To ensure safe and efficient operations, an autonomous vehicle needs to accurately predict the future motions of the traffic actors in its surroundings. We propose to rasterize high-definition maps and surroundings of each vehicle in vicinity of self-driving vehicle (SDV), thus providing complete context and information necessary for accurate prediction of future trajectory. We trained deep CNN called RasterNet to predict short-term vehicle trajectories, while accounting for an inherent uncertainty of the traffic motion. Large-scale evaluation on real-world data showed the system provides accurate predictions and well-calibrated uncertainties, indicating its practical benefits. Following extensive offline testing, the system was successfully tested onboard self-driving vehicles.

**Empirical evaluation**

**Experimental setup**
- We collected 240 hours of data by manually driving SDV in Pittsburgh, PA and Phoenix, AZ in various traffic conditions (e.g., varying times of day, days of the week), with collection rate of 10Hz, the same frequency the Unscented Kalman Filter (UKF) tracker was run on.
- Each actor at each discrete tracking time step amounts to one data point, with overall data comprising 7.8 million examples after removing static actors.
- We considered horizon of 3s (i.e., we set H = 30), and used 3:1:1 split to obtain train/validation/test data.

**Table 1. Comparison of average prediction errors for competing methods (in meters)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Raster</th>
<th>State</th>
<th>Loss</th>
<th>Displacement</th>
<th>Along-track</th>
<th>Cross-track</th>
</tr>
</thead>
<tbody>
<tr>
<td>UKF</td>
<td>–</td>
<td>yes</td>
<td>–</td>
<td>1.46</td>
<td>1.21</td>
<td>0.57</td>
</tr>
<tr>
<td>Linear model</td>
<td>–</td>
<td>yes</td>
<td>(2)</td>
<td>1.19</td>
<td>1.03</td>
<td>0.43</td>
</tr>
<tr>
<td>Lane-avoid</td>
<td>–</td>
<td>yes</td>
<td>–</td>
<td>1.09</td>
<td>1.00</td>
<td>0.19</td>
</tr>
<tr>
<td>AlexNet</td>
<td>w/o fading</td>
<td>no</td>
<td>(2)</td>
<td>3.14</td>
<td>3.11</td>
<td>0.35</td>
</tr>
<tr>
<td>AlexNet</td>
<td>w/ fading</td>
<td>no</td>
<td>(2)</td>
<td>1.24</td>
<td>1.23</td>
<td>0.22</td>
</tr>
<tr>
<td>AlexNet</td>
<td>w/ fading</td>
<td>yes</td>
<td>(2)</td>
<td>0.67</td>
<td>0.94</td>
<td>0.21</td>
</tr>
<tr>
<td>VGG-19</td>
<td>w/ fading</td>
<td>yes</td>
<td>(2)</td>
<td>0.86</td>
<td>0.83</td>
<td>0.20</td>
</tr>
<tr>
<td>R2Net-30</td>
<td>w/ fading</td>
<td>yes</td>
<td>(2)</td>
<td>0.73</td>
<td>0.70</td>
<td>0.18</td>
</tr>
<tr>
<td>MobileNet-v2</td>
<td>w/ fading</td>
<td>yes</td>
<td>(5)</td>
<td>0.71</td>
<td>0.68</td>
<td>0.18</td>
</tr>
<tr>
<td>MobileNet-v2</td>
<td>w/ fading</td>
<td>yes</td>
<td>(5)</td>
<td>0.62</td>
<td>0.60</td>
<td>0.14</td>
</tr>
</tbody>
</table>

**How well-calibrated are the uncertainty estimates?**
- We used reliability diagrams to quantify the calibration (note we are somewhat under-confident at 1s).

**Case studies**
- In Figure 5 we give analysis of three scenes commonly encountered in traffic. As we see, the model provided accurate short-term trajectories, as well as reasonable and intuitive uncertainty estimates.

**Future/work**
- Guarantee physical realism of predicted trajectories using vehicle models
- Model interaction between actors
- Learn better trajectory loss function with GANs
- Include explicit constraints on trajectory prediction (e.g. lane following, avoiding static obstacles)
- Combine models of different actor types
- Joint Perception-Prediction models for faster inference

**Proposed approach**
- Given info describing the scene around an actor and their recent history, the task is to predict actor movement as well as its movement uncertainty.
- In the following we define the loss function used to train the CNN.
- For an actor $i$ observed at time $j$, and assuming historical states of all actors $S_j$, map data $M$, and model parameters $\Theta$, we define prediction error at horizon $h$ as an $L_2$-displacement between predicted and ground-truth $x/y$ location:

$$d_{i(j+h)} = \left( (\hat{x}_{i(j+h)} - \hat{x}_{i(j+h)}(S_j, M, \Theta))^2 + (\hat{y}_{i(j+h)} - \hat{y}_{i(j+h)}(S_j, M, \Theta))^2 \right)^{1/2}$$

- Let us assume the displacement loss comes from a half-Gaussian distribution:

$$d_{i(j+h)} \sim \mathcal{N}(0, \sigma_{i(j+h)}(S_j, M, \Theta)^2)$$

- Then, per-actor loss is a negative log-likelihood computed over all horizons $H$:

$$L_{ij} = \sum_{h=1}^{H} \frac{d_{i(j+h)}^2}{2 \sigma_{i(j+h)}(S_j, M, \Theta)^2} + \log \sigma_{i(j+h)}(S_j, M, \Theta)$$

**Network architecture**
- We train a CNN model to minimize and solve the above problem.
- The proposed architecture is given below, outputting $x/y$-locations (total of $2H$ outputs), and standard deviation for each location (additional $H$ outputs).

**In-depth error analysis using error heatmaps**
- In Figure 6 we provide an additional analysis of cross- and along-track errors.
- At each timestamp of the event (x-axis), we color-code errors at each prediction horizon up to 3 seconds in the future (y-axis).
- The actor starts to approach the intersection at around 1s mark, and initiates the turn at around 3s mark.
- While both methods initially do not predict the turn, MN-2 is faster to capture the new behavior.