	155.1K	50.4K	50.5K
#NC	7.9K	1.5K	8.4K	110.7K	22.1K	76.8K	82.3K	29.9K	29.5K

TABLE II: Let's a *sample* be a particular pedestrian completing a C/NC sequence. Thus, different samples can overlap in the same frame. Let N_F^S be the number of labeled frames of the C/NC sequence of sample S. Let N_S be the number of samples in a particular subset of videos. For each dataset and split subset, this table reports $\#l = \sum_{s=1}^{N_S} \sum_{f=1}^{N_F^s} gt(s, f, l)$, where $l \in \{\text{C, NC}\}$ is the label, and $gt(s, f, l) = 1$ if l matches the C/NC GT associated to the pedestrian sample s at frame f, and $gt(s, f, l) = 0$ otherwise.

C/NC prediction [15], (x_j, y_j) are normalized at each frame to the range $[0, 1]$. Thus, we work with normalized 2D coordinates $(\hat{x}_j, \hat{y}_j)$ which add invariance to ego-vehicle to pedestrian distance variations. Overall, the input to PedGNN has dimensions (N_F,19,3), where N_F is the number of frames used to perform C/NC predictions, which is determined experimentally during the training of PedGNN. Obviously, the 19 comes from the number of joints, and 3 from the information per joint, *i.e.*, $(\hat{x}_j, \hat{y}_j, c_j)$.

Finally, we remark that we focus on having a lightweight and fast C/NC prediction model. PedGNN shows a memory footprint of 27KB and an inference time of ~ 0.6ms on an NVIDIA GTX 1080 GPU. As we will see in Section IV, this is one order of magnitude of improvement over other state-of-the-art methods such as [5].

IV. Experimental Results

A. Datasets, metrics, frameworks, pose estimation

For our experiments, as real-world datasets we use the de-facto standards for C/NC prediction, *i.e.*, JAAD [28] and PIE [27]. We use JAAD *all* version. As a synthetic dataset, we use[4] our PedSynth. Table I summarizes their main features. For JAAD and PIE datasets we also use their standard Train/Val./Test split. For PedSynth we performed a random split and fix it for all the experiments. Specifically, $\sim 80\%$ of PedSynth is allocated for training, $\sim 10\%$ for validation, and another $\sim 10\%$ for testing. Table II provides information on the corresponding splits in terms of C/NC labeling.

To report our results, we apply the metrics used in C/NC prediction literature, *i.e.*, standard *Accuracy*, *Precision*, *Recall*, and *F1-score*. For training models and running inferences, we use PyTorch. Since PedGNN is based on pedestrian skeletons adjusted on 2D images, we use the state-of-the-art deep model named AlphaPose [13] as an off-the-shelf method. AlphaPose does not return the 19 joints we use for PedGNN. Compared to Fig. 4, AlphaPose does not provide the Neck and CHip joints. To compute the Neck coordinates we average LShoulder and RShoulder coordinates. Analogously, to compute the CHip coordinates we average LHip and RHip.

B. Training protocol

As the training optimizer, we use AdamW with binary cross-entropy loss and default parameters except for the learning rate, l_r. We perform training runs for a maximum number of M_E epochs. We also apply a 50% dropout. In this optimization process, a training sample consists of a sequence of skeletons from the same pedestrian. In other words, for C/NC pedestrian prediction, we consider N_F consecutive frames. We use a training batch size of 500 samples. Since skeleton information has a really low memory footprint, this batch size fits well in a single 24GB memory GPU. Since some experiments rely on training images from different datasets, we utilized PyTorch's ConcatDataset and WeightedRandomSampler functions to ensure equal dataset sampling per batch.

To train a C/NC prediction model, we test different values for N_F and l_r. Regarding N_F, we consider values in the range $[4, \ldots, 32]$ with step=2. Since all datasets were recorded at 30fps (Table I), this is equivalent to considering a temporal window from $\sim$ 133ms to $\sim$ 1067ms. Regarding l_r, we consider values in $\{0.001, 0.005, 0.0002, 0.0005\}$. While training a model, we assess its F1-score at the end of each epoch with the help of the validation set associated with the targeted training set. We have set $M_E = 100$. To apply the C/NC prediction models we use a temporal sliding window with a step of 1 frame. Across the considered N_F and l_r values and epochs, we take as the final model the best-performing one.

Depending on the size of the training set, obtaining a trained model requires from $\sim 3-4$h (single training dataset) to $\sim 6-9$h (multiple training datasets) in a desktop PC with an NVIDIA RTX 3090 GPU. We remark that, for doing these experiments, pedestrian skeletons are computed and recorded on the hard disk once.

User

[4]At the moment of elaborating our work, CP2A and Virtual-Pedcross-4667 datasets do not seem downloadable anymore.

Train	Test	N_F	Accuracy	Precision	Recall	F1-score
JAAD	JAAD	18	**80.32**	84.72	**87.91**	85.29
PedSynth	JAAD	32	78.59	**89.48**	84.62	**85.45**
PIE	PIE	08	**68.11**	66.98	68.36	69.81
PedSynth	PIE	16	62.74	**69.37**	**79.31**	**74.01**

TABLE III: C/NC prediction performance with PedGNN.

Train	Test	N_F	Accuracy	Precision	Recall	F1-score
J	J	18	80.32	84.72	87.91	85.29
S	J	32	78.59	**89.48**	84.62	85.45
J + P	J	08	72.36	74.22	89.18	81.20
J + S	J	32	**86.22**	77.35	**96.19**	**85.96**
J + S + P	J	08	74.41	76.73	88.00	81.98
P	P	08	68.11	66.98	68.36	69.81
S	P	16	62.74	69.37	79.31	74.01
P + J	P	32	69.26	77.03	75.26	76.13
P + S	P	16	**70.52**	74.80	**82.73**	**79.12**
P + S + J	P	08	69.34	**78.24**	70.20	76.35

TABLE IV: Performance when combining different datasets for training PedGNN. J: JAAD, P: PIE, S: PedSynth.

Train	Test	N_F	Accuracy	Precision	Recall	F1-score
J	S	08	**72.23**	**77.33**	83.97	80.97
P	S	08	69.66	74.54	82.47	78.30
J + P	S	08	71.40	73.11	**85.74**	**82.98**
J + S	J	32	86.22	77.35	96.19	85.96
P + S	P	16	70.52	74.80	82.73	79.12

TABLE V: PedSynth (S) as testing dataset. J: JAAD, P: PIE.

Model	Train	Test	N_F	Accuracy	Precision	Recall	F1-score
PedGNN	S (GT)	S (GT)	08	89.29	95.85	88.69	92.14
PedGNN	J	J	18	80.32	**84.72**	**87.91**	**85.29**
PedGraph+*	J	J	32	**83.85**	53.76	59.21	56.36
PedGNN	P	P	08	68.11	66.98	**68.36**	**69.81**
PedGraph+*	P*	P*	32	**79.15**	**77.91**	36.51	49.72

TABLE VI: J: JAAD, P: PIE, S: PedSynth. GT refers to using ground truth skeletons from CARLA. PedGraph+* refers to PedGraph+ [5] but only considers pedestrian skeletons (from AlphaPose) as input information ($N_F = 32$ is used in [5]). For PIE, PedGraph+* only considers the $\sim 30\%$ of C/NC cases, which we denote as P*.

Model	Size (MB)	Inference time (ms)
PCPA [23]	118.8	38.6
Global PCPA [40]	374.2	70.83
FUSSI [26]	8.4	34.92
PedGraph [6]	0.22	29.01
PedGraph+ [5]	0.28	5.47
TEP [1]	12.8	2.85
V-PedCross [2]	4.8	-
PedGNN (Ours)	**0.027**	**0.58**

TABLE VII: Memory footprint and inference time of PedGNN and different models from the state-of-the-art. All times are computed on an NVIDIA GTX 1080 GPU. We have extracted these times from the respective papers.

C. Experiments and discussion

We start the experiments by evaluating how effective PedSynth is training our PedGNN model to perform on the JAAD and PIE testing sets. Table III shows the results. The N_F value refers to the number of input frames (per pedestrian skeletons) that was best for each case according to the previously described training protocol. Training on PedSynth requires considering more frames than using the respective JAAD/PIE training data. However, this does not affect the prediction latency since, as we have mentioned before, we use a temporal sliding window of a 1-frame step. Moreover, in an NVIDIA GTX 1080 GPU, for $N_F = 32$ PedGNN only takes ~ 0.6ms to perform the inference. In terms of accuracy, training on the respective real-world datasets is better than training on PedSynth. However, we can see how it is not the case for F1-score, which is a more unbiased metric than accuracy when the testing data distribution presents a class unbalanced. Table II shows that this is the case here since both JAAD and PIE have testing sets clearly biased toward the crossing (C) class. Thus, we think PedSynth is an effective dataset for training C/NC prediction models.

At this point, it is worth commenting that, as shown in Table I, PedSynth provides a skeleton GT for each pedestrian (coming from the CARLA simulator). The joints used by PedGNN are a subset of those provided by the CARLA simulator, so the mapping is straightforward. Moreover, fitting confidences c_j can be set to 1 since skeletons are perfectly fitting pedestrians. Therefore, this raises the question of using such skeletons as GT for training with PedSynth instead of applying AlphaPose to the synthetic pedestrians. We did the corresponding experiments, however, F1-score dropped ~ 10 points when testing on JAAD and ~ 6 for PIE. In other words, a synth-to-real domain gap is induced by the use of different skeleton sources at training (GT) and testing (Alphapose) time. We leave future work to investigate more in deep the underlying reasons for the domain gap and keep using AlphaPose for all the training runs involving PedSynth.

Sometimes we may have real-world training data labeled for C/NC prediction, as is the case of PIE and JAAD. Then, it is also interesting to see if the synthetic data at hand can act as a complement, giving rise to better-performing models. Note that by using the same skeleton fitting method we avoid the synth-to-real domain gap provided this method performs well in the real and synthetic domains. According to our experiments, AlphaPose fulfills so. Therefore, we have combined PedSynth training data with JAAD and/or PIE training data for assessing the complementarity of these datasets. Table IV presents the corresponding results, where we also include those in Table III for easier comparison. We can see that, when testing in JAAD, combining JAAD and PedSynth gives better results than using JAAD or PedSynth alone and even than combining JAAD and PIE. In fact, it seems that PIE produces negative transfer when combining the three datasets. On the other hand, since the testing set of PIE has around $4\times$ more C-frames than JAAD, and around $9\times$ more NC-frames, results on PIE are of special interest. We can see that, when testing in PIE, combining PIE and PedSynth gives rise to the best results in terms of accuracy and F1-score. Thus, we think that PedSynth can complement real-world datasets for training purposes.

We can also assume that we use PedSynth for testing purposes. Table V shows the results corresponding to training with the real-world training sets and testing in the PedSynth testing set. For easier visual comparison, we also include the best models obtained when testing in real-world testing sets. These correspond to training on the respective training set plus the training set of PedSynth. We can see that performance metrics report comparable values when testing in real-world sets and in our synthetic set. Thus, we think PedSynth can play the role of the testing set too; in other words, it is not easier than their real-world counterparts.

At this point, we put the focus on the PedGNN model. On the one hand, we assess its potential by using PedSynth and the associated pedestrian GT skeletons, so that results are not influenced by the skeleton fitting method in place (here AlphaPose). Moreover, we compare our results with the state-of-the-art method on C/NC prediction here named PedGraph+ [5]. Table VI shows the results. For PedGraph+ we have copied the results reported in [5] when only AlphaPose-based skeletons are considered as input. However, for the PIE dataset, only a portion of the data is considered in [5], roughly the 30%. We can observe that PedGNN has great potential of providing good performance, which can be seen when training and testing with perfect pedestrian skeletons (GT). Note that F1-score is $\sim 92\%$. Of course, there is room for improvement. Compared to PedGraph+ assuming the same input data (AlphaPose-based skeletons), we can see how PedGraph+ performs better in terms of accuracy, but significantly worse when using the more representative F1-score metric. Moreover, we can see in Table VII how in terms of memory footprint and inference speed PedGNN is significantly more lightweight and faster.

Finally, as an example of qualitative results, Fig. 5 illustrates the performance of PedGNN trained on JAAD+PedSynth and tested on JAAD. In case (a), while the ego-vehicle turns to the right, the intention of a pedestrian that started to cross in the left is properly predicted from the very beginning. In case (b), while the ego-vehicle moves straight forward, a pedestrian standing still at the border of the road is properly predicted as a non-crossing pedestrian. In cases (c) and (d), PedGNN requires more frames to reach the proper prediction. In case (c) the pedestrian seems to take the crossing decision later than in case (a), so predicting the intentions required some additional time. In case (d), the pedestrian seems to start crossing in front of the ego-vehicle in a parking area, but finally, it does not. In fact, for us it is unclear if the GT is right, after all, it is based on human labelers and, therefore, there is subjectivity. For instance, if the initial frames have been labeled after looking at what the pedestrian did at the final ones, this would be like using the future to predict the present/past, which cannot be done by the temporal sliding window mechanism used to process onboard continuous image sequences.

V. Conclusion

In this paper, we have introduced our framework ARCANE which allows the generation of synthetic datasets labeled for the pedestrian C/NC prediction task. It works on top of the CARLA simulator so being aligned with the autonomous driving research community. Advanced users can programmatically design their pedestrian C/NC scenarios, the rest can adjust a configuration file to use existing scenarios. For example, we have generated the PedSynth dataset by using ARCANE. It is diverse and contains a large amount of pedestrian C/NC cases. We have shown its usefulness by running an extensive set of experiments. We have seen that it can play the role of the training set alone, it can complement real-world training sets, and it can play the role of the testing set. Most experiments are based on our model PedGNN, also introduced in this paper. It processes sequences of pedestrian skeletons to produce C/NC predictions. Our experiments show that PedGNN produces state-of-the-art results, despite being significantly more lightweight and faster than previous C/NC prediction models. In future work, we plan to use PedSynth and PedGNN to address the synth-to-real unsupervised domain adaptation problem for the pedestrian C/NC prediction task.

References

[1] Lina Achaji, Julien Moreau, Thibault Fouqueray, Francois Aioun, and Francois Charpillet. Is attention to bounding boxes all you need for pedestrian action prediction? In *Intelligent Vehicles Symposium (IV)*, 2022.

[2] Jie Bai, Xing Fang, Jianwu Fang, Jianru Xue, and Changwei Yuan. Deep virtual-to-real distillation for pedestrian crossing prediction. In *Intelligent Transportation Systems Conference (ITSC)*, 2022.

[3] Smail Bouhsain, Saeed Saadatnejad, and Alexandre Alahi. Pedestrian intention prediction: A multi-task perspective. In *Symposium of the European Association for Research in Transportation (hEART)*, 2020.

[4] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. arXiv:2001.10773, 2020.

[5] Pablo Rodrigo Gantier Cadena, Yeqiang Qian, Chunxiang Wang, and Ming Yang. Pedestrian graph +: A fast pedestrian crossing prediction model based on graph convolutional networks. *IEEE Trans. on Intelligent Transportation Systems*, 23:21050–21061, 2022.

[6] Pablo Rodrigo Gantier Cadena, Ming Yang, Yeqiang Qian, and Chunxiang Wang. Pedestrian graph: Pedestrian crossing prediction based on 2D pose estimation and graph convolutional networks. In *Intelligent Transportation Systems Conference (ITSC)*, 2019.

[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yase Sheikh. OpenPose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 43:172–186, 2021.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Ernest Cheung, Anson Wong, Aniket Bera, and Dinesh Manocha. MixedPeds: Pedestrian detection in unannotated videos using synthetically generated human-agents for training. In *AAAI Conference on Artificial Intelligence*, 2017.

[10] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Semantic image segmentation: Two decades of research. *Foundations and Trends in Computer Graphics and Vision*, 14:1–162, 2022.

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.

[12] Volker Eiselein Erik Bochinski and Tomas Sikora. Training a convolutional neural network for multi-class object detection using solely virtual world data. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2016.

[13] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. AlphaPose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 45:7157–7173, 2023.

[14] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *International Conference on Computer Vision (ICCV)*, 2017.

[15] Zhijie Fang and Antonio M. López. Is the pedestrian going to cross? answering by 2d pose estimation. In *Intelligent Vehicles Symposium (IV)*, 2018.

[16] Joseph Gesnouin, Steve Pechberti, Bogdan Stanciulescu, and Fabien Moutarde. TrouSPI-Net: Spatio-temporal attention on parallel atrous convolutions and U-GRUs for skeletal pedestrian crossing prediction. In *International Conference on Automatic Face and Gesture Recognition(FG)*, 2021.

[17] Akhil Gurram, Ahmet F. Tuna, Fengyi Shen, Onay Urfalioglu, and Antonio M. López. Monocular depth estimation through virtual-world supervision and real-world SFM self-supervision. *IEEE Trans. on Intelligent Transportation Systems*, 23:12738–12751, 2021.

[18] Je-Seok Ham, Kangmin Bae, and Jinyoung Moon. MCIP: Multi-stream network for pedestrian crossing intention prediction. In *European Conference on Computer Vision (ECCV)–Workshops*, 2022.

[19] Hironori Hattori, Vishnu N. Boddeti, Kris Kitani, and Takeo Kanade. Learning scene-specific pedestrian detectors without real data. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] Christoph G. Keller and Dariu M. Gavrila. Will the pedestrian cross? a study on pedestrian path prediction. *IEEE Trans. on Intelligent Transportation Systems*, 15:494–506, 2014.

[21] Wonhui Kim, Manikandasriram Srinivasan Ramanagopal, Charlie Barto, Ming-Yuan Yu, Karl Rosaen, Nicholas Goumas, Ram Vasudevan, and Matthew Johnson-Roberson. PedX: Benchmark dataset for metric 3-D pose estimation of pedestrians in complex urban intersections. *IEEE Robotics and Automation Letters*, 4:1940–1947, 2018.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representation (ICLR)*, 2017.

[23] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. Benchmark for evaluating pedestrian action prediction. In *Winter conf. on Applications of Computer Vision (WACV)*, 2021.

[24] Google Auto LLC. Google self-driving car testing report on disengagements of autonomous mode, December 2015.

[25] Javier Marin, David Gerónimo, David Vázquez, and Antonio M. López. Learning appearance in virtual scenarios for pedestrian detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[26] Francesco Piccoli, Rajarathnam Balakrishnan, Maria Jesus Perez, Moraldeepsingh Sachdeo, Carlos Nunez, Matthew Tang, Kajsa Andreasson, Kalle Bjurek, Ria Dass Raj, Ebba Davidsson, Colin Eriksson, Victor Hagman, Jonas Sjoberg, Ying Li, L. Srikar Muppirisetty, and Sohini Roychowdhury. FuSSI-Net: Fusion of spatio-temporal skeletons for intention prediction network. In *Asilomar Conference on Signals, Systems, and Computers*, 2020.

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

[28] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *International Conference on Computer Vision (ICCV)–Workshops*, 2017.

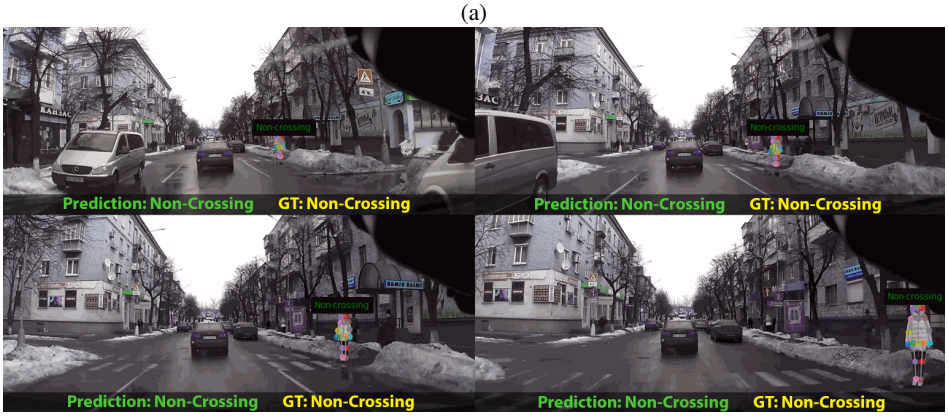
[29] Haziq Razali, Taylor Mordan, and Alexandre Alahi. Pedestrian intention prediction: A convolutional bottom-up multi-task approach. *Transportation Research Part C: Emerging Technologies*, 130:103259, 2021.

[30] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, 2016.

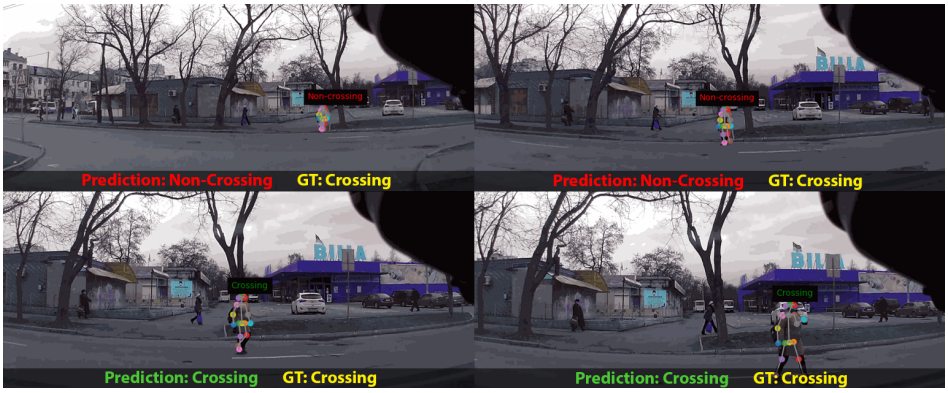
[31] Germán Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio M. López. The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] Friederike Schneemann and Patrick Heinemann. Context-based detection of pedestrian crossing intention for autonomous driving in urban environments. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.
[33] Nicolas Schneider and Dariu M. Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition (GCPR)*, 2013.
[34] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics (FSR)*, 2018.
[35] Thomas Stauner, Frederik Blank, Michael Fürst, Johannes Günther, Korbinian Hagn, Philipp Heidenreich, Markus Huber, Bastian Knerr, Thomas Schulik, and Karl-Ferdinand Lei. SynPeDS: A synthetic dataset for pedestrian detection in urban traffic scenes. In *ACM Computer Science in Cars Symposium*, 2022.
[36] Fan Wang, Jie Bai, and Jianwu Fang. Pedestrian crossing prediction based on invariant feature extraction of cross-spectral images. In *International Conference on Autonomous Unmanned Systems (ICAUS)*, 2023.
[37] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.
[38] Maciej Wielgosz, Antonio M López, and Muhammad Naveed Riaz. CARLA-BSP: a simulated dataset with pedestrians. arXiv:2305.00204, 2023.
[39] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. arXiv:1810.08705, 2018.
[40] Dongfang Yang, Haolin Zhang, Ekim Yurtsever, Keith A. Redmill, and Umit Ozguner. Predicting pedestrian crossing intention with feature fusion and spatio-temporal attention. *IEEE Trans. on Intelligent Transportation Systems*, 7:221–230, 2021.

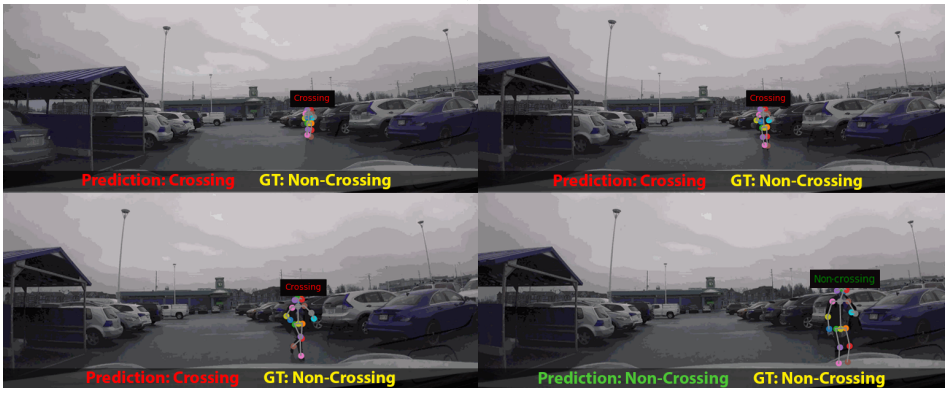
(a)



(b)



(c)



(d)

Fig. 5: Performance of PedGNN trained on JAAD+PedSynth and tested on JAAD. Cases (a) and (b) are fully successful, while in cases (c) and (d) there are C/NC prediction discrepancies with the labels provided by human labelers (GT). Time in each sequence runs from top-left to bottom-right.