Predicting Motion of Vulnerable Road Users using High-Definition Maps and Efficient ConvNets

Fang-Chieh Chou, Tsung-Han Lin, Henggang Cui, Vladan Radosavljevic, Thi Nguyen, Tzu-Kuo Huang, Matthew Niedoba, Jeff Schneider and Nemanja Djuric

Abstract—Following detection and tracking of traffic actors, prediction of their future motion is the next critical component of a self-driving vehicle (SDV) technology, allowing the SDV to operate safely and efficiently in its environment. This is particularly important when it comes to vulnerable road users (VRUs), such as pedestrians and bicyclists. These actors need to be handled with special care due to an increased risk of injury, as well as the fact that their behavior is less predictable than that of motorized actors. To address this issue, in this paper we present a deep learning-based method for predicting VRU movement, where we rasterize high-definition maps and actor’s surroundings into bird’s-eye view image used as an input to deep convolutional networks. In addition, we propose a fast architecture suitable for real-time inference, and present a detailed ablation study of various rasterization choices. The results strongly indicate benefits of using the proposed approach for motion prediction of VRUs, both in terms of accuracy and latency.

I. INTRODUCTION

Recent advances in high-performance hardware and software led to unprecedented breakthroughs in AI applications. Computers have surpassed human performance in centuries old games such as go [29], understand health conditions and suggest medical treatment [35], and can even reason about complex relationships conveyed through images [39]. This progress also prompted renewed enthusiasm and work on self-driving vehicles (SDVs), a nascent technology holding a potential to transform the way we live and work. While interest in SDVs goes as far back as the 1980s [24], only in the last decade government agencies and large industry players turned their focus towards the field, leading to a new era of research that transformed the way we live and work.

Predicting movement of traffic actors is a critical part of the autonomous technology. Once a self-driving vehicle successfully detects and tracks a traffic actor in its vicinity, it needs to understand how they will move in the near future in order for both actors and SDV to be safe during operation [6]. This holds particularly true for vulnerable road users (VRUs), defined as traffic actors with increased risk of injury, unprotected by an outside shield [21]. Road planners and policy makers have recognized this problem many decades ago, and attempted to mitigate it through several means. This included legal frameworks, designing new road types (e.g., segregating VRUs from motorized actors), educating both drivers and VRUs (with particular focus on children and elderly that are at an even greater risk than others [5 21]), to name a few. These efforts have however given limited results, and in the US proportion of VRU deaths within overall traffic fatalities has actually increased between 2008 and 2017 from 14% to 19% [19].

In the current study we address a critical aspect of the SDV technology, focusing on predicting motion for VRUs, namely pedestrians and bicyclists. The main contributions of the paper are summarized below:

- We present a system for motion prediction of VRU traffic actors, building upon recently proposed context rasterization techniques described in [6];
- We propose a fast and efficient convolutional neural network (CNN) architecture, suitable for running in real-time onboard the SDV platform;
- We present a detailed ablation study of various rasterization settings, providing critical insights into which parts of the system contribute the most to the prediction accuracy.

II. RELATED WORK

Efficient and accurate detection, tracking, and motion prediction of VRUs are one of the key factors and requirements for autonomous vehicles to be safely deployed in complex urban environments. With greatly improved detection and tracking of VRUs [22], research on motion prediction of VRUs has been gaining a lot of traction recently. Although most of the studies on the VRU motion prediction consider pedestrians with only a very few focusing on bicyclists, recent work showed importance of predicting bicyclists at crossings [3] and signalized intersections [32]. In this section we provide an overview of motion prediction of pedestrians and bicyclists as they relate to the problem of autonomous driving, while more comprehensive review that includes research on motion prediction of vehicles can be found in [6].

Motion prediction. A common approach for prediction of VRUs’ motion in an autonomous driving systems is to use motion model from tracking component to predict their future states. Most of the autonomous system’s pedestrian tracking components use either the Brownian or the constant velocity motion models [31]. These models do not take into account scene context, and therefore fail in long-term prediction tasks as VRU motion follows complex patterns constrained by static and dynamic obstacles along the path. Traditionally, handcrafted features were used for motion prediction of VRUs with respect to a surrounding context. The social force model for pedestrian motion prediction incorporated interactive forces that guide pedestrians towards their goals and enforce collision avoidance among pedestrians, as well as between pedestrians and static obstacles [10 41]. Similar approach was applied for bicyclist motion prediction [13]. In [25] authors introduced a motion model for bicyclist motion prediction that incorporates...
knowledge of the road topology. The authors were able to improve prediction accuracy by using specific motion models for a pre-specified set of canonical directions. Habibi et al. [8] obtained more accurate predictions of pedestrian trajectories by incorporating semantic features from a scene such as relative distance to curbside and status of pedestrian traffic lights in the Gaussian Process (GP) model. A significant number of studies has been devoted to modeling pedestrian motion using maximum entropy Inverse Reinforcement Learning (IRL) [44]. In a followup work [43] the authors introduced an IRL model based on a set of manually designed feature functions that capture interaction and collision avoidance behavior of pedestrians. While all of these approaches are capable of predicting pedestrian and bicyclist motions in many scenarios, the need for manual design of features makes them hard to scale in complex driving environments [37].

**Deep learning motion prediction.** Inspired by the success in the various areas of computer vision and robotics, many deep learning-based approaches have been proposed recently for the motion prediction task in order to model object-object and object-scene interactions which may not be straightforward to represent manually. Most of deep learning approaches are based on Long Short-Term Memory (LSTM) variant of recurrent neural networks (RNNs) [11]. Authors of [33] used a sequence-to-sequence LSTM encoder-decoder architecture to predict the pedestrian position and angle of direction. Incorporating the angular information in addition to the temporal features led to a significant improvement in the prediction accuracy. With respect to modeling of dynamic context, [2] proposed an approach for pedestrian motion prediction based on LSTM architecture and a “social pooling” layer that uses spatial information of nearby pedestrians to implicitly model interactions among them. Vemula et al. [38] proposed “social attention” method to predict future motion based on estimation of the relative importance of pedestrians through an attention layer. Recently, [7] proposed an LSTM-based Generative Adversarial Network (GAN) to generate and predict socially feasible motions. On the other hand, with respect to modeling of static context [23] proposed an LSTM-based model for pedestrian’s motion prediction that incorporates the map of static obstacles and position of surrounding pedestrians. Moreover, [20] presented SoPhie, an LSTM-based GAN system for predicting physically and socially acceptable pedestrian trajectories using an RGB image from the scene and the trajectory information of surrounding pedestrians. Recent work [18, 37] incorporated scene information as well as human movement trajectories in the pedestrian motion prediction process. In addition to LSTM-based approaches, [20] proposed CNN-based approach where convolutional layers are utilized to handle temporal dependencies. Authors of [16] use CNNs for joint detection, tracking, and prediction. However, these models do not take into account scene context information, which can provide a strong signal on how the actors would move. On the other hand, in [4] the authors include map information to predict high-level intent of vehicle actors, unlike our model that directly predicts future trajectories.

**Efficient CNN architectures.** Since the introduction of AlexNet [15], researchers made significant progress in improving CNN architectures to make them more accurate and efficient. State-of-the-art architectures, such as VGG [30] or ResNet [9], tend to have a large number of layers running expensive computations, making them unsuitable for real-time inference. Recent proposals such as MobileNet [12] and ShuffleNet [42] replace regular convolutional operator with more efficient depthwise separable or group convolutions, making them small and fast for mobile applications. MobileNet-v2 (MNv2) [27] further improves the original MobileNet by combining depthwise convolution with residual connections and bottleneck layers proposed in ResNet. One problem of this work is its focus on reducing number of floating point operations per second (FLOPS) instead of optimizing for actual inference latency on devices. More recently, MnasNet [34] applied network architecture search algorithms [45] to optimize MNv2 for both accuracy and inference latency on mobile devices, and is able to improve both while maintaining similar FLOPS. ShuffleNet v2 [17] proposed several guidelines for designing fast networks beyond counting FLOPS, and applied these guidelines to design architectures that are both fast and accurate on GPUs and mobile CPUs. In this paper we propose modifications to MNv2 model that make it much faster on GPU without compromising accuracy.

### III. Proposed Approach

In this section we discuss our approach to trajectory prediction of VRU actors. We build upon work described in [6], which considered vehicle actors instead and used rasterized images of actor context as an input to CNNs to predict future trajectories. However, in this study we demonstrate that the methodology can be successfully applied to VRU actors as well (see Figure 1 for an example of pedestrian motion prediction), and explore two aspects of the existing approach that are critical to model’s accuracy and inference speed. First, we experiment with different variations of the CNN architectures, and propose a novel architecture that significantly reduces inference latency without affecting accuracy. Second, we perform ablation study.
of different rasterization configurations, and measured their impact on the prediction accuracy for VRU actors.

Let us assume that we have access to real-time data streams coming from sensors such as lidar, radar, or camera, installed aboard a self-driving vehicle. In addition, assume that these inputs are used by an existing detection and tracking system, outputting state estimates $\hat{S}$ for all surrounding actors (state comprises the bounding box, position, velocity, acceleration, heading, and heading change rate). Denote a set of discrete times at which tracker outputs state estimates as $T = \{t_1, t_2, \ldots, t_p\}$, where time gap between consecutive time steps is constant (e.g., the gap is equal to 0.1s for tracker running at a frequency of $10Hz$). Then, we denote state output of a tracker for the $i$-th actor at time $t_j$ as $s_{ij}$, where $i = 1, \ldots, N_j$ with $N_j$ being a number of unique actors tracked at $t_j$. Moreover, we assume access to detailed, high-definition map information $\mathcal{M}$ of the SDV’s operating area, including road and crosswalk locations, lane directions, and other relevant map information.

Using the state estimates and high-definition map, we rasterize an actor-specific bird’s-eye view raster image encoding the actor’s surrounding map and traffic actors for each actor of interest (see Figure 1). Then, given the $i$-th actor’s raster image and state estimate $s_{ij}$ at time step $t_j$, we use a CNN model to predict its future state sequences up to $H$ steps of horizon $[s_{ij}(j+1), \ldots, s_{ij}(j+H)]$, trained to minimize the average displacement (or $te_2$) error of the predicted trajectory points. Without the loss of generality, in this work we simplify the task to infer the $i$-th actor’s future $x$- and $y$-positions instead of full state estimates, while the remaining states can be derived by considering the current state estimate $s_{ij}$ and the future predicted position estimates. Both past and future positions at time $t_j$ are represented in the actor-centric coordinate system derived from actor’s state at time $t_j$, where forward direction is $x$-axis, left-hand direction is $y$-axis, and actor’s bounding box centroid represents the origin.

Following the approach in [8] we use a MNv2 model as the CNN architecture to compute future positions from the input raster image. Below we describe improvements to this architecture, followed by discussion of variations to the rasterization process that were considered in the ablation study.

A. Improved MobileNet-v2 architecture for fast inference

1) Base CNN: In this section we propose several simple modifications to the MNv2 architecture that significantly speed up GPU inference. MNv2 is based on the inverted bottleneck block illustrated in Figure 2. In each block, the input feature map is first upsampled to $k$ times more channels ($k$ is set to 6 in the original MNv2) with $1 \times 1$ convolution, followed by $3 \times 3$ depthwise convolution (DwConv) applied to the upsampled feature map. Then, the feature map is compressed back to the original channel size using $1 \times 1$ convolution, and summed with the initial input through residual connection. Non-linear activation function (e.g., ReLU) is applied only in the upsampled phase, as non-linearity in the bottlenecked phase (before the upsampling or after the compression) causes too much information loss and hurts model performance. Batchnorm (BN) is used in all 3 layers. While the majority of the FLOPS are in the $1 \times 1$ convolutions (amounting to 87% of the total), the other operations still incur non-negligible cost. As discussed in [17], FLOPS itself is not an accurate metric of actual latency, and another important factor is the number of memory access operations (MAC). Operations such as DwConv, BatchNorm, ReLU, and BiasAdd, while having small FLOPS, typically incur heavy MAC. Especially in MNv2, the operations in the upsampled phase have $k$ times more MAC than the same operations in the bottlenecked phase.

Compared to MNv2, in the proposed novel CNN architecture called FastMobileNet (FMNet) we move most of the operations originally in the upsampled phase into the bottlenecked phase, reducing their FLOPS and MAC by a factor of $k$, see Figure 2. The only remaining operation in the upsampled phase is a ReLU. Similarly to MNv2, no ReLU is applied in the bottlenecked phase as applying non-linearity to the bottlenecked feature map causes significant information loss. As the layers are linear in the bottlenecked phase, we only apply one BiasAdd at the end of the block, as applying multiple BiasAdd in consecutive linear layers does not increase
model expressiveness. Note that we do not use batch-norm in FMNet as we found the model converges well during training without it, and excessive batch-norm operations cost additional computation time. In addition, we need to allow different strides in order to reduce height and width of the feature map during feature extraction. The FMNet block supporting this operation is similar to the regular block (see Figure [2]), except the original input is downsampled to the correct output size for residual connection. The base model architecture (further discussed and extended in the following section) is illustrated in Figure [3], and the FMNet architecture corresponding to the CNN part is shown in Table I, where the layer sizes and block repeats of the model are based on MNv2-0.5 [27] (i.e., MNv2 with halved channel sizes in all layers).

2) Fusion of auxiliary features: Previous work [6] showed that combining the raster input with other state features of actors (e.g., the current and past velocity, position, or actor heading) significantly improves the model accuracy. Thus, it is beneficial to build a network that fuses the raster image input (as a 3D-tensor of size $height \times width \times channels$) and other auxiliary features (as a 1D-vector) that include the actor states and/or other hand-engineered features. A straightforward way to achieve this, as done in [6], is to concatenate the flattened CNN output from the raster image with the 1D auxiliary features, then apply additional fully-connected layers to allow non-linear feature interactions, as shown in Fig. [3]. In this section we propose an alternative, more efficient way to fuse the raster CNN and the auxiliary features. We convert the 1D auxiliary features into a 3D feature map by a sequence of a fully-connected layer, reshaping, and $1 \times 1$ convolution, and fuse it into an intermediate feature map of the CNN by element-wise addition (see Fig. [3b]). In this way, we reuse existing downstream CNN computations to achieve non-linear interactions between raster features and the auxiliary features. This removes a need for the additional fully-connected layer that is used in the fusion through concatenation, thus saving valuable computation time. Furthermore, in this way we allow the feature pixels at different spatial locations of the CNN feature map to interact differently with auxiliary features. We perform feature fusion at the output of FMNet block 3, as described in Table I. As shown in the evaluation section, we found that this spatial feature fusion method improves both model accuracy and latency.

B. Exploring various rasterization settings

To describe rasterization, let us first introduce a concept of a vector layer, formed by a collection of polygons and lines that belong to a common type. For example, in the case of map elements we have vector layer of roads, of crosswalks, and so on. To rasterize vector layer into an RGB space, each vector layer is manually assigned a color from a set of distinct RGB colors that make a difference among layers more prominent. Once the colors are defined, vector layers are rasterized one by one on top of each other, in the order from layers that represent larger areas such as road polygons towards layers that represent finer structures such as lanes or actor bounding

![Diagram](https://via.placeholder.com/150)

Fig. 3: Feature fusion through (a) concatenation; and (b) spatial fusion

| Layer               | Output                  | Stride | Repeats |
|---------------------|-------------------------|--------|---------|
| Raster image        | $300 \times 300 \times 3$ | –      | –       |
| Conv $3 \times 3$   | $150 \times 150 \times 24$ | 2      | 1       |
| DwConv $3 \times 3$ | $75 \times 75 \times 24$  | 2      | 1       |
| FMNet block 1       | $75 \times 75 \times 12$  | 1      | 2       |
| FMNet block 2       | $38 \times 38 \times 16$  | 2      | 3       |
| FMNet block 3       | $19 \times 19 \times 32$  | 2      | 4       |
| FMNet block 4       | $19 \times 19 \times 48$  | 1      | 3       |
| FMNet block 5       | $10 \times 10 \times 80$  | 2      | 3       |
| FMNet block 6       | $10 \times 10 \times 160$ | 1      | 1       |
| Conv $1 \times 1$   | $10 \times 10 \times 640$ | 1      | 1       |
| Global average pooling | $1 \times 1 \times 640$   | 1      | 1       |

TABLE I: Base CNN architecture of FastMobileNet (upsample factor for all FMNet blocks is set to $k = 6$)
boxes. To represent context around the $i$-th actor tracked at time step $t_j$ we create a rasterized image $I_{ij}$ of size $n \times n$ such that the actor is positioned at pixel $(w, h)$ within $I_{ij}$, where $w$ represents width and $h$ represents height measured from the bottom-left corner of the image. We color the actor of interest differently so that it is distinguishable from other actors. More detailed explanation of rasterization can be found in [6].

In this study we evaluated several different choices of rasterization for the prediction of VRU actors, and their impact on the model performance. For all approaches we maintain a constant RGB raster dimension of $300 \times 300$ pixels (i.e., we set $n = 300$), and discuss the specifics of each choice below.

**Raster pixel resolution.** The resolution governs the extent of surrounding context seen by the model. At $0.1m$ resolution, the model sees $25m$ in front and $5m$ behind the actor (assuming the rotated raster discussed above). Larger resolution allows for larger context around the actor to be captured, however the raster loses finer details which may be critical for accuracy. To evaluate this hypothesis we experimented with resolutions of $0.1m$, $0.2m$ and $0.3m$, as illustrated in Figure 4.

**Raster frame rotation.** In the previous work [6] the raster was rotated separately for each actor, such that the heading of each actor points up and the target actor is placed at $w = 150, h = 50$ (as seen in Figure 4). In this way heading of the actor is encoded directly into the input, and the raster encodes more context in front of the actor. We tested an alternative scheme where the raster frame is unrotated such that north direction points up, and the actor is placed in the center (setting $w = h = 150$), as illustrated in Figure 5.

**Lane direction.** In the previous work [6] the direction of each lane segment was encoded as a hue value in HSV color space with saturation and value set to maximum, followed by conversion of HSV to RGB color space. Alternatively we can encode all lanes with a constant color, such that the raster does not contain lane direction information. This is represented in Figure 5b where raster does not encode lane direction, as opposed to Figure 4b where lane color indicates its heading.

**Traffic lights.** We use an existing in-house traffic light classification algorithm to extract current traffic light states from sensor inputs. To encode this info in the input raster, we plot traffic light states as a colored circle at location where lane meets a traffic-light controlled intersection. Furthermore, we identify inactive crosswalks and paint them green (compare Figure 5c where raster does not encode traffic light info, as opposed to Figure 4b).

**Learning raster colors.** In the previous work colors for each raster layer type were picked manually [6]. An alternative approach is to have the DNN learn the colors by itself, optimizing raster image for the prediction task. In this study, we provided all raster layers (e.g., road and crosswalk polygons, tracked objects) to the network as separate binary-valued channels, and added a $1 \times 1$ convolution layer with 3 output channels and linear activation to generate the RGB raster image (see Figure 5d for example of a learned raster). The generated RGB image is then passed to the rest of the network as before.

**Model pre-training.** We also evaluated one modification that is not related to rasterization choices. As the majority of road actors are vehicles, our training data has a much larger number of such actors. To make use of this data, we can initialize our VRU models with a pre-trained vehicle model trained using more examples. The models can then be fine-tuned using corresponding VRU training examples until convergence.
TABLE II: Comparison of various CNN architectures (all models except for the last row use the concatenation feature fusion)

| Architecture          | Pred. error [m] | Latency [ns] | FLOPS   | Num. parameters | MAC    | Num. ops |
|-----------------------|-----------------|--------------|---------|----------------|--------|----------|
| ResNet18              | 1.29            | 36.2         | 6.26G   | 11.7M          | 163 MB | 641      |
| MNv2-0.5              | 1.27            | 21.3         | 308M    | 598K           | 146 MB | 1542     |
| MnasNet-0.5           | 1.28            | 18.3         | 323M    | 844K           | 113 MB | 1490     |
| FMNet                 | 1.28            | 12.1         | 340M    | 565K           | 55 MB  | 336      |
| FMNet with spatial fusion | 1.24          | 10.4         | 285M    | 558K           | 47 MB  | 370      |

IV. EXPERIMENTS

We collected 240 hours of data by manually driving SDV in various traffic conditions (e.g., varying times of day, days of the week). The data contains significantly different number of examples for various actor types, namely 7.8 million vehicles, 2.4 million pedestrians, and 520 thousand bicyclists. Traffic actors were tracked using Unscented Kalman filter (UKF) [40], taking raw sensor data from the camera, lidar, and radar, and outputting state estimates for each object at 10Hz. We considered prediction horizon of 6s (i.e., $H = 60$) for VRU actors. For the default rasterization scheme (used in the architecture experiments and as a base setting in the ablation study), we rotated raster to actor frame with resolution of 0.2m, including both lane heading and traffic light raster layers.

We implemented models in TensorFlow [1] and trained on 16 Nvidia Titan X GPU cards. We used open-source distributed framework Horovod [23] for training, completing in around 24 hours. We used a per-GPU batch size of 64 and Adam optimizer [14], setting initial learning rate to $10^{-4}$ further decreased by a factor of 0.9 every 20,000 iterations.

A. Comparison of CNN architectures

In the first set of experiments we compared a number of CNN architectures, summarizing results in Table II. To ensure fair comparison and avoid potential issues with small data sets, we trained all models on vehicle actors where we set the prediction horizon to 6s. Average prediction error and latency are reported in the table. We skip the feature fusion layers when computing the number of parameters, as the concatenation feature fusion adds a large amount of parameters which complicates the comparison (an $1024 \times 4096$ fully-connected layer for feature fusion adds 4M extra parameters). MAC is approximated by the sum of tensor sizes of all graph nodes. Column “Num. ops” refers to the total number of operations in the TensorFlow graph of each model. The inference latency is measured at a batch of 32 actors on a GTX 1080Ti GPU. As our prediction algorithm performs inference for each actor in the scene, having such a large batch size is not uncommon when SDV is driving on crowded streets. Note that the model latency depends on its implementation, as fusing graph operations manually with TensorFlow custom ops or automatically using Nvidia TensorRT might improve inference latency. For simplicity and to facilitate fair comparison, we implemented all models using TensorFlow built-in operations without additional optimization.

First, we compared the prediction accuracy and inference latency on several base CNN architectures. We found that the proposed FMNet gives similar prediction accuracy as other modern architectures such as ResNet, MNv2, and MnasNet, while being much faster during inference. In terms of the number of FLOPS and parameters, FMNet is similar to MnasNet-0.5 and MNv2-0.5 (which it is based on), while AlexNet and ResNet18 have many more FLOPS and parameters. Fast inference of FMNet can be explained by low MAC and operation counts, which we specifically optimized for during the model design phase. It is interesting to note that AlexNet is the second fastest CNN while having the second largest FLOPS. This is possibly due to it having the smallest number of layers, as evidenced by its lowest operation counts, although its accuracy is not on par with the competing networks.

Secondly, we found that FMNet with spatial feature fusion further improves the accuracy and inference time when compared to the model with feature fusion through concatenation. As discussed in previous section, the spatial fusion allows interactions between raster and state features with awareness of spatial locations, and removes an expensive fully-connected layer used in the original architecture. This resulted in lower FLOPS, as well as lower number of parameters and MAC. Following these results we use the best performing FMNet with spatial fusion as the model architecture in the following ablation studies on rasterization choices.

B. Rasterization ablation studies

We conducted ablation study regarding the rasterization setup, modifying various parameters of the rasterization configuration of the base setup. Empirical results in terms of average prediction error, as well as short- and long-term errors are given in Table III. We can see that the all variants of the proposed CNN approach (referred to as RasterNet) significantly outperform UKF, especially at longer horizons.

First, we analyzed accuracy of the base 0.2m-resolution, as compared to other resolution choices. Resolution of 0.1m has smaller coverage of the surrounding context, and is expected to benefit slow-moving objects such as pedestrians. On the other hand, resolution of 0.3m may benefit faster-moving objects requiring larger coverage, while also resulting in loss of finer details useful for slower-moving actors. As can be seen, for pedestrians 0.1m-resolution indeed resulted in lower error, while setting 0.3m gave the worst performance. We observed that resolution of 0.1m showed no significant difference for bicyclists, while 0.3m resulted in slightly higher error. This may be attributed to the fact that bicyclists are not fast enough to benefit from wider context.

Next, we evaluated the impact of not rotating raster such that actor heading points up, as discussed in Section III-B.
TABLE III: Comparison of prediction errors (in meters) for various experimental settings

| Approach                        | Resolution | Bicyclists |             |          | Pedestrians |             |          |
|---------------------------------|------------|------------|-------------|----------|-------------|-------------|----------|
|                                 |            | Average @1s| 2.80        | 0.67     | 0.22        | 1.22        |          |
|                                 |            | @5s        | 6.60        | 0.51     | 0.17        | 0.90        |          |
|                                 |            | UKF        |             |          |             |             |          |
| RasterNet                       | 0.1m       | 1.07       | 0.43        | 2.73     | 0.51        | 0.17        | 0.90     |
|                                 | 0.2m       | 1.07       | 0.44        | 2.72     | 0.52        | 0.18        | 0.93     |
|                                 | 0.3m       | 1.09       | 0.45        | 2.80     | 0.53        | 0.18        | 0.95     |
|                                 | 0.2m       | 1.29       | 0.49        | 3.30     | 0.58        | 0.20        | 1.02     |
|                                 | 0.2m       | 1.11       | 0.44        | 2.86     | 0.55        | 0.20        | 0.96     |
|                                 | 0.2m       | 1.07       | 0.43        | 2.72     | 0.52        | 0.18        | 0.93     |
|                                 | 0.2m       | 1.05       | 0.42        | 2.70     | 0.53        | 0.18        | 0.93     |
|                                 | 0.2m       | 3.11       | 0.89        | 8.47     | 1.96        | 0.40        | 3.82     |
|                                 | 0.2m       | 1.05       | 0.42        | 2.70     | 0.59        | 0.20        | 1.05     |

Fig. 6: Bicyclist model before and after traffic light turning red; ground-truth (green) and predicted (blue) trajectories overlaid which resulted in a significant drop of accuracy for both actor types. This can be explained by the fact that, when raster is not rotated such that actor heading points up, there is a large number of input data variations that network needs to observe to learn how actors move (so the input data could be augmented by randomly rotating each example). In other words, for such setup actors may move in any direction, which is not the case for rotated raster where actors initially always move upwards, resulting in a simplified prediction problem.

We further investigated the effect of encoding traffic light info. The traffic light is important for predicting longitudinal movement, as it can provide info about whether an actor may or may not pass through an intersection. We observed error increase without traffic light rasterization for both actor types, matching this intuition. Figure 6 gives an illustrative example of how additional traffic information improves prediction, where the actor’s predicted trajectory changed from crossing to not-crossing due to different traffic light states. Furthermore, we removed lane heading information provided in raster images, encoded by using different colors to indicate different directions. Without lane heading bicyclist model degraded slightly in performance, while pedestrian model was unaffected. This again matches intuition, as bicyclists may behave as vehicles at times and follow lane direction, while pedestrians do not use lane heading when in traffic. Finally, we tried learning raster colors instead of setting them manually. The results indicate that learned colors slightly improved accuracy for bicyclists, whereas the pedestrian model slightly degraded compared to the baseline. This suggests that our manual rasterization setup captured sufficient signal when it comes to motion prediction of pedestrians, while for bicyclists it could be suboptimal.

Lastly, due to significantly larger amount of vehicle data as compared to VRUs, instead of training from scratch we fine-tuned VRU models using preloaded weights from a pretrained vehicle model. As observed in the second-to-last row, directly applying the vehicle model to bicyclists and pedestrians without fine-tuning results in poor performance across the board, even worse than UKF, due to a different nature of these actor types as compared to vehicles. On the other hand, with additional fine-tuning using training data of corresponding actor types the bicyclist performance improved over the baseline, indicating that bicyclists may exhibit similar behavior to vehicles. We can also see that the pedestrian model regressed, explained by the fact that pedestrian motion is very different from vehicle motion, thus making vehicle pretraining ineffective.

V. CONCLUSION

We presented an efficient and effective solution to motion prediction of VRU actors. This is a critical problem in autonomous driving, as such actors have higher risk of injury and are less predictable since they may change behavior faster than vehicles. We applied recently proposed rasterization technique to generate raster images of actors’ surroundings encoding their context, used as input to deep CNN trained to predict actor trajectory. Moreover, we proposed a fast architecture suitable for real-time operations, and finally presented a detailed ablation study of various rasterization choices. The results strongly indicate benefits of the proposed approaches, and provide useful insights into the task of motion prediction.
REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/.

[2] Alexandre Alahi, Kratarth Goel, et al. Social LSTM: Human Trajectory Prediction in Crowded Spaces. IEEE, Jun 2016. doi: 10.1109/cvpr.2016.110. URL http://dx.doi.org/10.1109/CVPR.2016.110.

[3] I. Cara and E. d. Gelder. Classification for safety-critical car-cyclist scenarios using machine learning. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pages 1995–2000, Sept 2015. doi: 10.1109/ITSC.2015.323.

[4] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In Conference on Robot Learning (CoRL), 2018.

[5] Aymerly Constant and Emmanuel Lagarde. Protecting vulnerable road users from injury. PLOS Medicine, 7(3):1–4, 03 2010. doi: 10.1371/journal.pmed.1000228. URL https://doi.org/10.1371/journal.pmed.1000228.

[6] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, and Jeff Schneider. Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. arXiv preprint arXiv:1808.05819, 2018.

[7] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. CoRR, abs/1806.09453, 2018.

[8] Golnaz Habibi, Nikita Jaipuria, and Jonathan P. How. Context-aware pedestrian motion prediction in urban intersections. CoRR, abs/1806.09453, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[10] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. Phys. Rev. E, 51:4282–4286, May 1995. doi: 10.1103/PhysRevE.51.4282. URL https://link.aps.org/doi/10.1103/PhysRevE.51.4282.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

[13] L. Huang, J. Wu, F. You, Z. Lv, and H. Song. Cyclist social force model at unsignalized intersections with heterogeneous traffic. IEEE Transactions on Industrial Informatics, 13(2):782–792, April 2017. ISSN 1551-3203. doi: 10.1109/TII.2016.2597744.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[16] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3569–3577, 2018.

[17] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. CoRR, abs/1807.11164, 2018. URL http://arxiv.org/abs/1807.11164.

[18] H. Manh and G. Alaghband. Scene-LSTM: A Model for Human Trajectory Prediction. ArXiv e-prints, August 2018.

[19] NCSA. 2017 fatal motor vehicle crashes: Overview. Technical Report DOT HS 812 603, National Center for Statistics and Analysis, October 2018.

[20] N. Nikhil and B. Tran Morris. Convolutional Neural Network for Trajectory Prediction. ArXiv e-prints, September 2018.

[21] OECD. Safety of vulnerable road users. Technical Report 68074, Organisation for Economic Co-operation and Development, Augist 1998.

[22] E. Ohn-Bar and M. M. Trivedi. Looking at humans in the age of self-driving and highly automated vehicles. IEEE Transactions on Intelligent Vehicles, 1(1):90–104, March 2016. ISSN 2379-8904. doi: 10.1109/TIV.2016.2571067.

[23] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena. A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–8, May 2018. doi: 10.1109/ICRA.2018.8461157.

[24] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In Advances in neural information processing systems, pages 305–313, 1989.

[25] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila. Using road topology to improve cyclist path prediction. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 289–296, June 2017. doi: 10.1109/IVS.2017.7995734.

[26] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, S. H. Rezaatofghi, and S. Savarese. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. ArXiv e-prints, June 2018.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[28] Alexander Sergeev and Mike Del Balso. Horovod: fast
and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[29] David Silver, Aja Huang, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Luciano Spinello, Rudolph Triebel, and Roland Siegwart. Multimodal people detection and tracking in crowded scenes. In *AAAI*, 2008.

[32] Jillian Strauss, Luis F. Miranda-Moreno, and Patrick Morency. Mapping cyclist activity and injury risk in a network combining smartphone gps data and bicycle counts. *Accident; analysis and prevention*, 83:132–42, 2015.

[33] Li Sun, Zhi Yan, Sergi Molina Mellado, Marc Hanheide, and Tom Duckett. 3dof pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2018.

[34] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018. URL [http://arxiv.org/abs/1807.11626](http://arxiv.org/abs/1807.11626).

[35] Eric J Topol. *The patient will see you now: the future of medicine is in your hands*. Tantor Media, 2015.

[36] Chris Urmson et al. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2), 2008.

[37] Daksh Varshneya and G. Srinivasaraghavan. Human trajectory prediction using spatially aware neural architecture search for mobile. *CoRR*, abs/1705.09436, 2017.

[38] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2018.

[39] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663, 2017.

[40] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.

[41] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *CVPR 2011*, pages 1345–1352, June 2011. doi: 10.1109/CVPR.2011.5995468.

[42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[43] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936, Oct 2009. doi: 10.1109/IROS.2009.5354147.

[44] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI’08, pages 1433–1438. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL [http://dl.acm.org/citation.cfm?id=1620270.1620297](http://dl.acm.org/citation.cfm?id=1620270.1620297).

[45] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL [http://arxiv.org/abs/1611.01578](http://arxiv.org/abs/1611.01578).