Surrounding Motion Predict Model for Autonomous Vehicle

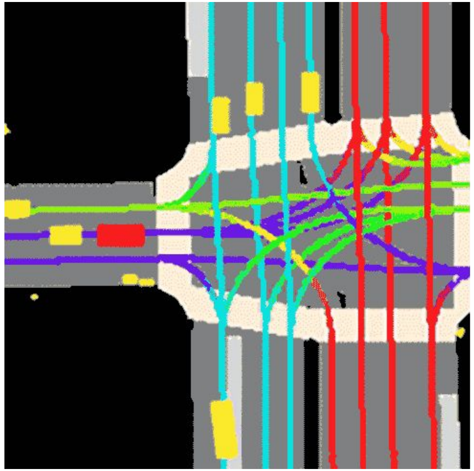
Tieming Sun, Wenhao Cui, Guangrui Shen

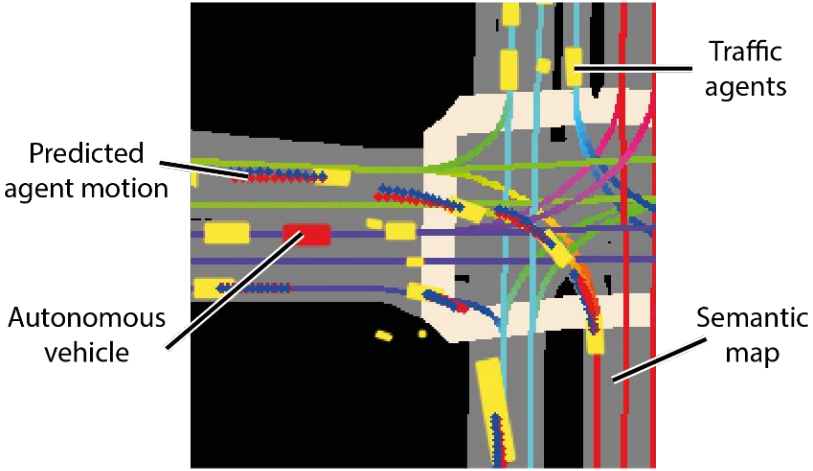
Mentor: Olaoluwa Adigun

EE599 - Deep Learning - Fall 2020

USC Viterbi School of Engineering

Motivation





- Predict surrounding agents motions of the autonomous vehicle over 5s given their current and historical positions
  - Useful for planning self driving vehicle's movement
- Deep learning techniques (CNNs) + Ensemble Models

https://self-driving.lyft.com/level5/prediction/

Outline

- Overview
- Challenges
- Architecture
- Metrics
- Implementation
- Conclusions

Project Overview

- Use L5kit library to access and visualize the dataset
- Build motion prediction model using ResNet 34 (Baseline), MixNet-m, MixNet-l, MixNet-xl, Ensemble models
- Choose negative multi-log-likelihood as evaluate metric

https://arxiv.org/pdf/2006.14480v2.pd
f

Challenges

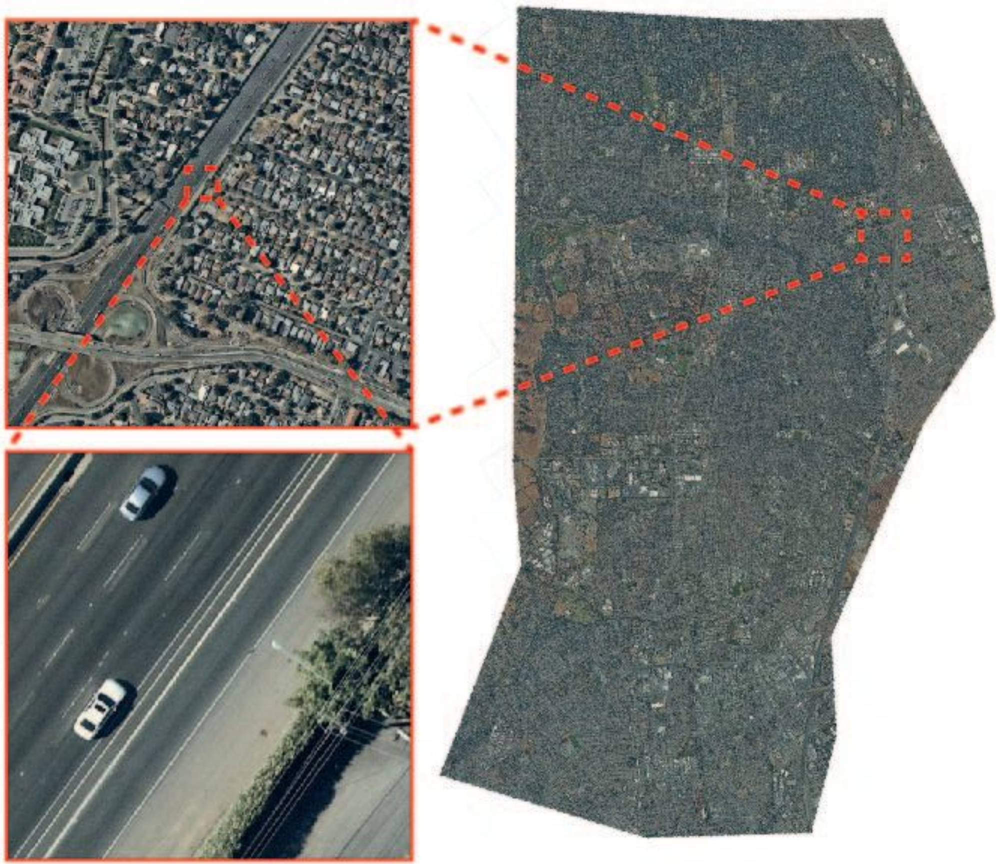
USC Viterbi School of Engineering

- Choose best architecture compromising model speed and prediction capability
- Understand MixNet architecture
- Decide which loss function to use on different training stages
- Ensemble models into multiple trajectories due to the high ambiguity of real world road environment
- Understand the influence of hyperparameters
  - Raster size
  - Pixel size
  - Batch size
  - Traffic lights

Dataset Introduction

USC Viterbi School of Engineering

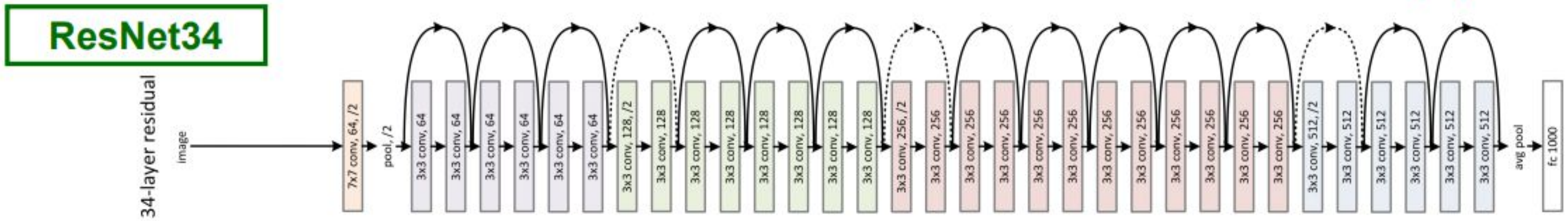
- No. of Scenes: **170,000** scenes
- Period: **25** seconds long
- Total time: **1,118** hours
- Semantic map: **8500** lane segments and **15000** annotated traffic agents.
- Agents: Cars (**92.47%**), pedestrians (**5.91%**) and cyclists (**1.62%**).
- Aerial image spanning **74 km²** at a resolution of **6 cm** per pixel



https://arxiv.org/pdf/2006.14480v2.pdf

Architectures

Baseline Model: ResNet 34



- Input channels : 2 * (10 + 1) + 3 = 25
- Output size : 50 * 2 = 100

[He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).]

Architectures

MixNet

Intuition: The smaller kernels (3x3, 5x5) serve to capture lower resolution details, while the larger kernels (7x7, 9x9) capture higher resolution patterns and thus ultimately build a more efficient network.

MixConvs: Blend multiple kernels sizes into a single layer

MixNets: Ultimately using a range of 1-5 kernels was deemed to be the best standard. By leveraging Neural Architecture Search (NAS), the final MixNets models were built.

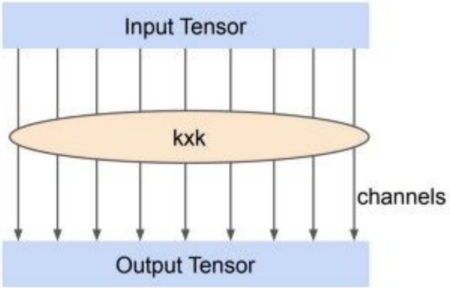


Figure: Vanilla Depthwise Convolution

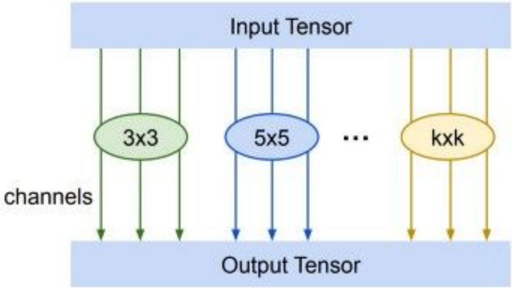


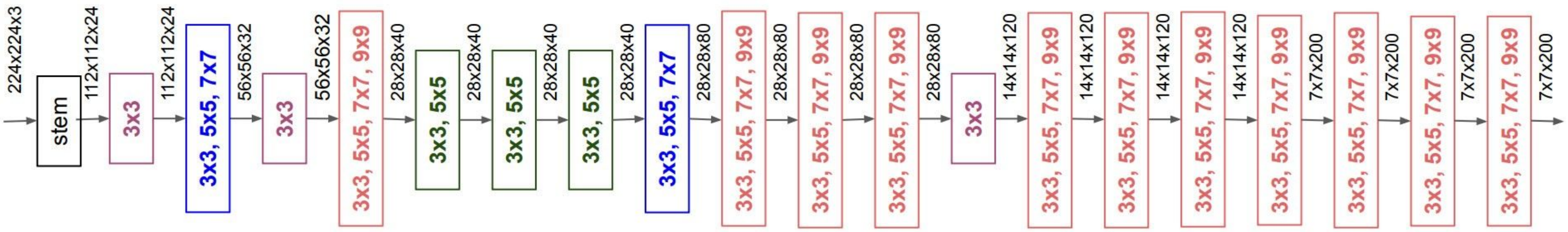
Figure: MixConv

TAN, LE: MIXCONV: MIXED DEPTHWISE CONVOLUTIONAL KERNELS

Architectures

3 hypotheses: MixNet-m, MixNet-l, MixNet-xl

- Input channels: 25
- Output size: 100



MixNet-M

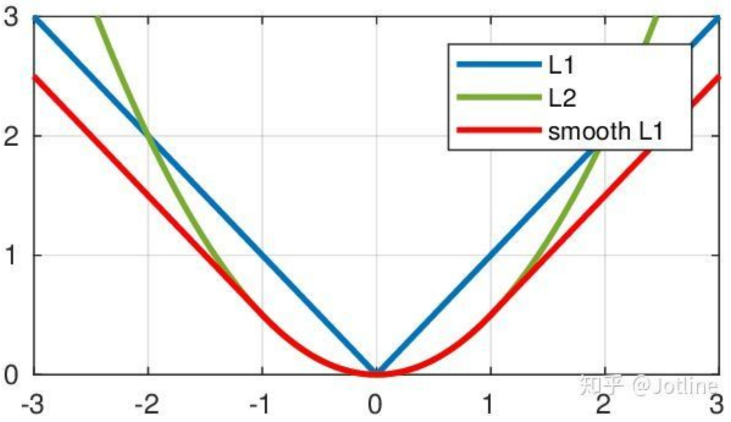
- m, l, xl are degrees of MixNet layers depth.
- E.g. MixNet-l is simply a 1.3 depth multiplier of -m

TAN, LE: MIXCONV: MIXED DEPTHWISE CONVOLUTIONAL KERNELS

Metrics

Training loss function 1: SmoothL1Loss

$$loss(x,y)=\frac{1}{n}\sum_{i=1}^{n}\begin{cases}0.5\times(y_i-f(x_i))^2, & if\ |y_i-f(x_i)|<1\\ |y_i-f(x_i)|-0.5, & otherwise\end{cases}$$



- Often used in regression problem or in the occasion that exist relatively big numbers in features
- Smooth gradient when prediction and ground truth are close

USC Viterbi School of Engineering

Metrics

Training loss function 2: Negative Multi-Log-Likelihood

Ground truth positions of a sample trajectory: $x_1, x_2, \ldots, x_{50}, y_1, y_2, \ldots, y_{50}$

3 predicted positions, represented by means: $\bar{x}_1^3, \bar{x}_2^3, \ldots, \bar{x}_{50}^3, \bar{y}_1^3, \bar{y}_2^3, \ldots, \bar{y}_{50}^3$

Set 3 confidences c of the 3 hypotheses.

Assumptions: Independent Gaussian mixture model, yielding the likelihood:

$$P(x_{1,\ldots,50}, y_{1,\ldots,50} \mid c^{1,\ldots,3}, \bar{x}_{1,\ldots,50}^{1,2,3}, \bar{y}_{1,\ldots,50}^{1,2,3})$$
$$= \sum_{k=1}^{3} c^k N(x_{1,\ldots,50} \mid \bar{x}_{1,\ldots,50}^{k}, \Sigma = 1) N(y_{1,\ldots,50} \mid \bar{y}_{1,\ldots,50}^{k}, \Sigma = 1)$$
$$= \sum_{k=1}^{3} c^k \prod_{t=1}^{50} N(x_t \mid \bar{x}_t^k, \sigma = 1) N(y_t \mid \bar{y}_t^k, \sigma = 1)$$

$$NLL = -\log P(x_{1,\ldots,50}, y_{1,\ldots,50} \mid c^{1,2,3}, \bar{x}_{1,\ldots,50}^{1,2,3}, \bar{y}_{1,\ldots,50}^{1,2,3})$$
$$= -\log \sum_{k=1}^{3} e^{\log(c^k) - \frac{1}{2}\sum_{t=1}^{50}((\bar{x}_t^k - x_t)^2 + (\bar{y}_t^k - y_t)^2)}$$

https://github.com/lyft/l5kit/blob/master/competition.md

Implementation

1. Data generation and visualization using released Python toolkit l5kit by Lyft

Figure: Satellite View: Ground Truth Trajectory of Autonomous Vehicle

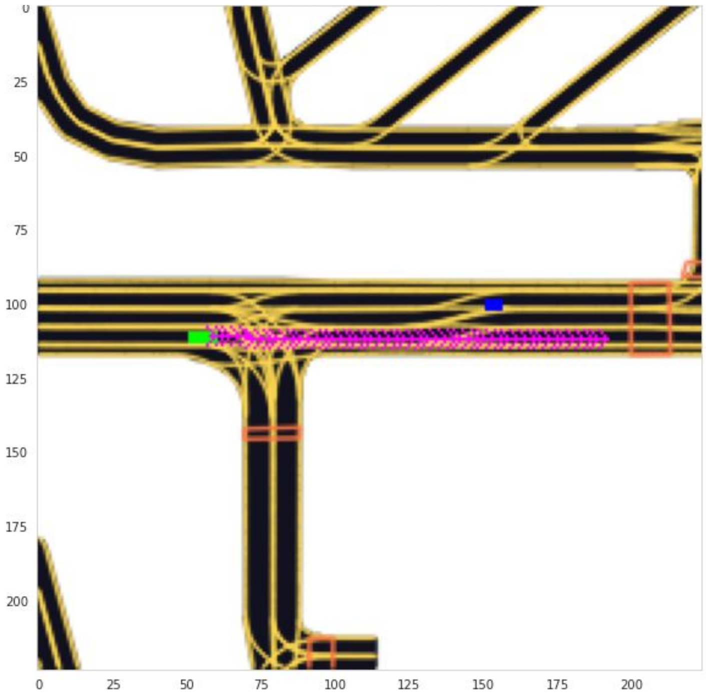


Figure: Semantic View: Ground Truth Trajectory of Autonomous Vehicle

Implementation

1. Data generation and visualization using released Python toolkit l5kit by Lyft

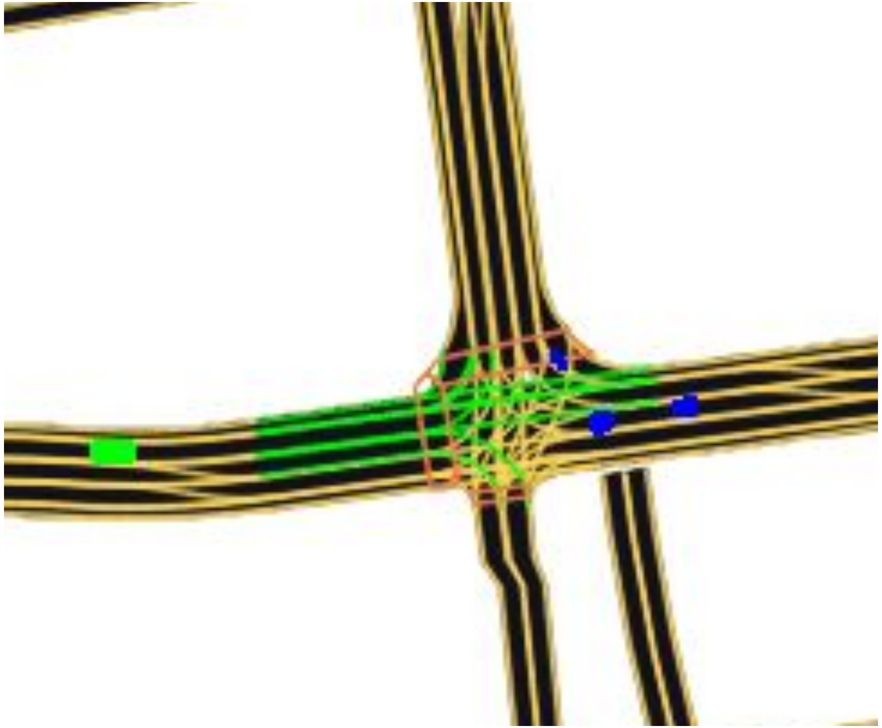


Figure. Self-driving Vehicle Movement during a scene

Implementation

USC Viterbi School of Engineering

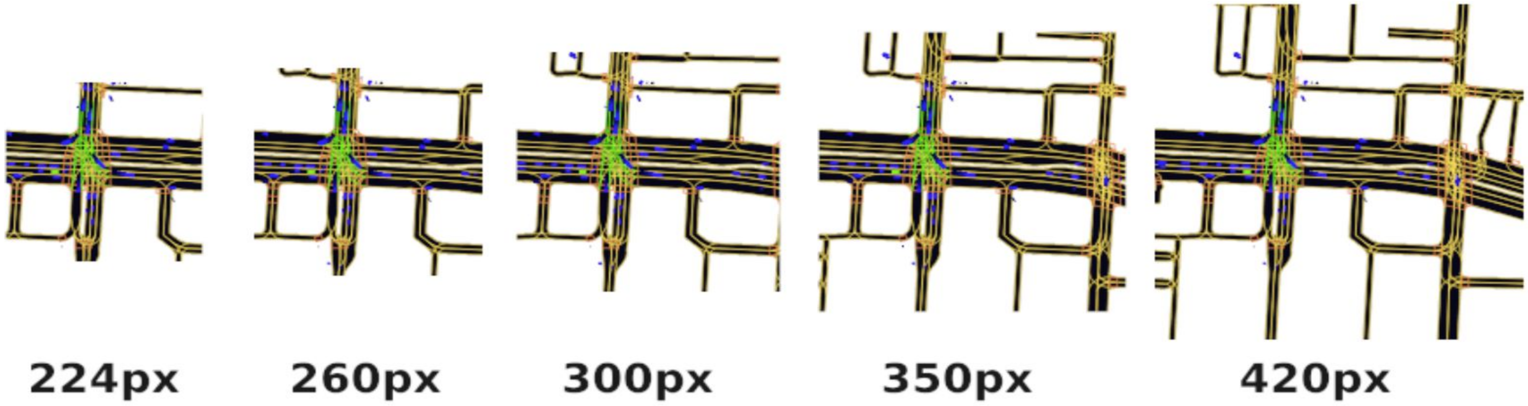
2. Set Configurations for training, validate dataset

- Raster Size: [300, 300]
- Pixel Size: [0.5, 0.5]
- Ego Center: [0.25, 0.5]
- Filter Agents Threshold: 0.5
- History Frames: 10
- Future Frames: 50
- Delta Time: 0.1
- Batch Size: 32
- Disable Traffic Light Faces: False

Implementation

2. Set Configurations for training, validate dataset

- Remove traffic light Mask Layer
    - Traffic light histories has counterproductive effects.
    - Agents velocity has already been considered.
- Image and raster size selection
    - raster_size decides the rasterized image final size in pixels (eg: [300, 300]).



Implementation

3. Training

- Model : **mixnet_xl**
  - batch size: **32**
  - Iterations: **43500**
- Learning Rate **1e-4** with L2 regularizer (weight decay rate) **1e-6**
- Adam optimizer
- Loss Function: use Smooth L1 to converge easier, then Negative Log Likelihood (NLL)

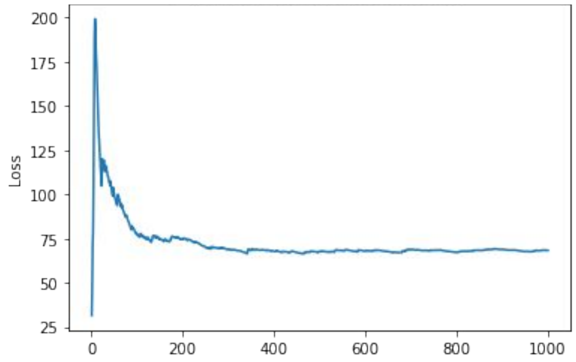


Figure. Average Loss during training, changing from L1 loss to NLL

Implementation

USC Viterbi School of Engineering

4. Performance

Table. Negative Log Likelihood Loss vs. Several Configurations

Model	train	validation
mixnet_m, 1 traj, 35000 iterations	55.6	61.4
mixnet_l, 1 traj, 50000 iterations	52.9	53.0
mixnet_xl, 1 traj, 43500 iterations	48.3	**51.8**
Ensembled (with confidence), 3 traj	NULL	34.9
Ensembled (K-means), 3 traj	NULL	**31.4**

Implementation

4. Performance

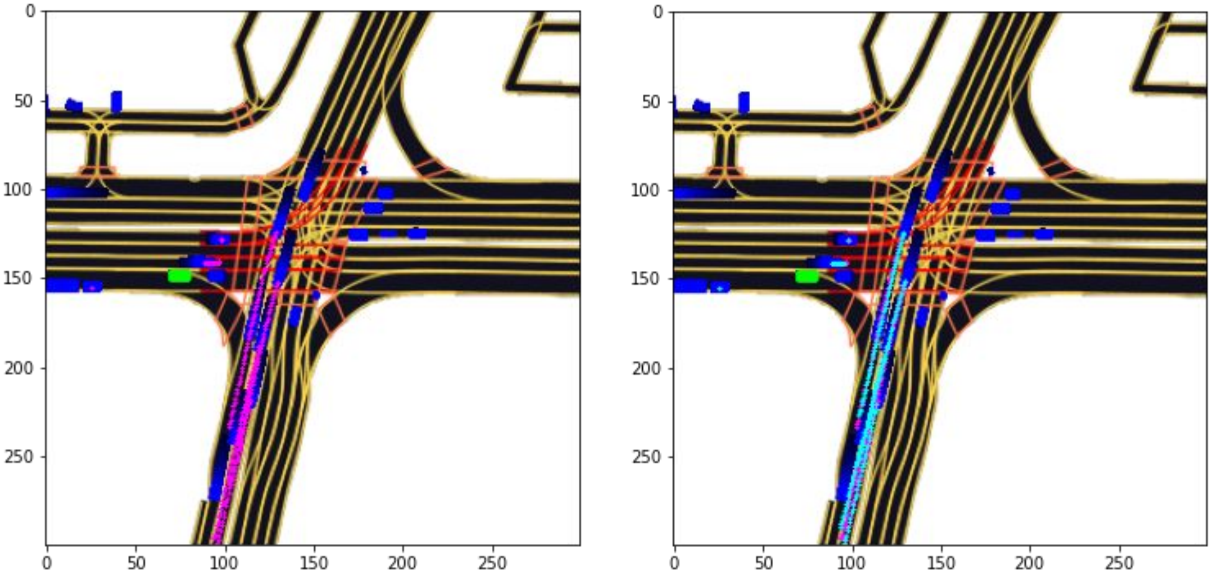


Figure. Example for ground truth (pink line, left) and model prediction (light blue line, right)

Implementation

4. Performance

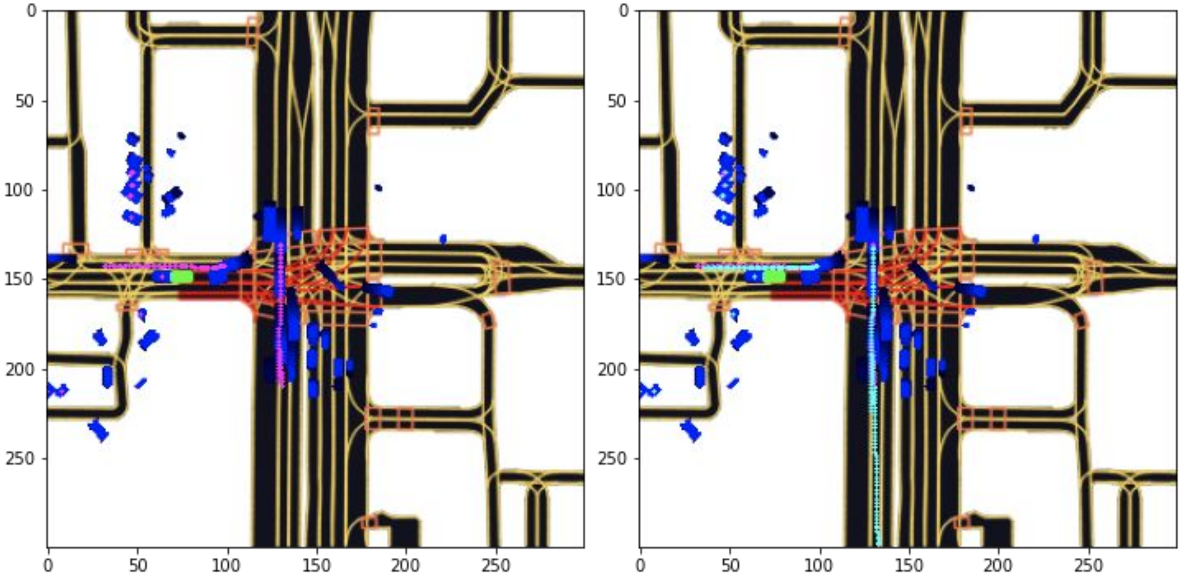


Figure. Example for ground truth (pink line, left) and model prediction (light blue line, right)

Implementation

4. Performance

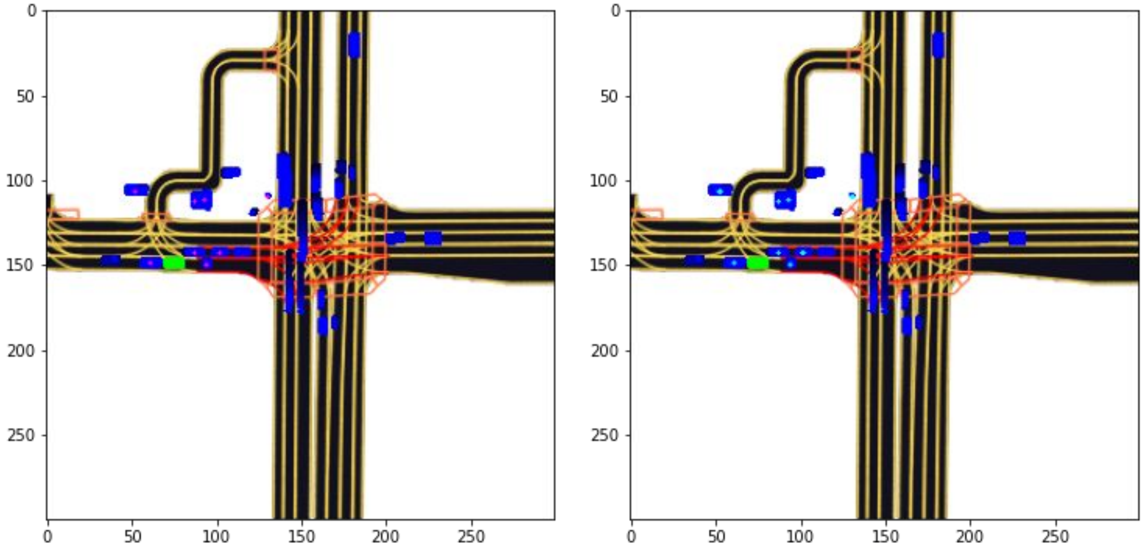


Figure. Example for ground truth (pink line, left) and model prediction (light blue line, right)

Tricks We Tried

- Penalize more on corner cases (pull over, turn, U-turn)
    - If yaw(deviation angle) approximately < 30 degree, set 0.5

$$Custom\ NLL = \begin{cases} NLL \times penalize \times arccos\left(\frac{|dx|}{\sqrt{dx^2+dy^2}}\right) & arccos\left(\frac{|dx|}{\sqrt{dx^2+dy^2}}\right) \geq 0.5 \\ NLL \times penalize \times 0.5 & arccos\left(\frac{|dx|}{\sqrt{dx^2+dy^2}}\right) < 0.5 \end{cases}$$

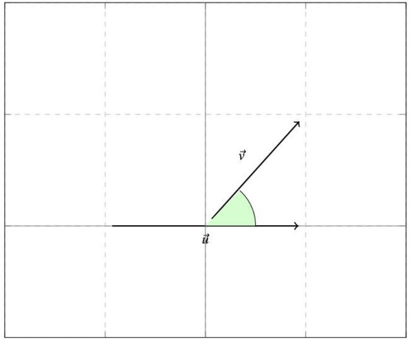


Figure. Penalize is proportion with arc-cosine function

- K-means algorithm: Ensemble 10 Models into 3
    - Clustering several models, vote 3 cluster centroids to represent them
    - Useful when checkpoints are set, not sure whether models are underfitting or overfitting
    - Outperform confidence ensemble by 3.5

Conclusions

- Do not quit too early
  - Train more than 200k samples to make sure performance
  - Be cautious about corner cases
- Be patient when fine-tuning hyperparameters
  - Underfitting model in most cases
  - Improve batch size
- Worth it to try different metrics as the loss function
  - Different metrics have different objectives
  - Smooth L1 VS. NLL VS. Customed NLL
- Future Work
  - GAN
  - LSTM Encoder & Decoder + Social Pooling Layer

[DEO, TRIVEDI: CONVOLUTIONAL SOCIAL POOLING FOR VEHICLE TRAJECTORY PREDICTION]
[KOSARAJU, SADEGHIAN: SOCIAL-BIGAT: MULTIMODAL TRAJECTORY FORECASTING USING BICYCLE-GAN AND GRAPH ATTENTION NETWORKS]