**Motion Prediction for Autonomous Vehicles**

Highly automated technology, self-driving cars, promise a range of potential benefits like greater road safely, enhanced independence for seniors and disables, ability to recapture driving time, fewer roadway crashes and reduced traffic congestion. With huge amount of data and leveraging deep learning paradigms, it is possible to tackle these problems and promote human like control and safety.

**Objective:** This project aims to embed Level 1 motion planning within vehicles using various algorithms to predict best trajectories and paths of advancement by critically evaluating movements of other traffic agents. By designing Residual Neural Network (CNN) and Long short-term memory LSTM (RNN) form the base for surveyed driving scene perception, path planning and behavior algorithms. Compare and contrast evaluation metrics like learning curve, time efficiency and visual representations for these models.

**Data:** The data was collected by a fleet of 20 autonomous vehicles over a four-month period which consists of scenes, frames and agents. On top of this, the dataset consists of high-definition semantic map and aerial view over the area. The data is packed in .zarr files and divided into train, test, valid and sample datasets. The analysis of train, test and validate frame count over time are present in the ratio 83:7:10. The L5Kit package is used to load, rasterize, and visualize this data.

* Scenes: Driving episodes acquired from a given vehicle for capturing aerial and semantic views.
* Frames: Snapshots in time of the pose of the vehicle in the form of time frames.
* Agents: Four agent label probabilities captured by the vehicle's sensors.
* Aerial map: An aerial map used when rasterization is performed
* Semantic map : A high-definition semantic map used for rasterization

**Preprocessing: ­**

* Rasterizer takes images in constructor, maps them on world coordinates and transforms it into a series of vectors. Rasterization crops images around the agent of interest to generate a forward vector pointing right for the current timestep. Using 224x224 image size for rasterizing py-semantic map types.
* Agent Dataset function maps the agents to valid agent indices.
* Chunked Dataset is a dataset that lives on disk in compressed chunks, it has easy to use data loading and writing interfaces that involves making numpy-like slices.

**Algorithms:** We worked on our baseline model ResNet18 in the initial phase and later extended this work to ResNet50 and LSTM.

**1. ResNet:** Convolutional Neural Network (CNN) architecture with thousands of convolutional layers. The problem with CNN that repeated backward passes diminish the computed gradients resulting in performance degradation which is called called as the Vanishing Gradient problem. ResNet solution is that it uses skip connections that skip gradients through layers and reuse activation from previous layers. The ResNet50 model consists of 5 convolution layers dollowed by a dully connected layer. This model has 2048 input features and 100 output features.

* Batch size (Train, Valid, Test): 16, 32, 8
* Learning rate: 0.001
* Criterion: Mean Squared Error
* Optimizer: AdamW
* Number of steps: 10000

**2. LSTM:** Recurrent Neural Network (RNN) architecture with encoder and decoder elements. Encoder reads entire input sequence of input image vectors, processes it and captures contextual information. Retains a good summary of historic scene information through hidden and cell states. Decoder reads entire target sequence, offsets it by one timestamp along with last hidden and cell state of encoder. Also predicts coordinates of target vehicle for next timestamp. The model consisted of encoder and a decoder followed by a fully connected layer. The model has 128 input features and 100 output features.

* Batch size (Train, Valid, Test): 16, 32, 8
* Learning rate: 0.001
* Criterion: Mean Squared Error
* Optimizer: AdamW
* Number of steps: 10000
* Schedular: LR Schedular

**Distributed Environment:** The two approaches used are Data Parallelization and Model Parallelization that evaluate the processing time and learning curves to infer single and multiple GPU processing. To do distributed training, the model would just have to be wrapped using DistributedDataParallel and the training script would just have to be launched using dist.init\_process\_group(). The data was sampled and distributed using DistributedSampler. The node communication bandwidth is extremely important for multi-node distributed training. This is achieved using --local\_rank for argparse to specify the different parallel processes of the same load to be processed on 6 nodes for 10 epochs of train and validation. For each epoch, the data load was distributed along 6 nodes. Training for given batches across all the nodes were done simultaneously. The results from all the nodes were aggregated after training.

Results:

1. Learning loss

Loss curves for the three models were recorded for 10000 steps for above mentioned batch configurations.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Model | Split | Loss |
| 1. | ResNet 18 | Train | 10.172 |
| Validation | 12.817 |
| 2. | ResNet 50 | Train | 6.446 |
| Validation | 9.056 |
| 3. | LSTM | Train | 5.425 |
| Validation | 5.097 |

**2. Time efficiency**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Model | Split | Time/iteration | Total time |
| 1. | ResNet18 | Train | 1.06s | 14580s |
| Validation | 1.13s |
| 2. | ResNet50 | Train | 1.69s | 15120s |
| Validation | 1.91s |
| 3. | LSTM | Train | 1.45s | 13680s |
| Validation | 1.62s |

3. Visualizations: In the below result images, following elements infer:

* Autonomous vehicle represented by green rectangles
* Blue rectangles represent agents. Some agents like vehicles are spotted on the road. Taking a closer look, we can spot few blue dots which represents other vehicles, pedestrians, and bicycles.
* The semantic colored yellow lines represent paths along which we must predict the motions of external agents.
* The green dotted line depicts predicted agent’s motion

**A picture containing diagram

Description automatically generated**

**Diagram, engineering drawing

Description automatically generated**