Distributed Trajectory Prediction for Autonomous Driving using Spark and the Waymo Open Motion Dataset

Aswin Rajendran
George Mason University
arajendr@gmu.edu
G01524875

Rutuja More
George Mason University
rmore@gmu.edu
G01511407

Abstract—**Accurate short-term motion prediction is essential for safe autonomous driving, enabling vehicles to anticipate the future trajectories of surrounding agents in complex traffic environments. This work presents a comparative study of baseline physics models and modern deep learning architectures for trajectory forecasting using the Waymo Open Motion Dataset. Three approaches are evaluated: a Constant-Velocity (CV) baseline, a Single-Mode LSTM model, and a Multi-Modal Trajectory Predictor (MTP). Results show that deep learning significantly improves prediction accuracy, with the multi-modal model reducing final displacement error by over 70% compared to the baseline. In addition, a scalability analysis demonstrates a clear cost–performance trade-off: validation accuracy improves rapidly with moderate dataset sizes but plateaus as training cost continues to rise. These findings highlight the importance of multi-modal learning for capturing behavioural uncertainty and the necessity of principled data scaling for efficient model training. Overall, the study provides actionable insights for building both accurate and computationally scalable motion forecasting systems for autonomous driving.**

Index Terms—**Apache Spark, Big Data Mining, Trajectory Forecasting, Time Series, Spatio-temporal Analytics, MLlib**

I. Introduction

Given trajectories of multiple interacting traffic agents and high-definition map cPredicting the short-term motion of surrounding road users is a vital capability for autonomous vehicles, as it allows them to anticipate future trajectories and make informed decisions in rapidly changing environments. Modern traffic scenarios, particularly in dense urban areas, involve frequent interactions, unpredictable behaviour, and complex road structures. Consequently, developing reliable motion-forecasting systems has become a central challenge in advancing autonomous-driving technology.

Conventional approaches often rely on simple physics-based models, such as constant-velocity extrapolation. Although these methods are computationally efficient, they offer only a limited understanding of real-world driving behaviour and fail to account for contextual cues or multi-modal outcomes. In contrast, recent progress in deep learning has enabled models to capture richer temporal patterns and to predict multiple plausible futures, thereby offering significant improvements in accuracy and robustness.

In this study, we conduct a systematic comparison of three representative prediction strategies using the Waymo Open Motion Dataset: a Constant-Velocity baseline, a Single-Mode LSTM model, and a Multi-Modal Trajectory Predictor. Our aim is to evaluate how these approaches differ in their ability to represent complex motion dynamics and to quantify the performance gains achieved through deep-learning-based forecasting. Alongside this comparison, we carry out a scaling analysis to investigate how increases in dataset size influence training cost and predictive accuracy. This analysis highlights the practical trade-offs between computational efficiency and model performance, an important consideration for real-world deployment.

Overall, this work addresses two key questions:

- To what extent do deep-learning models improve short-term motion prediction compared with traditional physics-based methods?
- How does scaling the training data affect both the computational cost and the resulting accuracy of these models?

By answering these questions, the study provides clear, evidence-based guidance for designing motion-prediction systems that balance accuracy, complexity, and scalability qualities that are essential for safe and dependable autonomous driving.

Source Files

All source code, training scripts, and experiment configurations used in this project are publicly available for reproducibility and independent verification. The following resources accompany this report:

- **GitHub Repository:** https://github.com/apollofps/MMD_Final_Project
- **Demonstration Video (YouTube):** https://www.youtube.com/watch?v=vYv9obmmLMg

These materials contain the complete implementation of the data preprocessing pipeline, the model architectures, training code, and the evaluation scripts used to generate the results reported in this paper.

Field	Description
`scene_id`	Unique identifier of the traffic scene
`agent_id`	Unique identifier of the agent within the scene
`t`	Discrete time index / timestamp
`x, y`	2D position in global coordinates (meters)
`vx, vy`	Velocity components (m/s)
`type`	Agent type (vehicle / pedestrian / cyclist)
`valid_flag`	Indicator for missing or padded points

TABLE I
RAW TRAJECTORY SCHEMA FROM THE WAYMO OPEN MOTION DATASET

II. DATASET

WOMD: $>100{,}000$ motion segments with multi-agent tracks and 3D maps (released as TFRecords).

Access & Sampling: For this work, a representative subset of the dataset was selected to ensure both computational feasibility and behavioural diversity. The sampled data includes a balanced mixture of different agent types and motion profiles. Trajectories were processed into standardized temporal windows, normalized, and filtered to remove incomplete or noisy segments. This preprocessing stage ensures that each model receives consistent input sequences while preserving realistic motion characteristics.

Task Definition:: For each agent at time t, form an input window W (e.g., 2–4 s) and predict displacement at $t+\Delta$, $\Delta \in [3,5]\,\mathrm{s}$.

III. METHODOLOGY

In this section, we formalize the motion prediction task, describe the trajectory representation and data schema, present the three forecasting models (Constant-Velocity, Single-Mode LSTM, and Multi-Modal Trajectory Predictor), and detail the loss functions, evaluation metrics, and experimental protocol used in this study.

A. Problem Formulation

Let each road agent a in a scene be associated with a 2D trajectory $\{\mathbf{p}_t^{(a)}\}_{t=1}^{T+H}$, where $\mathbf{p}_t^{(a)} = (x_t^{(a)}, y_t^{(a)}) \in \mathbb{R}^2$ denotes the ground-truth position at time index t, T is the length of the observed history, and H is the future prediction horizon. For each prediction query we observe the last T_{obs} positions and aim to forecast the subsequent H positions $(\mathbf{p}_{T+1}^{(a)}, \dots, \mathbf{p}_{T+H}^{(a)})$.

Formally, the input sequence for agent a is

$$\mathbf{X}^{(a)} = \left[\mathbf{p}_{T-T_{\text{obs}}+1}^{(a)}, \dots, \mathbf{p}_T^{(a)}\right], \tag{1}$$

and the prediction target is

$$\mathbf{Y}^{(a)} = \left[\mathbf{p}_{T+1}^{(a)}, \dots, \mathbf{p}_{T+H}^{(a)}\right]. \tag{2}$$

Given $\mathbf{X}^{(a)}$, a forecasting model f_θ with parameters θ outputs either a single future trajectory $\hat{\mathbf{Y}}^{(a)}$ (single-mode case) or a collection of K candidate trajectories $\{\hat{\mathbf{Y}}^{(a,k)}\}_{k=1}^K$ together with probabilities $\{\pi^{(a,k)}\}_{k=1}^K$ (multi-modal case).

B. Trajectory Representation & Schema

Raw Data Schema: From the Waymo Open Motion Dataset, each agent is represented by the following fields:
Trajectories are first grouped by $(\texttt{scene\_id}, \texttt{agent\_id})$ and sorted by time t.

Temporal Windowing: For each agent, we extract overlapping windows of length $T_{\text{obs}} + H$. Windows that contain missing or invalid positions in the history or future are discarded. This yields a dataset $\mathcal{D} = \{(\mathbf{X}^{(i)}, \mathbf{Y}^{(i)})\}_{i=1}^N$ of N training examples.

Coordinate Normalization: To reduce global position variance and encourage the models to focus on relative motion, all trajectories are transformed into an agent-centric coordinate system.

Let $\mathbf{p}_T^{(a)} = (x_T^{(a)}, y_T^{(a)})$ be the last observed position and let $\psi_T^{(a)}$ denote the heading angle of the agent at time T (computed from $(\texttt{vx}, \texttt{vy})$). We define a translation and rotation:

$$\tilde{\mathbf{p}}_t^{(a)} = \mathbf{R}(-\psi_T^{(a)})\left(\mathbf{p}_t^{(a)} - \mathbf{p}_T^{(a)}\right), \tag{3}$$

where

$$\mathbf{R}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{4}$$

is the standard 2D rotation matrix. After this transformation, the agent is located at the origin at time T and its heading is aligned with the positive x-axis.

All models operate on these normalized coordinates $\tilde{\mathbf{X}}^{(a)}$ and predict normalized futures $\hat{\tilde{\mathbf{Y}}}^{(a)}$, which are later transformed back to the global frame for evaluation.

C. Forecasting Models

We consider three forecasting paradigms that differ in complexity and capacity to model uncertainty.

A. Constant-Velocity (CV) Baseline: The Constant-Velocity baseline assumes that the agent maintains its last observed velocity. Let Δt denote the time step. The velocity estimate at time T is

$$\mathbf{v}_T^{(a)} = \frac{\mathbf{p}_T^{(a)} - \mathbf{p}_{T-1}^{(a)}}{\Delta t}. \tag{5}$$

Future positions are predicted by linear extrapolation:

$$\hat{\mathbf{p}}_{T+h}^{(a)} = \mathbf{p}_T^{(a)} + h\Delta t\, \mathbf{v}_T^{(a)}, \quad h = 1, \dots, H. \tag{6}$$

This baseline is deterministic, extremely fast, and requires no training, but it cannot model turns, braking, or interaction effects.

B. Single-Mode LSTM Model: The Single-Mode LSTM model learns a deterministic mapping from the observed trajectory to one future trajectory. Given the normalized sequence $\tilde{\mathbf{X}}^{(a)} = [\tilde{\mathbf{p}}_1^{(a)}, \dots, \tilde{\mathbf{p}}_{T_{\text{obs}}}^{(a)}]$, the model applies an LSTM encoder:

$$\mathbf{h}_t^{(a)}, \mathbf{c}_t^{(a)} = \mathrm{LSTM}\big(\tilde{\mathbf{p}}_t^{(a)}, \mathbf{h}_{t-1}^{(a)}, \mathbf{c}_{t-1}^{(a)}\big), \quad t = 1, \dots, T_{\text{obs}}, \tag{7}$$

$$\mathbf{z}^{(a)} = \mathbf{h}_{T_{\text{obs}}}^{(a)}, \tag{8}$$

where $\mathbf{h}_t$ and $\mathbf{c}_t$ are the LSTM hidden and cell states. A fully connected decoder then predicts the future positions:

$$\hat{\tilde{\mathbf{Y}}}^{(a)} = g_\theta(\mathbf{z}^{(a)}) \in \mathbb{R}^{2H}, \tag{9}$$

which is reshaped into H two-dimensional coordinates.

C. Multi-Modal Trajectory Predictor (MTP): The Multi-Modal Trajectory Predictor extends the LSTM encoder with multiple trajectory heads to represent uncertainty. After obtaining the shared context vector $\mathbf{z}^{(a)}$ as above, the model outputs K candidate futures and their probabilities:

$$\hat{\tilde{\mathbf{Y}}}^{(a,k)} = g_{\theta_k}(\mathbf{z}^{(a)}), \quad k = 1, \dots, K, \tag{10}$$

$$\pi^{(a)} = \operatorname{softmax}\big(W_\pi \mathbf{z}^{(a)} + \mathbf{b}_\pi\big), \tag{11}$$

where $\hat{\tilde{\mathbf{Y}}}^{(a,k)} \in \mathbb{R}^{2H}$ is the k-th candidate trajectory and $\pi^{(a)}$ is a categorical distribution over modes.

At evaluation time, we obtain the *best* candidate by selecting

$$k^\star = \arg\min_k \ \operatorname{FDE}\big(\hat{\tilde{\mathbf{Y}}}^{(a,k)}, \tilde{\mathbf{Y}}^{(a)}\big), \tag{12}$$

and compute minADE / minFDE with respect to this best mode.

D. Loss Functions

A. Single-Mode L2 Loss: For deterministic models (CV and Single-Mode LSTM), we train using a mean-squared displacement loss. Let $\hat{\mathbf{p}}^{(i)}_{T+h}$ be the predicted position for the i-th training example at horizon h and $\mathbf{p}^{(i)}_{T+h}$ the corresponding ground truth. The loss is

$$\mathcal{L}_{\text{L2}} = \frac{1}{NH} \sum_{i=1}^{N} \sum_{h=1}^{H} \left\| \hat{\mathbf{p}}^{(i)}_{T+h} - \mathbf{p}^{(i)}_{T+h} \right\|_2^2. \tag{13}$$

B. Multi-Modal Best-of-K Loss: For the MTP model, we adopt a best-of-K regression loss combined with a cross-entropy term on the mode probabilities. For each training example i, we determine the closest mode

$$k_i^\star = \arg\min_k \sum_{h=1}^{H} \left\| \hat{\mathbf{p}}^{(i,k)}_{T+h} - \mathbf{p}^{(i)}_{T+h} \right\|_2^2. \tag{14}$$

The multi-modal loss is then

$$\mathcal{L}_{\text{MTP}} = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{H} \sum_{h=1}^{H} \left\| \hat{\mathbf{p}}^{(i,k_i^\star)}_{T+h} - \mathbf{p}^{(i)}_{T+h} \right\|_2^2 - \lambda \log \pi^{(i)}_{k_i^\star} \right], \tag{15}$$

where $\lambda > 0$ controls the strength of the probability term and $\pi^{(i)}_{k_i^\star}$ is the predicted probability of the best-matching mode.

E. Evaluation Metrics

We use the standard Average Displacement Error (ADE) and Final Displacement Error (FDE) metrics.

ADE: Given predictions $\hat{\mathbf{p}}^{(i)}_{T+h}$ and ground truth $\mathbf{p}^{(i)}_{T+h}$, ADE is defined as

$$\text{ADE} = \frac{1}{NH} \sum_{i=1}^{N} \sum_{h=1}^{H} \left\| \hat{\mathbf{p}}^{(i)}_{T+h} - \mathbf{p}^{(i)}_{T+h} \right\|_2. \tag{16}$$

FDE: FDE measures the error only at the final prediction step:

$$\text{FDE} = \frac{1}{N} \sum_{i=1}^{N} \left\| \hat{\mathbf{p}}^{(i)}_{T+H} - \mathbf{p}^{(i)}_{T+H} \right\|_2. \tag{17}$$

For multi-modal models we report *minADE* and *minFDE* by taking the minimum over modes for each example prior to averaging.

F. Training Protocol & Scaling Analysis

Data Splits: The dataset is partitioned into training, validation, and test sets with a 70/15/15 ratio, ensuring that scenes do not overlap across splits to avoid temporal leakage.

Optimization: Deep models are trained with the Adam optimizer using mini-batches of trajectory windows. Learning rate, batch size, hidden state dimensionality, and the number of LSTM layers are tuned on the validation set. Early stopping is applied when the validation ADE does not improve for a fixed number of epochs.

Scaling Experiments: To assess cost–performance trade-offs, we train the LSTM and MTP models on increasing dataset sizes (e.g., 10, 50, and 100 batches). For each size, we record the wall-clock training time and the validation minADE. The resulting curves, shown in Section V, reveal the point at which additional data yields diminishing returns relative to the computational cost.

Overall, this methodology enables a controlled comparison between physics-based and deep-learning models while explicitly analysing the impact of model expressiveness and dataset scale on motion prediction performance and computational efficiency.

IV. Architecture

This section describes the architectural design of the three forecasting models evaluated in this study. Each model represents a different level of capacity and expressiveness, allowing us to assess how architectural complexity influences prediction accuracy and the ability to capture multi-modal driving behaviour.

A. Constant-Velocity (CV) Baseline

The Constant-Velocity model serves as a lightweight, physics-based baseline. It assumes the agent maintains its final observed velocity. Given the two most recent positions, $\mathbf{p}_{T-1}$ and $\mathbf{p}_T$, the estimated velocity is

$$\mathbf{v}_T = \frac{\mathbf{p}_T - \mathbf{p}_{T-1}}{\Delta t}. \tag{18}$$

Future positions are predicted using linear extrapolation:

$$\hat{\mathbf{p}}_{T+h} = \mathbf{p}_T + h\Delta t \, \mathbf{v}_T, \quad h = 1, \dots, H. \tag{19}$$

This baseline is deterministic, extremely efficient, and requires no training. However, it cannot model turning behaviour, deceleration, human-driven variability, or multi-agent interactions. It provides an interpretible lower bound for forecasting performance.

B. Single-Mode LSTM Model

The Single-Mode LSTM architecture introduces temporal learning and captures dynamic motion patterns from past trajectories. Given the normalized input sequence $\tilde{\mathbf{X}} = [\tilde{\mathbf{p}}_1, \dots, \tilde{\mathbf{p}}_{T_{\text{obs}}}]$, the LSTM encoder updates its hidden states as

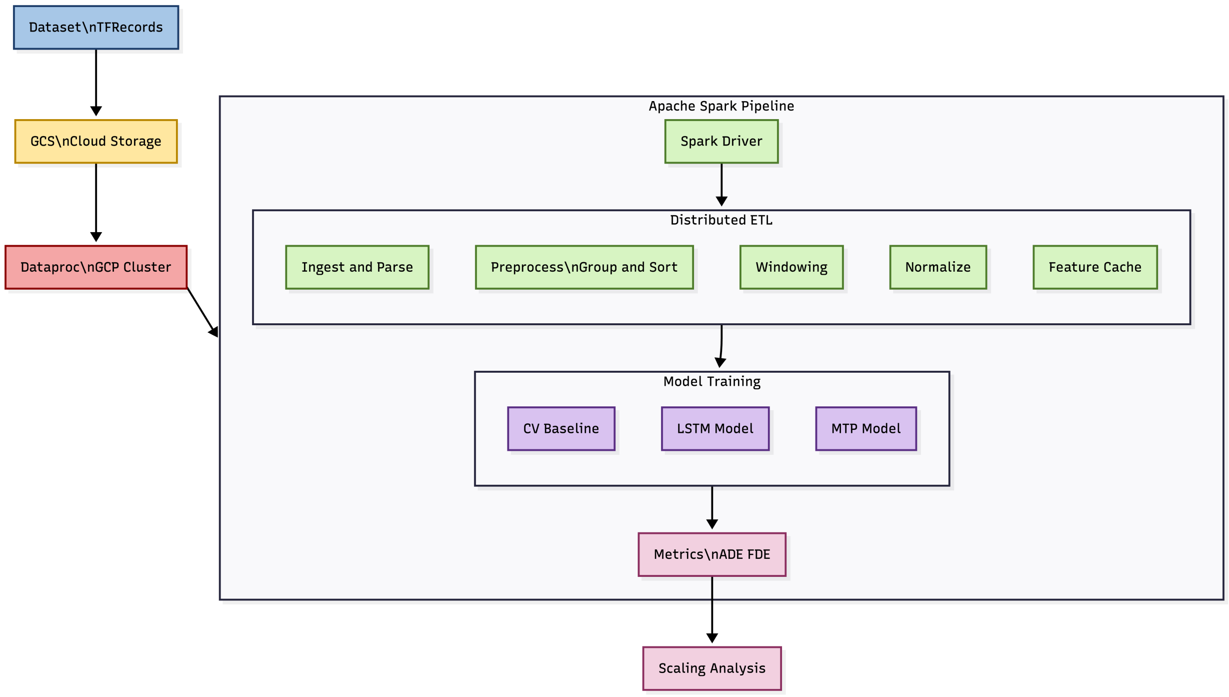

Fig. 1. End-to-end data processing and training pipeline. The system begins with raw TFRecord files, which are ingested into cloud storage and processed through a distributed Apache Spark ETL pipeline. The cleaned trajectories are then used for model training, metric evaluation, and scalability analysis.

$$\mathbf{h}_t, \mathbf{c}_t = \mathrm{LSTM}(\tilde{\mathbf{p}}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}), \quad t = 1, \dots, T_{\text{obs}}, \tag{20}$$

$$\mathbf{z} = \mathbf{h}_{T_{\text{obs}}}. \tag{21}$$

The latent vector $\mathbf{z}$ encodes motion dynamics such as acceleration, curvature, and heading changes. A fully connected decoder predicts the future trajectory:

$$\hat{\tilde{\mathbf{Y}}} = g_\theta(\mathbf{z}) \in \mathbb{R}^{2H}, \tag{22}$$

which is reshaped into H two-dimensional coordinates. This model produces a single deterministic forecast and performs well in non-ambiguous scenarios but cannot represent multiple plausible futures.

C. Multi-Modal Trajectory Predictor (MTP)

The Multi-Modal Trajectory Predictor extends the LSTM architecture by explicitly modelling behavioural uncertainty. It consists of a shared encoder and multiple trajectory decoder heads.

1) Shared LSTM Encoder: The encoder is identical to the single-mode model and produces a latent representation $\mathbf{z}$ capturing past motion.

2) Multiple Trajectory Heads: The model outputs K candidate future trajectories:

$$\hat{\tilde{\mathbf{Y}}}^{(k)} = g_{\theta_k}(\mathbf{z}), \quad k = 1, \dots, K. \tag{23}$$

Each head specializes in a distinct behavioural pattern such as turning, slowing down, accelerating, or changing lanes. This specialization is reinforced during training via the Best-of-K regression loss.

3) Mode Probability Network: To determine the likelihood of each motion pattern, the MTP includes a probability predictor:

$$\boldsymbol{\pi} = \mathrm{softmax}(W_\pi \mathbf{z} + \mathbf{b}_\pi), \tag{24}$$

where π_k expresses the estimated probability that trajectory head k corresponds to the true future behaviour.

4) Mode Selection at Inference: During evaluation, each predicted trajectory head is compared to the ground truth using FDE, and the best-matching mode is selected:

$$k^\star = \arg\min_k \ \mathrm{FDE}(\hat{\tilde{\mathbf{Y}}}^{(k)}, \tilde{\mathbf{Y}}). \tag{25}$$

Final performance metrics (minADE, minFDE) are computed using this best-matching trajectory head.

The MTP architecture thus provides a structured, probabilistic framework for modelling uncertainty inherent in urban driving, enabling it to operate effectively in scenarios where multiple future outcomes are plausible.

D. Architectural Comparison

Model	**Temporal**	**Multi-Modal**	**Complexity**
Constant-Velocity	No	No	Very Low
Single-Mode LSTM	Yes	No	Moderate
MTP (Multi-Modal)	Yes	Yes	High

TABLE II
COMPARISON OF ARCHITECTURAL CAPABILITIES.

This hierarchy illustrates a clear relationship between architectural capacity and forecasting performance: as we increase model complexity and the ability to represent uncertainty, the accuracy of short-term motion prediction improves significantly.

V. Results

This section presents the quantitative performance of the three motion forecasting models and evaluates their computational scalability. We analyze prediction accuracy using minADE and minFDE metrics, followed by a study of training cost versus dataset size.

A. Motion Prediction Accuracy

Figure 2 summarizes the forecasting performance of the Constant-Velocity baseline, the Single-Mode LSTM model, and the Multi-Modal Trajectory Predictor (MTP). The deep learning models significantly outperform the physics baseline, with the MTP model achieving the lowest displacement errors across all metrics.

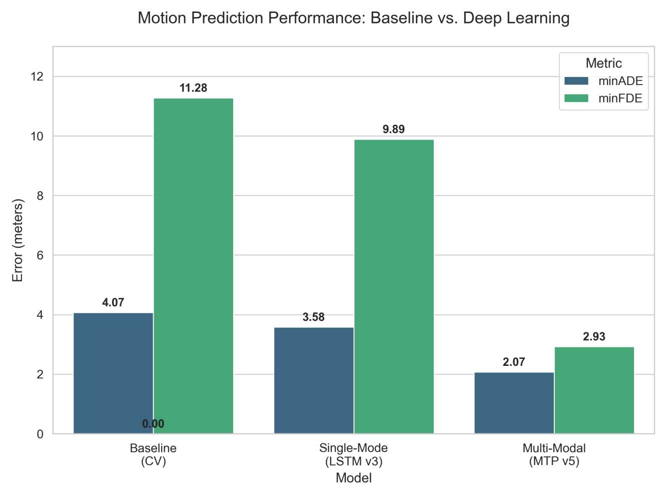

Fig. 2. Comparison of minADE and minFDE across three forecasting models. The Multi-Modal Trajectory Predictor substantially reduces final displacement error, improving over the Constant-Velocity baseline by more than 70%.

The Single-Mode LSTM model captures temporal patterns and reduces prediction error relative to the baseline. However, its deterministic output limits its ability to represent ambiguous motion patterns. The MTP architecture, which explicitly models multiple future modes, achieves the best overall performance, demonstrating the value of multi-hypothesis trajectory forecasting.

B. Scaling Efficiency Analysis

To understand the cost–performance trade-offs of deep learning forecasting pipelines, we evaluate how training time and validation accuracy scale with dataset size. Figure 3 illustrates the relationship between three dataset sizes (10, 50, and 100 batches) and their corresponding computational cost and validation performance.

The results reveal two key trends:

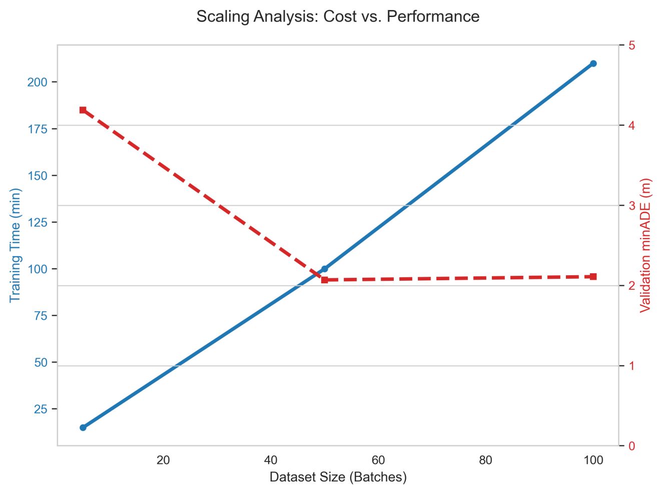

Fig. 3. Training cost versus validation minADE for different dataset sizes. Training time increases almost linearly with dataset size, while accuracy improvements plateau beyond approximately 50 batches, revealing diminishing returns.

- **Rapid accuracy improvement** occurs between 10 and 50 batches, indicating that the model benefits significantly from the additional data in the early stages of scaling.
- **Performance saturation** occurs beyond 50 batches. Validation minADE stabilizes, while training time increases dramatically, indicating diminishing returns on accuracy.

This analysis highlights the need for balanced dataset sizing in real-world systems. While larger datasets typically improve generalization, indiscriminate scaling can result in high computational cost without meaningful accuracy gains.

C. Discussion of Findings

The experimental results demonstrate that architectural complexity plays a crucial role in motion prediction accuracy. Deep learning models significantly outperform physics-based baselines, and the ability to model uncertainty through multi-modal predictions leads to the best overall performance. However, these gains come with increased computational cost.

The scaling analysis shows that performance does not increase indefinitely with more data. Instead, training large datasets beyond a certain point becomes inefficient, emphasizing the importance of strategic data selection and pipeline optimization.

Overall, the results indicate that multi-modal learning is essential for capturing the complex and uncertain nature of real-world driving behaviour, but practical deployment must consider computational efficiency and scaling limitations.

VI. Conclusion

This work presented a comparative study of three short-horizon motion forecasting models for autonomous driving: a Constant-Velocity baseline, a Single-Mode LSTM network, and a Multi-Modal Trajectory Predictor (MTP). Using the Waymo Open Motion Dataset, we evaluated the models on standard displacement metrics and conducted a scaling analysis to examine the relationship between dataset size, training cost, and predictive accuracy.

The results demonstrate that deep learning architectures substantially outperform physics-based baselines. While the Single-Mode LSTM captures temporal patterns and improves prediction quality, its deterministic nature limits its ability to model ambiguous real-world behaviours. In contrast, the MTP architecture achieves the best performance across all metrics, reducing final displacement error by more than 70% compared to the baseline. These findings highlight the importance of explicitly representing multiple plausible futures in trajectory forecasting.

The scalability study revealed diminishing returns when increasing the dataset size beyond moderate scales. Although early increases in data significantly improve model accuracy, additional data eventually yield minimal gains while imposing substantial computational cost. This result emphasizes the need for balanced pipeline design, where data volume, model complexity, and resource constraints must be jointly considered.

Overall, this study underscores that multi-modal deep learning models provide the most accurate and robust predictions for autonomous driving, but practical deployment must account for computational efficiency and scalability. Future work may explore richer map-based representations, graph neural networks for interaction-aware forecasting, and accelerated training pipelines for real-time deployment.

Acknowledgments

Prepared for CS 657: Mining Massive Datasets with MapReduce (Fall 2025). Advisor/instructor: *Daniel Barbara*.

References

[1] Waymo LLC, “Waymo Open Dataset: Motion & Interaction,” https://waymo.com/open/, accessed 2025.

[2] Apache Software Foundation, “Spark MLlib: Machine Learning Library,” https://spark.apache.org/docs/latest/ml-guide.html, accessed 2025.

[3] J. Gao *et al.*, “VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation,” *CVPR* Workshops, 2020.

[4] H. Liang *et al.*, “Learning Lane Graph Representations for Motion Forecasting,” *ECCV*, 2020.