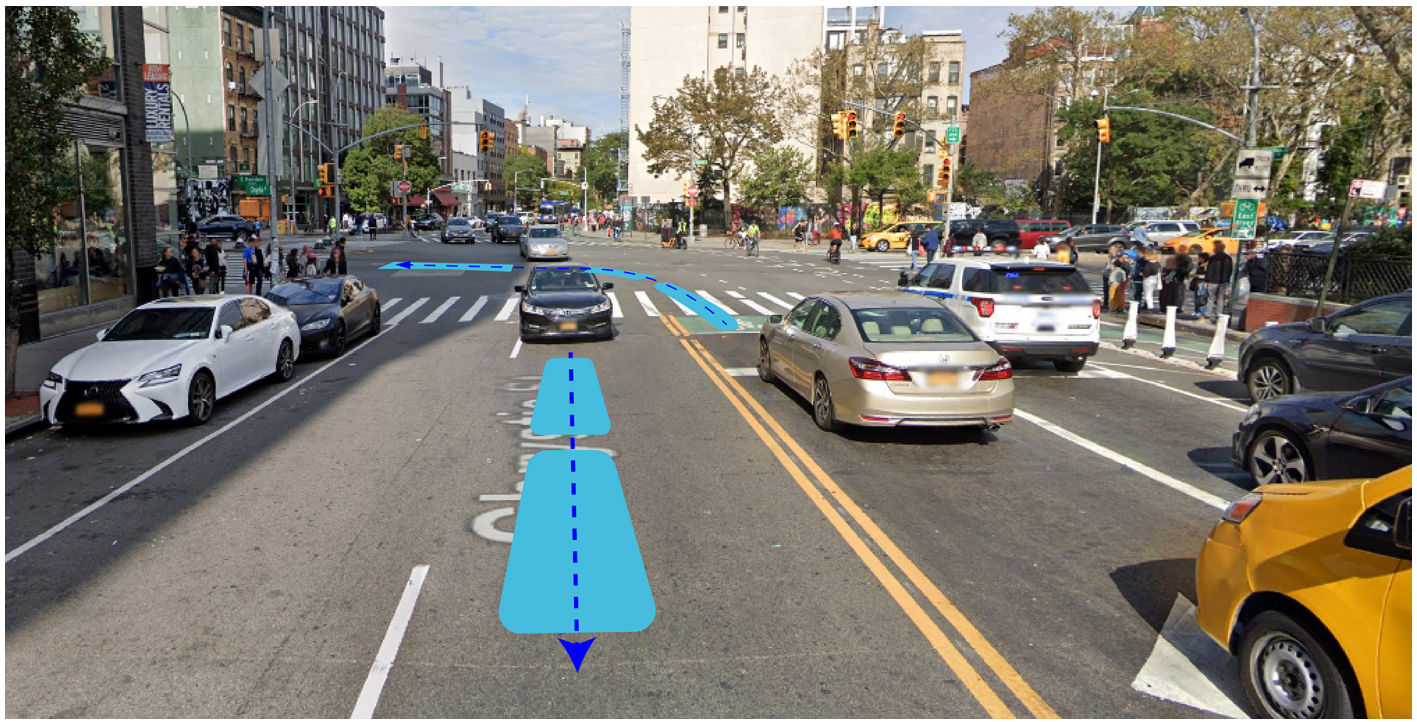
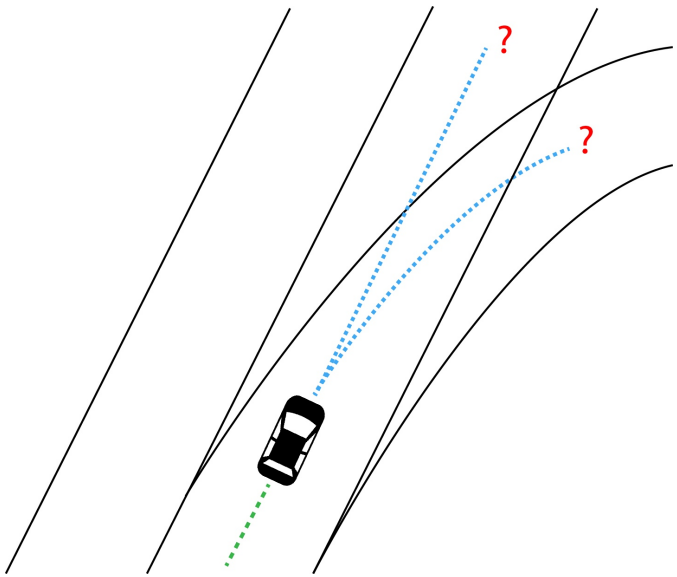
Scene-Compliant GAN for Movement Predictions of Traffic Actors

Dmitry Eremeev, Alexey Pustynnikov



Motion Forecasting

- Predicting future movement of traffic actors is important in context of Self Driving Car (SDC) motion safety.
- Various companies collect huge amounts of traffic data using cars equipped with special tools (radars, lidars, motion sensors, etc.)
- Traffic data contains information about SDC, surrounding agents, lanes and traffic lights.
- Given a tracked actor's state (trajectory and yaws) over the several past timestamps along with rasterized scene image, the task is to predict future positions.



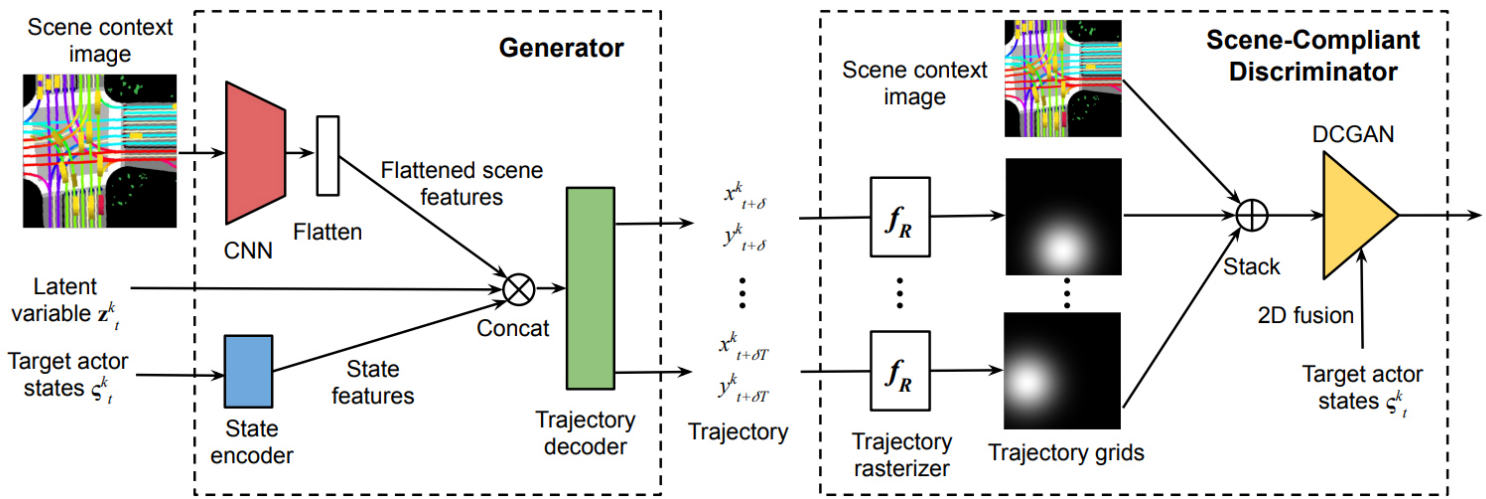
Scene Compliant GAN

arXiv:2004.06247 [cs.LG]

Improving Movement Predictions of Traffic Actors in Bird's-Eye View Models using GANs and Differentiable Trajectory Rasterization

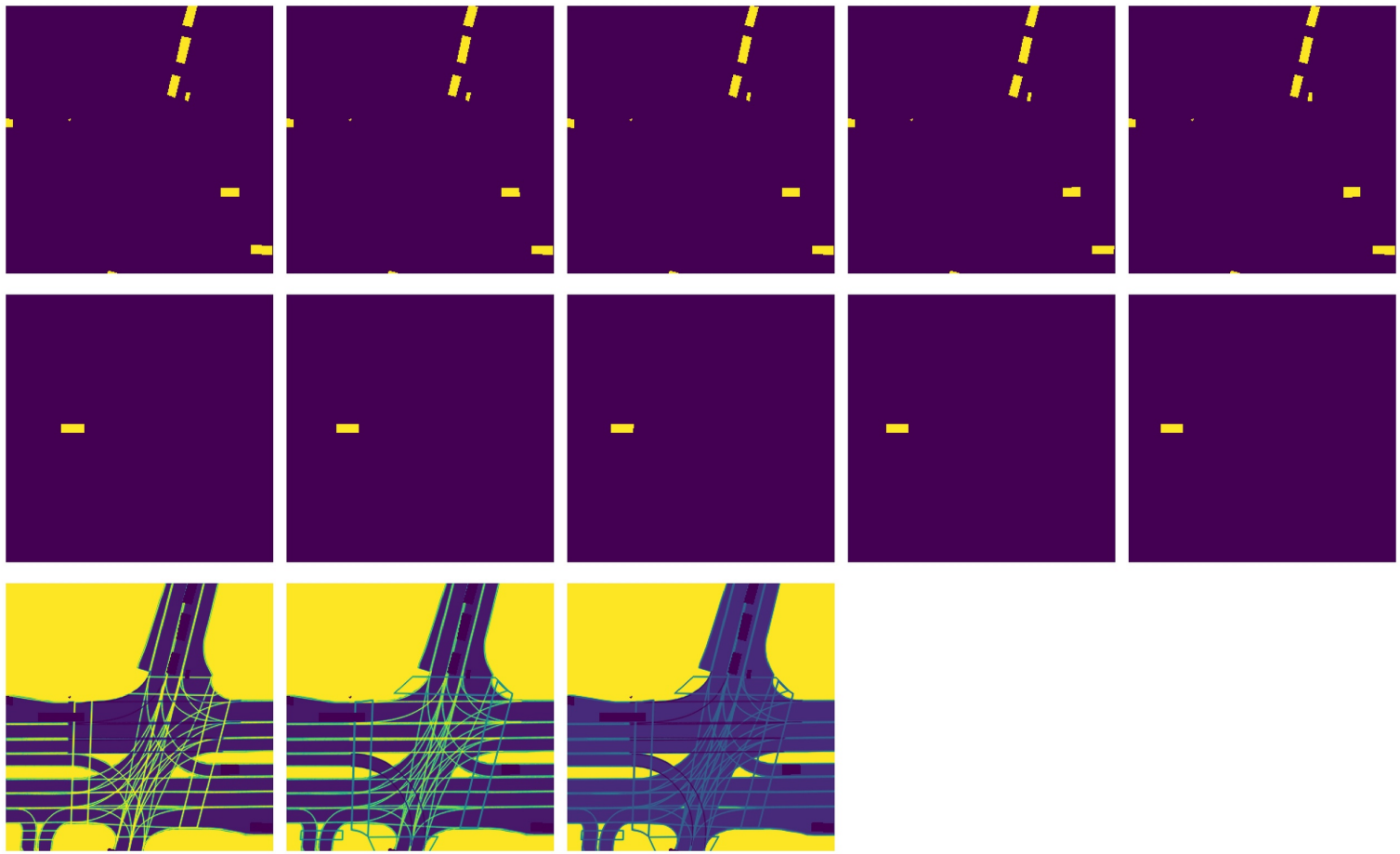
Eason Wang*, Henggang Cui*, Sai Yalamanchi, Mohana Moorthy, Fang-Chieh Chou, Nemanja Djuric
Uber Advanced Technologies Group
{junheng, hcui2, syalamanchi, mmoorthy, fchou, ndjuric}@uber.com

- Relies on Wasserstein GAN with Gradient Penalty.
- Input: history of actor's states (positions and yaws) + rasterized scene image. The generator extracts the scene context features using a convolutional neural network.
- Fully convolutional DCGAN as Discriminator with Differentiable Rasterizer.
- 2D Fusion of past observed states of the actor in discriminator.



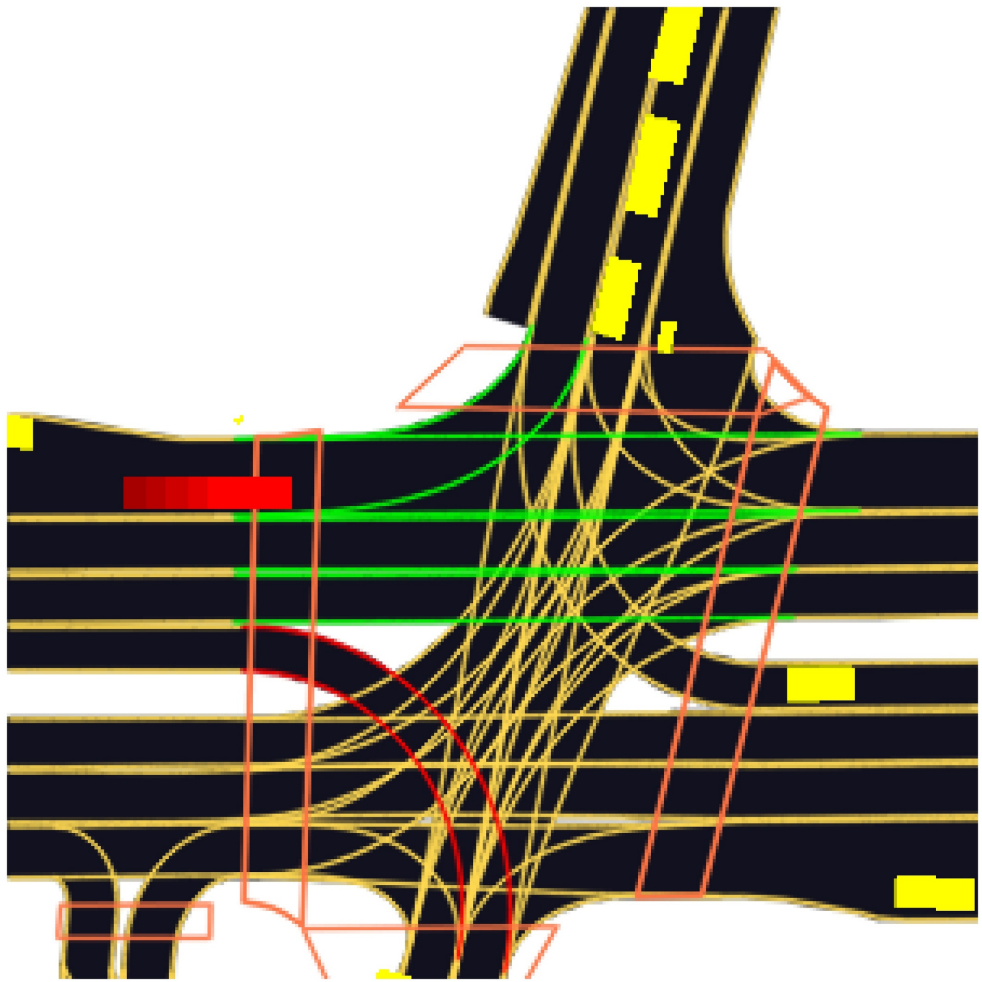
Lyft Level 5

- L5Kit package, motion prediction dataset arXiv:2006.14480
  https://github.com/lyft/l5kit
  https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/data
- Provides a sequences of images for agents and ego-car + images for a scene in different channels in form of Pytorch Dataset



Rasterization for CNN

- Apply a custom rasterizer to get a single RGB image for a scene.
- We use fading to incorporate the history of ego-car and other agents into the image: states that are more distant in the past are more transparent.



Generative Adversarial Networks

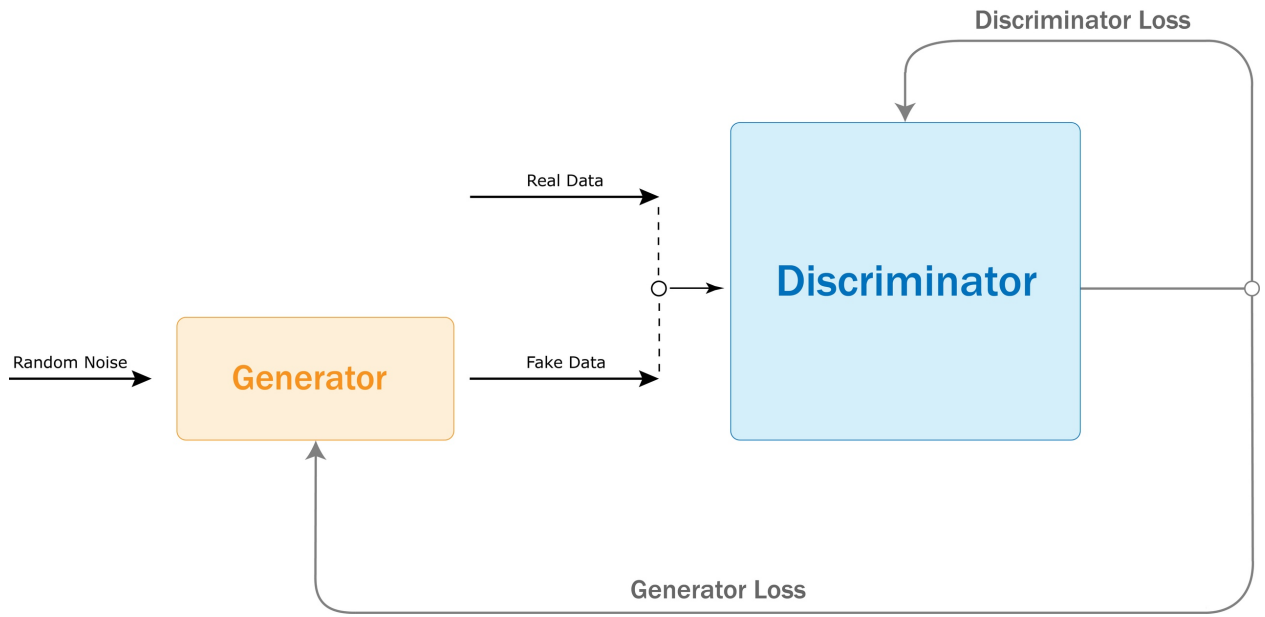
- Generator (G) produces fake samples using noise **z**, Discriminator (D) tries to guess whether given samples are real or fake. arXiv:1406.2661
- D is trained to maximize the probability of assigning the correct label to real training examples and samples from G:

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim \mathcal{N}(0,1)} [\log(1 - D(G(z)))] \longrightarrow \max$$

- G learns to generate samples that have a low probability of being fake:

$$\mathcal{L}_G = \mathbb{E}_{z \sim \mathcal{N}(0,1)} [\log(1 - D(G(z)))] \longrightarrow \min$$

- Commonly implemented as binary classification: minimizing **binary cross entropy** using **1** for real and **0** for generated samples.



Wasserstein GAN

- Vanilla GAN is unstable. For a stable learning process, Wasserstein GAN proposes a new cost function based on Wasserstein distance: arXiv:1701.07875

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim p_{data}} [D(x)] - \mathbb{E}_{\tilde{x} \sim p_{gen}} [D(\tilde{x})], \quad \mathcal{D} \text{ is 1 - Lipschitz.}$$

- Critic D now scores with a **real value instead of probability**.
- Originally Lipschitz constraint was enforced by critic's weight clipping:

$$w \longleftarrow \text{clip}(w, -c, c).$$

Wasserstein GAN with Gradient Penalty

- Another way to satisfy the constraint is to add Gradient Penalty that penalizes gradient norm for random samples $\hat{x} \sim \hat{p}$ arXiv:1704.00028
- Sample $\hat{x}$ uniformly along straight lines between pairs of points that sampled from the data distribution $\mathbf{p_{data}}$ and G distribution $\mathbf{p_{gen}}$:

$$\hat{x} = \alpha x + (1 - \alpha)\tilde{x}, \quad \alpha \sim U[0, 1],\ x \sim p_{data},\ \tilde{x} \sim p_{gen}.$$

- Loss functions:

$$\mathcal{L}_D = \mathbb{E}_{z \sim \mathcal{N}(0,1)} [D(G(z))] - \mathbb{E}_{x \sim p_{data}} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim \hat{p}} \left(\|\nabla_{\hat{x}} D(\hat{x})\| - 1 \right)^2 \longrightarrow \text{min}$$

$$\mathcal{L}_G = -\mathbb{E}_{z \sim \mathcal{N}(0,1)} D(G(z)) \longrightarrow \text{min}$$

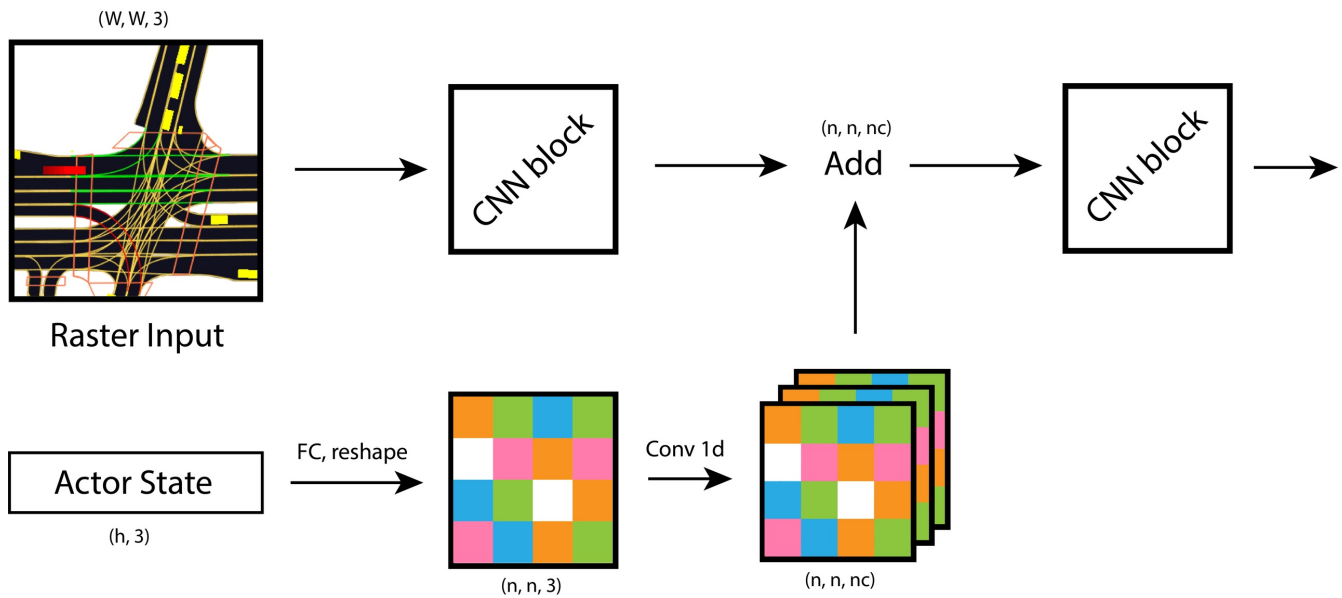
Variety Loss, 2D Fusion

- We add to generator loss an additional best-of-K ℓ_2 variety loss between the ground truth $\mathbf{Y}$ and one of K generated predictions $\hat{\mathbf{Y}}^{(K)}$:

$$\mathcal{L}_{\text{variety}} = \min_K \|\mathbf{Y} - \hat{\mathbf{Y}}^{(K)}\|_2^2$$

- We use the prediction with **lowest loss to update** the model parameters.
- D uses 2D-Fusion to incorporate the past observed states into the image:

arXiv:1906.08469



Differentiable Rasterizer

- Simplifies the Discriminator's job by rasterizing the future trajectory $\{[x_t, y_t], t \in T\}$ into a sequence of 2D grids $\mathcal{G}_t$.
- Each grid is represented by a 2D Gaussian distribution around the particular point of future trajectory. The grids are stacked along channels.
- For a cell $[i, j]$ of a grid $\mathcal{G}_t$:

$$\mathcal{G}_t^{ij} = \mathcal{N}(\Delta_t^{ij} | 0, \Sigma),$$

where covariance matrix is $\Sigma = \text{diag}([\sigma^2, \sigma^2])$,
Δ_t^{ij} is a distance between a grid's cell $[i, j]$ and the trajectory point $[x_t, y_t]$ (given in the actor's frame of reference):

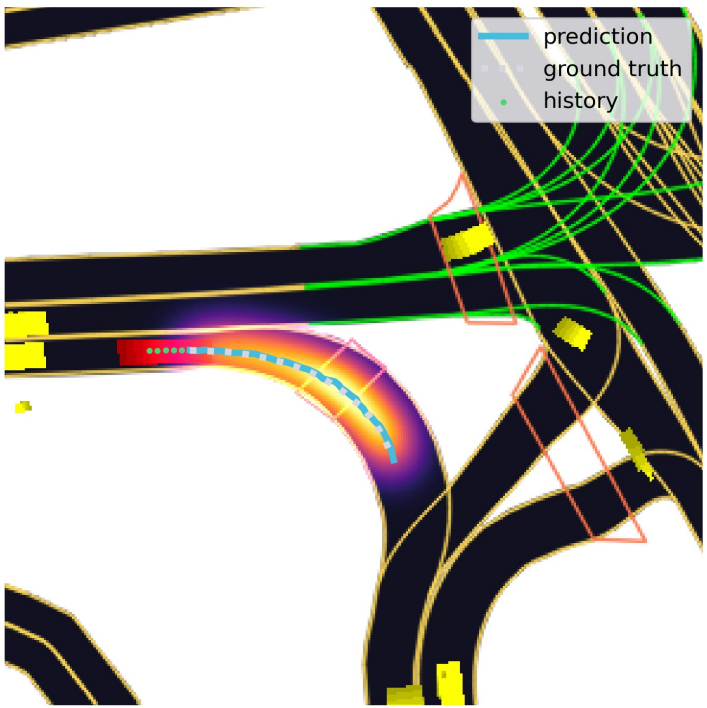
$$\Delta_t^{ij} = [(i - h_0)r - x_t, (j - w_0)r - y_t],$$
$$r - \text{raster resolution},$$
$$[h_0, w_0] - \text{original cell of the actor in the image}.$$

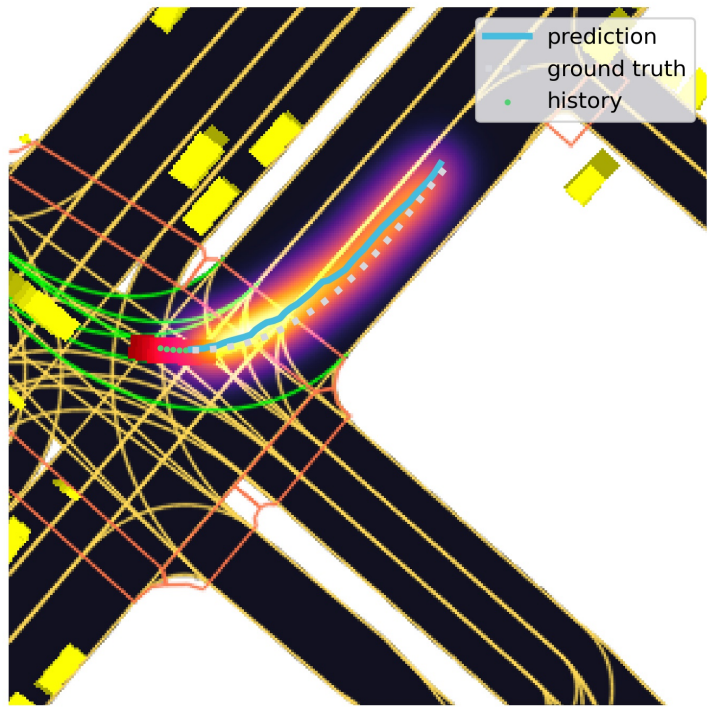
- One can easily calculate the gradient:

$$\text{grad}\, \mathcal{G}_t^{ij} = \frac{\mathcal{G}_t^{ij}}{\sigma^2} \Delta_t^{ij}.$$

Differentiable Rasterizer Visualization

Below are examples of differentiable rasterizer's work on predictions produced by Generator. For visual presentation we combined all channels with grids to one by summing them.

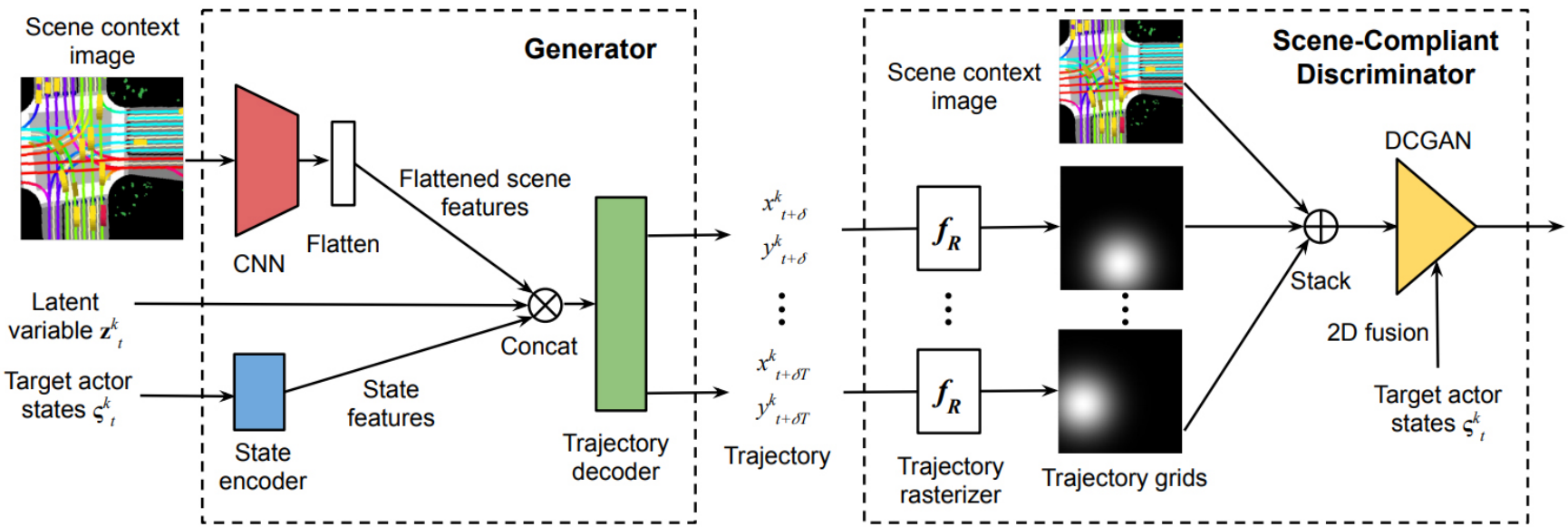




SC-GAN architecture

arXiv:2004.06247 [cs.LG]

- Run **three** D steps for **every** G update step.
- Use MobileNet v2 as CNN in G.
- Our implementation is based on PyTorch
  https://github.com/d-eremeev

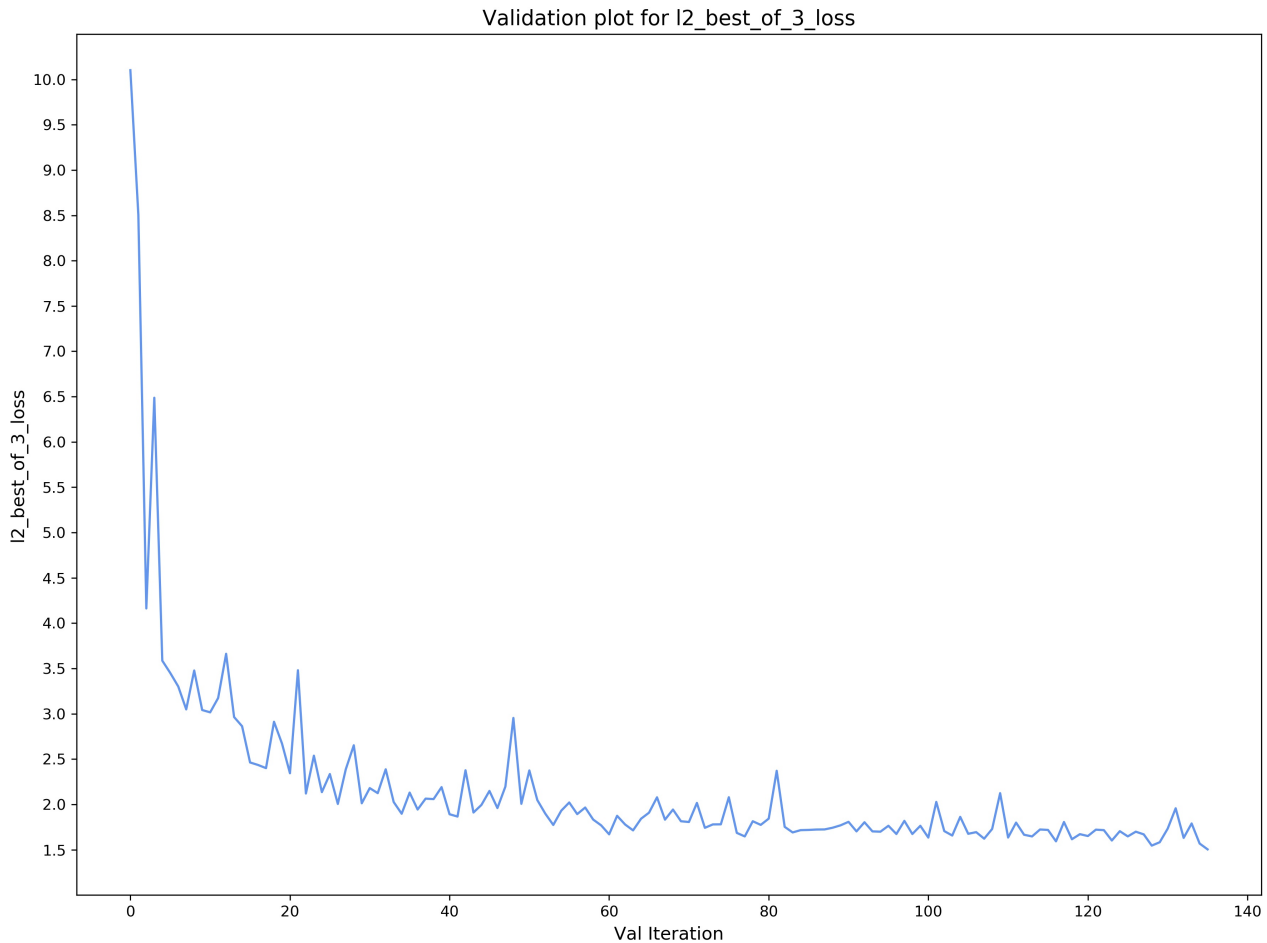


Results

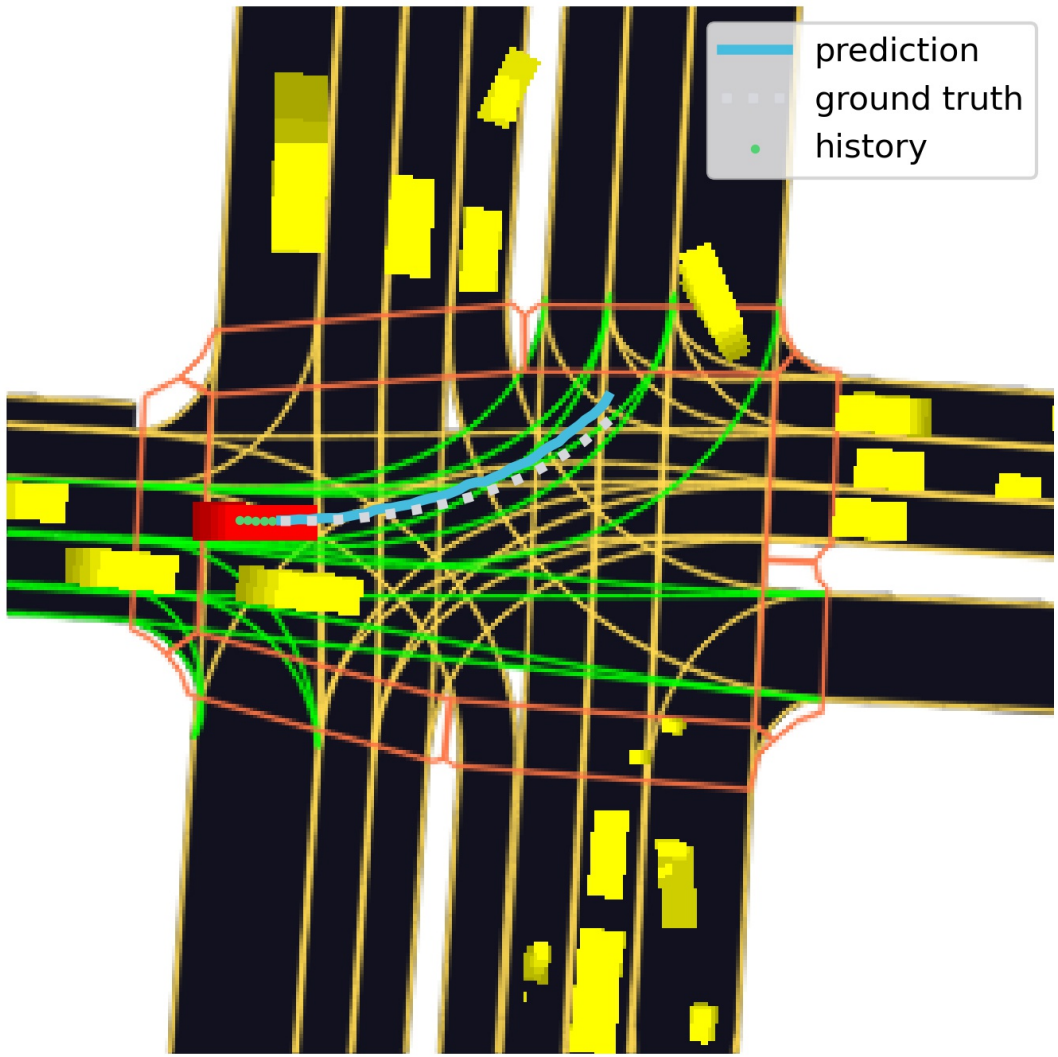
- We validate on 20000 random scenes with **non-static** ego-cars.
- We use average ℓ_2 error **per** trajectory **point** as metric.

Results (ℓ_2, m)

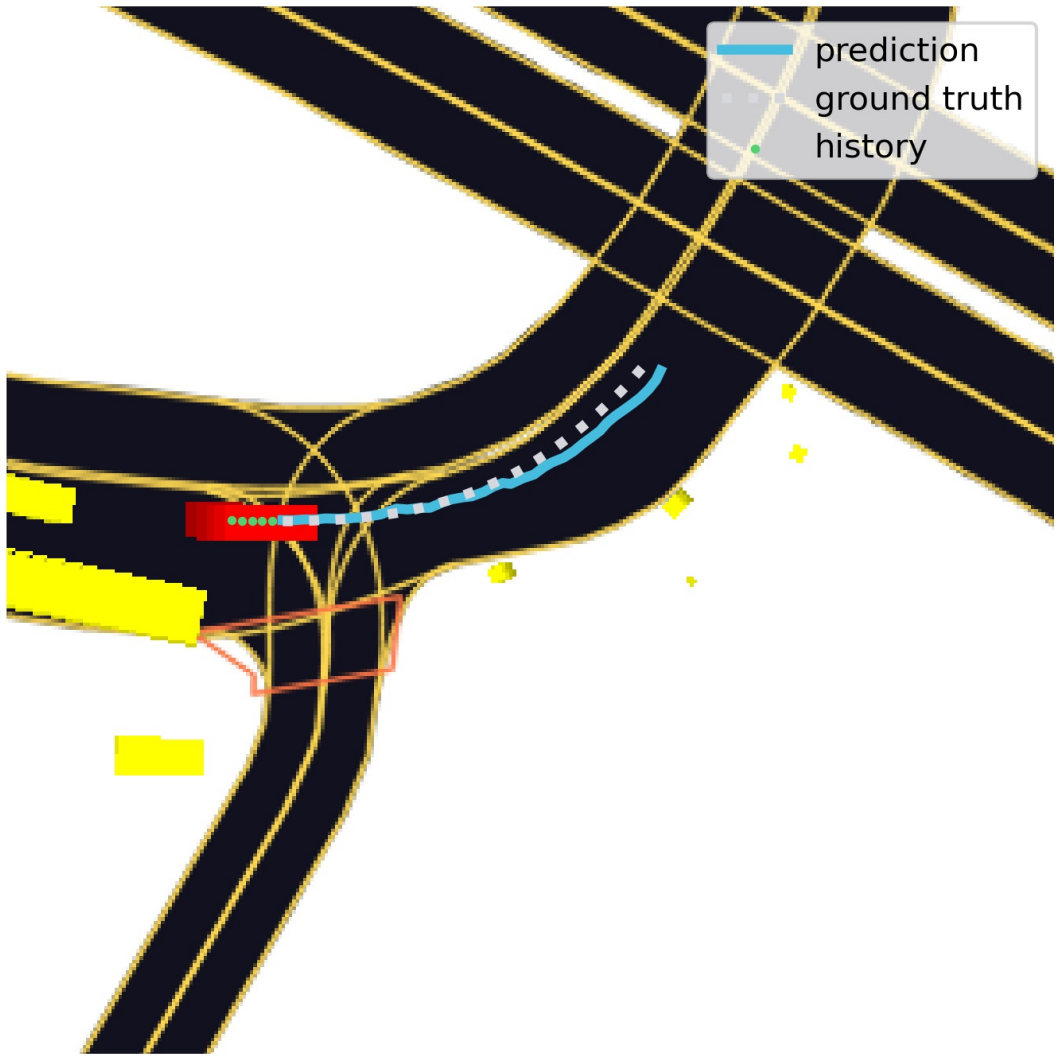
best–of–3	best–of–20
1.503	0.684



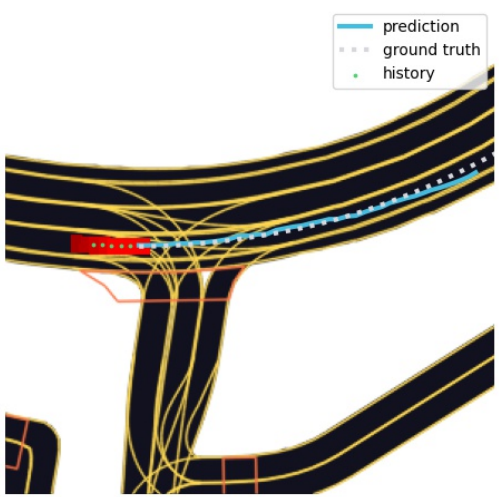
Examples

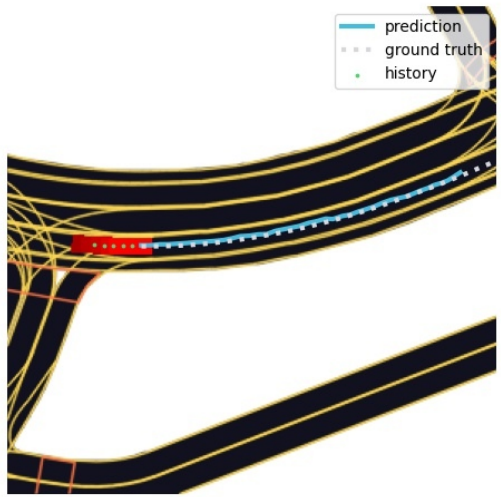


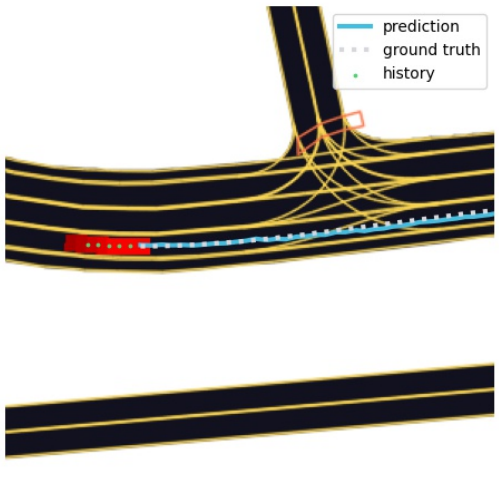
Examples



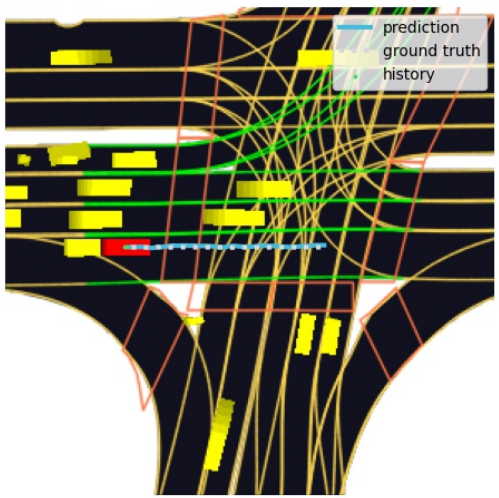
Examples

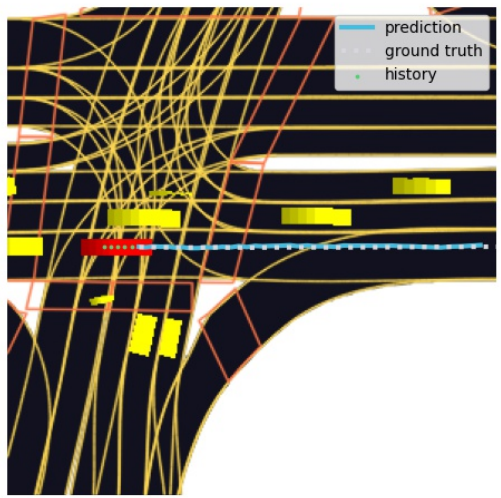

prediction
ground truth
history

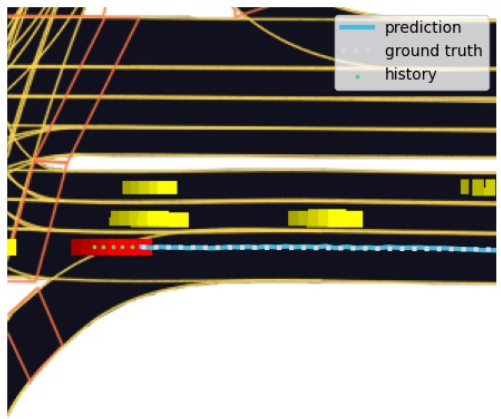

prediction
ground truth
history


prediction
ground truth
history

Examples


prediction
ground truth
history


prediction
ground truth
history


prediction
ground truth
history