EEL6938: Artificial Intelligence for Autonomous Systems

Final Exstension Trajectory Prediction Using DNNS

By Alexander Green

Trajectory Prediction Using DNNs

## Introduction

This project implements and evaluates multiple loss functions for multi-trajectory prediction using the model architecture from "TUTR: Trajectory Unified Transformer for Pedestrian Trajectory Prediction." The initial implementation focused on a Minimum Average Displacement Error (MinADE) loss function for basic multi-trajectory prediction. This extended work explores two additional loss functions: a diversity-promoting MinADE variant and a Gaussian Mixture Model (GMM) Negative Log-Likelihood approach.

Using the NuScenes dataset, the model predicts vehicle trajectories for 4 seconds (8 time indices) based on 2 seconds (5 time indices) of historical data. This report compares three distinct multi-trajectory prediction approaches:

1. **Standard MinADE loss:** Selects the best trajectory from multiple candidates
2. **Diversity-promoting MinADE loss:** Encourages variety in predicted trajectories
3. **Probabilistic GMM NLL loss:** Models predictions as a mixture of Gaussians

By implementing these advanced loss functions, we aim to improve both the accuracy and diversity of trajectory predictions, better capturing the inherent multimodal nature of future trajectories in complex environments.

## Windows Conversion

When starting this assignment, I decided to make the code compatible with Windows operating systems rather than setting up a compatible Linux environment on my computer. The primary change was adding the ***if \_\_name\_\_ == "\_\_main\_\_":*** conditional statement at the beginning of the main execution block and indenting all subsequent code. This modification prevents code execution during import while maintaining functionality when run as a script, addressing Python's different module handling behavior on Windows platforms.

*A screen shot of a computer program

AI-generated content may be incorrect.*

***Code 1:*** *Main Function Conversion for Windows*

## How to Use

The code can be run on any platform after installing the required dependencies. First, download and unzip the project folder. Open a terminal in the project directory and install the requirements with ***pip install -r requirements.txt***.

## To run the model with different loss functions, use the following commands:

## MinADE loss: ***python train\_eval.py --lossFunction minADE***

## MinADE div loss: ***python train\_eval.py --lossFunction minADEdiv***

## GMM NLL loss: ***python train\_eval.py --lossFunction GMM\_NLL***

## Additional parameters can be modified, such as the number of clusters (***--n\_clusters***), epochs (***--epoch***), or learning rate (***--lr***). For example: ***python train\_eval.py --lossFunction minADEdiv --lambda\_diversity 0.7 --n\_clusters 75 --epoch 100*** would run the model with the diversity-promoting loss function, diversity weight of 0.7, 75 clusters, and 100 training epochs.

Standard MinADE Loss

The baseline implementation uses Minimum Average Displacement Error (MinADE) which generates multiple candidate trajectories but only penalizes the one closest to the ground truth:



***Code 2:*** *MinADE Loss Calculations*

This approach computes the error between each predicted trajectory and the ground truth, then only penalizes the trajectory with the minimum error. This allows the model to generate multiple predictions without forcing all predictions to match a single ground truth.

## MinADE with Diversity Loss

The diversity-promoting variant also generates multiple trajectories but adds a term that penalizes similar trajectories, encouraging the model to generate diverse predictions:



***Code 3:*** *MinADE Diversity Loss Calculations*

This implementation calculates pairwise distances between all predicted trajectories and encourages them to be different from each other. The ***lambda\_diversity*** parameter controls the trade-off between accuracy and diversity.

## GMM Negative Log-Likelihood Loss

The GMM NLL loss treats multi-trajectory prediction as a probabilistic model with weighted mixture components:

  
***Code 4:*** *GNN NLL Loss Calculations*

This probabilistic approach models each of the multiple predictions as a component in a Gaussian mixture model, weighting them by predicted probabilities. It encourages accurate predictions with appropriate confidence levels, allowing the model to express uncertainty across its multiple predictions.

## Mathematical Formulation

Each loss function represents a different approach to the multi-trajectory prediction problem. This code implementation can be broken down mathematically for each of the following loss functions.

Where:

* ***B*** is the batch size
* ***K*** is the number of predicted trajectories (num\_modes)
* ***T*** is the sequence length
* & are the predicted coordinates at time ***t*** for batch element ***b*** and trajectory ***k***
* & are the ground truth coordinates at time ***t*** for batch element ***b***
* are the predicted mode probabilities
* is the fixed variance parameter

### MinADE Loss

1. Individual position error at each timestep:

*The Euclidean distance between predicted and ground truth coordinates at time* ***t***

1. Average Displacement Error (ADE) for trajectory k:

*Average error across all timesteps for batch element* ***b*** *and trajectory* ***k***

1. Minimum ADE across all trajectories:

*Best trajectory error for batch element* ***b***

1. Final Loss:

*Average of minimum errors across the batch*

### MinADE with Diversity Loss

*Adds a diversity term to the multi-trajectory approach*

### GMM NLL Loss

1. Individual position error at each timestep:

*The Euclidean distance between predicted and ground truth coordinates at time* ***t***

1. Joint likelihood over all timesteps for a trajectory:

*Product of Gaussian densities across all timesteps*

1. Weighted mixture likelihood using predicted mode probabilities:

*Weighted sum of likelihoods from each mixture component*

1. Final Loss:

*Negative mean log-likelihood across the batch*

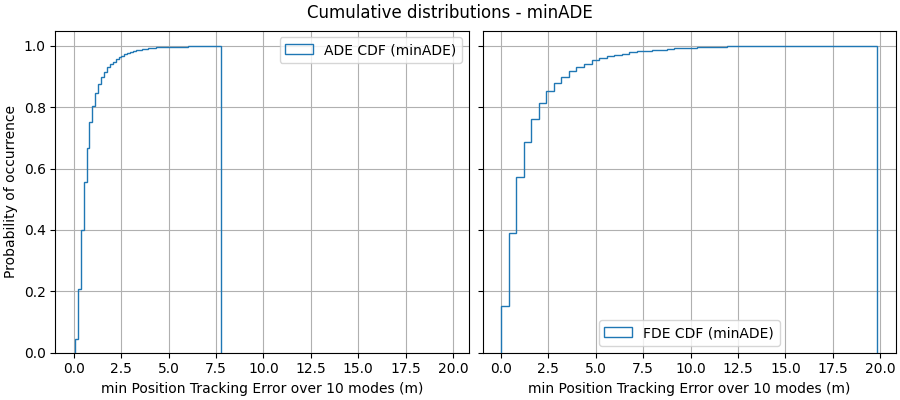
## Results

The results demonstrate significant performance differences between the three multi-trajectory prediction approaches.

|  |  |  |  |
| --- | --- | --- | --- |
|  | MinADE | MinADE w/ Div | GNN NLL |
| Best Average minADE | 0.8302 | 0.8596 | 0.5931 |
| Best Average minFDE | 1.6144 | 1.7145 | 0.9238 |
| Best median minADE | 0.6154 | 0.6088 | 0.4998 |
| Best median minFDE | 1.0359 | 1.0582 | 0.6180 |
| Best 10th percentile minADE | 0.2707 | 0.2657 | 0.2089 |
| Best 10th percentile minFDE | 0.3306 | 0.3272 | 0.1845 |
| Best 90th percentile minADE | 1.6025 | 1.7359 | 1.0640 |
| Best 90th percentile minFDE | 3.5806 | 3.9422 | 2.0265 |

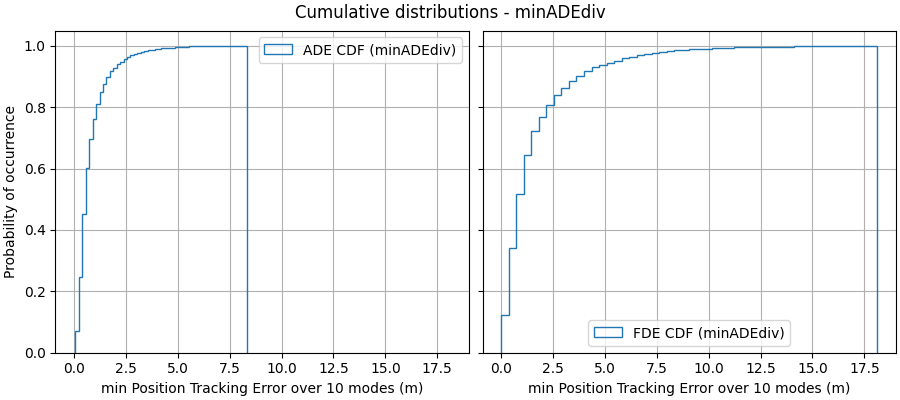
***Table 1:*** *Loss Function Results*

Standard MinADE Loss shows solid performance with an average minADE of 0.8302 and minFDE of 1.6144. While it generates multiple trajectories, it doesn't effectively encourage diversity or accurate probability estimation.



***Figure 1:*** *Standard MinADE Loss Cumulative Distribution*

MinADE with Diversity Loss performs somewhat better than standard MinADE in some metrics (particularly in the 10th percentile) but slightly worse in average performance with minADE of 0.8596 and minFDE of 1.7145. The diversity term helps capture different possible future paths but may sometimes sacrifice accuracy.



***Figure 2:*** *MinADE with Diversity Loss Cumulative Distribution*

GMM NLL Loss achieves the best performance across all metrics, with an average minADE of 0.5931 and minFDE of 0.9238. This probabilistic approach effectively models uncertainty distribution across multiple trajectories, resulting in a 29% improvement over the standard MinADE approach.

A graph of a number of numbers

AI-generated content may be incorrect.

***Figure 3:*** *GMM NLL Loss Cumulative Distribution*

The GMM NLL approach shows a 28.6% improvement in average minADE and a 42.8% improvement in average minFDE compared to the standard approach. The cumulative distribution plots further illustrate this improvement, showing that the probabilistic approach has much higher probabilities of achieving low error values.

## Conclusion

This study demonstrates the importance of appropriate loss function design for multi-trajectory prediction. All three tested approaches offer valuable insights into handling the inherent multimodality of trajectory prediction:

The standard MinADE loss provides a solid baseline for multi-trajectory prediction but lacks the sophistication to fully model trajectory uncertainties or ensure diversity among predictions.

The diversity-promoting MinADE loss shows improvements over the standard approach in capturing varied predictions, particularly in the best-case scenarios (10th percentile metrics), demonstrating the value of explicitly encouraging diversity.

The GMM NLL loss significantly outperforms other approaches by providing a principled probabilistic framework that effectively models the uncertainty distribution. Its superior performance across all metrics suggests that explicitly modeling prediction confidence through a mixture model better captures the multimodal nature of possible future trajectories.

These findings highlight how sophisticated loss functions can dramatically improve multi-trajectory prediction models for autonomous systems. Future work could explore combining the probabilistic framework of GMM with explicit diversity measures, or investigating dynamic variance parameters in the GMM model to further improve prediction accuracy and the coverage of possible futures.