ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

ENHANCING BIG DATA ANALYSIS IN TWO-WHEELED VEHICLES THROUGH MACHINE LEARNING TECHNIQUES

Presentata da: Federico Pennino

Coordinatore Dottorato

Paola Salomoni

Supervisore

Maurizio Gabbrielli

Co-supervisore

David Attisano

Esame finale anno 2026

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria

Enhancing Big Data Analysis in Two-Wheeled Vehicles through Machine Learning Techniques

Dottorato di Ricerca in Computer Science and Engineering
Ciclo XXXVIII

Relatore
Prof. Maurizio Gabbrielli

Candidato
Federico Pennino

Correlatori
Dott. David Attisano
Dott. Davide Sette

Abstract

Modern motorcycles generate vast, high-dimensional data streams that hold immense potential but whose raw, complex, and largely unlabeled nature presents a significant barrier to extracting meaningful intelligence. This thesis introduces a suite of self-supervised learning techniques to systematically transform this data into structured and interpretable knowledge.

The research begins by addressing rider identification, employing a triplet loss framework on weakly supervised data to create discriminative "behavioral fingerprints" and impose a first layer of semantic structure. Building on this, a contextual contrastive learning framework is developed to understand the dynamic nature of a ride by modeling the temporal relationships between adjacent segments, thereby capturing sequential riding patterns. The analysis then models the direct interplay between rider and vehicle using a novel dual-encoder architecture that learns a shared latent space for rider inputs and vehicle states, explicitly capturing their relationship.

To address the practical challenge of efficient data retrieval at scale, the thesis introduces Trajectory-Embedded Matryoshka Representation Learning, which creates nested, multi-scale embeddings to accelerate similarity searches without compromising performance. Finally, the work challenges the "more data is better" paradigm with a data-centric optimization framework. By leveraging a foundation model to curate smaller, higher-quality "training diets," this approach demonstrates superior performance on a downstream time-series forecasting task.

Collectively, these contributions demonstrate a methodological progression — from structuring raw signals and learning complex behaviors to optimizing data retrieval and engineering the training set itself — providing a comprehensive framework for turning unstructured sensor data from two-wheeled vehicles into actionable intelligence.

To my whole family.

Contents

Chapter 1

Introduction

Modern vehicles have evolved from primarily mechanical systems into sophisticated electronic platforms that generate and process vast amounts of data. The on-board sensor suite of a contemporary motorcycle extends far beyond classical telematics [Fas24, Big18]. It typically includes both sensors that measure data from the vehicle and virtual sensors that calculate values not easily acquired through hardware. These sensors generate a continuous, multi-modal data stream that intrinsically represents the rider, the machine, and the road [RM22]. This data-centric trend is a driver of the rapidly expanding "Connected Motorcycle", a sector projected to experience exponential growth, with market size estimates expanding from around $100 million in the early 2020s to exceed $1 billion by 2030, potentially [Mor25, AIn25]. This economic momentum underscores a fundamental shift in the industry's value proposition.

While the mechanical part and the electronic factors remain crucial, a new asset has emerged as a primary driver of competitive advantage: the vast amount of data generated, and the intelligence that can be extracted from it [MAEP15, KTZ19]. The ability to collect, process, and analyze these massive data streams [ABSP23] — often aggregated into vast, unstructured "data lakes" [NM22, AAO24] — enables a new frontier of applications, from real-time safety interventions to hyper-personalized rider experiences. The value of a motorcycle is increasingly found in the data it generates and the insights it can provide, rather than solely in its physical engineering [BBB+23a].

This thesis addresses the problem of extracting intelligence and identifying patterns from these data streams without relying on prior knowledge or existing biases, leveraging self-supervised learning techniques [BIS+23] that can automatically discover meaningful representations and relationships within the data itself.

The torrent of data generated from instrumented motorcycles is not inherently practical [HHLR15]. In its raw form, this data constitutes a high-dimensional, unstructured labyrinth of signals that requires the knowledge of a domain expert to be analyzed [HZL+21]. The vast majority of data collected from real-world riding is not annotated with the high-level semantic information required for traditional supervised machine learning [ERC+23]. The core of this thesis's research problem is to develop a suite of machine learning techniques that can systematically transform the vast, complex, and largely unlabeled sensor data generated by modern motorcycles into structured and interpretable knowledge.

The thesis is organized as follows: Chapter 2 provides a comprehensive overview of the fundamental concepts leveraged throughout this work, including the Transformer architecture, machine learning paradigms, and contrastive learning frameworks. Subsequently, Chapter 3 reviews related works in driving style analysis, the application of contrastive learning to time series, and the principles of data-centric AI, contextualizing the contributions that follow. Chapter 4 describes DMLogger, a cost-effective proof-of-concept for a remote data acquisition system developed to address the critical bottleneck of collecting large-scale, diverse real-world operational data from motorcycles. This system functions as a prototype connected ECU, reliably capturing and transmitting high-frequency CAN bus data during regular vehicle operation.

In Chapter 5, we present an approach that frames the problem of driver identification within a weakly supervised context. We address the issue of insufficient annotations by utilizing triplet loss [SKP15] to construct triplets from the coarse-grained information extracted from the raw data. By learning to distinguish between riding segments from the same day versus different days, the model successfully extracts discriminative "behavioral fingerprints" for individual riders, imposing a first and crucial layer of semantic structure onto the chaotic raw data.

In Chapter 6, we moved beyond the static identification of a rider. In this chapter, we investigate the dynamic nature of their behavior over time by asking,

"How are they driving?". This goal is accomplished through the introduction of a contextual contrastive learning framework, inspired by [ERC+21]. This approach models the temporal flow and causal relationships between adjacent segments of a ride, creating rich representations that encode sequential patterns of acceleration, braking, and cornering. This contribution transforms the analysis from static snapshots to a dynamic understanding of the ride itself.

In the same direction but with a different perspective, Chapter 7 shifts the focus to modeling the direct, real-time interplay between the rider and the motorcycle. A novel dual-encoder framework, architecturally inspired by Contrastive Language-Image Pre-training (CLIP) [RKH+21], is proposed. This framework learns a shared latent space that jointly embeds rider inputs (e.g., throttle, gear changes) and the resulting vehicle states (e.g., speed, lean angle). By explicitly modeling this cause-and-effect relationship, the framework provides a powerful tool for mission profiling and comprehensive characterization of driving behavior.

While this approach effectively models the interplay between rider and machine, using these fixed-size embeddings for similarity searches can lead to significant lookup times, especially as the dataset grows. In Chapter 8, we shift the focus to accelerating this retrieval process by introducing Trajectory-Embedded Matryoshka Representation Learning. Using trajectory data from Porto Dataset [MMJ13], we incorporate in [JPR+23] a Matryoshka loss [KBR+22], which creates nested vector representations that maintain strong semantic capabilities even when truncated. This structure enables a two-stage retrieval pipeline. In the first stage, a smaller, lower-dimensional version of the embedding is used to quickly narrow down the search space to a subset of candidates. In the second stage, the full-dimensional representation is used to find the most similar match within this much smaller subset, drastically reducing the overall lookup time without compromising performance.

In conclusion, the final contribution (Chapter 9) challenges the "more data is better" paradigm. This work — drawing inspiration from [DYF+25] — introduces a data-centric optimization framework that intelligently curates smaller, yet more effective, training datasets. By leveraging a large foundation model (MOMENT-1 [GSC+24]) to partition the data into behaviorally consistent clusters and then treating the data composition as a hyperparameter to be optimized, this framework demonstrates that a smaller, higher-quality "training diet" can achieve superior

performance and generalization on a downstream time-series forecasting task. This contribution offers a powerful and practical validation of the data-centric AI philosophy.

Collectively, these contributions demonstrate a clear path from grappling with raw, unlabeled signals to extracting high-level knowledge and finally to engineering the data itself for optimal model performance, providing a robust and novel framework for enhancing big data analysis in two-wheeled vehicles.

Chapter 2

Background

This chapter lays the theoretical groundwork for the research in this thesis by reviewing the core machine learning concepts and architectures that will be utilized in the subsequent chapters. The discussion begins in Section 2.1 with a detailed overview of the Transformer model, covering its network architecture, attention mechanism, and specific adaptations for time-series analysis. Following this, Section 2.2 outlines the fundamental learning paradigms in machine learning, focusing on supervised, weakly supervised, and self-supervised approaches. The chapter then explores time-series embedding in Section 2.3, a critical technique for converting raw temporal data into meaningful vector representations. Finally, Section 2.4 examines the framework of contrastive learning, including methodologies such as Triplet Loss and InfoNCE, and their applications in models like CLIP.

2.1 Transformer

The Transformer model [VSP+17] has gained prominence in machine learning due to its ability to efficiently handle sequential data. Unlike traditional models such as recurrent neural networks (RNNs) [Jor86, RHW86] or long short-term memory networks (LSTMs) [HS97], Transformers rely on a self-attention mechanism that captures dependencies across the entire sequence simultaneously. This mechanism enables the model to assess the importance of various elements in the sequence, making it particularly effective for tasks that involve long-range relationships.

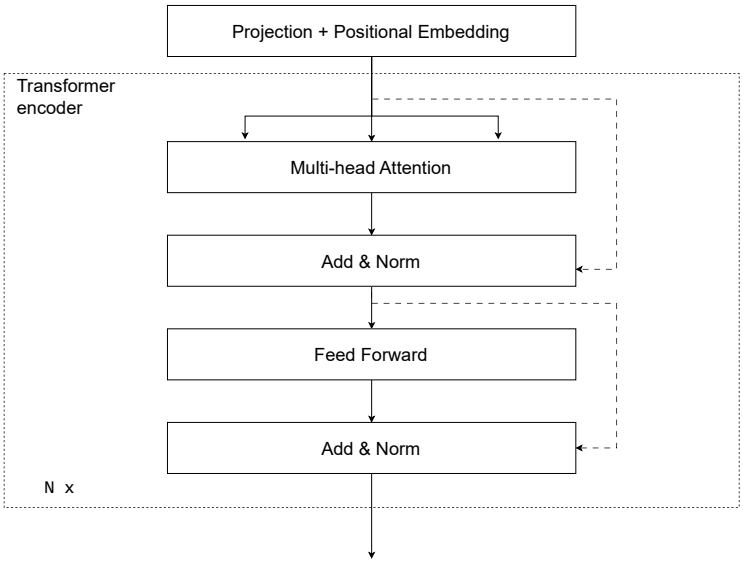


Figure 2.1: Diagram of the Transformer Encoder Architecture: Illustration of the key components including multi-head attention, feed-forward layers, residual connections with normalization, and positional embedding. The structure is repeated N times to capture complex dependencies in sequential data.

The architecture - illustrated in Figure 2.1 - comprises multiple encoder layers, each containing multi-head self-attention and feed-forward sublayers, augmented with residual connections and normalization to ensure stability during training.

Before being processed by the main encoder stack, the input data $X = (x_1, x_2, ..., x_T)$ is first transformed into a sequence of high-dimensional vectors. These vector embeddings are achieved through a linear layer, or an "embedding" layer, which projects the input features into a space of dimension d model.

As the Transformer architecture does not inherently capture ordering, positional information must be added explicitly. This is accomplished by adding positional encodings to the input embeddings. The positional encoding typologies and their behavior are detailed in Section 2.1.1.

The core of the network is a stack of N identical encoder layers. Each layer is composed of two primary sub-layers: a multi-head self-attention (MHSA) mechanism and a position-wise feed-forward network (FFN).

The Multi-head self-attention is the key innovation of the Transformer. It allows

the model to weigh the importance of different time steps when representing a single time step. Instead of treating all past data points equally, it can, for example, pay more attention to a recent sharp impact when predicting the subsequent rebound. The "multi-head" aspect enhances this capability by performing the self-attention process multiple times in parallel across different learned "representation subspaces." This allows different heads to focus on various temporal patterns simultaneously—for instance, one head might capture short-term, high-frequency vibrations while another tracks longer-term oscillations. The self-attention mechanism itself is detailed in section 2.1.2.

Another important component is the *Feed-Forward Network* sub-layer, it consists of a two-layer, fully connected network with a non-linear activation function (typically GELU or ReLU) applied between the layers. It processes the output of the attention mechanism, introducing additional non-linearity and allowing for more complex feature extraction at each time step independently.

To ensure stable training of such a deep network, each of these two sub-layers has a residual connection around it, followed by layer normalization. This means the input to a sub-layer is added to its output, preventing the loss of information and mitigating the vanishing gradient problem. The subsequent normalization stabilizes the distributions of activations within the model, accelerating convergence.

2.1.1 Positional encoding

Positional encoding plays a crucial role in Transformer architectures by providing temporal order information to the self-attention mechanism, which is inherently permutation invariant. The choice of positional encoding scheme has been identified as a major factor influencing length generalization and model performance, particularly in sequence modeling tasks [ZGZ+24, KPN+23, Guo24]. Recent comprehensive studies have systematically evaluated various positional encoding approaches, revealing significant differences in their effectiveness for different applications.

Absolute Positional Encoding. In this approach, each position p in an input sequence is assigned a unique embedding vector $\mathbf{e}_p$, which may be learned jointly with the model parameters or initialized and kept fixed. These position embeddings are added to the token embeddings $\mathbf{x}_p$ to form position-aware inputs

$\mathbf{x}_p + \mathbf{e}_p$, thereby breaking the permutation invariance of self-attention and enabling the model to distinguish tokens at different sequence positions. Learned absolute encodings offer flexibility to adapt to data during training, while fixed encodings avoid overfitting to specific sequence lengths but may lack adaptability.

Sinusoidal Positional Encoding. This approach was originally introduced in [VSP+17], it is a deterministic scheme where each position p is encoded as a vector whose components are given by (2.1) and (2.2) for dimension index i and model dimensionality d.

$$\mathrm{PE}(p, 2i) = \sin\big(p/10000^{2i/d}\big) \tag{2.1}$$

$$\mathrm{PE}(p, 2i+1) = \cos\big(p/10000^{2i/d}\big) \tag{2.2}$$

This construction yields position vectors with smoothly varying frequency content, ensuring that similar positions have correlated encodings and that the model can, in principle, generalize to positions beyond those seen during training. Using sinusoidal functions also facilitates gradient flow and obviates the need to learn extra parameters for position information.

Rotary Positional Encoding. Rotary positional encoding (RoPE) [SLP+21] embeds position information directly into the attention mechanism by applying complex rotations to the query and key vectors. Given a base rotation matrix R_p derived from sinusoidal angles for position p, the transformed vectors $\widetilde{\mathbf{q}}_p = R_p \mathbf{q}_p$ and $\widetilde{\mathbf{k}}_p = R_p \mathbf{k}_p$ encode both absolute and relative positional relationships. This method seamlessly integrates position into the dot-product attention score $\widetilde{\mathbf{q}}_p^\top \widetilde{\mathbf{k}}_{p'}$, improving extrapolation to longer sequences and capturing distance-dependent interactions without separate learnable bias terms.

2.1.2 Attention Mechanism

Self-attention is a mechanism that enables a model to create context-aware representations of elements in a sequence by weighing the importance of all other elements relative to a specific one. It works by projecting each input vector into three distinct vectors: a Query (Q), a Key (K), and a Value (V). The Query vector represents the current element seeking to aggregate context. The Key vectors

represent all elements in the sequence that provide context, and the Value vectors represent the content of these contextual elements.

The process begins by calculating a score for each element in the sequence with respect to the current element. This score is the dot product of the current element's Query vector with the Key vectors of all elements (including itself). These scores are then scaled down by dividing by the square root of the dimension of the key vectors, $\sqrt{d_k}$, to ensure numerical stability. A *softmax* function is applied to these scaled scores to transform them into attention weights—a set of positive values that sum to 1. Finally, the output for the current element is a weighted sum of all Value vectors in the sequence, where the weights are the computed attention scores. This entire operation can be expressed concisely by the formula presented in Eq. 2.3.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{2.3}$$

This mechanism allows the model to dynamically decide which parts of the input sequence are most relevant for understanding and representing each individual part, effectively capturing complex dependencies regardless of their distance in the sequence.

2.1.3 Transformer for time-series

In time-series applications, Transformers have demonstrated superior performance by leveraging their self-attention capabilities to model both short-term variations and long-term trends [SZL+25, AHLH24]. The self-attention mechanism enables Transformers to capture semantic correlations among elements within long sequences, effectively addressing temporal dependencies that span across different time horizons [KIK25]. This advantage is further enhanced by the parallelized processing of input data, which accelerates computation and allows the model to scale effectively to large datasets.

The vanilla Time Series Transformer [WZZ+23] serves as the foundational encoder-decoder architecture for probabilistic forecasting, establishing the baseline approach for applying Transformers to temporal data. Informer [ZZP+20] pioneered efficient long sequence time-series forecasting by introducing ProbSparse attention

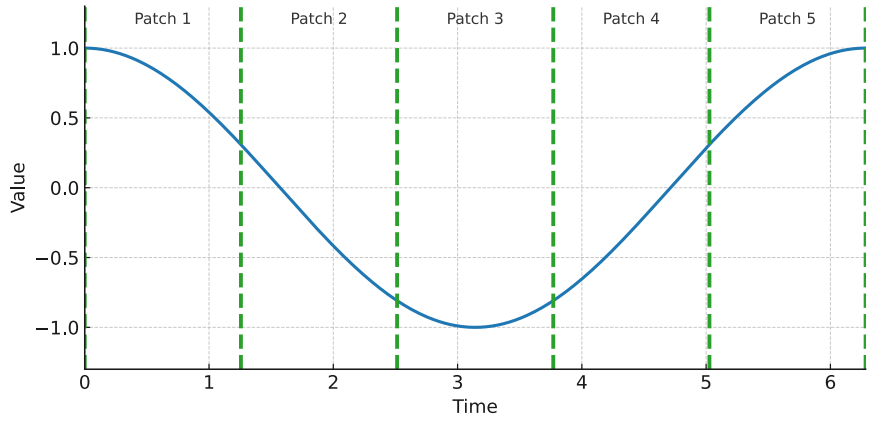


Figure 2.2: This figure illustrates the patchification process in the PatchTST model, where the time series data (represented by the cosine wave) is segmented into patches. Each colored region represents a distinct patch.

to achieve $O(n\ log(n))$ complexity and addresses the quadratic computational bottleneck of vanilla Transformers. Building upon this foundation, Autoformer [WXWL21] employed a novel decomposition architecture with Auto-Correlation mechanism to progressively decompose trend and seasonal components, moving beyond traditional attention mechanisms.

PatchTST [NHNSK23] - following the trend of [DBK+20] - proposes a new approach in the field by segmenting time series into subseries-level patches served as input tokens, achieving significant improvements in long-term forecasting accuracy while maintaining channel independence. The patching process is shown in Figure 2.2.

These findings open the possibility of training a foundational model that takes time series as input. Most recently, TimesFM [DKSZ23], Google's foundation model trained on 100 billion real-world time points, demonstrates impressive zero-shot performance across diverse datasets and temporal granularities, representing the latest advancement toward universal time series models. In a similar direction, MOMENT-1 [GSC+24] introduced a family of open-source foundation models for universal time-series analysis, capable of performing forecasting, imputation, and classification tasks through in-context learning without requiring task-specific fine-tuning.

2.2 Learning paradigm

This section outlines the fundamental learning paradigms that form the basis for the methodologies employed in this thesis. The discussion will focus on three key approaches: supervised learning (section 2.2.1), which relies on labeled data; weakly supervised learning (section 2.2.2), which handles imperfect supervision; and self-supervised learning (section 2.2.3), which leverages unlabeled data to learn powerful representations. It is important to note that two other major paradigms, unsupervised learning [CMZA22] and reinforcement learning [KLM96, WWL+24], are intentionally omitted because they fall outside the scope of this thesis.

2.2.1 Supervised Learning

Supervised learning forms the foundational paradigm for machine learning applications in vehicular sensor data analysis [SV25]. Given an input space $\mathcal{X}$ (representing all possible sensor measurements), an output space $\mathcal{Y}$ (representing all possible labels or predictions), and an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (the true underlying relationship between inputs and outputs), supervised learning seeks to find a hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$ (a function mapping inputs to outputs) from a hypothesis class $\mathcal{H}$ (the set of all possible functions we consider) that minimizes the expected risk in Eq. 2.4, where ℓ denotes the loss function measuring prediction accuracy, $R(h)$ represents the expected prediction error, and $\mathbb{E}_{(x,y)\sim\mathcal{D}}$ denotes the expected value over all possible input-output pairs (x, y) drawn from the distribution $\mathcal{D}$.

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)] \tag{2.4}$$

The intuition behind this formulation is straightforward: we want to find a model that makes accurate predictions on average across all possible scenarios, not just the specific examples we've seen during training.

The theoretical foundations rest on PAC (Probably Approximately Correct) learning theory [Val84], which provides sample complexity bounds through the Vapnik-Chervonenkis (VC) dimension [VC15]. This theory ensures that with probability at least $1 - \delta$ (where δ represents the acceptable failure probability), the learned hypothesis achieves error at most ϵ (the maximum acceptable error)

when the training sample size m satisfies Eq. 2.5, where $VC(\mathcal{H})$ denotes the VC dimension of the hypothesis class, a measure of the model's capacity.

$$m \geq \frac{1}{\epsilon}\left(\log\left(\frac{1}{\delta}\right) + VC(\mathcal{H})\right) \tag{2.5}$$

The intuition behind this bound is that more complex models (higher VC dimension) require more training examples to achieve the same level of confidence in their generalization ability.

The mathematical framework for time series supervised learning structures temporal data as sequences of sensor measurements over time as show in Eq. 2.6 where each x_i represents a D-dimensional sensor measurement vector (e.g., speed, RPM, throttle position, and temperature readings from D different sensors) captured at timestamp t_i, and T denotes the total number of time steps in the sequence. For instance, in a motorcycle ECU system with $D = 10$ sensors sampling data every second for $T = 60$ seconds, we obtain 60 measurements, each containing 10 sensor values.

$$X = \{(x_1, t_1), (x_2, t_2), \ldots, (x_T, t_T)\} \quad \text{where} \quad x_i \in \mathbb{R}^D \tag{2.6}$$

The intuition behind this representation is that by organizing data as ordered sequences $(x_1, t_1) \rightarrow (x_2, t_2) \rightarrow \ldots \rightarrow (x_T, t_T)$, supervised learning algorithms can learn how sensor patterns evolve over time, capturing dependencies such as "when RPM increases rapidly followed by sudden throttle closure, a gear shift is likely occurring." This temporal structure enables the model to map input sequences $X \in \mathcal{X}$ to output predictions $y \in \mathcal{Y}$ (such as riding style classification or fuel consumption estimation) by learning from labeled examples in the training data. In this direction, [PSAG24b] presents a supervised learning approach based on ECU data to predict coolant engine temperature, leveraging temporal patterns to forecast future temperatures from historical sensor readings.

This representation enables supervised learning algorithms to capture complex temporal dependencies in motorcycle ECU data streams. Recent developments in supervised learning for time series analysis have demonstrated significant advances in automotive applications, particularly for ECU data processing and driving behavior classification. Work such as [CRG24] achieves remarkable results in real-

time fuel consumption prediction using ECU data, reporting R^2 values (coefficients of determination) of 0.97 for vehicle speed-mass air flow models and 0.96 for RPM-throttle position sensor relationships.

The supervised learning framework enables the real-time processing of multi-dimensional sensor streams for applications such as collision detection, predictive maintenance, and rider behavior profiling, with machine learning approaches scaling effectively to accommodate the increasing complexity of modern motorcycle sensor arrays.

2.2.2 Weakly supervised Learning

Weakly supervised learning addresses the fundamental challenge of learning from imperfect supervision, as formalized by [Zho17]. The paradigm encompasses three distinct supervision degradation scenarios: incomplete, inexact and inaccurate.

Incomplete supervision addresses scenarios where only a subset $\mathcal{D}_L \subset \mathcal{D}$ of training instances possesses labels, where $\mathcal{D}$ denotes the complete dataset and $\mathcal{D}_L$ the labeled subset, while the remainder $\mathcal{D}_U = \mathcal{D} \setminus \mathcal{D}_L$ remains unlabeled, with $|\mathcal{D}_L| \ll |\mathcal{D}|$ indicating that the cardinality of labeled instances is much smaller than the total dataset size. Intuitively, this scenario mirrors the common situation where labeling data is expensive or time-consuming—such as requiring domain experts to manually annotate thousands of sensor readings—resulting in most data remaining unlabeled. The learning algorithm must leverage the small labeled set to extract patterns applicable to the abundant unlabeled data, similar to a student learning from a few solved examples and then applying that knowledge to many unsolved problems.

Inexact supervision occurs when labels are provided at a coarser granularity than required for the learning task. In the multi-instance learning (MIL) formulation, training data consists of bags $X_i = \{x_{i1}, \ldots, x_{im_i}\}$, where X_i represents the i-th bag containing m_i instances x_{ij}, with bag-level labels $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the label space. The objective is to learn an instance-level classifier $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ represents the instance feature space and f maps individual instances to their predicted labels. The intuition behind this formulation is that we often have labels for groups or collections of data points rather than individual points. For instance,

we might know that a motorcycle riding session was overall "aggressive," but we lack precise labels indicating which specific moments within that session exhibited aggressive behavior. The algorithm must infer fine-grained patterns from these coarse-grained annotations, analogous to knowing a movie is "action-packed" and trying to identify which specific scenes contain action sequences.

Inaccurate supervision addresses the presence of label noise in training data, formally modeled through a noise transition matrix $T \in [0,1]^{|\mathcal{Y}|\times|\mathcal{Y}|}$, where T is a square matrix of dimension $|\mathcal{Y}| \times |\mathcal{Y}|$ with entries in the interval $[0,1]$, and $T_{ij} = P(\tilde{y} = j \mid y = i)$ represents the probability of observing corrupted label $\tilde{y} = j$ given that the true label is $y = i$. This scenario reflects the reality that labels are often imperfect: human annotators make mistakes, automated labeling systems introduce errors, or sensor malfunctions produce incorrect readings. The noise transition matrix quantifies how likely each type of error is—for example, how often "normal braking" gets mislabeled as "hard braking." Learning algorithms must be robust to these systematic errors, distinguishing genuine patterns from labeling artifacts, much like a teacher who can identify which test answers are likely mistakes versus genuine misconceptions.

The relevance of weakly supervised learning to motorcycle ECU data analysis is particularly compelling given the inherent challenges of obtaining comprehensive labeled datasets for diverse riding conditions and behavioral patterns. Noise-robust learning techniques become essential for ECU sensor data, which exhibits varying reliability across different sensor modalities and environmental conditions.

The application of co-teaching frameworks [HYY+18] and adaptive sample selection strategies — such as triplet loss [SKP15] — addresses instance-dependent label noise characteristic of real-world sensor deployments. Recent work on federated weakly supervised learning [SYL+25] demonstrates robust two-stage frameworks specifically designed for sensor-based activity recognition with label noise, incorporating Gaussian Mixture Model-based client quality assessment and prototype regularization for distributed motorcycle fleet analysis while preserving rider privacy.

2.2.3 Self-supervised Learning

Self-supervised learning (SSL) [BIS+23] provides a framework for learning from vast amounts of unlabeled data, thereby bypassing the costly and time-consuming manual annotation required by supervised methods. The core principle of SSL is to devise a pretext task where the supervision signal is generated from the input data itself. By solving this pretext task [DCLT18] — for instance, predicting a masked portion of an input from the visible parts — the model is compelled to learn meaningful and transferable representations of the data.

To illustrate this concept concretely: in natural language processing, a model might be trained to predict masked words in a sentence (e.g., predicting "motorcycle" in "The `[MASK]` accelerated rapidly") [DCLT18], learning linguistic patterns without requiring labeled examples. In the context of time-series data from motorcycles, this principle can be applied by masking segments of sensor readings — such as a portion of the speed, throttle position, or lean angle — and training the model to reconstruct these missing values from the surrounding temporal context. Through this reconstruction task, the model learns to capture the patterns and dependencies in riding behavior without requiring expensive manual labels like "aggressive braking" or "smooth cornering" for each segment.

The field of SSL has evolved into several distinct families of methods, each with its own strategy for generating these self-supervisory signals. While all SSL approaches share the common goal of learning from unlabeled data, they differ fundamentally in their learning mechanisms and architectural choices. The Deep Metric Learning family (Section 2.2.3), with its roots in supervised contrastive methods, has evolved to define similarity through data augmentation rather than explicit labels, organizing the embedding space by contrasting multiple views of the data. The Self-distillation family (Section 2.2.3) takes a different approach, using asymmetric teacher-student architectures to generate stable learning signals without requiring negative samples. Finally, the Canonical Correlation Analysis family (Section 2.2.3) draws from classical statistics to directly optimize the statistical properties of learned representations, promoting invariance and decorrelation through structured objectives. Each of these paradigms offers advantages and trade-offs in terms of scalability and the quality of learned representations.

Deep Metric Learning

The Deep Metric Learning (DML) family of methods is built on the foundational goal of training a model to organize an embedding space where distance directly corresponds to semantic similarity. The history of this approach traces back to early explorations of contrastive frameworks for learning (Section 3.2), such as [BGL+93], which laid the conceptual groundwork. In its initial formulation, DML operated in a fully supervised setting. The objective was to train a network that would map semantically similar inputs to nearby points in the embedding space while pushing dissimilar inputs far apart, where pre-existing class labels from fully-annotated datasets explicitly defined semantic similarity. A significant refinement of this idea was the development of the "triplet"-based objective, explored in [SKP15], which structured the learning task around a query anchor, a positive sample, and a negative sample. These triplets were constructed from labeled datasets: given an anchor image of a particular class, a positive was sampled from the same class, and a negative was drawn from a different class. The model's task was to ensure the anchor's representation was closer to the positive's than to the negative's by a specified margin. This supervised paradigm proved effective for tasks like face recognition and image retrieval where fine-grained labeled data was available, but it required substantial annotation effort and struggled to scale to domains where labels were expensive or unavailable. As the field evolved, researchers sought to reduce the dependency on expensive annotations. A weakly-supervised variant emerged to address scenarios with limited or noisy supervision, where approximate similarity relationships could be derived from sources such as image-caption pairs, GPS coordinates, or temporal ordering, rather than precise class labels. However, the true paradigm shift came with self-supervised learning (SSL), which eliminated the dependency on any form of explicit supervision. A significant step in this evolution from classical supervised DML approaches to the modern SSL paradigm was the development of more efficient training objectives capable of handling numerous negative examples simultaneously, such as Contrastive Predictive Coding (CPC) [vLV18]. This transition was characterized by several key conceptual shifts that redefined the field. The most fundamental change was in how positive and negative pairs were generated: instead of relying on explicit labels or weak

supervision signals, modern self-supervised contrastive methods create pairs on-the-fly through stochastic data augmentation applied to unlabeled data. Two differently augmented versions of the same image form a positive pair (as they share semantic content), while augmented versions of different images serve as negatives. This approach enabled learning from vast amounts of unlabeled data without any annotation cost. Furthermore, a crucial architectural innovation was the introduction of a small "projector" network on top of the main encoder, which proved essential for boosting the performance of the final representations. Finally, the need for complex "hard-negative mining" was largely eliminated in favor of a simpler and more scalable approach that leverages large, randomly-sampled training batches.

Self-distillation

The Self-distillation family of methods offers a powerful alternative to contrastive techniques by using an asymmetric teacher-student architecture to generate learning signals, thereby avoiding the need for negative sampling. The lineage of this approach can be traced to earlier self-labeling methods, such as DeepCluster [CBJD18], which presents an iterative process of clustering data into pseudo-labels and then training a network to predict these assignments. A major breakthrough occurred with the introduction of BYOL (Bootstrap Your Own Latent) [GSA+20], which established the modern self-distillation framework. BYOL proposed an online (student) network that learns to predict the representations of a target (teacher) network. Crucially, it prevented model collapse by utilizing a momentum encoder, where the teacher's weights are updated as a slowly moving average of the student's weights, thereby providing stable regression targets. SimSiam further simplified this architecture [CH20], which demonstrated that the momentum encoder was not strictly necessary and that a simple stop-gradient operation on the teacher branch was sufficient to prevent collapse. Subsequent work, such as DINO [CTM+21], successfully adapted this paradigm for Vision Transformers, reintroducing a momentum encoder while also adding centering and sharpening of the teacher's outputs. This process functions as a form of online clustering.

Canonical Correlation Analysis

The Canonical Correlation Analysis (CCA) family of methods is distinctly rooted in classical statistics, drawing its core principles from the CCA framework originally proposed in [Hot36]. The initial goal of this statistical technique was to infer the relationship between two sets of variables by finding linear transformations that maximized their correlation. These ideas were later adapted for deep learning through architectures like Deep Canonically Correlated Autoencoders (DCCAE) [WALB16] and in work as [AABL13], which focused on jointly learning parameters for two networks to make their outputs maximally correlated. This foundation gave rise to modern SSL methods like Barlow Twins [ZJM+21] and VICReg [BPL21]. These contemporary methods learn distortion-invariant representations by directly optimizing the statistical properties of embeddings derived from two augmented views of an input. The central objective is to force the cross-covariance matrix between the two sets of representations towards the identity matrix. This structured goal implicitly promotes multiple beneficial properties, as explicitly stated in VICReg: it encourages invariance by aligning the views, maintains variance along each feature dimension to prevent collapse, and reduces redundancy by minimizing covariance (decorrelating) between different feature dimensions. These methods offer a statistically principled approach to learning rich features without relying on contrastive pairs or teacher-student dynamics.

2.3 Time-series Embedding

Time series embedding is a technique for converting raw, variable-length time series data into fixed-length numerical vectors, often referred to as embeddings [FBA23, IGKM25]. The primary purpose of this transformation is to capture the key temporal dynamics and characteristic patterns of the series in a dense and compact format. This process organizes the raw data into a structured latent space where the proximity between vectors reflects the similarity between the original time series, making subsequent downstream tasks like classification, forecasting, or clustering more effective and computationally efficient. This conversion is typically achieved using neural network encoders, which are trained to extract features into

these vectors, usually called an embedding representation.

2.3.1 Domain-specific Representation

Traditionally, time series analysis — contrary to NLP [TY24] — has relied on domain-specific embedding representations. This approach involves either manual feature engineering or training specialized models tailored to the characteristics of a particular data domain. For example, in clinical applications, models for electrocardiogram (ECG) analysis [LOW+25] might be designed to specifically detect P-Q-R-S-T waves, while models for financial data would focus on capturing volatility and momentum indicators [DSD24].

These methods can achieve high performance on their target task because they incorporate explicit domain knowledge. However, their primary limitation is a lack of generalizability. A feature set or model architecture optimized for one domain, such as industrial sensor data, performs poorly when transferred to another, like human activity recognition, without significant redesign and retraining. This necessitates a costly and time-consuming development cycle for each new application, motivating the shift towards more universal approaches.

2.3.2 Universal Representation

The development of universal representations for time series data has recently become an active and promising area of research [TSK+24, KCL+25]. Unlike domain-specific approaches, universal representation learning, often powered by foundation models, aims to create powerful feature extractors that can generalize across a wide variety of time series domains and downstream tasks [TSK+24]. These models are typically pre-trained on vast and diverse datasets, learning a rich set of patterns that can be fine-tuned for specific applications.

This paradigm has led to the development of large-scale foundation models that instantiate these design principles. An example is **MOMENT** [ASY+19] — the model used in Section 9 — which employs a Transformer-based encoder pre-trained on a massive corpus of public time series data. Its pretext task is masked modeling, where it learns versatile patterns by reconstructing randomly masked patches of a series. The resulting embeddings have demonstrated strong zero-shot and few-shot

performance across a range of downstream benchmarks. Another notable direction is represented by models like **Moirai** [LLW+24, AWL+24, WLK+24], which extend the universal concept to multi-variate scenarios.

The core challenge for all such models remains the diversity of temporal patterns across domains. Unlike images or text, time series data exhibit vastly different characteristics in terms of scale, frequency, seasonality, and noise, making the design of a truly universal feature extractor a complex undertaking.

2.3.3 Hierarchical and Multi-Scale Representations

A key insight into creating robust universal representations is that temporal patterns often exist at multiple scales. For instance, traffic data has daily, weekly, and seasonal patterns. To this end, the concept of hierarchical and multi-scale representations has gained significant attention [PGG25]. These methods are designed to capture information at different resolutions simultaneously.

A novel technique in this area is Matryoshka Representation Learning (MRL) [KBR+22], which provides a framework for creating nested embeddings. In MRL, a high-dimensional representation contains within it a set of lower-dimensional, yet still meaningful, representations. This "Matryoshka doll" structure enables highly adaptive deployment based on computational constraints. The application of MRL to time series offers the ability to dynamically trade off between embedding accuracy and computational cost during inference. This is particularly relevant for real-time applications like vehicle behavior monitoring, where a coarse embedding can be used for low-latency alerts on an edge device, while the full, high-fidelity embedding can be used for detailed offline analysis on a server.

2.4 Contrastive learning

Contrastive learning typically involves learning representations by maximizing the similarity between similar pairs (positive pairs) and minimizing the similarity between dissimilar pairs (negative pairs). It is a general framework for representation learning and can be applied to various tasks. Its usage has been highly successful in various domains, including computer vision, natural language processing, and

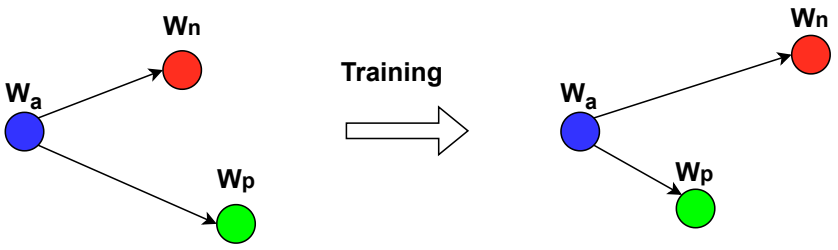


Figure 2.3: How Triplet loss works. After the training, the positive example W_p is getting closer to W_a than the negative example W_n.

recommendation systems. Deep metric learning, in the same direction, specifically focuses on learning embeddings (feature representations) where the distance between data points reflects their similarity or dissimilarity. Usually deep metric learning methods employ specific loss functions - such as triplet loss or contrastive loss - thought for learning embeddings that preserve semantic similarity.

2.4.1 Triplet Loss

Triplet loss is a loss function that aims to embed data points into a latent space where similar instances are placed closer than dissimilar ones. For each input data point, a "positive pair" is sampled, which is treated as similar examples, and the model is expected to bring them closer in the feature space. Negative pairs are also created by randomly selecting other data points from the dataset. The model is trained to minimize a contrastive loss function, which encourages the positive pairs to be close in the feature space while pushing the negative pairs apart. As the model is trained, it learns to create meaningful representations (embeddings) of the input data, which can be used for various downstream tasks such as image classification, object detection, or clustering. In Figure 2.3 an example of how this process works is shown. The triplet loss formula is reported in Eq. 2.7 where a is the anchor sample, p is the positive sample, n is the negative sample and m is the margin.

$$L(a, p, n) = \max\left(\|\mathbf{a} - \mathbf{p}\|_2 - \|\mathbf{a} - \mathbf{n}\|_2 + m, 0\right) \tag{2.7}$$

This forces the embedding network to learn a representation of driving samples that captures the unique patterns of each data sample.

2.4.2 The Challenge of Triplet Selection

Since its introduction, triplet loss [SKP15] has become one of the most widely used loss functions for metric learning. While triplet loss provides a framework for learning discriminative embeddings, its effectiveness heavily depends on the careful selection of triplets during training. Triplet Loss to effectively works in practice needs Negative Samples Mining [RCSJ20] — on each training step, we sample triplets (x_a, x_p, x_n) that satisfy Eq. 2.8.

$$D_{f_\theta(x_a,x_n)} < D_{f_\theta(xa,x_p)} + \alpha \tag{2.8}$$

This sampling strategy is crucial because randomly selected triplets often satisfy the margin constraint trivially, providing little learning signal. The challenge becomes more pronounced as training progresses. Towards the end of the training, most randomly selected negative examples can no longer yield non-zero triplet loss error. This leads to a fundamental sampling dilemma: easy triplets contribute minimal gradients, while maintaining computational overhead, whereas hard triplets risk introducing noisy or outlier samples that could destabilize training. Furthermore, During one update, the triplet loss only compares a sample with one negative sample while ignoring negative samples from the rest of the classes. As consequence, the embedding vector for a sample is only guaranteed to be far from the selected negative class but not necessarily the others.

These sampling limitations have motivated the development of more sophisticated mining strategies and alternative loss formulations. Over the years, numerous efforts have emerged to address these fundamental issues, including Quadruplet Loss [CCZH17], Structured Loss [SXJS15], and N-Pair Loss [Soh16], all building on the same core idea while attempting to improve sampling effectiveness. However, despite these improvements, the computational complexity of effective hard negative mining and the need for careful hyperparameter tuning in distributed training environments remain significant practical challenges.

These persistent limitations, along with the expansion and sampling issues inherent to contrastive approaches, led researchers to explore different paradigms. Rather than continuing to refine triplet-based methods, the field began moving toward approaches that could leverage larger sets of negative samples more efficiently and

avoid the complex mining procedures altogether. InfoNCE [CKNH20] emerged from this context as part of the broader contrastive representation learning framework, offering a principled way to learn from multiple negatives simultaneously.

2.4.3 InfoNCE Loss

InfoNCE (Information Noise Contrastive Estimation) [CKNH20] — Originally introduced in [vLV18] — is grounded in mutual information theory and designed to learn representations by distinguishing between positive pairs and a large set of negative samples simultaneously. InfoNCE has proven particularly effective in self-supervised learning scenarios, where the positive pairs are typically generated through data augmentation strategies rather than explicit labels.

The idea behind InfoNCE is: given a query sample and one positive example, the objective is to identify the positive among K negative samples drawn from a noise distribution. Formally, for a query representation q, a positive key k^+, and a set of negative keys $\{k_1^-, k_2^-, ..., k_k^-\}$, the InfoNCE loss is defined as Eq. 2.9 where τ is a temperature parameter that controls the concentration of the distribution, and the dot product $q \cdot k$ represents the similarity between query and key representations in the learned embedding space.

$$\mathcal{L}_{\text{InfoNCE}} = -\log\left(\frac{\exp(q \cdot k_+/\tau)}{\exp(q \cdot k_+/\tau) + \sum_{i=1}^{K} \exp(q \cdot k_i^-/\tau)}\right) \tag{2.9}$$

What makes InfoNCE particularly attractive is its ability to leverage large numbers of negative samples without the computational overhead of explicit hard negative mining. Unlike triplet loss, which compares against only one negative sample per update, InfoNCE naturally incorporates information from all K negatives in a single forward pass. This "many-vs-one" comparison provides richer learning signals and better gradient flow, especially in the early stages of training when most samples would produce zero triplet loss.

From an information-theoretic perspective, InfoNCE provides a lower bound on the mutual information between the query and positive key, with the bound becoming tighter as the number of negative samples K increases. This theoretical foundation offers guarantees about representation quality that were absent in earlier

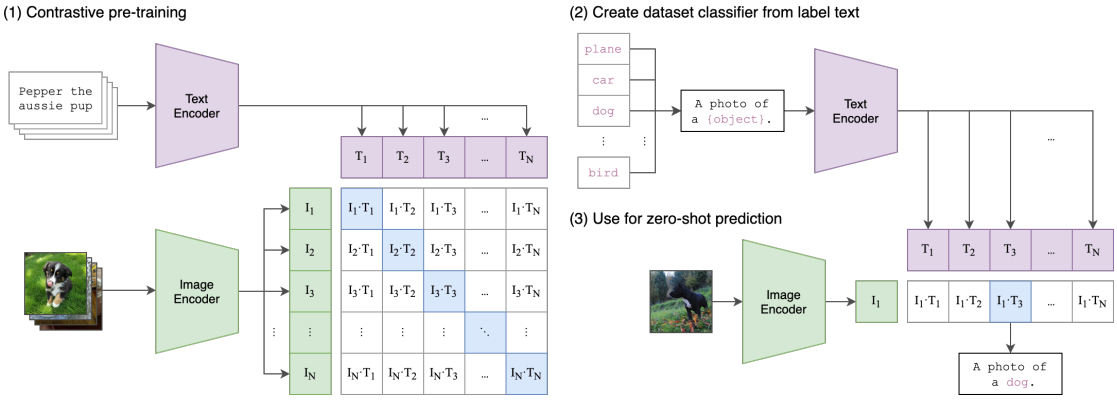


Figure 2.4: Original image from [RKH+21] describing the CLIP framework. This diagram illustrates the training and inference process of the CLIP (Contrastive Language-Image Pre-training) model. On the left, it learns to match images with their corresponding text descriptions from a large batch. On the right, it uses this knowledge to perform zero-shot classification by finding the most likely text description for a given image.

heuristic approaches. Moreover, the method scales naturally to large batch sizes and distributed training scenarios, as negative samples can be efficiently drawn from within-batch examples or maintained in memory.

The temperature parameter τ plays a crucial role in controlling the learning dynamics. Lower temperatures create sharper distributions that emphasize hard negatives, while higher temperatures lead to softer distributions that consider all negatives more equally. This provides a principled way to balance between focusing on difficult examples and maintaining stable training, addressing one of the key challenges in triplet loss optimization.

2.4.4 Contrastive Language-Image Pre-training (CLIP)

CLIP (Contrastive Language-Image Pre-training) — introduced in [RKH+21] — represents an application of contrastive learning that bridges the gap between computer vision and natural language processing. CLIP demonstrates how contrastive learning principles can be scaled to learn multimodal representations from web-scale data, fundamentally changing how we approach vision-language understanding tasks.

CLIP employs a dual-encoder architecture consisting of separate neural networks for processing images and text. The image encoder can be either a Vision Transformer (ViT) or a ResNet-based convolutional network, while the text encoder is typically a Transformer architecture similar to GPT. Both encoders map their respective inputs to a shared d-dimensional embedding space where semantic similarity can be measured through cosine similarity or dot product. Given a batch of N (image, text) pairs, CLIP learns to maximize the cosine similarity between correct image-text pairs while minimizing it for incorrect pairings. This creates an $N \times N$ classification problem where each image must be matched with its corresponding text description from among N possibilities, and vice versa. The training pipeline is shown in Figure 2.4.

CLIP's training procedure directly leverages the InfoNCE loss formulation adapted for the multimodal setting. For a batch containing image embeddings $I = i_1, i_2, ..., i_n$ and text embeddings $T = t_1, t_2, ..., t_n$, where (i_j, t_j) forms a positive pair, the loss function can be expressed as shown in Eq 2.12 where L_{T2I} (Eq 2.10) is the treating text as queries and images as keys and L_{I2T} (Eq 2.11) is the symmetric version treating images as queries and text as keys.

$$\mathcal{L}_{\text{I2T}} = -\frac{1}{N}\sum_{j=1}^{N}\log\left(\frac{\exp(\cos(i_j, t_j)/\tau)}{\sum_{k=1}^{N}\exp(\cos(i_j, t_k)/\tau)}\right) \tag{2.10}$$

$$\mathcal{L}_{\text{T2I}} = -\frac{1}{N}\sum_{j=1}^{N}\log\left(\frac{\exp(\cos(t_j, i_j)/\tau)}{\sum_{k=1}^{N}\exp(\cos(t_j, i_k)/\tau)}\right) \tag{2.11}$$

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{2}[\mathcal{L}_{\text{I2T}} + \mathcal{L}_{\text{T2I}}] \tag{2.12}$$

This bidirectional contrastive learning ensures that both modalities learn to encode semantically meaningful information that transfers across domains.

CLIP's contrastive pre-training enables remarkable zero-shot transfer capabilities across a wide range of vision tasks. By formulating classification problems as image-text matching tasks—where class names are converted to natural language descriptions—CLIP can perform competitively on numerous benchmarks without task-specific fine-tuning. This demonstrates how contrastive learning can lead to more generalizable representations compared to traditional supervised

approaches. The model's ability to understand compositional concepts, handle out-of-distribution examples, and perform reasoning about visual scenes stems directly from its contrastive training objective. By learning to associate images with natural language descriptions, CLIP develops rich semantic understanding that goes beyond simple object recognition.

Chapter 3

Related Work

The research presented in this thesis builds upon several areas of machine learning and vehicle behavior analysis. This chapter provides a comprehensive review of the relevant literature, organized as follows: Section 3.1 presents a review of methodologies for diving Style Recognition and Vehicle Behavior Analysis, Section 3.2 presents an overview of current methods for Contrastive Learning in the Time Series domain and Section 3.3 make an overview on the data-centric paradigm for machine learning.

3.1 Driving Style Recognition and Vehicle Behavior Analysis

Deep learning has shown great promise in handling time series data, making it an ideal candidate for analyzing and recognizing driving styles [ZYW+25, VFCG25]. Time series modeling for driving style recognition involves organizing driver data coming from sensors by timestamps and using machine learning models to capture temporal patterns. These models can be used to classify drivers or extract features that discriminate driving style – e.g. aggressive or cautious – by learning from the data [CXZ+21]. Existing works assume the availability of accurate, labeled data that has been acquired in ad-hoc sessions. Usually, the approaches to driver style recognition existing in the literature can be divided essentially into three classes:

- Trajectory-based approaches: that rely on the ability to analyze the trajectory

of the vehicle to identify patterns that are characteristic of different driving styles;

- Context-aware: these approaches, which consider the traffic context when classifying the driver's style, e.g. whether a driver is more aggressive in heavy traffic than in light traffic;
- Feature-based: they extract a set of features from the vehicle's motion data, such as thrusting, braking, and steering angle.

Trajectory-based methods have emerged as a fundamental approach for driving behavior analysis. [CXZ+21] proposed a measurement of risk (MOR) method to recognize risky driving behavior based on trajectory data extracted from surveillance videos, studying three types of dangerous driving behavior: speed-unstable driving, serpentine driving, and risky car-following driving. Their method achieved recognition accuracies of 91% using boxplot-based methods and 86% using distribution-based methods. Similarly, [AAGG+22] implemented a methodology for automatically detecting vehicle maneuvers from vehicle telemetry data under naturalistic driving settings, employing an energy-maximization algorithm (EMA) capable of extracting driving events of varying durations from continuous signal data. Recent advances in trajectory analysis have incorporated deep learning techniques for enhanced pattern recognition. [ZYW+25] conducted an analysis of possible quantifications of driving style in trajectory forecasting, emphasizing that human driving style is a correlating factor to decision making, especially in edge-case scenarios where risk is nontrivial. However, most trajectory-based approaches remain focused on four-wheeled vehicles, with limited application to motorcycle dynamics, which present unique challenges due to their lean angle dynamics and different interaction patterns with the road environment.

Context-aware systems provide intelligent recommendations by considering environmental and traffic conditions when assessing driving behavior. [RM05] defined a framework for a context-aware driving behavior model capable of predicting driver's behavior by integrating contextual information related to the vehicle, environment, driver, and traffic conditions. Their approach used Bayesian networks to model driver behavior while considering factors such as risk aversion, age, and environmental conditions like lane occupancy.

[DAA21] developed a context-aware driver behavior monitoring system in vehicular ad-hoc networks (VANET) that monitors irregular actions such as high speed, alcohol consumption, and driver pressure, sending alerts to nearby vehicles and roadside units. The system adopted a real-time VANET prototype involving three entities: the driver, vehicle, and environment, achieving detection accuracies above 85% for various risky behaviors [FAMC24]. Similarly, context-aware systems have been applied to detect abnormal driving behaviors by incorporating traffic flow characteristics, weather conditions, and time-of-day factors [Alg12]. However, context-aware approaches for motorcycles remain underexplored in the literature. The unique riding dynamics of motorcycles, including their vulnerability to weather conditions and different traffic interaction patterns, require specialized context-aware models that current automotive-focused research has not adequately addressed.

Feature-based methods constitute the most prevalent approach in driving style recognition, extracting discriminative features from sensor data to classify driving behaviors. [SEN20] proposed a systematic methodology to evaluate prediction models for driving style classification, comparing five different models (Support Vector Machines, Artificial Neural Networks, fuzzy logic, k-Nearest Neighbor, and Random Forests) using features extracted from vehicle motion data. Their results demonstrated that SVM outperformed other models, achieving an average accuracy of 96% with features including acceleration, deceleration, and steering patterns. Traditional feature extraction approaches focus on statistical measures derived from sensor data. [WX22] explored feature selection for driving style recognition using multi-classification and supervised learning, identifying maximum speed as the most important feature with differences up to 10 m/s among driving style groups. Their neural network-based classification model achieved 86.1% accuracy using only maximum speed as input, demonstrating the effectiveness of carefully selected features for driving style discrimination. Recent advances have incorporated deep learning architectures for automatic feature extraction from time series data. [SGAQ23] reviewed deep learning models for driver behavior detection in time series data, highlighting that Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) have shown superior performance compared to traditional machine learning approaches.

The CNN models achieved accuracy rates above 98%, while LSTM models reached 97.75% accuracy in driving maneuver classification tasks [AAGG+22].

3.2 Contrastive Learning for Time Series

Contrastive learning, initially developed for computer vision applications [ZZTZ22, LALZ24], operates on the principle of learning representations by contrasting positive and negative sample pairs. The core objective is to maximize similarity between positive pairs (similar samples) while minimizing similarity between negative pairs (dissimilar samples). When adapted to time series data, this paradigm faces challenges stemming from the temporal nature of the data, complex multivariate relationships, and domain-specific characteristics that differ significantly from visual or textual data.

Recent advances have demonstrated the effectiveness of contrastive learning in various time series tasks, including classification, forecasting, and anomaly detection [ERC+21, YWD+21]. Methods such as TS-TCC (Time-Series Temporal and Contextual Contrasting) [ERC+21], TS2Vec [YWD+21], and CoST (Contrastive Learning of Disentangled Seasonal-Trend Representations) [WLS+22] have established new state-of-the-art performance across multiple benchmarks, showcasing the potential of contrastive approaches in time series analysis. Specifically, TS-TCC employs a Temporal Contrasting Module that learns robust temporal representations by designing cross-view prediction tasks. Given two different augmented views of the same time series, the module utilizes past latent features from one augmentation to predict future representations of the other augmentation. This approach forces the model to learn robust representations that are invariant to perturbations while maintaining temporal consistency. Building upon the temporal module, the contextual component maximizes similarity among different contexts of the same sample while minimizing similarity among contexts of different samples. This dual-level approach ensures that learned representations capture both local temporal patterns and global contextual information.

In the same direction as the presented works, [LKL+23] introduces FOCAL, a contrastive learning framework tailored for multimodal time-series sensing signals. FOCAL effectively captures modality-specific and cross-modal features by factor-

izing the latent space into shared and private components, using orthogonality constraints to preserve modality-exclusive information and a temporal locality constraint to maintain relevant temporal relationships. Similarly, [LC23] presents TimesURL, a framework for universal time-series representation learning. It addresses challenges in constructing effective augmentations and hard negatives by introducing double Universums, enhancing contrastive learning with temporally consistent and discriminative samples, and jointly optimizing contrastive learning and time reconstruction to capture universal patterns.

3.3 Data-Centric Machine Learning

The composition and quality of the training dataset are universally acknowledged as critical factors influencing the performance and generalization capabilities of machine learning models. While random sampling has been a long-standing default, a growing consensus is emerging that more sophisticated data selection and scheduling strategies can yield significant improvements.

Recent research increasingly uses clustering of data points (in an embedding or feature space) to inform data sampling strategies or curricula during training. One prominent example in NLP is the `ClusterClip` method for large language models [SLF+24]. `ClusterClip` clusters the training corpus by semantic similarity (using pre-trained text embeddings) to approximate the data distribution, then adjusts the sampling of mini-batches to be initially uniform across clusters.

A conceptually related approach in LLM pre-training is CLIMB [DYF+25], which focuses on optimizing the mixture of data from different sources or clusters. While CLIMB's primary goal is to determine the ideal blend of data for pre-training (e.g., using token counts from different data clusters), its core principle of cluster-driven data composition is highly relevant. In their framework, a separate "predictor" model is trained to map data mixture compositions (as input features) to the final LLM performance metrics (as target labels).

In other machine learning fields, such as computer vision, the same idea of improving the quality of the training dataset has led to the use of clustering to design curricula for generative models. In [ZZGZ19], the authors propose a clustering-based curriculum for GAN training: data points from dense, central

clusters (i.e., prototypical examples) are fed to the GAN early on, whereas more peripheral (outlier) clusters are introduced gradually in later rounds. This strategy of progressing from "easy" core samples to "harder" outliers improved GAN convergence on noisy image datasets.

Despite these advances in text and image domains, time series data selection remains largely underexplored. Time series present unique challenges that distinguish them from other data modalities: they are inherently high-dimensional with complex temporal dependencies, often multivariate with intricate cross-channel relationships, and contain latent behavioral regimes that do not decompose cleanly into discrete semantic categories like text.

Chapter 4

Designing a Remote Data Acquisition System

The development and validation of robust machine learning models for next-generation motorcycle applications—such as predictive maintenance, adaptive traction control, and Advanced Rider-Assistance Systems (ARAS)—are critically dependent on extensive and varied real-world operational data. Acquiring such large-scale datasets presents a significant bottleneck, as data from simulations or controlled test tracks fails to capture the full spectrum of unpredictable variables, environmental conditions, and diverse scenarios encountered in actual riding.

To address this gap, a cost-effective remote data logging system — called DMLogger — has been developed. This system is not merely a data logger; it functions as a prototype for a motorcycle-connected Electronic Control Unit (ECU) [L+14, Ara05].

As a proof of concept, the system is engineered to reliably capture and transmit high-frequency CAN (Controller Area Network) bus [ISO15, ISO24, BEKL90] data directly from the vehicle during regular operation. This includes critical parameters such as throttle position, engine RPM, wheel speed, gear position, lean angle, and ABS/Traction Control interventions. Its cost-effective design is essential, as it enables scalable deployment across a fleet of test vehicles — the only way to achieve the necessary volume and diversity (e.g., different riders, road types, weather) for robust machine learning.

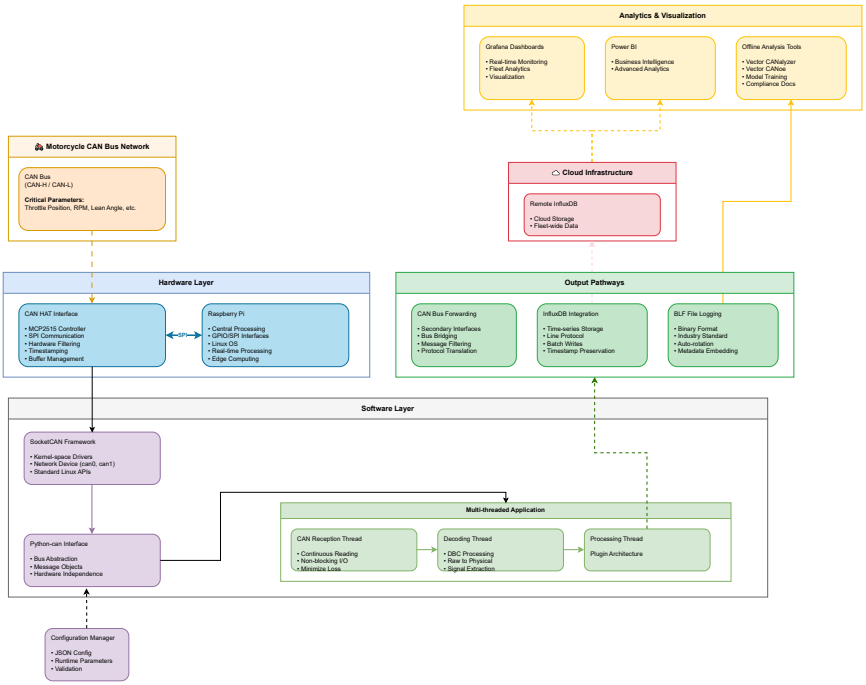


Figure 4.1: DMLogger System Architecture. End-to-end data acquisition pipeline showing hardware components (Raspberry Pi with CAN HAT interface), software stack (SocketCAN, python-can library, multi-threaded application), and output pathways (CAN forwarding, InfluxDB integration, BLF logging). The system enables real-time monitoring via Grafana dashboards and offline analysis through industry-standard tools, supporting both edge computing and cloud-based fleet analytics.

Based on testing conducted with the company, the system supports dual operational modes with no packet loss and consistent logging across all pathways: real-time monitoring through customizable Grafana dashboards and offline analysis using industry-standard tools such as CANape for signal inspection. The data acquisition and logging architecture has demonstrated data integrity from capture to analysis, making the system reliable for both monitoring and post-processing.

4.1 System Architecture

This section presents the design and implementation of the data-acquisition system for capturing and analyzing telemetry data from the motorcycle's Controller Area

Network (CAN) bus [AM18, BEKL90]. The system addresses the need for real-time monitoring and historical analysis of vehicle performance parameters, enabling both immediate operational insights and long-term analysis. The data acquisition pipeline — showed in Figure 4.1 — comprises three primary components:

- A hardware layer — described in Section 4.2 — composed of a Raspberry Pi single-board computer [Jol21, JC17, Ras25] and a CAN interface HAT (Hardware Attached on Top);
- A Software level — described in Section 4.3 — built on top of the python-can software library, InfluxDB [Inf25], Grafana [K+21, Gra25], and Power BI [Mic25];
- A processing thread — described in Section 4.4 — implementing a plugin-based architecture for multiple output pathways, including CAN bus forwarding, time-series logging, and binary file persistence.

These two components work together to create an end-to-end data pipeline from the motorcycle's CAN bus to cloud-based analytics platforms. The hardware layer ensures reliable physical connectivity and real-time data capture, while the software layer provides the intelligence for data processing, storage, and visualization.

4.2 Hardware Architecture

The Raspberry Pi was selected as the central processing unit for several strategic reasons aligned with the proof-of-concept objectives. Its computational capabilities are sufficient for real-time CAN message processing, local data buffering, and edge computing tasks, while maintaining a compact form factor suitable for motorcycle integration. The platform's GPIO and SPI interfaces provide direct hardware communication capabilities essential for CAN interface integration. We used a Linux environment, which enables the deployment of standard networking protocols, database systems, and Python runtime environments without the constraints typical of microcontroller-based systems. The CAN HAT interface provides the physical layer connection to the motorcycle's CAN bus network. This hardware module handles the electrical signal conversion between the CAN bus differential signaling

(CAN-H and CAN-L) and the digital interface accessible to the Raspberry Pi via SPI (Serial Peripheral Interface). The module incorporates an MCP2515 CAN controller [Mic21], providing hardware-level message filtering, timestamping, and buffer management capabilities that reduce the computational load on the main processor.

4.3 Software Architecture

The software stack is organized in a layered architecture that promotes modularity and maintainability. At the lowest level, the SocketCAN framework [Lin25, HRS12] provides kernel-space drivers for CAN hardware, exposing CAN interfaces as network devices (typically can0, can1, etc.) accessible through standard Linux networking APIs. This abstraction allows CAN buses to be treated similarly to Ethernet interfaces, enabling the use of standard networking tools for debugging and monitoring.

The python-can library [Pyt24] provides the application-level interface to SocketCAN, abstracting hardware-specific details and offering a consistent API regardless of the underlying CAN adapter. This library implements the Bus abstraction for sending and receiving messages, along with Message objects that encapsulate CAN frame data, including arbitration ID, data payload, timestamp, and flags (extended frame, remote frame, error frame).

The data logging application itself is structured as a multi-threaded Python application with distinct responsibilities:

- CAN Reception Thread: Continuously reads messages from the CAN bus using non-blocking or asynchronous I/O patterns to minimize message loss;
- Decoding Thread: Decodes raw CAN messages into physical values using database files (DBC format) when available, or maintains raw format for undocumented signals;
- Processing Thread: Performs configurable operations on decoded messages, including forwarding to secondary CAN buses, transmitting data to remote

endpoints via network protocols, and persisting messages to Binary Logging Format (BLF) [Vec25] files for offline analysis;

The combination of these three processing pathways provides comprehensive data coverage for different analytical needs. Real-time monitoring through InfluxDB and Grafana enables immediate data analytics through visualization. Simultaneously, BLF file persistence ensures that complete raw data is preserved for detailed post-processing, model training, and compliance documentation. The CAN bus forwarding capability facilitates integration with existing vehicle diagnostic systems or additional data acquisition equipment.

4.4 Processing Thread

The Processing Thread serves as the application's action layer, implementing multiple output pathways for CAN data based on runtime configuration. Its design employs a plugin-based architecture that allows individual processing modules to be enabled or disabled independently.

CAN Bus Forwarding

Messages can be selectively forwarded to one or more secondary CAN interfaces, enabling use cases such as bus bridging, message filtering, and protocol translation. The forwarding logic supports configurable filtering based on arbitration ID ranges, message content patterns, or timing criteria.

InfluxDB Time-Series Logging

Decoded CAN signals are transmitted to an InfluxDB instance for real-time visualization through Grafana dashboards. Each signal is mapped to a measurement with appropriate tags, and field values representing the physical quantities extracted during decoding. The integration leverages InfluxDB's line protocol for efficient batch writes, with configurable buffering intervals to balance write frequency against data granularity. Timestamps from CAN messages are preserved to maintain temporal accuracy, enabling precise correlation of events across multiple signals.

This pipeline supports both local InfluxDB deployments for on-site monitoring and remote instances for cloud-based fleet analytics.

Binary Logging Format (BLF) Persistence

All received messages can be logged to BLF files, the industry-standard format for automotive bus data supported by tools such as Vector CANalyzer and CANoe. Log files are automatically rotated based on size or time thresholds, with filenames incorporating timestamps for easy organization. Metadata, including bus configuration, sample rate, and system information, is embedded in the file header for complete traceability.

4.5 Configuration Management

The application's behavior is fully controlled through a JSON configuration file that defines all operational parameters without requiring code modifications. This approach enables rapid deployment across different vehicle platforms and use cases while maintaining a single codebase. The configuration file specifies:

- CAN Interface Settings: Physical interface names (e.g., can0, can1), bitrates, and bus-specific parameters such as sample point and timing configurations;
- DBC Database Paths: References to message definition files used by the decoding thread to translate raw CAN frames into named signals with physical units;
- Processing Module Activation: Boolean flags to enable or disable each output pathway (CAN forwarding, InfluxDB logging, BLF persistence);
- Logging and Diagnostics: Verbosity levels for application logging, performance metric collection intervals, and health monitoring parameters.

The configuration loader validates the JSON structure at startup, checking for required fields, valid data types, and logical consistency (such as ensuring referenced CAN interfaces exist in the system).

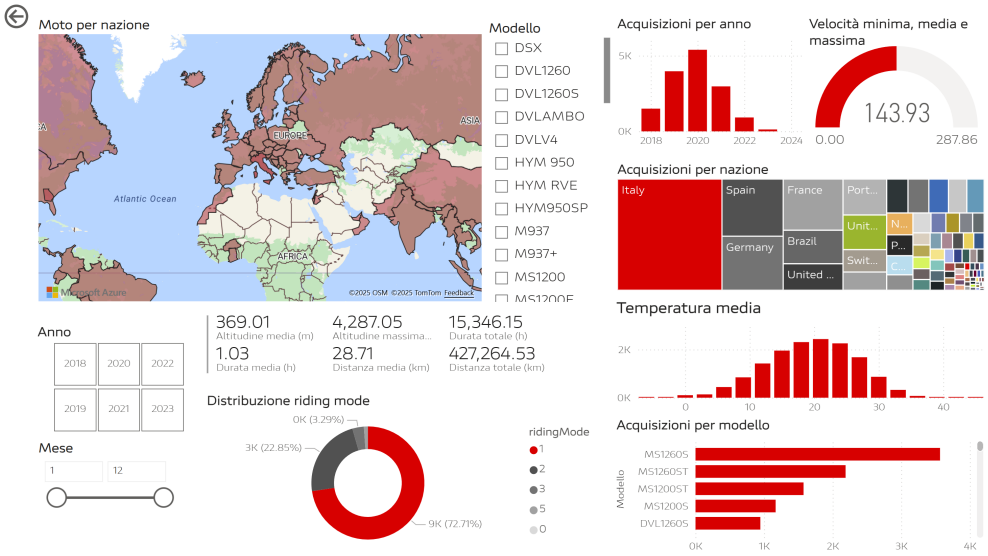


Figure 4.2: Power BI dashboard visualization. It enables users to navigate data by year, model, and geographic locality.

4.6 Data Visualization

A Power BI dashboard was developed to provide analytical insights from aggregated CAN bus data. The interface implements a data pipeline that extracts and preprocesses raw signals, computing aggregate statistics across time, vehicle, and geographic dimensions. Interactive filtering capabilities enable users to explore fleet-wide patterns by year, model, and geographic locality, supporting operational analysis. This interface transforms high-frequency time-series data into actionable insights for stakeholders.

This back-end computes key aggregate statistics across crucial dimensions — including temporal trends, vehicle performance, and geospatial patterns. The intuitive, interactive filtering capabilities empower stakeholders to dynamically segment and explore fleet-wide patterns by model year, specific vehicle models, and geographic localities. This tool directly supports advanced operational analysis, transforming vast streams of high-velocity time-series data into clear, actionable intelligence for optimizing fleet efficiency and informing maintenance strategies.

Chapter 5

Driving Style Recognition

Modern motorcycles are now equipped with a sophisticated array of sensors that continuously generate vast amounts of data. This data stream offers new opportunities to move beyond classic performance analysis and get fine grained information from the interaction between the rider and the machine. One of the most significant applications of this data is Driving Style Recognition, a field focused on identifying and analyzing the unique behavioral patterns of individual riders.

The core challenge lies in the nature of this data; it is often high-dimensional, complex, and lacks clear labels, making manual analysis challenging. This chapter uses triplet loss, discussed in 2.4.1, - and a weakly supervised approach to build the input tuple - to sift through these massive datasets and extract distinct "behavioral fingerprints" for each user. By doing so, it is possible to transform overwhelming and raw sensor readings into structured knowledge.

This chapter focuses on how representation learning can create robust models that capture the patterns of a person's riding habits—from how aggressively they accelerate and brake to their stability and confidence on the road. Ultimately, the ability to recognize and analyze driving styles opens the door to significant enhancements in user experience, vehicle personalization, and performance analysis.

This work has been published in the proceedings of the 2024 IEEE Intelligent Transportation Systems Conference (ITSC 2024) [PSAG24a].

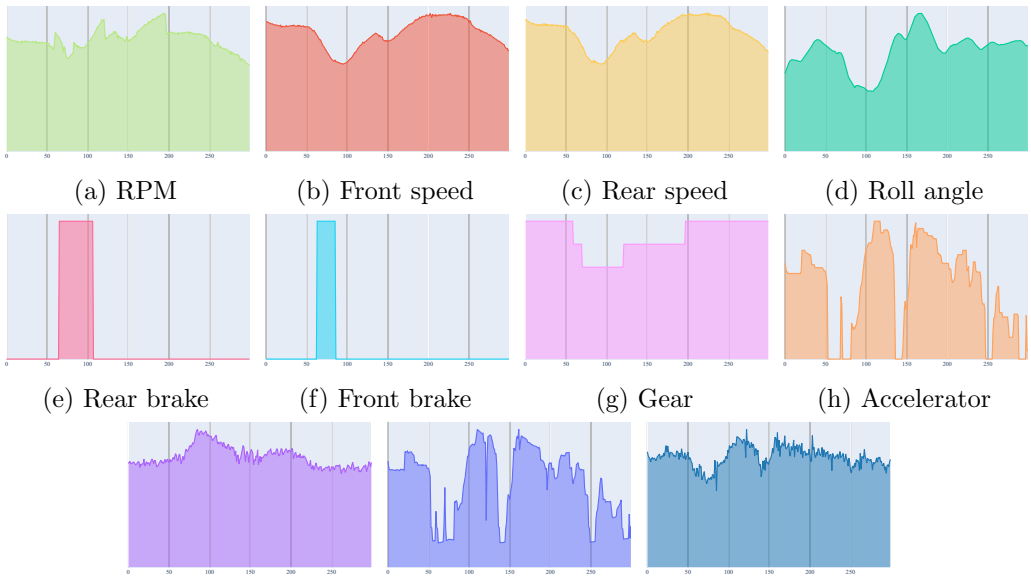
5.1 Research question

Motorcycles are equipped with sensors and various ECUs that produce and calculate quantities over the time that are meant to represent its state. Our motorcycle state is represented by the signals reported in Figure 5.1. This set of signals are logged to produce an acquisition that is used to record the events of the ride. This acquisition serves as a tool for analysing the motorcycle's performance, diagnosing potential issues, and gaining insights into its behaviour throughout the duration of the ride. We have a set $S_1 = \{A_1, A_2, ...A_n\}$ of acquisitions, whereas each of these acquisitions A_t is labelled with a day D_t, in which the acquisition happens. Furthermore, we have a list of drivers $Dr(S_1) = \{Dr_1, Dr_2, \ ... \ Dr_n\}$ where it is indicated who was driving on a given day. There is no direct correlation between drivers and acquisition. Usually on a given date D_t we have that $\|S_1\| \gg \|Dr(S_1)\|$. The dataset S_1 acquired on the road was not collected in a controlled environment. On the other side, we have a set S_2 of acquisitions where each acquisition is labelled with the correct driver in $Dr(S_2) = \{Dr_1, Dr_2, \ ... \ Dr_m\}$. However, the number of occurrences in S_2 is smaller than the number of occurrences in S_1. We have the information that $Dr(S_1) \cap Dr(S_2) = \emptyset$. The goal of this work is to build — using the data in S_1 - a model able to describe drivers in a multidimensional space. Furthermore, we want to evaluate our model on the task of driver recognition using S_2. In conclusion, we are interested in analysing the properties of the built representation space, extracting information on driver's riding style using a dataset S_3 coming from the same data lake of S_1.

5.2 Methodology

5.2.1 Dataset

The dataset we used during the development of this work was provided by *Ducati Motor Holding*. We have two different groups of data: the first one contains data coming from the road, while the second one contains data coming from a racetrack. The training set data was recorded from five *Multistrada V4* motorcycles, with eighteen different drivers during two different years. The total time of the



(a) RPM (b) Front speed (c) Rear speed (d) Roll angle

(e) Rear brake (f) Front brake (g) Gear (h) Accelerator

(i) Pitch angle (j) Clutch (k) Long. Acc.

Figure 5.1: Example of an input time series fed through the network. Each input reflects a sensor acquired from the motorcycle.

acquisition is 1420 h. In this data, there is no accurate annotation about which rider was driving in a specific acquisition. We have been able to extract the information about the list of drivers who were driving on the day of a given acquisition (usually more than 2), but it wasn't possible to assign a specific acquisition to a specific driver. For this reason, we used this set of data as training set: indeed, we can use a weakly supervised approach to extract knowledge from this mass of data, even though we cannot use them in the evaluation phase. Furthermore, a subset of the data for which we have not been able to extract any information on the drivers was used for clustering analysis. These data comprise in total 48 h and we use them to look at how the network was discriminating in road conditions.

On the other hand, we have been able to collect racetrack acquisitions where the driver was annotated in detail and we could use this data as test set. In this case, we have 14 drivers and a total of 104 h of acquisitions. It is worth mentioning that the set of drivers in the training set and the set of drivers in the test set are not overlapping. This allows us to have a final dataset with large variation in usage conditions. In Figure 5.5 further information are reported on the different datasets.

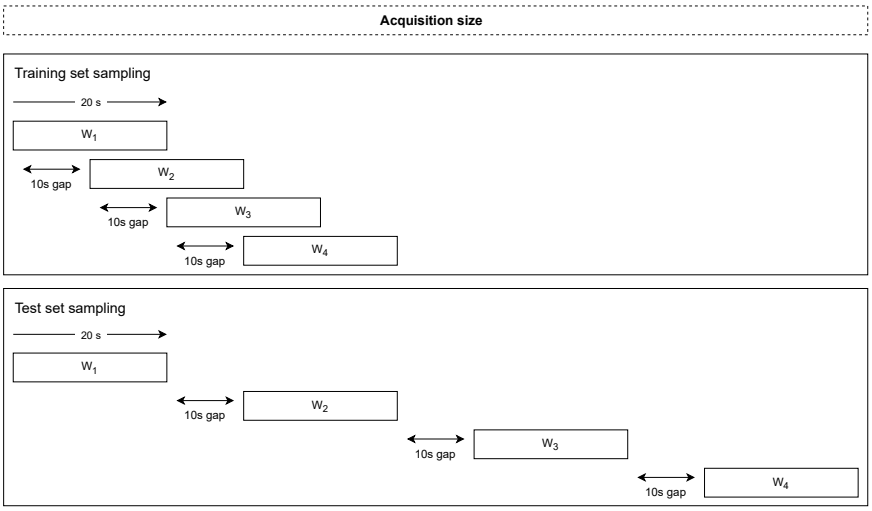


Figure 5.2: This diagram illustrates the sampling methodology for training and test sets.

No specific limits were placed on vehicle use and data has been acquired directly from the vehicle's *Engine Control Unit* (ECU) using a logger, a component that records and stores events, messages, and data relevant for monitoring. The strategy of acquiring data directly from ECUs allowed us to have good quality data recorder at high frequency. We split the dataset into time series windows of 20 s sampled at 10 Hz. We decided on this range of downsampling because it is compatible with the range of reaction of a human being. The typical human response time is around a quarter of a second, which is roughly 250 milliseconds (ms). This is the time it takes for someone to perceive a stimulus, process it in the brain, and then perform a corresponding action. Since our goal is to recognize characteristic behaviors of the driver using higher frequencies would then have deviated from a range compatible with our target. Furthermore, this assumption allowed us even to reduce the quantity of noise in our data. In the training set case, windows are sliding by 10 s. This is done to augment the quantity of data we have. Instead, to avoid overlapping, we slide the windows by a value of 30 s in the test set. A visual example of the different windowing strategies between training and test set is shown in Figure 5.2.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.1}$$

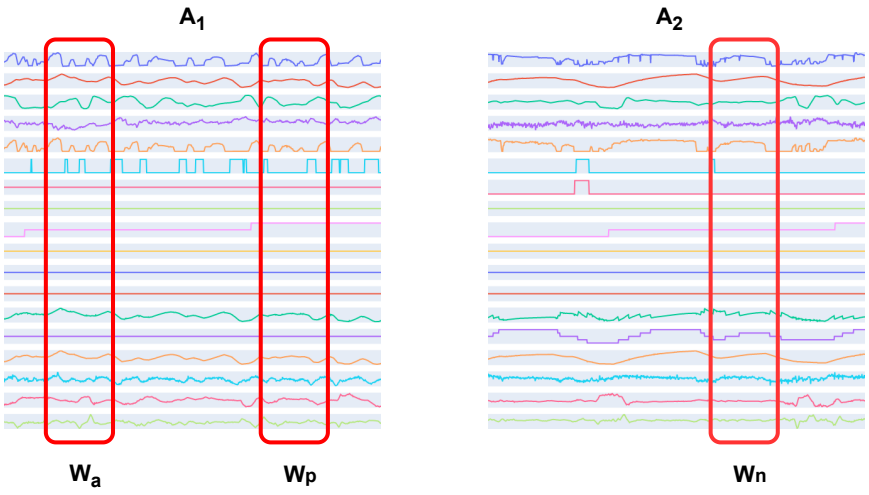


Figure 5.3: The anchor example W_a and the positive example W_p are sampled from the same acquisition A_1. Whereas the negative example is sampled from another file W_n.

Data values usually vary in a fixed range. For this reason, we decided to scale them in a range $[0; 1]$. Each feature has been scaled independently using a *min-max scaler*, the formula is shown in (5.1). This normalization method is highly useful because it scales down the original data range to a specific range, avoiding distortion, thereby preserving the proportional relationships between variables. By employing the *min-max scaler*, the data is uniformly transformed, allowing machine learning algorithms to converge more efficiently and better adapt to the data. This enhances model performance and helps prevent divergence and overfitting.

5.3 Experiment

5.3.1 Triplet sampling

In the use of triplet loss, the process of constructing data triplets plays an important role. In fact, as previously mentioned, the purpose of triplet loss is to bring the positive (similar) points closer while trying to push away negative (dissimilar) ones. Each triplet is composed of an anchor, a positive example, and a negative example. The anchor (a) – acting as the anchor point for learning – represents the data

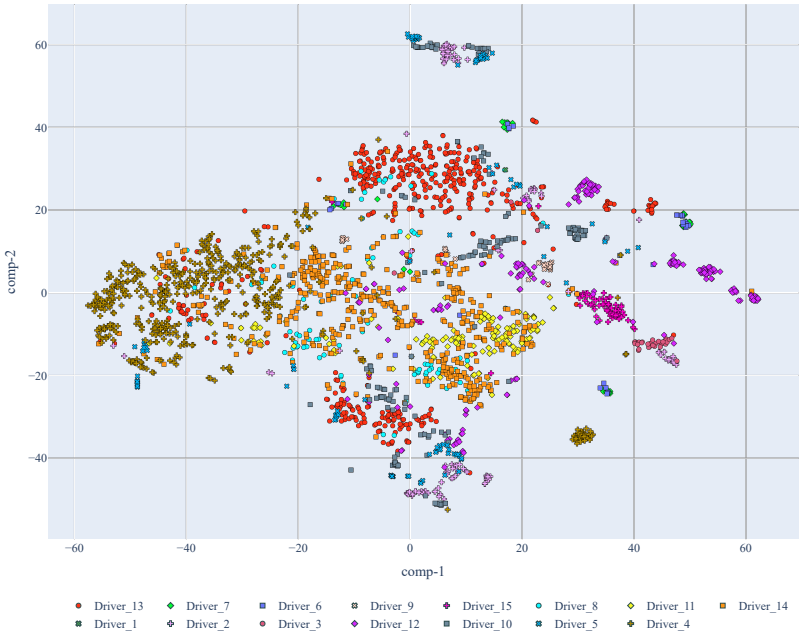


Figure 5.4: t-SNE visualization [vdMH08] of our embedding on the test-set S_2.

instance for which the model should learn a robust and discriminative embedding. On the other hand, the positive example (p) mirrors the anchor but represents a data point that shares conceptual similarity while, conversely, the negative example (n) is intentionally dissimilar and serves as a representative of data that differs significantly from the anchor. In the context of facial recognition, for instance, the anchor could correspond to an image of a person's face, while a positive example could take the form of another image of the same individual's face and a negative example could be the image of the face of a different person.

In this work, the anchor is a multivariate time series window W_a sampled from an acquisition A_1. The positive example is then a window W_p sampled from the

	Duration	Distance	Type
S_1	1420 h	102.580 km	Road
S_2	104 h	4.765 km	Circuit
S_3	48 h	4.166 km	Road

Figure 5.5: Duration and type for each dataset

same acquisition A_1, while the negative example is a window W_n sampled from an entirely different acquisition A_2. In Figure 5.3 it is visually shown how tuple are formed from the dataset.

$$Drivers(G_1) \cap Drivers(G_2) = \emptyset \tag{5.2}$$

We don't have any information about who was driving in a given acquisition, but we have just the information on which drivers were riding on a given day (usually up to 2). For this reason, we sample A_1 and A_2 look at who were riding on a given day. So a tuple is made following the formula in (5.2), where G_i is the day an acquisition A_i has been logged.

5.3.2 Training configuration

In this work, we decided to adopt a modification of *XceptionTime* [RZF+19] as the neural network on which to perform pre-training. The model was trained for $\approx 80,000$ iterations with a batch size of 256, utilizing the Adam optimizer and a learning rate of 0.001. The training time took 18 hours. We used such a large batch size because it reduces the noise in gradient updates, which can be particularly beneficial in situations where the triplet loss landscape can be noisy.

When we discuss triplet loss, embedding size and margin are key components for training. Embedding size determines the dimensionality of the feature vectors

Table 5.1: Latent space dimension fine-tuning

Latent space	**Metrics**		
	mAP	***Recall@1***	***Recall@5***
16	0.38	0.81	0.94
32	**0.40**	0.82	0.94
64	**0.40**	**0.83**	**0.95**
128	0.39	0.82	0.94
256	0.39	0.82	0.94
512	0.39	0.81	0.94

learned by the network, and this impacts the discriminative power and the generalization ability of the model. On the other side, the margin is a hyperparameter that plays a crucial role in determining how the loss encourages the learning of feature representations in the network. Both parameters embedding size and margin have been fine-tuned during the development of the project. Furthermore, we used the constraint presented in [SKP15] and showed in (5.3) to bind the embedding to reside on a hypersphere.

$$\|f(x)\| = 1 \tag{5.3}$$

The machine we used to train the model mounts an Intel i7-11850H, an NVIDIA RTX A2000 and 32 GB of RAM. The operating system was Windows 10 and PyTorch 2.0 [PGM+19] was used as a Machine Learning framework to build the project.

5.3.3 Evaluation metrics

The scope of our evaluation spans multiple tasks, beginning with a focus on the pre-training stage to enhance the model's initial learning and following with an application to driver recognition system.

Pre-training is crucial for processing and interpreting essential data, which subsequently improves the model's accuracy in identifying driving patterns. For this stage, we applied metrics such as *mean average precision*, reported in (5.4), and Recall@k [JDS11]. The goal of Mean Average Precision (mAP) as a metric is to assess the precision of a ranking algorithm by considering the average precision across multiple queries or classes. Instead, Recall@k metric is useful for assessing the model's ability to retrieve relevant instances within the top-k recommendations.

$$\text{mAP} = \frac{1}{N} \sum_{i=0}^{n} AP_i \tag{5.4}$$

In addition to understanding the potential of our model, we are interested in assessing the quality of the learned embedding in the driver recognition task. *K-nearest neighbour* (kNN) is known for its role as a machine learning algorithm for classification tasks. We stacked on top of the neural network trained with triplet

Table 5.2: Triplet loss margin fine-tuning

	Metrics		
Margin	***mAP***	***Recall@1***	***Recall@5***
0.1	0.39	0.83	0.94
0.2	**0.40**	**0.83**	**0.95**
0.3	**0.40**	0.80	**0.95**
0.4	0.38	0.79	0.93
0.5	0.38	0.78	0.92

loss a kNN model to evaluate the quality of the learned representation in the task of driver recognition. The approach we used is described here:

- We embed our data points into a lower dimensional space using our model;
- For each data point, we perform a k-NN search to find the k nearest neighbours based on the learned representations;
- a majority vote on the k retrieved values is applied.

We use this model because it gives us a measure of how well the model has learned to group the data points.

5.4 Results

We decided to start our result analysis from the pre-training task. This is done to assess the quality of the representation space built by the model. The entire evaluation process has been done on S_2, this means that none of the drivers used in the training phase is part of this evaluation.

Firstly, we fine-tuned the latent space dimension and the margin parameters. Latent space dimension and triplet loss margin are important hyperparameters in machine learning models that utilize triplet loss for metric learning. Latent space dimension refers to the dimensionality of the latent space, which is the space in which the model embeds the input data. Margin instead specifies the maximum distance that has to be kept between the anchor and the positive example and the

Table 5.3: Classification results

Model	Metrics
	Acc.
Random Forest [HSS+17]	0.55
ARNet [DYY+17]	0.71
Driver2vec [YZZ+21]	0.73
D-CRNN [MMP+21]	0.75
Our	**0.83**

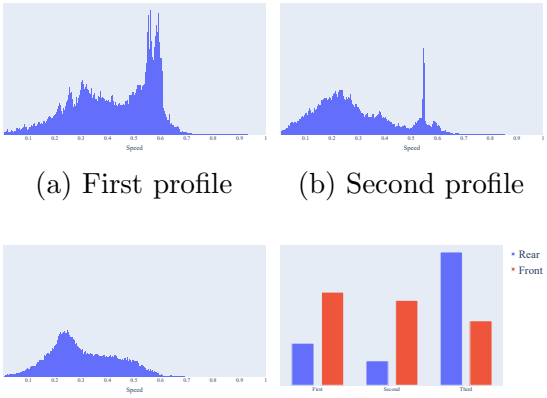
Table 5.4: K-NN results

Model	Metrics			
	Acc.	**Prec**	**Rec.**	**F1-score**
1-NN	0.82	**0.80**	**0.79**	**0.79**
3-NN	**0.83**	0.79	**0.79**	0.78
5-NN	**0.83**	0.79	0.76	0.77
8-NN	0.82	0.77	0.76	0.77
15-NN	0.81	0.77	0.73	0.75

minimum distance that has to be retained between the anchor and the negative example. Both latent space dimension and triplet loss margin can significantly impact the model's performance. As we can see from Table 5.1 and Table 5.2 we found our best setup with a latent space dimension equal to 64 and a margin distance of 0.2. The model's performance is characterized by moderate accuracy, with a 0.4 for mAP across all possible rankings. The model has shown interesting results at retrieving the most relevant item, with a value of 0.83, and performs consistently in identifying one of the top five relevant items, with a value of 0.95. This means that the retrieval system can return relevant documents, but it is more likely to return the most relevant document when the query contains some specific information. The model's performance demonstrates promising potential, but further refinements could lead to even better precision.

We then studied the ability to enhance spatial information from the built representation space for the classification task. We trained on top of the learning space a KNN model to classify who is driving in a given window. The results on different K values are reported in Table 5.4. This solution has shown to be resilient even to larger values of K. We cannot see any drop in the quality of the results. These values on precision, recall and F1-score underline the model's capacity to effectively capture a significant percentage of actual positive instances, demonstrating its ability to discern motorcycle drivers. Furthermore, as reported in Table 5.3 our model set a significant step ahead compared to other solutions. These demonstrate how our model can build a representation space that describes driver behavior features even on unknown drivers at training time.

5.5 Driving style Analysis



(a) First profile (b) Second profile

(c) Third profile (d) Braking

Figure 5.6: Data distribution for speed and braking based on the driver styles extracted by K-Means

We are interested in studying the properties on the road data of our model. We decided then to extract from the same data lake of S_1 another dataset called S_3. Data presented in $S3$ are not part of the training set. We performed K-Means on the dimensionality reduction produced by our model. We set $K = 3$. We are reporting in Figure 5.6 the speed distribution for each driver profile. As we can see from Figure 5.6a this user profile is definitely the more aggressive. In fact, the data distribution is jagged, and the user prefers high speeds. Instead, the profile showed in Figure 5.6b can be considered a "moderate" driver. Its curve of data is smoother and, as demonstrated from the peak of data focused around 0.5, he is more inclined to use cruise control. Looking at the third profile, we can observe it is "quieter" than the others two. The curve is really smooth, and he is pushing moderately to higher speed. In

Figure 5.6d are reported the braking events for each driving profile. It is interesting to note that the first two profiles are more prone to use the front brake, whereas the third one uses the rear more intensively. On a motorcycle, there is a tendency to brake with the front brake, as it is the most powerful brake and allows you to reduce speed more quickly and safely. The rear brake, on the other hand, is used to stabilize the bike during cornering. Furthermore, it allows you to brake more easily on slippery surfaces than the front brake. Then, the third profile's intensive use of the rear brake demonstrates a tendency not to trust the motorcycle's stability and to be less confident.

Chapter 6

Contextual Contrastive Learning for Rider Behavior Analysis

In Chapter 5 we introduced a method using triplet loss to sift through massive datasets and extract distinct "behavioral fingerprints" for each user, allowing us to identify individual riders. In this chapter the analysis shifts from who is driving to how they are driving in a sequential context.

While the previous chapter successfully demonstrated that triplet loss and a weakly supervised approach can create robust representations for driver identification, it primarily treated riding segments as independent samples. This approach, while effective for classification, does not fully capture the temporal dynamics and causal relationships inherent in a continuous riding experience.

To gain a deeper understanding of rider behavior, it is essential to model the flow and context of a journey. A rider's actions are not isolated events; they are part of a continuous stream of decisions and reactions influenced by preceding maneuvers and anticipating future road conditions. This chapter, therefore, introduces a more advanced contextual contrastive learning framework designed to capture these sequential patterns.

By modeling the relationships between adjacent time segments, this approach moves beyond static behavioral traits to learn representations that encode the nuances of how a rider transitions between different states. This allows for a more granular analysis of riding styles, such as identifying patterns of acceleration,

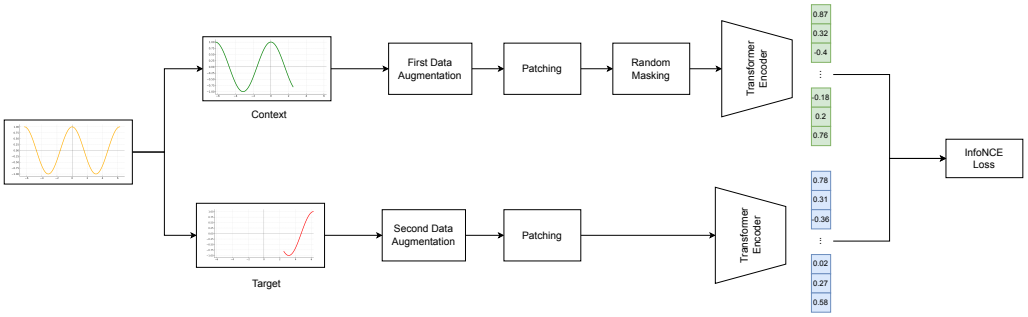


Figure 6.1: Overview of the contextual contrastive learning framework. Each riding data window is split into a context and a target segment. Independent augmentation pipelines are applied, including patching, random masking, and signal transformations. Both segments are encoded using a shared Transformer encoder, and the resulting embeddings are optimized using the InfoNCE loss to align contextually similar pairs while separating dissimilar ones.

braking, and gear shifting that unfold over time.

This work has been accepted in the proceedings of the 2025 IEEE Intelligent Transportation Systems Conference (ITSC 2025) [PSAG25a].

6.1 Methodologies

6.1.1 Framework

The proposed framework utilizes a contrastive learning approach inspired by [ERC+21], particularly in its strong and weak augmentations to create correlated views of the input time-series data. In Figure 6.1, the framework pipeline is shown.

Our pipeline begins by splitting each riding data window into a `context` segment and a subsequent `target` segment. These views undergo separate data augmentation processes. While [ERC+21] defines specific strong (e.g., permutation-and-jitter) and weak (e.g., jitter-and-scale) augmentation strategies, our approach differs in implementation. In our setup, the "context" view receives the 'strong' augmentation, which consists of applying a first data augmentation, followed by the patching process, and then some patches are randomly masked. The "target" view receives the "weak" augmentation, involving only one data augmentation

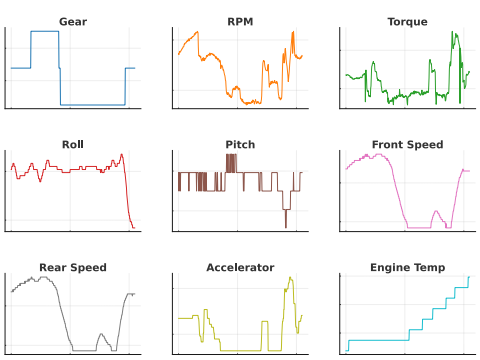


Figure 6.2: Time Series Visualization of Vehicle Dynamics. Each subplot represents a distinct input signal.

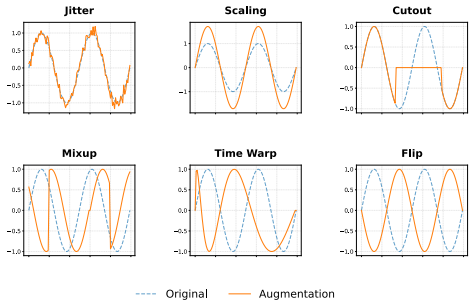


Figure 6.3: Visualization of data augmentation techniques used in our framework for time series data. The plot shows the effect of various augmentations.

technique followed by the patching process. The patching process, where the time series is divided into patches, is required for the PatchTST model [NNSK22], the model we chose as the encoder. A grid search is performed to identify the optimal combination of these data augmentation techniques for the specific time series data.

Following augmentation and patching, both views are fed into the PatchTST (Patch Time Series Transformer) encoder model to extract high-dimensional latent representations. The goal is to learn representations where different views (augmentations/contexts) of the same sample are similar, while views from different samples are dissimilar. This is achieved using an InfoNCE loss function, contrasting the representations from the two augmented views. This contrastive process encourages the model to learn robust and discriminative features from the time-series data.

6.1.2 Evaluation metrics

To quantitatively assess the quality and effectiveness of the learned embeddings in capturing meaningful patterns, we employed two standard information retrieval metrics: Recall at k (specifically for k=1, 5, and 10) and Normalized Discounted Cumulative Gain at k (specifically for k=10). These metrics evaluate the model's ability to retrieve and rank the contextually appropriate target segment for a given query segment.

Recall@1, *Recall*@5, and *Recall*@10 evaluate whether the specific target segment t corresponding to a given context query segment q is retrieved within the top-k results. In our contrastive setup, where there is typically a single designated target t for each context q, the metric is calculated as shown in Eq. 6.1, where $\text{Ret}_k(q)$ is the set of top-k retrieved segments for query q.

$$\text{Recall@}k = \begin{cases} 1 & \text{if } t \in \text{Ret}_k(q) \\ 0 & \text{if } t \notin \text{Ret}_k(q) \end{cases} \tag{6.1}$$

We report this for k=1, 5, and 10 to assess the model's sensitivity in identifying the correct target segment at different retrieval depths.

NDCG@10 measures the ranking quality within the top 10 results, assigning higher scores if the target segment t is ranked higher. It normalizes the cumulative gain obtained from the ranking against the ideal ranking score.

$$\text{NDCG@10} = \frac{1}{\text{IDCG@10}} \sum_{i=1}^{10} \frac{\text{rel}_i}{\log_2(i+1)} \tag{6.2}$$

The used formula is shown in Eq. 6.2, where rel_i is the relevance score (1 if the segment at rank i is the target t, 0 otherwise), and IDCG@10 is the Discounted Cumulative Gain score of the ideal ranking (effectively 1 if the target t exists and is relevant within the top 10 scope), ensuring normalization between 0 and 1. This metric accounts for the position of the retrieved target segment, rewarding higher ranks more.

Together, Recall@k and NDCG@10 provide a comprehensive evaluation, assessing both the successful retrieval and the ranking quality of the learned representations for identifying contextually similar rider behaviors.

6.2 Experimental Setup

6.2.1 Dataset

The dataset used in this article is based on a multivariate time series dataset obtained from Ducati Motor Holding. This dataset comprises sensor measurements

Table 6.1: Aggregated Best Metrics by Embedding Size

Embedding	Recall@10	Recall@30	Recall@50	NDCG@10
16	0.451	0.671	0.762	0.266
32	0.658	0.845	0.906	0.438
64	0.846	0.950	0.973	0.623
128	0.912	0.973	0.984	0.725
256	0.967	0.989	0.994	0.849
512	0.976	**0.993**	**0.996**	0.880
1024	**0.977**	0.992	0.995	**0.884**
2048	0.898	0.963	0.977	0.728

collected from instrumented motorcycles operating under real-world conditions. We selected a set of nine sensors, shown in Figure 6.2, each recording at a sampling frequency of 10 Hz. The data collection spanned two years and incorporated contributions from 18 riders, capturing inter-subject variability in driving patterns. The recordings were geographically sampled across diverse zones within Northern Italy. A methodological control was implemented during dataset partitioning: training and test subsets were constructed using data originating from mutually exclusive sets of motorcycles. This strict separation ensures the assessment of model generalization capabilities independent of specific vehicle characteristics. Following segmentation of the time series into analysis windows — where each window has a total length of 600 timesteps, structured as a 420-timestep context segment followed by a 180-timestep target segment — the finalized dataset includes a training partition of 170,000 windows and a test partition containing 20,000 windows, reserved exclusively for final model performance evaluation.

6.2.2 Training configuration

The model processes input windows of 600 time steps for each of the 10 time series channels. Each window is divided into a context segment of 420 timesteps (70%) and a target segment of 180 timesteps (30%). As the PatchTST model requires, the time series within these segments are further divided into patches of length 20. During the strong augmentation applied to the context view, a random masking strategy masks 75% of the context patches.

The PatchTST architecture consists of 2 encoder layers, utilizing eight attention heads per layer. The feedforward network dimension within each layer is 256, and a dropout rate of 0.1 is applied for regularization. An ablation study was conducted to determine the optimal configuration for the model's embedding size and the training batch size used in the experiments.

6.3 Results

6.3.1 Ablation study

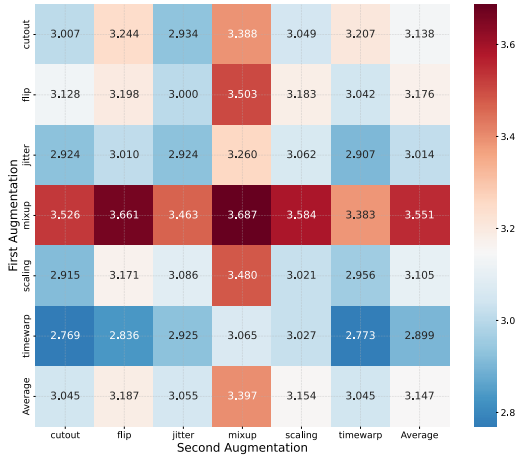


Figure 6.4: Heatmap illustrating the Test Loss obtained from combining data augmentation techniques. In this plot lower is better. Columns represent augmentation applied on the context stream. Rows represent the augmentation applied on the target branch. The color intensity reflects the average loss value.

In this section, we investigate the impact of the design choices in our proposed framework, focusing on (1) the combination of strong (context) and weak (target) augmentation strategies, and (2) the effect of hyperparameter selections such as embedding dimension and batch size. We aim to highlight how each component contributes to model performance and identify the most effective configuration for robust time-series representation learning.

Because our contrastive framework relies on creating complementary `context` and `target` views, we conducted experiments to determine which augmentations lead to the best downstream performance. Figure 6.3 summarizes the test losses for all pairwise combinations of strong augmentations (rows) and weak augmentations (columns). Each cell reports the resulting average test loss after training, where lower values indicate better performance.

From these results, applying *timewarp* as the strong augmentation (for the context view) and *cutout* as the weak augmentation (for the target view) achieved the best performance, yielding a test loss of 2.769. By reshaping the temporal structure, we hypothesize that timewarp exposes the network to large-scale temporal variations, while cutout enforces local invariance by masking time series segments. This combination balances global and local perturbations, ultimately enhancing the learned representations.

Alongside data augmentation strategies, we also evaluated the influence of model hyperparameters. Figure 6.5 depicts a contour plot (in log scale) of performance across varying embedding dimensions and batch sizes. A "spot" region exists where moderately large embedding dimensions and batch sizes converge to consistently high performance (measured by the NDCG@10 metric). Beyond that region, performance tends to plateau or degrade slightly, implying diminishing returns for overly large hyperparameter values. The best value is obtained with batch size at 4096 and embedding dimensionality at 1024.

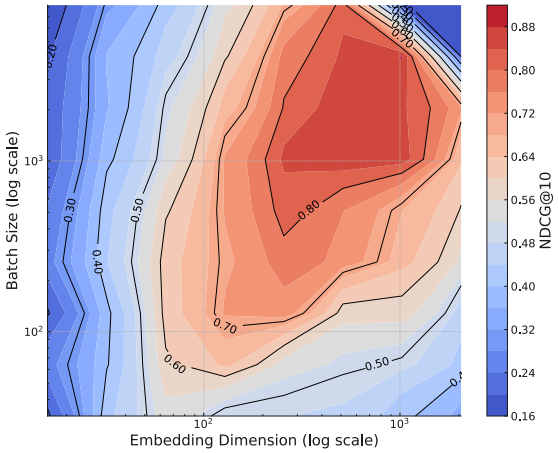


Figure 6.5: This contour plot visualizes NDCG@10 performance as a function of two hyperparameters — embedding dimension and batch size — both measured on a logarithmic scale. In this plot higher is better.

We report additional metrics in Table 6.1. The results confirm that while increasing the embedding dimension from small to moderately large values yields clear improvements, the gains taper off once a sufficiently expressive feature space is reached. For instance, going from 64 to 256 dimensions leads to consistent gains across metrics, whereas moving beyond 256 typically shows only marginal improvement.

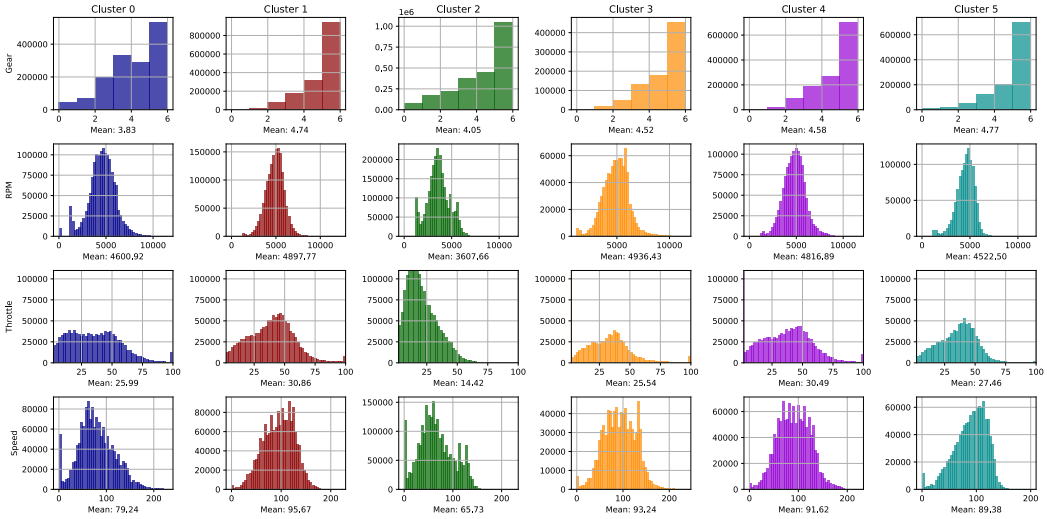


Figure 6.6: Variable distributions for the six behavioural clusters identified via K-Means clustering of embeddings learned by our framework. Each column represents a cluster (0-5), and rows show histograms for Gear, RPM, Throttle (%), and Speed (units). Mean values are indicated below each distribution.

6.4 Discussion

Employing our framework with the optimal configuration determined through ablation studies — specifically, a batch size of 4096 and an embedding size of 1024 — we generated embeddings from the motorcycle time series data that effectively captured distinct operational patterns. Subsequent application of the K-Means clustering algorithm [BN06] to these embeddings revealed six distinct behavioural clusters, as visualized by the distributions of Gear, RPM, Throttle (%), and Speed (units) in Figure 6.6.

Cluster 0 captured dynamic riding conditions, displaying a greater mix of gears skewed lower (mean 3.83) alongside moderate speeds (mean 79.24 units) and throttle levels (mean 25.99%), potentially representing riding in variable conditions such as urban environments or mixed secondary roads. Following this, Cluster 1 represented the most performance-oriented behaviour, exhibiting the highest average speed (95.67 units) and high RPM (mean 4897.77), coupled with high gear usage (mean 4.74) and moderate throttle (mean 30.86%), indicative of active acceleration or sustained high-speed riding.

In contrast, Cluster 2 identified phases characterized by low speed (mean 65.73 units), low RPM (mean 3607.66), and minimal throttle (mean 14.42%), indicative of deceleration, coasting, or very low-speed maneuvering (mean gear 4.05). Cluster 3 represented cruising at moderate speeds, maintaining high gear usage (mean 4.52) but at average speeds of 83.24 units and with lower throttle input (mean 25.54%), possibly reflecting operation on secondary roads or steady non-highway travel. Similarly representing cruising, Cluster 4 suggested slightly more active cruising or gentle acceleration, maintaining high gear usage (mean 4.58), high RPM (mean 4816.89), moderate throttle (30.49%), and high speed (91.62 units).

Finally, Cluster 5 shows steady-state highway cruising, characterized by operation predominantly in high gears (mean 4.77), low-to-mid RPM (mean 4522.50), minimal throttle (mean 27.46%), and speeds averaging 89.38 units. Taken together, the clear separation of these diverse operating modes demonstrates the effectiveness of the contrastive learning framework in extracting meaningful behavioral signatures from the complex, multivariate motorcycle time series data.

Chapter 7

A Dual-Encoder framework for Driving Behavior and Mission Profiling

In Chapter 5 and Chapter 6, the analysis progressed from identifying individual riders via static "behavioral fingerprints" to modeling the temporal flow of their actions using contextual contrastive learning. While these approaches successfully captured who is driving and the sequential patterns of how they drive, they analyzed the vehicle's data as a single, unified stream. This overlooks the distinct, yet fundamentally linked, relationship between a rider's specific inputs and the motorcycle's resulting dynamic responses.

To achieve a deeper understanding of this interplay, this chapter introduces a dual-encoder framework. This architecture is designed to create a joint embedding of two separate domains: rider behavior and vehicle state. By processing signals like throttle and gear changes in one encoder and vehicle dynamics like speed and lean angle in another, the framework learns to map the direct relationship between a rider's actions and the motorcycle's reaction.

Inspired by methodologies like CLIP (Contrastive Language-Image Pre-training), InfoNCE loss function aligns the corresponding pairs of rider actions and vehicle states in a shared latent space. This explicit modeling of the cause-and-effect relationship between the two data streams provides a powerful foundation for

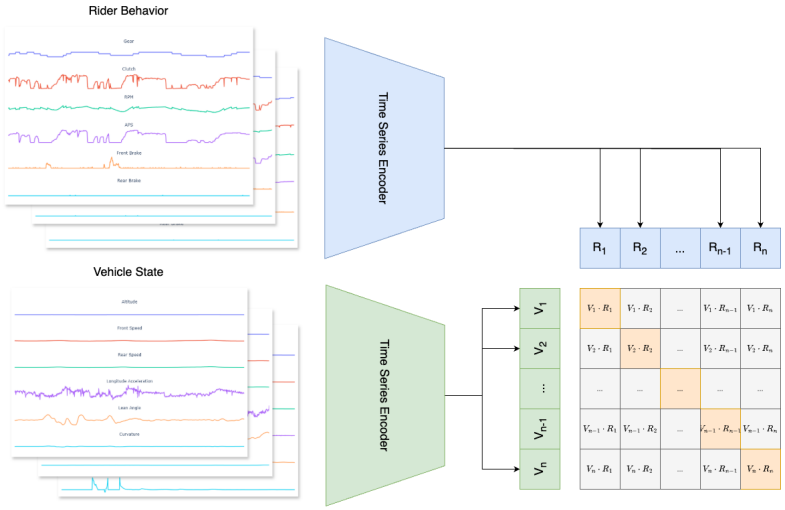


Figure 7.1: Dual-Encoder Architecture for Joint Embedding of Rider Behavior and Vehicle State Time Series. The framework employs two Time Series Encoders to process rider behavior (top) and vehicle state (bottom). Each encoder generates a set of latent representations. A contrastive loss function is applied to the resulting joint embedding matrix.

robust mission profiling and a more comprehensive characterization of driving behavior.

This work has been accepted in the proceedings of the 2025 INNS International Joint Conference on Neural Networks (IJCNN 2025) [PSAG25b].

7.1 Methodology

7.1.1 Framework Design

The proposed framework, illustrated in Figure 7.1, adopts a dual-encoder architecture inspired by Contrastive Language-Image Pre-training (CLIP) [RKH+21] principles to achieve a joint embedding of rider behavior and vehicle state. This architecture comprises two distinct Time Series Encoders, each dedicated to processing multivariate time-series data representing one of these two domains. As depicted in the upper portion of Figure 7.1, the Rider Behavior Encoder receives inputs including Gear, Clutch, throttle position, RPM, and brake pressures, capturing the rider's control actions. Concurrently, the Vehicle State Encoder, shown in the

lower portion of Figure 7.1, processes time-series data speed, Altitude, longitudinal acceleration, lean angle and road curvature, thus characterizing the motorcycle's dynamic response.

During the training, each encoder independently generates a sequence of latent representations for each time series window within a batch. In Figure 7.1, these are denoted as R_1 to R_n for rider behavior and V_1 to V_n for vehicle state, where the subscript indicates different windows from the input batch. These representations are then combined to form a joint embedding matrix. Within this matrix, a contrastive loss function is applied, designed to minimize the distance between embeddings of corresponding rider behavior and vehicle state windows from the same point in time (represented by the diagonal elements in the matrix visualization), while maximizing the distance between embeddings of non-corresponding windows (off-diagonal elements).

This training paradigm effectively forces the network to learn a unified representation space where the relationship between rider actions and the resulting vehicle dynamics is explicitly encoded, providing a robust foundation for mission profile analysis and characterization. Adopting a dual-encoder structure - inspired by CLIP - allows for the effective fusion of heterogeneous data streams, providing a novel approach to understanding the complex interplay between the rider and the motorcycle.

7.1.2 Feature Importance

This study uses GradientSHAP (SHapley Additive exPlanations) [LL17] values to quantify feature importance. Specifically, we calculate feature importance as the averaged sum of absolute SHAP values across the test set. We used this methodology to capture the temporal dynamics inherent in time series data.

For a given feature x_i, its importance I (x_i) is defined in Eq. 7.1 where N denotes the total number of samples in the test set, T represents the number of time steps in each time series and $\text{SHAP}_i^{(s,t)}$ is the SHAP value of feature x_i for sample s at time step t.

$$\mathrm{I}\left(x_i\right) = \frac{1}{N}\sum_{s=1}^{N}\left(\sum_{t=1}^{T}\left|\mathrm{SHAP}_i^{(s,t)}\right|\right) \tag{7.1}$$

This metric is computed by first calculating the absolute SHAP values for each feature at every time step within each sample, ensuring that both positive and negative contributions are accounted for without cancellation. The absolute values are then summed across all time steps for each sample, capturing the cumulative influence of the feature throughout the temporal sequence. Finally, these sums are averaged across all samples in the dataset, providing a normalized measure of feature importance that facilitates comparison between different features irrespective of the dataset's size or the temporal length of the time series.

The rationale for using the averaged sum of absolute SHAP values is multifaceted. Firstly, summing the absolute SHAP values across time steps for each sample effectively captures the temporal dynamics of feature contributions, reflecting both transient and sustained impacts on the model's predictions. This is particularly pertinent in time series analysis, where the relevance of features may fluctuate or persist across different temporal contexts. Secondly, averaging these sums across all samples normalizes the feature importance scores, mitigating biases introduced by differing numbers of samples. This normalization ensures that the importance metrics are comparable across features, providing a standardized measure of their general significance within the model.

7.2 Experimental Setup

7.2.1 Dataset

The dataset used in this study was acquired using a data logger connected to a motorcycle's Electronic Control Units (ECUs) via the Controller Area Network (CAN) bus. The data logger passively monitors and records the data transmitted over the CAN bus. Rider inputs include gear, throttle position, rpm and brake pressures, while vehicle dynamics are represented by speed, longitudinal acceleration, lean angle, and altitude.

The raw data were sampled initially by the logger at 100 Hz, providing fine-

Metric	32	64	128	256
Recall@1	0.64	0.83	0.93	**0.94**
Recall@3	0.79	0.91	0.96	**0.97**
Recall@5	0.83	0.93	**0.97**	**0.97**
Recall@10	0.88	0.95	**0.98**	**0.98**
Recall@15	0.90	0.96	**0.98**	**0.98**
MRR	0.73	0.87	0.95	**0.96**

Table 7.1: Performance of the dual-encoder framework with varying embedding dimensions. Metrics include Recall at different levels and Mean Reciprocal Rank (MRR). Results demonstrate improved performance with increasing embedding dimensions.

grained temporal resolution. However, for this research, the dataset was downsampled to 10 Hz. The various acquisitions were windowed to have fixed-size inputs. Each window encompasses 1000 consecutive data points, corresponding to 100 seconds of riding data at the downsampled rate. This window size was chosen to balance the need for capturing sufficient contextual information to characterize riding behavior and mission profile while avoiding excessively long sequences that could introduce noise or irrelevant information.

The data was collected and tested under various riding conditions, encompassing diverse scenarios and maneuvers, to ensure the generalizability of the findings. The dataset was partitioned into a training set and a test set. The training set comprises $\approx 1060\ h$ hours of riding data, providing a substantial basis for model training. We sampled 100.000 windows from this data source. The test set consists of $\approx 196\ h$ hours of data reserved for evaluating the performance and generalization ability of the trained model on unseen data. We sampled 30.000 windows from this data lake.

7.2.2 Implementation Details

The dataset used in this study was acquired using a data logger connected to a motorcycle's Electronic Control Units (ECUs) via the Controller Area Network (CAN) bus. The data logger passively monitors and records the data transmitted over the CAN bus. Rider inputs include gear, throttle position, rpm and brake pressures, while vehicle dynamics are represented by speed, longitudinal acceleration, lean angle, and altitude.

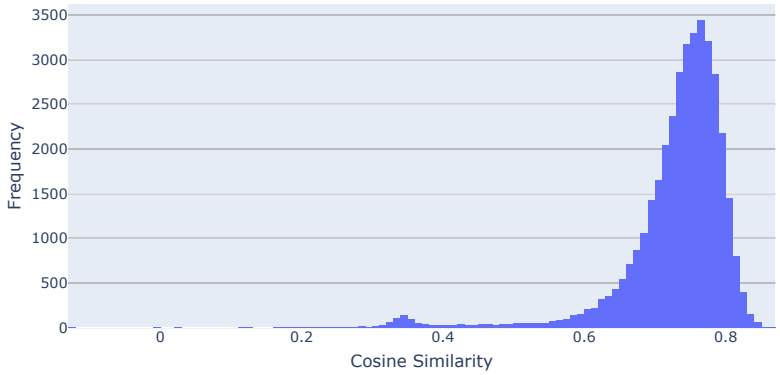


Figure 7.2: This figure visualizes the distribution of cosine similarity scores between rider behavior embeddings and vehicle state embeddings.

The raw data were sampled initially by the logger at 100 Hz, providing fine-grained temporal resolution. However, for this research, the dataset was downsampled to 10 Hz. The various acquisitions were windowed to have fixed-size inputs. Each window encompasses 1000 consecutive data points, corresponding to 100 seconds of riding data at the downsampled rate. This window size was chosen to balance the need for capturing sufficient contextual information to characterize riding behavior and mission profile while avoiding excessively long sequences that could introduce noise or irrelevant information.

The data was collected and tested under various riding conditions, encompassing diverse scenarios and maneuvers, to ensure the generalizability of the findings. The dataset was partitioned into a training set and a test set. The training set comprises $\approx 1060\ h$ hours of riding data, providing a substantial basis for model training. We sampled 100.000 windows from this data source. The test set consists of $\approx 196\ h$ hours of data reserved for evaluating the performance and generalization ability of the trained model on unseen data. We sampled 30.000 windows from this data lake.

In this work, we developed a dual-encoder framework inspired by CLIP, leveraging a time-series transformer model to process data with high temporal granularity. We used PatchTST [NNSK22] as backbone for the framework training. The model integrates patch-based processing with Transformer architectures, providing a lightweight approach to time-series analysis. The patch length, set to 30 with a

stride of 30, controls the temporal granularity of input sequences, balancing the model's ability to capture both short- and long-term dependencies. The embedding dimension, a crucial parameter for representation richness, was evaluated with values of 32, 64, 128, and 256 in an ablation study.

The HuggingFace [WDS+19] implementation of PatchTST was utilized, configured with three attention layers, 16 attention heads, a feed-forward dimension of 256, and dropout rates of 0.2 for both general and head-specific layers. The model uses GELU as the activation function and incorporates sinusoidal positional encodings to enhance temporal structure representation. The context length was set to 1000 to handle extensive temporal dependencies.

Optimization was performed using AdamW, with an initial learning rate of $1 \cdot 10^{-5}$ and a cosine scheduler applied over two million iterations. Training was conducted for 1.000.000 iterations on a machine equipped with an Intel i7-11850H processor, 32 GB of RAM, and an Nvidia RTX 3060 GPU with 12 GB of VRAM, using PyTorch 2.0 [PGM+19] on Windows 10.

7.2.3 Evaluation Metrics

The proposed dual-encoder framework's performance is evaluated using Recall@k and Mean Reciprocal Rank (MRR), widely adopted in information retrieval and recommendation systems. Recall@k, defined in Eq. 7.2, evaluates the proportion of relevant items retrieved within the top-k results, where R is the set of relevant items, and K is the set of retrieved items.

$$Recall@k = \frac{\text{Number of relevant items in top } k}{\text{Total number of relevant items}} \tag{7.2}$$

MRR measures the quality of rankings by taking the reciprocal of the rank of the first relevant item retrieved, averaged across all queries. The formula is presented in Eq. 7.3, where Q is the total set of queries, and r_i is the rank of the first relevant item for the $i-th$ query.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{r_i} \tag{7.3}$$

These metrics comprehensively evaluate retrieval accuracy and ranking quality,

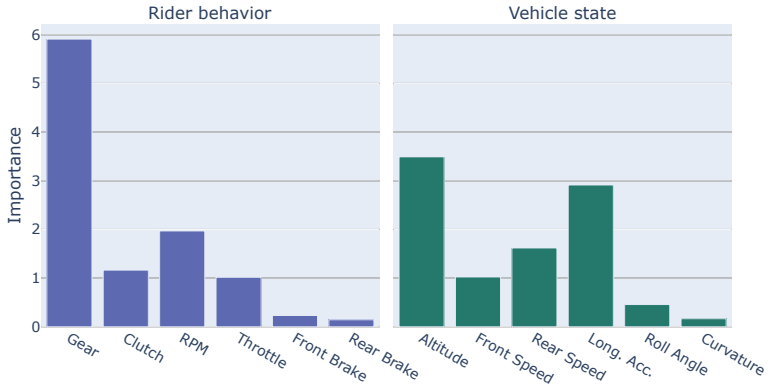


Figure 7.3: The bar plot illustrates the relative importance of features representing rider behavior and vehicle state. Features like gear and altitude show the highest contributions, indicating their role in encoding the interaction between rider inputs and motorcycle dynamics.

ensuring a robust analysis of the framework's performance.

7.3 Results

7.3.1 Framework Performance

The performance of the proposed dual-encoder framework was systematically evaluated across different embedding dimensions (32, 64, 128, 256) to assess its ability to characterize driving behavior and mission profiling. Table 7.1 summarizes that increasing the embedding dimension consistently improves performance across all metrics. Notably, Recall@1 increased from 0.64 at 32 dimensions to 0.94 at 256, reflecting a significant enhancement in the model's ability to correctly match vehicle states with corresponding rider behaviors. Similar improvements were observed in Recall@3, Recall@5, and Recall@15, with all metrics demonstrating robust convergence at higher embedding sizes. The Mean Reciprocal Rank (MRR) followed a similar trend, rising from 0.73 to 0.96, further underscoring the framework's capacity to rank relevant pairs accurately in the embedding space.

Complementing these results, the cosine similarity distribution - shown in Figure 7.2 - provides qualitative insights into the framework's embedding performance. Additionally, the joint embedding space visualized in Figure 7.4 demonstrates

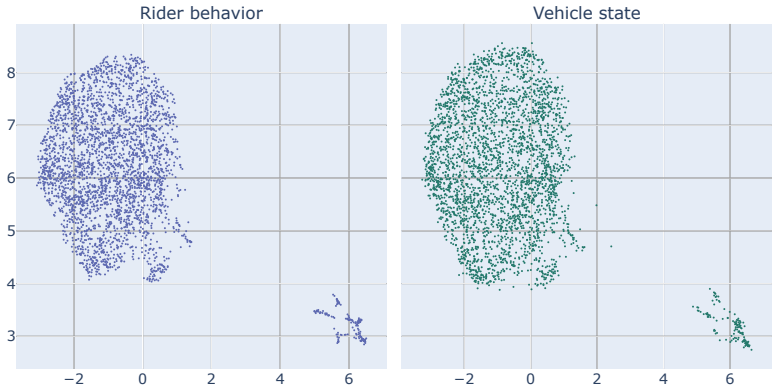


Figure 7.4: UMAP [MHM18] visualization of the joint embedding space for rider behavior and vehicle state.

the alignment of rider behavior and vehicle state embeddings, illustrating their clear separation and meaningful relationships in the latent space. The distribution reveals a pronounced density of high similarity values, indicating that the model effectively aligns embeddings of corresponding vehicle states and rider behaviors while maintaining sufficient separation from mismatched pairs. Interestingly, an analysis of low cosine similarity scores uncovered that these scores are often associated with specific operational conditions, such as the initial stages of starting the motorcycle or scenarios where the rider's interaction with the vehicle is minimal. These conditions are characterized by lower variability in rider inputs and vehicle dynamics.

The ability to detect and highlight embeddings with lower cosine similarity provides a mechanism for identifying potential anomalies or atypical riding stages. For example, low interaction states, such as prolonged idling or transitions during startup, could serve as markers for detecting unusual or unexpected behaviors. This capability positions the dual-encoder framework as a tool for characterizing mission profiles and a potential anomaly detection method, further expanding its applicability in enhancing safety and performance monitoring.

7.3.2 Feature importance

The feature importance analysis calculated using GradientSHAP is visualized in Figure 7.3. It reveals insights into the dual-encoder framework's ability to capture

key aspects of rider behavior and vehicle state. For rider behavior, gear and RPM exhibit the highest importance scores, suggesting their dominant role in encoding the rider's operational inputs. The relationship between these features is particularly noteworthy; gear and RPM are intrinsically correlated, as gear changes directly influence engine speed. This correlation reflects the framework's sensitivity to the mechanical interplay between rider control and engine response, a fundamental aspect of motorcycle dynamics.

In vehicle state, altitude stands out with the highest importance score, emphasizing the model's sensitivity to environmental and terrain-related variations. This indicates the framework's ability to encode contextual factors for capturing the motorcycle's operational scenarios. Longitudinal acceleration is the second most important feature, reflecting the dynamic forces experienced by the vehicle during acceleration and braking phases. Furthermore, the relative importance of rear speed and front speed, although not as high as other features, may still contribute to a consistent representation of vehicle state, as these speeds are generally correlated during regular operation.

This feature importance analysis validates the relevance of the selected inputs and underscores the framework's capability to focus on the most meaningful aspects of rider-vehicle interaction. Moreover, the high importance scores for key features highlight the dual-encoder framework's potential for applications like rider profiling and detecting deviations from normal operating conditions. By focusing on these relationships, the dual-encoder framework represents the riding process, capturing direct effects (e.g., gear influencing RPM) and contextual adjustments (e.g., altitude impacting longitudinal acceleration).

7.3.3 Cluster analysis

We performed the clustering analysis using k-means with $k = 10$. While the model incorporates a broader set of features, for ease of visualization, in Figure 7.5 we present the four with the highest importance: gear, RPM, altitude, and longitudinal acceleration.

Notably, Profile 3 represents the start engine phase or idling. This is evidenced by its sharply concentrated RPM distribution near the lower range, indicative of

Figure 7.5: Distributions of gear, RPM, altitude, and longitudinal acceleration across the ten clusters obtained using k-means clustering. The displayed features were selected based on the high-importance scores to provide insights into the interaction between rider behavior and vehicle state.

minimal engine activity during idle, its gear distribution predominantly fixed at neutral or the lowest gear, and its flat, narrowly distributed longitudinal acceleration values, reflecting the absence of significant vehicle movement. These features underscore the model's capability to isolate and identify specific operational phases, such as idling.

In contrast, other profiles exhibit markedly different shapes in their distributions. Profiles 1, 4, and 7 show bell-shaped RPM distributions extending to higher ranges and a wider spread of gear usage, indicating dynamic riding patterns with frequent gear shifts and significant vehicle motion typical of highway or open-road conditions. The longitudinal acceleration in these profiles reflects consistent braking and acceleration phases, as seen in its broader, more symmetric distribution. Conversely, Profiles 2 and 5, emphasizing lower to mid-range RPMs and gears, reflect urban or stop-and-go traffic scenarios, where gear changes are less frequent but correlate with moderate bursts of longitudinal acceleration. Profiles like 6 and 10 exhibit features with slightly skewed RPM distributions and mid-range gear usage, suggesting transitional riding styles or mixed driving conditions.

These profile variations emphasize the framework's ability to capture granular patterns in gear, RPM, and longitudinal acceleration distributions. Such insights highlight the potential of this framework for mission profiling and rider behavior

characterization, as well as for detecting atypical operational phases like idling or unexpected transitions.

Chapter 8

Trajectory-Embedded Matryoshka Representation Learning

In the previous chapter, we introduced a dual-encoder framework to create a joint embedding space for rider behavior and vehicle state. While this approach effectively models the interplay between rider and machine, using these fixed-size embeddings for similarity searches can lead to significant lookup times, especially as the dataset grows.

This chapter shifts the focus to accelerating this retrieval process by introducing Trajectory-Embedded Matryoshka Representation Learning. We adapt the methodology of [JPR+23] by incorporating a Matryoshka loss [KBR+22], which creates nested vector representations that maintain strong semantic capabilities even when truncated. This unique structure enables a two-stage retrieval pipeline. In the first stage, a smaller, lower-dimensional version of the embedding is used to narrow down the search space to a candidate subset quickly. In the second stage, the full-dimensional representation is used to find the most similar match within this much smaller subset, drastically reducing the overall lookup time without compromising performance. This chapter evaluates the effectiveness of this approach in optimizing the trade-off between search speed and accuracy.

This work has been published in the proceedings of the 34rd European Symposium on Artificial Neural Networks (ESANN 2025) [PGG25].

8.1 Methodology

We adapted the methodology of Jiang et al. [JPR+23], embedding spatial and temporal information with a modified loss function using the Matryoshka technique to optimize multiple output dimensions. The model performance was evaluated in relation to embedding dimensionalities and memory usage.

8.1.1 Spatial-Temporal architecture

Jiang et al. [JPR+23] introduced *"START"*, an architecture to encode temporal (time of the day) and spatial information (trajectory information and road network) jointly. The architecture encodes trajectory information by adding time-of-day and day-of-week information to the representation. The architecture uses a graph neural network to convert the road network into road representation vectors. To support multiple downstream tasks, the work implemented two different losses:

- $\mathcal{L}^{mask}$: *Span-Masked Trajectory Recovery*, where part of the input road network is masked, the model tries to recover it. This loss captures co-occurrence relationships between roads and contextual information of the road network.

- $\mathcal{L}^{con}$: *Trajectory Contrastive Learning*, where similar vectorial representations are brought closer in latent space. This objective is meant to improve the modeling of the spatial-temporal characteristics and travel semantics.

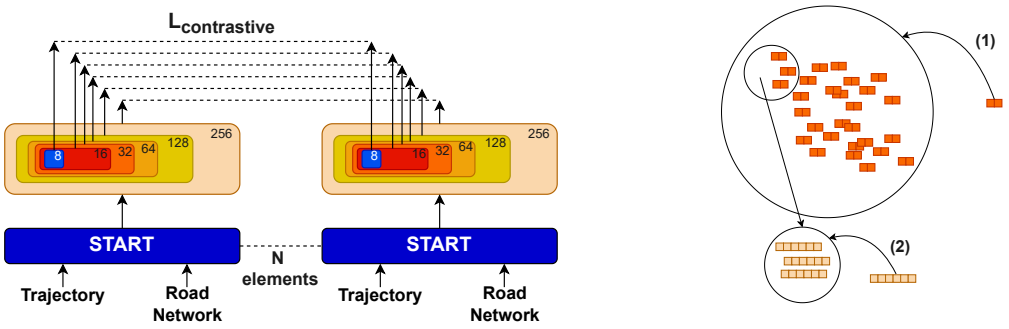
The final pre-training loss $\mathcal{L}^{pre}$ is defined by the formula in Eq. 8.1:

$$\mathcal{L}^{pre} = (1 - \lambda)\, \mathcal{L}^{mask} + \lambda\, \mathcal{L}^{con} \tag{8.1}$$

where λ acts as a weight, balancing the two tasks.

8.1.2 Efficiency analysis with Matryoshka loss

We adapted the former architecture to obtain a model capable of producing a vector that maintains good semantic representation capabilities even when truncated by



(a) $\mathcal{L}^{con}$ Contrastive loss

(b) Two-stage retrieval

Figure 8.1: Main modification to Jiang et al. [JPR+23] work. (a) shows $\mathcal{L}^{con}$, the Matryoshka contrastive loss. (b) presents the two-stage retrieval process.

the last dimensions. In these terms, we modified the *Trajectory Contrastive Learning* objective by incorporating Matryoshka loss [KBR+22].

This is illustrated in Figure 8.1a where the $\mathcal{L}^{con}$ in-batch contrastive loss used by Jiang et al. [JPR+23] aligns similar trajectories with the same dimensionality while moving away from the other representations—also with the same dimensionality—present in the batch. We have composed each batch of $N = 32$ elements, each containing a couple of similar trajectories treated as ground truth, where each trajectory is represented with the trajectory and the road network representation (see Section 8.1.1). We applied the Matryoshka loss [KBR+22] to a set d of different dimensionalities where $d = \{256, 128, 32, 16, 8\}$, obtaining $|d|$ different losses, subsequently summated in $\sum_{dim \in d} \mathcal{L}^{con}_{dim}$. We have that the final loss is represented by Eq. 8.2:

$$\mathcal{L}^{pre} = (1 - \lambda)\, \mathcal{L}^{mask} + \lambda \sum_{dim \in d} \mathcal{L}^{con}_{dim} \tag{8.2}$$

The parameter λ balances the importance of the masked contrastive loss, $\mathcal{L}^{mask}$, against the summation of contrastive losses across different embedding dimensions, $\mathcal{L}^{con}_{d}$, ensuring that the network does not overfit a specific task.

We trained the model with AdamW as an optimizer, with a batch size of 32 elements for 20 training epochs. We initially set the learning rate at $2 \cdot 10^{-4}$ with a warmup for the first four epochs and decreases using a cosine annealing

Model	Most Similar Search					Trajectory Search
	MR	MRR	HRQ1	HRQ5	HR@10	Precision
No Matryoshka	**1.3**	0.98	0.97	**0.99**	**0.99**	**0.80**
256	**1.3**	**0.99**	**0.98**	**0.99**	**0.99**	0.79
128	1.4	**0.99**	**0.98**	**0.99**	**0.99**	0.78
64	1.5	0.99	0.98	0.995	0.997	0.75
32	2.0	0.98	0.97	0.99	0.994	0.70
16	12	0.95	0.94	0.97	0.98	0.61
8	31	0.85	0.80	0.90	0.93	0.50

Table 8.1: Performance comparison of model metrics with and without Matryoshka Loss across various embedding sizes in trajectory analysis. The tasks are Most Similar Search and 5-Nearest Trajectory Search.

schedule. The λ weight is settled at 0.6. We evaluated the model with different dimensionalities on the Trajectory Similarity Search task and k-nearest trajectory search with the Euclidean distance. We assessed the efficiency by relating the model's performance to time and memory consumption.

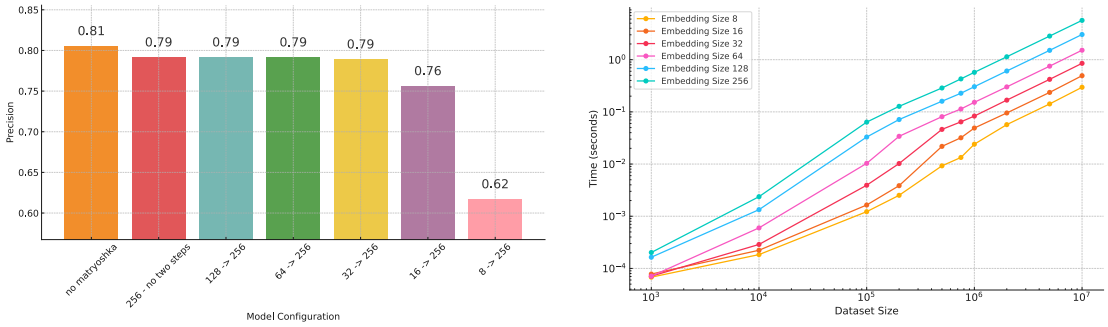
8.1.3 Two-stage retrieval pipeline

We designed a two-step retrieval (Figure 8.1b) to optimize the time search while maintaining the same model performance. We define the stages as follows: (1) First lookup phase: we use a lower dimensionality to shrink the search space to a small subset of 1000 samples. (2) Second lookup phase: we use the full dimensionality to retrieve the most similar sample out of the 1000 subset.

We conducted a grid search over the vectors' dimensionality and the subset's dimension to obtain the best-performing algorithm.

8.2 Discussion

The results presented in this study underscore the efficacy of Trajectory-Embedded Matryoshka Representation Learning (TE-MRL) in managing the demands of computational resources in trajectory data analysis. By incorporating MRL into trajectory analysis, TE-MRL facilitates a hierarchical yet computationally efficient approach to understanding spatial-temporal dynamics, as evidenced by the results



(a) Precision Two-Step Approach

(b) Execution time over dataset sizes

Figure 8.2: Performance analysis of embedding sizes using matryoshka Loss. (a) shows the precision achieved at various embedding sizes with Two-Step. (b) charts the computational time required at different dataset sizes for varying embedding sizes.

on the Porto dataset. As shown in Table 8.1, TE-MRL maintains high levels of accuracy in Most Similar Search and Trajectory Search tasks across various embedding sizes. Even at reduced dimensions (e.g., 16 and 32), TE-MRL achieves good results in terms of Mean Reciprocal Rank (MRR) and Hit Rates (HR). This adaptability is crucial for applications where computational resources are limited or where real-time data processing is required.

Figure 8.2a shows the performance of the two-stage search approach within the TE-MRL framework, focusing mainly on the precision achieved with initial embeddings of varying dimensions transitioning to a 256-dimensional space.

We showed that by employing 32 dimensions during the initial retrieval stage and subsequently transitioning to 256 dimensions for the second stage, we matched the performance of the 256-dimensional matryoshka single-stage approach with a 8× speed-up in time. This precision result underscores the efficacy of the TE-MRL framework in utilizing lower-dimensional embeddings to achieve computational efficiency without compromising the quality of the results.

Figure 8.2b demonstrates the scalability of TE-MRL across different dataset sizes and embedding dimensions. The ratio of computational time related to the size of the dataset decreases by half with each dimensionality reduction. Hence, the two-step retrieval process helps to moderate the time complexity of computation by

using smaller embeddings in the initial retrieval phase while ensuring that precision is not compromised.

Chapter 9

Optimizing the Training Diet

In the preceding chapters, the focus was on developing models to interpret complex data streams from two-wheeled vehicles. This included the dual-encoder framework in Chapter 7, which successfully created a joint embedding space to understand the interplay between rider behavior and vehicle state. While these approaches are powerful for analyzing how data represents behavior, they operate on the assumption that the entire dataset is beneficial for training.

This chapter shifts the focus from the model's architecture to the composition of the training data itself, challenging the "more data is better" paradigm. Real-world sensor data often contains imbalances and redundancies, where simply increasing data volume can hinder, rather than help, a model's ability to learn critical patterns. Inspired by advances in data-centric learning, this chapter proposes a novel optimization framework to intelligently curate training datasets for time series analysis. In this chapter we target the forecasting task in time-series domain.

Unlike methods such as CLIMB [DYF+25], which use iterative bootstrapping and proxy models primarily in the text domain, this framework takes a more direct approach. It employs a large foundational encoder to transform raw time series into meaningful, task-agnostic embeddings. These embeddings allow the dataset to be partitioned into behaviorally consistent clusters using K-Means. The core of the methodology is to treat the data composition as a hyperparameter optimization problem, where a tool like Optuna is used to find the optimal sampling allocation from these clusters to maximize the final model's performance on a specific task.

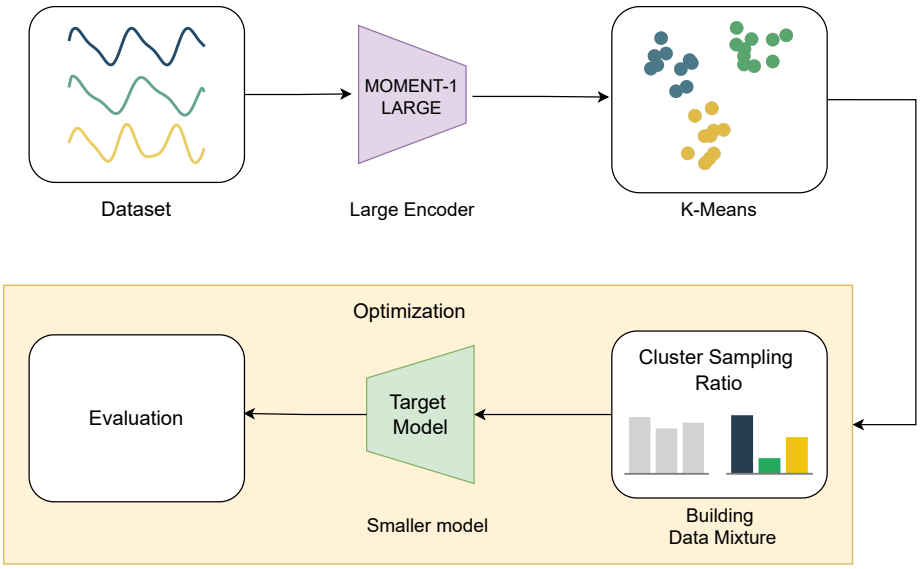


Figure 9.1: Pipeline of the training data mixture optimization. The full time-series dataset is embedded using a large pre-trained encoder and clustered via k-means into consistent groups. The search for optimal cluster sampling ratios is performed using Optuna's optimization, generating multiple candidate training sets. For each mixture, a fixed target model is trained and evaluated to guide the search toward data compositions that enhance model generalization.

This direct optimization loop creates a training "diet" specifically tailored to the model and task, avoiding the need for intermediate surrogate models.

9.1 Methodology

9.1.1 Pipeline overview

As illustrated in Figure 9.1, the process starts by using a large pre-trained encoder, `MOMENT-1 large` [GSC+24], to transform the entire dataset into dense, task-agnostic embeddings. Subsequently, these embeddings are partitioned into distinct clusters using K-Means.

The optimization process, managed by Optuna [ASY+19], is formulated as a search for the optimal set of sampling weights, $\{w_1, w_2, ..., w_k\}$, corresponding to each of the K clusters derived from the dataset. Each weight, w_k, is a continuous

hyperparameter within the search space $[0, 1]$.

For each trial within the optimization loop, Optuna proposes a vector of weights. A new training data mixture is then constructed by sampling n_k elements from each cluster C_k. The number of elements to be sampled from a given cluster is calculated as the product of the cluster's total size and its assigned weight, formally expressed as shown in Equation 9.1.

$$n_k = C_k \cdot w_k \tag{9.1}$$

The resulting data mixture is then used to train and evaluate the target model, providing a performance score that guides Optuna's subsequent search for an improved weight configuration.

9.1.2 Dataset and Preprocessing

In our study, we used the publicly available PMSM dataset [KWB21a], which provides time-series data from an electric motor operating under diverse loads. The goal is to predict four key thermal metrics — specifically the temperatures of the stator winding, tooth, yoke, and the permanent magnet — using measurements such as d- and q-axis current and voltage, motor speed, and torque. As a preprocessing strategy, we implement the one presented in [KWB21b]. However, compared to the original work, we divided the entire dataset into windows of 300 timesteps to enhance the model's capability to capture long-range relationships.

We computed Exponentially Weighted Moving Averages (EWMA) for measured and derived features using spans equal to [200, 500, 1500, 4000] to incorporate temporal context and model trends over various time scales. All features and target variables were normalized to a consistent range of zero to one using a `min-max scaler`, which was fitted only on the training data to prevent any information leakage.

The data corresponding to `profile_id=65` was used as the test set, and the data corresponding to `profile_id=[18, 39, 46, 56, 75]` was used for the validation set, ensuring our evaluation is performed on a completely unseen motor duty cycle. The remaining profiles were used for training. This results in a total $\simeq 1.1$ million of training windows. The evaluation set was composed of $\simeq 85.000$ windows and

the test set is composed by $\simeq 40.000$ windows.

9.1.3 Experiment setup

Our experimental setup is designed to effectively evaluate the data mixture optimization framework presented in Figure 9.1.

Firstly, we encode the time series into a task-agnostic embedding space using the `MOMENT-1 large` encoder. The resulting embeddings were then partitioned into 36 distinct groups using K-Means clustering, with $k = 36$ chosen through manual tuning.

Optuna proposed a set of 36 cluster weights for each trial, each a floating-point number between 0.0 and 1.0. Based on these weights, a new training data mixture was sampled. A fixed-architecture target model was then trained on this data mixture. The target model is a PatchTST [NHNSK23] with sinusoidal position embeddings, configured with an embedding dimension of 256, 8 attention heads, a patch size of 30, and a convolutional layer used for patching the input time series. We decided to use this architecture to reduce the model's training throughput.

The optimization was configured to run for 100 trials. We utilized Optuna's Tree-structured Parzen Estimator (TPE) sampler to navigate the search space and find the optimal mixture efficiently. The search was parallelized across 4 jobs to accelerate the discovery process. The study's objective was to minimize the Mean Squared Error (MSE).

Training for each trial was performed using the Adam optimizer with a learning rate of $1 \cdot 10^{-4}$ and a batch size of 1024. A learning rate scheduler was employed, featuring a linear warmup for the first 30% of the total training steps, followed by a linear decay.

The total number of training epochs was dynamically adjusted for each trial to ensure that the model consistently processed a total of 360 million training tokens, where a token is one patch passed through the model. This normalization of the training budget means that models trained on smaller, more targeted data mixtures iterate over their respective datasets more frequently.

9.2 Discussion

9.2.1 Performance

The optimization produces a new dataset 2.3 times smaller than the original full training set. The results, summarized in Table 9.1, demonstrate the effectiveness of our proposed data-mixture optimization pipeline. The PatchTST model trained on the optimized data mixture achieves the best performance across nearly all evaluation metrics. It recorded the lowest Average MSE of 1.37, a 19.41% improvement over the full dataset-trained model. We report the values presented in [KWB21a, LA22] for the baselines.

Furthermore, as shown in the Figure 9.2, our optimized data-mixture (indicated by the red star) achieves a lower MSE than the same model trained on various random percentages of the original dataset, including the full training set. This confirms that the optimized subset isn't only better than using a random subset of the same size, but also better than using any random subset.

Notably, our data-mixture approach significantly outperforms all the other models trained on the entire dataset. This highlights the benefit of our data selection strategy, which enables more efficient and targeted training. The model trained on the data mixture excels far beyond in predicting the temperature for the Yoke and Tooth, achieving the best scores for both MSE and MAE. While the MSE for the Winding component (1.33) is slightly higher than that of the model trained on the full dataset (1.31), its corresponding MAE (0.87) is the lowest among all models, indicating strong predictive accuracy. However, the LSTMs are the ones that perform best in predicting the temperature of the Permanent Magnet (PM) components, even though our solution significantly improves on PatchTST trained on the full dataset.

9.2.2 Cluster weights analysis

To gain insight into the produced data mixture, we performed an analysis of the weight distribution. The data-centric pipeline proposed in this work produced two complementary outcomes that, together, explain the observed improvement in performance of the PatchTST target model. First, the Optuna search discovered a

Table 9.1: Mean Squared Error (MSE) and Mean Absolute Error (MAE) for each model across engine components.

	MSE				MAE				Avg. MSE
Model	**Yoke**	**Tooth**	**Winding**	**PM**	**Yoke**	**Tooth**	**Winding**	**PM**	
TCN [KWB21b]	6.90	2.84	1.80	0.65	5.24	6.24	7.92	5.84	3.04
LSTM [LA22]	1.34	2.23	4.86	1.52	4.03	4.89	8.47	5.62	2.49
BiLSTM [LA22]	2.04	2.82	5.28	5.61	4.58	4.89	10.09	6.00	3.94
AttentionLSTM [LA22]	1.04	1.84	2.82	**1.17**	3.51	6.05	8.75	3.05	1.72
Our (Full dataset)	0.84	1.13	**1.31**	3.53	0.68	0.77	0.89	1.53	1.70
Our (Data-Mixture)	**0.65**	**0.91**	1.33	2.60	**0.60**	**0.68**	**0.87**	**1.32**	**1.37**

non-uniform set of sampling weights (see the radar plot in Figure 9.3a), effectively pruning entire clusters while strongly amplifying others. Second, those weights reshaped the empirical training distribution into one that is markedly less skewed than the raw data (compare the paired bar charts in Figure 9.3b). Taken together, these effects illuminate how a relatively small change in *which* samples are shown to the model can matter as much as architectural ingenuity or longer training schedules.

The optimization process produced a highly differentiated set of weights for the 36 data clusters, as illustrated in Figure 9.3a. The weights, representing the sampling proportion for each cluster, span from nearly 0.0 to almost 1.0. This variance underscores that the value of data for the given prediction task is not evenly distributed across the dataset's operational modes. Specific clusters were identified as highly valuable, receiving weights close to unity (e.g., clusters 0, 14, 15, 19, 24, 25, 31), ensuring their near-complete inclusion in the training mixture. Conversely, other clusters were assigned very low weights, indicating that their contribution to the model's ability to generalize was minimal or even detrimental.

The most striking example is Cluster 11, which received a weight approaching zero. An examination of the original data distribution (Figure 9.3b) reveals that Cluster 11 is one of the most populous clusters in the entire dataset. The optimization framework effectively determined that, despite its large volume, the information contained within this cluster was redundant or irrelevant for predicting the thermal metrics on the unseen validation profiles. By drastically down-sampling this cluster, the framework prioritized data quality over sheer quantity.

In general, the dataset transformation is visualized in Figure 9.3b. The initial

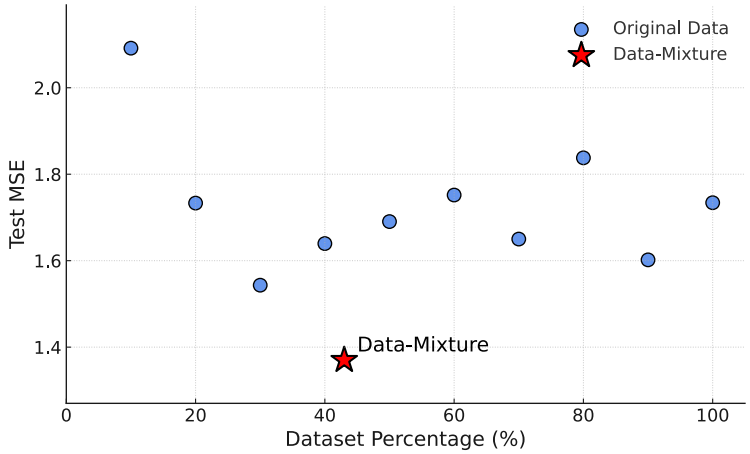
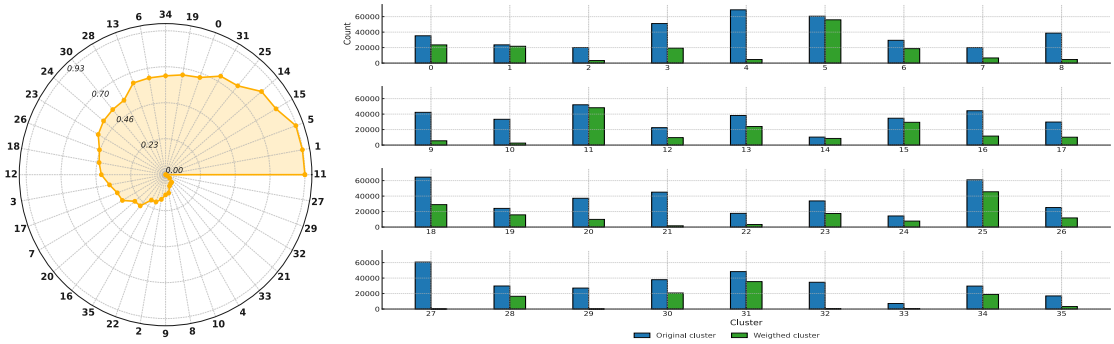


Figure 9.2: Test MSE as a function of training dataset size. The blue circles show the performance of the PatchTST model trained on various percentages of the original dataset, while the red star represents the model's superior performance when trained on our optimized data-mixture. The data-mixture, which is only 43% of the size of the full dataset, achieves a significantly lower MSE (1.37) compared to training on 100% of the original data (MSE of 1.73).

dataset was heavily skewed, with a few clusters containing the vast majority of data points while many others were sparsely populated. After applying the optimal weights, the resulting training set is significantly more balanced. The initially dominant clusters (e.g., 11, 5, 31) are reduced in size to be comparable with other, more informative clusters.

Table 9.2: Average MSE value for distilled models

	MSE		
Model	Full dataset	Data-Mixture	Gain (%)
TCN [KWB21b]	3.50	4.24	-21.14
LSTM [LA22]	5.64	5.52	2.13
BiLSTM [LA22]	4.18	3.32	20.57
AttentionLSTM [LA22]	3.89	3.59	7.71
PatchTST	1.70	1.37	19.41



(a) (b)

Figure 9.3: **(a)** Radar chart of optimized cluster weights arranged in descending order. Each axis corresponds to a cluster, and radial distance represents the relative weight. The chart highlights the distribution of weight values across all clusters, facilitating comparison of their relative importance. **(b)** Comparison of original versus weighted sample counts across the 36 clusters (0–35).

9.2.3 Distillation

To evaluate the generalizability of our data-mixture optimization, we retrained the state-of-the-art models presented in [KWB21a, LA22], including TCN, LSTM, BiLSTM, and AttentionLSTM.

The reported MSE values differ from those in prior work [LA22, KWB21a], as our models were trained using longer input windows (300 vs. 180 timesteps), which were chosen to capture long-term dependencies better and improve temporal context modeling—particularly beneficial for architectures like PatchTST that excel at leveraging extended temporal patterns.

This section aimed to assess whether the optimized subset — originally selected to improve PatchTST — could yield similar performance gains when applied to different architectures. This evaluation, analogous to a distillation process, probes the transferability of the optimized "training diet."

As shown in Table 9.2, the results indicate a strong dependency between model architecture and the effectiveness of the curated data. While PatchTST achieved a substantial 19.41% reduction in MSE, other models responded differently: BiLSTM improved by 20.57%, AttentionLSTM by 7.71%, and LSTM by 2.13%. In contrast, TCN exhibited a 21.14% performance drop when trained on the optimized data.

These outcomes suggest that models best suited for capturing long-range dependencies—such as PatchTST and BiLSTM—benefit most from the optimized subset. Conversely, the TCN model's performance degraded by 21.14% when trained on the data-mixture compared to the full dataset. This negative result is particularly insightful. The models that benefited most (PatchTST and BiLSTM) are architecturally better suited to capturing long-range dependencies. The 300-timestep windows used in this study, longer than the 180-step windows used in some baseline studies, likely contain such long-term patterns.

Therefore, the curated dataset was most effective for models capable of leveraging this specific type of information. Architectures like TCN, which may not model these extended dependencies as effectively, did not benefit and were potentially hindered by the removal of other, more redundant patterns they might have relied upon. This suggests that the advantages of data-mixture optimization are not entirely model-agnostic but are instead closely coupled with the learning capabilities of the chosen architecture.

This analysis underscores that effective data curation strategies must account for the specific inductive biases and learning mechanisms of the target architecture to achieve optimal performance gains.

9.3 Interpretability

To deepen our understanding of why certain clusters drive model performance, we conducted a qualitative "LLM review" [GJS+24] of cluster-specific behavior. We used `o4-mini` [Ope22] for this analysis. The system prompt is presented in Figure 9.4.

For each of the three most heavily weighted clusters (11, 1, and 5) and the three least weighted clusters (27, 29, and 32), we randomly sampled ten representative time-series plots and fed them to a large language model, asking it to characterize their dynamics and potential informational value. The LLM consistently highlighted that samples from clusters 11, 1, and 5 contain rich, structured variations—pronounced startup transients, clear load-change events, and well-defined steady-state plateaus—suggesting these data segments capture the fundamental modes of system operation.

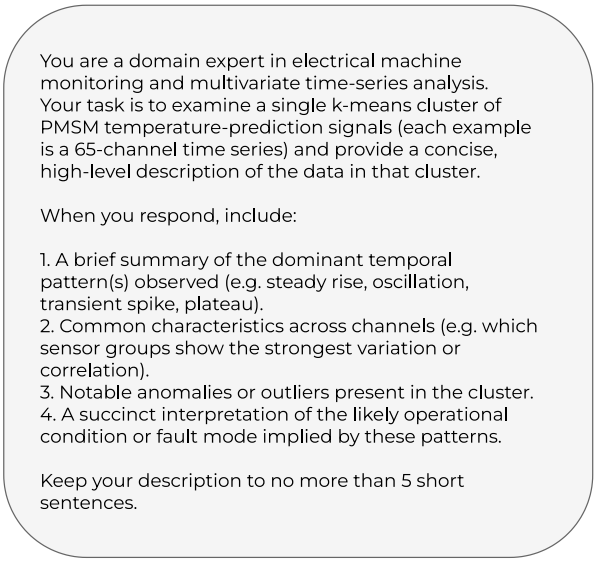


Figure 9.4: In this figure, we show the prompt used for the LLM-as-reviewer. The expert is required to summarize the information in the signals.

In contrast, the LLM described samples from clusters 27, 29, and 32 as largely uninformative: predominantly flatlines, subtle sensor noise, or simple shutdown spikes with little dynamic range. By presenting these insights alongside our Optuna-derived sampling weights, we not only reinforce why clusters 11, 1, and 5 are essential ingredients in the optimal "training diet" but also offer an interpretable rationale for deprioritizing clusters 27, 29, and 32.

This qualitative LLM-driven analysis thus bridges the gap between raw performance metrics and human intuition, demonstrating that the clusters we up-weight truly embody informative machine behaviors while those we down-weight contribute little beyond monotony.

Chapter 10

Conclusion

This dissertation addresses a central challenge in the modern automotive industry: transforming vast, unstructured sensor data from two-wheeled vehicles into structured, actionable knowledge. While modern motorcycles collect extensive data through their sensors and systems, the value of this data remains latent until it can be effectively organized, interpreted, and utilized. Through a series of machine learning methodologies, this work has demonstrated a clear, practical pathway to unlocking this potential.

The foundational prerequisite for any data-driven analysis is the ability to acquire and manage the data itself reliably. Chapter 4 addresses this critical requirement by presenting the design and implementation of a remote logging system capable of capturing, transmitting, and storing the high-volume sensor data streams generated by modern two-wheeled vehicles. This contribution establishes the technical infrastructure that bridges the gap between raw sensor outputs and analyzable datasets, ensuring data integrity and availability across distributed vehicle fleets.

The second contribution, detailed in Chapter 5, investigates a primary layer of organization by answering the question of "who". By employing a weakly supervised triplet loss framework, the system learned to extract "behavioral fingerprints", successfully identifying individual riders from their distinct interaction patterns with the machine. This initial step transformed a chaotic collection of time-series data into a set of attributable records, providing the foundational structure for all

subsequent analysis.

Building upon this foundation of identity, the research focus shifts to enrich this structure with dynamic context by addressing the question of "how." The contextual contrastive learning framework introduced in Chapter 6 moved beyond static identification to model the temporal flow and sequential nature of riding behavior. By learning the relationships between adjacent segments of a journey, this contribution transformed static rider profiles into dynamic models that capture how a ride unfolds over time. The $NDCG@10 = 0.88$ demonstrates that the model has successfully learned to encode the temporal dependencies and causal relationships between successive segments of a ride. In essence, the high retrieval accuracy confirms that a given "context" segment (the recent past of a ride) contains sufficient information to uniquely identify its corresponding "target" segment (the immediate future) from a large pool of alternatives.

The analysis then achieved a deeper level of granularity by explicitly modeling the dialogue between the rider and the machine. The dual-encoder framework presented in Chapter 7 creates a unified, joint embedding space that decodes how specific rider actions directly translate into vehicle responses. This provided a powerful lens for understanding the cause-and-effect dynamics of the rider-vehicle system, enabling robust mission profiling and a more comprehensive characterization of driving behavior.

Finally, Chapter 9 challenged the conventional wisdom that more data invariably leads to better models. By developing a data-centric optimization framework, this research demonstrated that the quality, not merely the quantity, of training data is the ultimate driver of performance. By systematically engineering an optimized, smaller training dataset, the framework achieved superior forecasting results with significantly less data, providing a robust and practical validation of the data-centric AI paradigm in a complex industrial setting. The results of this final chapter — achieving a 19.41% improvement in forecasting performance with a dataset that is 2.3 times smaller than the original — provide robust validation of the data-centric approach. It demonstrates a modern and mature perspective on applied machine learning, arguing that in many real-world scenarios, the most significant performance gains are to be found not in tweaking the model, but in meticulously curating the data it learns from.

The research presented in this dissertation makes several principal contributions to the fields of machine learning, time-series analysis, and intelligent transportation systems. The significance of these contributions lies not only in their specific methodological novelty but also in their potential to enable a new generation of intelligent vehicle technologies. The techniques developed the groundwork for a tangible future of motorcycling that is not only higher-performing but also fundamentally safer and more deeply personalized. The outcomes of this research enable us to envision a new generation of intelligent motorcycles that can dynamically adapt their performance characteristics, proactively schedule their maintenance, and offer intelligent, non-intrusive assistance.

Bibliography

[AABL13] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1247–1255, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[AAGG+22] Armstrong Aboah, Yaw Adu-Gyamfi, Senem Velipasalar Gursoy, Jennifer Merickel, Matt Rizzo, and Anuj Sharma. Driver maneuver detection and analysis using time series segmentation and classification. *J Transp Eng A Syst*, 149(3), dec 2022.

[AAO24] Sarah Azzabi, Zakiya Alfughi, and Abdelkader Ouda. Data lakes: A survey of concepts and architectures. *Computers*, 13(183), 2024.

[ABOE15] Ferhat Attal, Abderrahmane Boubezoul, Latifa Oukhellou, and Stéphane Espié. Powered two-wheeler riding pattern recognition using a machine-learning framework. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):475–487, 2015.

[ABSP23] Ana Almeida, Susana Brás, Susana Sargento, and Filipe Cabral Pinto. Time series big data: a survey on data stream frameworks, analysis and algorithms. *Journal of Big Data*, 10(1):83, May 2023.

[ADM+23] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas.

Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. *arXiv e-prints*, page arXiv:2301.08243, jan 2023.

[AHLH24] Ignacio Aguilera-Martos, Andrés Herrera-Poyatos, Julián Luengo, and Francisco Herrera. Local Attention Mechanism: Boosting the Transformer Architecture for Long-Sequence Time Series Forecasting. *arXiv e-prints*, page arXiv:2410.03805, oct 2024.

[AIn25] AInvest. Global connected motorcycle market to reach $967.71 million by 2030: Report, 2025.

[Alg12] Wael Alghamdi. Improving driver's behavior using context-aware systems. *Procedia Computer Science*, 10:1213–1216, 2012. ANT 2012 and MobiWIS 2012.

[AM18] Omid Avatefipour and Hafiz Malik. State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities. *arXiv preprint arXiv:1802.01725*, 2018.

[AMI+22] Sarah Najm Abdulwahid, Moamin A Mahmoud, Nazrita Ibrahim, Bilal Bahaa Zaidan, and Hussein Ali Ameen. Modeling motorcyclists' aggressive driving behavior using computational and statistical analysis of Real-Time driving data to improve road safety and reduce accidents. *Int J Environ Res Public Health*, 19(13), jun 2022.

[AMZ+22] Sarah Najm Abdulwahid, Moamin A Mahmoud, Bilal Bahaa Zaidan, Abdullah Hussein Alamoodi, Salem Garfan, Mohammed Talal, and Aws Alaa Zaidan. A comprehensive review on the behaviour of motorcyclists: Motivations, issues, challenges, substantial analysis and recommendations. *Int J Environ Res Public Health*, 19(6), mar 2022.

[Ara05] Vineet P. Aras. Design of electronic control unit (ecu) for automobiles-electronic engine management system. In *SAE Technical Paper*. SAE International, 2005.

[ASY+19] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *arXiv e-prints*, page arXiv:1907.10902, jul 2019.

[AWL+24] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

[BBB+23a] Mirco Bartolozzi, Abderrahmane Boubezoul, Samir Bouaziz, Giovanni Savino, and Stéphane Espié. Data-driven methodology for the investigation of riding dynamics: A motorcycle case study. *IEEE Transactions on Intelligent Transportation Systems*, 24(9):10224–10237, 2023.

[BBB+23b] Mirco Bartolozzi, Abderrahmane Boubezoul, Samir Bouaziz, Giovanni Savino, and Stéphane Espié. Understanding the behaviour of motorcycle riders: An objective investigation of riding style and capability. *Transportation Research Interdisciplinary Perspectives*, 22:100971, 2023.

[BEKL90] J. Bräuninger, R. Emig, T. Küttner, and A. Löffler. Controller area network for truck and bus applications. In *SAE Technical Paper*, number 902211. SAE International, 1990.

[BGL+93] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.

[Big18] Bigstep. Ai, machine learning and sexy motogp bikes, 2018.

[BIS+23] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes,

Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A Cookbook of Self-Supervised Learning. *arXiv e-prints*, page arXiv:2304.12210, apr 2023.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[BPL21] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. *arXiv e-prints*, page arXiv:2105.04906, may 2021.

[BRF+22] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6), dec 2022.

[CBJD18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep Clustering for Unsupervised Learning of Visual Features. *arXiv e-prints*, page arXiv:1807.05520, jul 2018.

[CCV19] Rishu Chhabra, Rama Challa, and Seema Verma. Smartphone based context-aware driver behavior classification using dynamic bayesian network. *Journal of Intelligent Fuzzy Systems*, 36:1–14, 01 2019.

[CCZH17] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. *arXiv e-prints*, page arXiv:1704.01719, apr 2017.

[CDL+24] Jinhui Cao, Xiaoqiang Di, Xu Liu, Jinqing Li, Zhi Li, Liang Zhao, Ammar Hawbani, and Mohsen Guizani. Anomaly detection for in-vehicle network using self-supervised learning with vehicle-cloud collaboration update. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):7454–7466, 2024.

[CGVB+24] Hui Chen, Charles Gouin-Vallerand, Kévin Bouchard, Sébastien Gaboury, Mélanie Couture, Nathalie Bier, and Sylvain Giroux. Contrastive self-supervised learning for sensor-based human activity recognition: A review. *IEEE Access*, 12:152511–152531, 2024.

[CH20] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv e-prints*, page arXiv:2002.05709, feb 2020.

[CKS23] Shivam Singh Chouhan, Ankit Kathuria, and Chalumuri Ravi Sekhar. The motorcycle rider behaviour questionnaire as a predictor of crashes: A systematic review and meta-analysis. *IATSS Research*, 47(1):61–72, 2023.

[CLC+21] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. Robust road network representation learning: When traffic patterns meet traveling semantics. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 211–220. ACM, 2021.

[CLW+22] Chao Chen, Qiang Liu, Xingchen Wang, Chengwu Liao, and Daqing Zhang. semi-traj2graph identifying fine-grained driving style with gps trajectory data via multi-task learning. *IEEE Transactions on Big Data*, 8(6):1550–1565, 2022.

[CMZA22] Yanbei Chen, Massimiliano Mancini, Xiatian Zhu, and Zeynep Akata. Semi-Supervised and Unsupervised Deep Visual Learning: A Survey. *arXiv e-prints*, page arXiv:2208.11296, aug 2022.

[CRG23] Rafael Canal, Felipe Kaminsky Riffel, and Giovani Gracioli. Driving profile analysis using machine learning techniques and ecu data. In

2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), pages 1–6, 2023.

[CRG24] Rafael Canal, Felipe K. Riffel, and Giovani Gracioli. Machine learning for real-time fuel consumption prediction and driving profile classification based on ecu data. *IEEE Access*, 12:68586–68600, 2024.

[CTM+21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. *arXiv e-prints*, page arXiv:2104.14294, apr 2021.

[CXZ+21] Shengdi Chen, Qingwen Xue, Xiaochen Zhao, Yingying Xing, and Jian John Lu. Risky driving behavior recognition based on vehicle trajectory. *Int J Environ Res Public Health*, 18(23), nov 2021.

[CZW+23] Hongqing Chu, Hejian Zhuang, Wenshuo Wang, Xiaoxiang Na, Lulu Guo, Jia Zhang, Bingzhao Gao, and Hong Chen. A review of driving style recognition methods from short-term and long-term perspectives. *IEEE Transactions on Intelligent Vehicles*, pages 1–15, 2023.

[DAA21] O. A. Daramola, O. S. Adewale, and B. O. Ayeni. Context-aware driver's behaviour monitoring system in vehicular ad-hoc network. *International Journal of Advanced Trends in Computer Science and Engineering*, 10(2):701–709, mar 2021.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv e-prints*, page arXiv:2010.11929, oct 2020.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805, October 2018.

[DKSZ23] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv e-prints*, page arXiv:2310.10688, oct 2023.

[DLY+16] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. Characterizing Driving Styles with Deep Learning. *arXiv e-prints*, page arXiv:1607.03611, jul 2016.

[DSD24] Rian Dolphin, Barry Smyth, and Ruihai Dong. Contrastive Learning of Asset Embeddings from Financial Time Series. *arXiv e-prints*, page arXiv:2407.18645, jul 2024.

[DYF+25] Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan, Lin, Jan Kautz, and Pavlo Molchanov. CLIMB: CLustering-based Iterative Data Mixture Bootstrapping for Language Model Pre-training. *arXiv e-prints*, page arXiv:2504.13161, apr 2025.

[DYY+17] Weishan Dong, Ting Yuan, Kai Yang, Changsheng Li, and Shilei Zhang. Autoencoder Regularized Network For Driving Style Representation Learning. *arXiv e-prints*, page arXiv:1701.01272, jan 2017.

[EMP17] Laura Eboli, Gabriella Mazzulla, and Giuseppe Pungillo. How drivers' characteristics can affect driving style. *Transportation Research Procedia*, 27:945–952, 2017. 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary.

[ERC+21] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[ERC+23] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Label-efficient Time Series Representation Learning: A Review. *arXiv e-prints*, page arXiv:2302.06433, feb 2023.

[ETKK16] Miro Enev, Alex Takakuwa, Karl Koscher, and Tadayoshi Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(1):34–50, Jan 2016.

[FAMC24] Adebamigbe Fasanmade, Ali H. Al-Bayatti, Jarrad Neil Morden, and Fabio Caraffini. Context-Aware Quantitative Risk Assessment Machine Learning Model for Drivers Distraction. *arXiv e-prints*, page arXiv:2402.13421, feb 2024.

[Fas24] FasterCapital. Motorcycle data analytics: Driving innovation: Exploring the intersection of motorcycle data analytics and entrepreneurship, 2024.

[FBA23] Archibald Fraikin, Adrien Bennetot, and Stéphanie Allassonnière. T-Rep: Representation Learning for Time Series using Time-Embeddings. *arXiv e-prints*, page arXiv:2310.04486, oct 2023.

[FD07] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[FFGS20] Todd A Frank, Graeme Fowler, Christina Garman, and Sarah Sharpe. Motorcycle rider inputs during typical maneuvers. In *SAE Technical Paper Series*, number 2020-01-1000, 400 Commonwealth Drive, Warrendale, PA, United States, apr 2020. SAE International.

[FL20] Tao-Yang Fu and Wang-Chien Lee. Trembr: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.*, 11(1):10:1–10:25, 2020.

[FPC+18] Yuxiang Feng, Simon Pickering, Edward Chappell, Pejman Iravani, and Chris Brace. Driving style analysis by classifying real-world data with support vector clustering. In *2018 3rd IEEE International*

Conference on Intelligent Transportation Engineering (ICITE), pages 264–268, 2018.

[GCZ+23] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends. *arXiv e-prints*, page arXiv:2301.05712, jan 2023.

[GDG+17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[GJS+24] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A Survey on LLM-as-a-Judge. *arXiv e-prints*, page arXiv:2411.15594, nov 2024.

[Gra25] Grafana Labs. Grafana: The open and composable observability and data visualization platform, 2025.

[GSA+20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv e-prints*, page arXiv:2006.07733, jun 2020.

[GSC+24] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

[Guo24] Shengran Guo. Application and impact of position embedding in natural language processing. In *2024 IEEE 4th International Con-*

ference on Electronic Communications, Internet of Things and Big Data (ICEIB), pages 246–250, 2024.

[GXL+24] Wentao Gao, Ziqi Xu, Jiuyong Li, Lin Liu, Jixue Liu, Thuc Duy Le, Debo Cheng, Yanchang Zhao, and Yun Chen. TSI: A Multi-View Representation Learning Approach for Time Series Forecasting. *arXiv e-prints*, page arXiv:2409.19871, sep 2024.

[HHLR15] Claudio Hartmann, Martin Hahmann, Wolfgang Lehner, and Frank Rosenthal. Exploiting big data in time series forecasting: A cross-sectional approach. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[Hot36] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[HRS12] Oliver Hartkopp, Christoph Reuber, and Roland Schilling. Socketcan – the official can api of the linux kernel. In *Proceedings of the 13th International CAN Conference*. iCC, 2012.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSMS21] Arsalan Hekmati, Iman Sadeghi Mahalli, and Mohammad Siamaki. A survey on permanent magnet synchronous machines: Recent applications and trends. *Electromechanical Energy Conversion Systems*, 1(3):42–50, 2021.

[HSS+17] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Rok Sosic, and Jure Leskovec. Driver Identification Using Automobile Sensor Data from a Single Turn. *arXiv e-prints*, page arXiv:1708.04636, jun 2017.

[HYY+18] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. *arXiv e-prints*, page arXiv:1804.06872, apr 2018.

[HZL+21] Zhongyang Han, Jun Zhao, Henry Leung, King Fai Ma, and Wei Wang. A review of deep learning models for time series prediction. *IEEE Sensors Journal*, 21(6):7833–7848, 2021.

[IGKM25] Habib Irani, Yasamin Ghahremani, Arshia Kermani, and Vangelis Metsis. Time Series Embedding Methods for Classification Tasks: A Review. *arXiv e-prints*, page arXiv:2501.13392, jan 2025.

[Inf25] InfluxData Inc. Influxdb: Scalable datastore for metrics, events, and real-time analytics, 2025. Open-source time-series database.

[ISO15] ISO. ISO 11898-1:2015 Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling. Standard, International Organization for Standardization, 2015.

[ISO24] ISO. ISO 11898-1:2024 Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical coding sublayer. Standard, International Organization for Standardization, 2024.

[JBZ+20] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A Survey on Contrastive Self-supervised Learning. *arXiv e-prints*, page arXiv:2011.00362, oct 2020.

[JC17] Steven J Johnston and Simon J Cox. The raspberry pi: A technology disrupter, and the enabler of dreams. *Electronics*, 6(3):51, 2017.

[JDS11] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[Jol21] Jolle W Jolles. Broad-scale applications of the raspberry pi: A review and guide for biologists. *Methods in Ecology and Evolution*, 12(9):1562–1579, 2021.

[Jor86] M I Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California

Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 05 1986.

[JPR+23] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*, pages 843–855. IEEE, 2023.

[K+21] Ashish Kumar et al. A comprehensive study on data visualization tool-grafana. *International Journal of Emerging Technologies and Innovative Research (JETIR)*, 8(5):f908–f914, 2021.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KBR+22] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham M. Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9*, 2022.

[KCL+25] Xiangjie Kong, Zhenghao Chen, Weiyao Liu, Kaili Ning, Lechao Zhang, Syauqie Muhammad Marier, Yichen Liu, Yuhao Chen, and Feng Xia. Deep learning for time series forecasting: a survey. *International Journal of Machine Learning and Cybernetics*, 16(7):5079–5112, Aug 2025.

[KIK25] Daichi Kimura, Tomonori Izumitani, and Hisashi Kashima. XicorAttention: Time Series Transformer Using Attention with Nonlinear Correlation. *arXiv e-prints*, page arXiv:2506.02694, jun 2025.

[KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *arXiv e-prints*, page cs/9605103, apr 1996.

[KNF+22] Neha Kalibhat, Kanika Narang, Hamed Firooz, Maziar Sanjabi, and Soheil Feizi. Measuring Self-Supervised Representation Quality for Downstream Classification using Discriminative Features. *arXiv e-prints*, page arXiv:2203.01881, mar 2022.

[KPN+23] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The Impact of Positional Encoding on Length Generalization in Transformers. *arXiv e-prints*, page arXiv:2305.19466, may 2023.

[KTZ19] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. Internet of things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data*, 6(1):111, Dec 2019.

[KWB21a] Wilhelm Kirchgässner, Oliver Wallscheid, and Joachim Böcker. Electric motor temperature dataset, 2021.

[KWB21b] Wilhelm Kirchgässner, Oliver Wallscheid, and Joachim Böcker. Estimating electric motor temperatures with deep residual machine learning. *IEEE Transactions on Power Electronics*, 36(7):7480–7488, 2021.

[L+14] F. Landhäußer et al. Electronic control unit (ecu). In K. Reif, editor, *Diesel Engine Management*, Bosch Professional Automotive Information, pages 332–351. Springer Vieweg, Wiesbaden, 2014.

[LA22] Jun Li and Thangarajah Akilan. Global Attention-based Encoder-Decoder LSTM Model for Temperature Prediction of Permanent Magnet Synchronous Motors. *arXiv e-prints*, page arXiv:2208.00293, jul 2022.

[LALZ24] Ziyu Liu, Azadeh Alavi, Minyi Li, and Xiang Zhang. Self-Supervised Learning for Time Series: Contrastive or Generative? *arXiv e-prints*, page arXiv:2403.09809, mar 2024.

[LC23] Jiexi Liu and Songcan Chen. TimesURL: Self-supervised Contrastive Learning for Universal Time Series Representation Learning. *arXiv e-prints*, page arXiv:2312.15709, dec 2023.

[Lin25] Linux Kernel Organization. Socketcan – controller area network. `https://www.kernel.org/doc/Documentation/networking/can.txt`, 2025. Linux Kernel Documentation.

[LKL+23] Shengzhong Liu, Tomoyoshi Kimura, Dongxin Liu, Ruijie Wang, Jinyang Li, Suhas Diggavi, Mani Srivastava, and Tarek Abdelzaher. FOCAL: Contrastive Learning for Multimodal Time-Series Sensing Signals in Factorized Orthogonal Latent Space. *arXiv e-prints*, page arXiv:2310.20071, oct 2023.

[LL17] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. *arXiv e-prints*, page arXiv:1705.07874, may 2017.

[LLW+24] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024.

[LOW+25] Che Liu, Cheng Ouyang, Zhongwei Wan, Haozhe Wang, Wenjia Bai, and Rossella Arcucci. Knowledge-enhanced Multimodal ECG Representation Learning with Arbitrary-Lead Inputs. *arXiv e-prints*, page arXiv:2502.17900, feb 2025.

[LZC+18] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 617–628. IEEE Computer Society, 2018.

[LZS+16] Jiajun Liu, Kun Zhao, Philipp Sommer, Shuo Shang, Brano Kusy, Jae-Gil Lee, and Raja Jurdak. A novel framework for online amnesic

trajectory compression in resource-constrained environments. *IEEE Trans. Knowl. Data Eng.*, 28(11):2827–2841, 2016.

[MAEP15] Mahdi H. Miraz, Maaruf Ali, Peter S. Excell, and Rich Picking. A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). In *2015 Internet Technologies and Applications (ITA)*, pages 219–224, 2015.

[MAS+24] John A. Miller, Mohammed Aldosari, Farah Saeed, Nasid Habib Barna, Subas Rana, I. Budak Arpinar, and Ninghao Liu. A Survey of Deep Learning and Foundation Models for Time Series Forecasting. *arXiv e-prints*, page arXiv:2401.13912, jan 2024.

[MH25] Abdul Majeed and Seong Oun Hwang. Data compactness versus prediction performance: Achieving both by pruning redundant samples with dominant patterns and hamming distance based sampling scheme. *IEEE Access*, 13:79655–79677, 2025.

[MHM18] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[Mic21] Microchip Technology Inc. *MCP2515 Stand-Alone CAN Controller with SPI Interface*. Microchip Technology Inc., 2021. Datasheet DS20001801K.

[Mic25] Microsoft Corporation. Microsoft power bi, 2025. Business analytics service.

[MKAW16] Rishi Menon, Arvind H. Kadam, Najath Abdul Azeez, and Sheldon S. Williamson. A comprehensive survey on permanent magnet synchronous motor drive systems for electric transportation applications. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6627–6632, 2016.

[MMHW+18] Clara Marina Martinez, Mira Heucke, Fei-Yue Wang, Bo Gao, and Dongpu Cao. Driving style recognition for intelligent vehicle control

and advanced driver assistance: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):666–676, 2018.

[MMJ13] Ferreira Michel Mendes-Moreira Joao L. L. Moreira-Matias, Luis and J. J. Taxi Service Trajectory - Prediction Challenge, ECML PKDD 2015. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C55W25.

[MMO+20] Fabio Martinelli, Francesco Mercaldo, Albina Orlando, Vittoria Nardone, Antonella Santone, and Arun Kumar Sangaiah. Human behavior characterization for driving style recognition in vehicle system. *Computers Electrical Engineering*, 83:102504, 2020.

[MMP+21] Sobhan Moosavi, Pravar D. Mahajan, Srinivasan Parthasarathy, Colleen Saunders-Chukwu, and Rajiv Ramnath. Driving Style Representation in Convolutional Recurrent Neural Network Model of Driver Identification. *arXiv e-prints*, page arXiv:2102.05843, feb 2021.

[Mor25] Mordor Intelligence. Connected motorcycle market size share analysis - growth trends forecasts (2025 - 2030) source: https://www.mordorintelligence.com/industry-reports/connected-motorcycle-market. https://www.mordorintelligence.com/industry-reports/connected-motorcycle-market, 2025. Accessed: 2025-08-17.

[MPS+24] Prakamya Mishra, Lincy Pattanaik, Arunima Sundar, Nishant Yadav, and Mayank Kulkarni. Clustering-based sampling for few-shot cross-domain keyphrase extraction. In Yvette Graham and Matthew Purver, editors, *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1232–1250, St. Julian's, Malta, mar 2024. Association for Computational Linguistics.

[MTMY24] Yasuhiro Mitsuhashi, Hitoshi Takeshita, Yoshitaka Momiyama, and Noboru Yabe. Study on motorcycle rider model using reinforcement

learning. *Transactions of Society of Automotive Engineers of Japan*, 55(3), 2024.

[NHNSK23] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[NM22] Athira Nambiar and Divyansh Mundra. An overview of data warehouse and data lake in modern enterprise data management. *Big Data and Cognitive Computing*, 6(132), 2022.

[NNSK22] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv e-prints*, page arXiv:2211.14730, nov 2022.

[NZW+23] Yao Nie, Sicong Zhang, Biao Wang, Wei Xu, Bin Liu, Zhou Yang, and Jianmin Lv. A time series is worth 64^2 tokens: Transformers for long-horizon time-series forecasting. *arXiv preprint arXiv:2211.14730*, 2023.

[Ope22] OpenAI. Introducing openai o3 and o4-mini. `https://openai.com/index/introducing-o3-and-o4-mini/`, 2022. Accessed: 2025-07-01.

[PAAD21] Nikolaos Peppes, Theodoros Alexakis, Evgenia Adamopoulou, and Konstantinos Demestichas. Driving behaviour analysis using machine and deep learning methods for continuous streams of vehicular data. *Sensors*, 21(4704), 2021.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[PGG25] Federico Pennino, Andrea Gurioli, and Maurizio Gabbrielli. Trajectory-embedded matryoshka representation learning for enhanced similarity analysis. In *ESANN 2025 proceesdings*, ESANN 2025, page 597–602. Ciaco - i6doc.com, 2025.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PSAG24a] Federico Pennino, Davide Sette, David Attisano, and Maurizio Gabbrielli. Driving style representation via convolutional neural networks: A contrastive learning approach. In *27th IEEE International Conference on Intelligent Transportation Systems, ITSC 2024, Edmonton, AB, Canada, September 24-27, 2024*, pages 3532–3538. IEEE, 2024.

[PSAG24b] Federico Pennino, Davide Sette, David Attisano, and Maurizio Gabbrielli. A machine learning based tool to estimate coolant engine temperature based on motorcycle riding data. In *2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 729–733, 2024.

[PSAG25a] Federico Pennino, Davide Sette, David Attisano, and Maurizio Gabbrielli. Contextual contrastive learning for rider behavior modeling using motorcycle sensor data. In *28th IEEE International Conference on Intelligent Transportation Systems, ITSC 2025, Gold Coast, Australia, November 18 – 21, 2025*. IEEE, 2025. Accepted.

[PSAG25b] Federico Pennino, Davide Sette, David Attisano, and Maurizio Gabbrielli. A dual-encoder framework for enhancing driving behavior and mission profiling characterization. In *INNS International Joint*

Conference on Neural Networks 2025, IJCNN 2025, Rome, Italy, June 30 to July 5, 2025. IEEE, 2025. In press.

[Pyt24] Python-CAN Contributors. python-can: Controller area network (can) support for python, 2024. https://github.com/hardbyte/python-can - version 4.6.1.

[Ras25] Raspberry Pi Foundation. Raspberry pi. https://www.raspberrypi.org/, 2025. Accessed: 2025-10-26.

[RCSJ20] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive Learning with Hard Negative Samples. *arXiv e-prints*, page arXiv:2010.04592, oct 2020.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[Riz14] Véronique Rizzi. *Motorcycle Riders' Acceptance of Advanced Rider Assistance Systems*. 01 2014.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv e-prints*, page arXiv:2103.00020, feb 2021.

[RM05] Andry Rakotonirainy and Frederic Maire. Context-aware driving behaviour model. In D Gilmore, editor, *Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles*, pages 1–6. National Highway Traffic Safety Administration, http://www.nhtsa.gov/ESV, 2005.

[RM22] Shashi Ravi and Mohan Rao Mamdikar. A review on its (intelligent transportation systems) technology. In *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 155–159, 2022.

[RZF+19] Elahe Rahimian, Soheil Zabihi, Seyed Farokh Atashzar, Amir Asif, and Arash Mohammadi. XceptionTime: A Novel Deep Architecture based on Depthwise Separable Convolutions for Hand Gesture Classification. *arXiv e-prints*, page arXiv:1911.03803, nov 2019.

[SEN20] Iván Silva and José Eugenio Naranjo. A systematic methodology to evaluate prediction models for driving style classification. *Sensors (Basel)*, 20(6), mar 2020.

[SFK+21] Christoph Sohrmann, Robert Fischbach, Andreas Krinke, Thomas Nirmaier, Volker Meyer zu Bexten, Goeran Jerke, and Jan Novacek. Towards a standardized format for automotive mission profiles. In *AmE 2021 - Automotive meets Electronics; 12th GMM-Symposium*, pages 1–6, 2021.

[SGAQ23] Rabab Saber, Said Ghoniemy, and Mirvat Al-Qutt. Driver behavior detection in time series decade review. *International Journal of Intelligent Computing and Information Sciences*, 23(3):114–140, 2023.

[SI84] Shokri Z. Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–87, 1984.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv e-prints*, page arXiv:1503.03832, 2015.

[SLF+24] Yunfan Shao, Linyang Li, Zhaoye Fei, Hang Yan, Dahua Lin, and Xipeng Qiu. Balanced Data Sampling for Language Model Training with Clustering. *arXiv e-prints*, page arXiv:2402.14526, feb 2024.

[SLLH24] Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. Review of Data-centric Time Series Analysis from Sample, Feature, and Period. *arXiv e-prints*, page arXiv:2404.16886, apr 2024.

[SLP+21] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv e-prints*, page arXiv:2104.09864, apr 2021.

[Soh16] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[SSL+14] Chika Sakashita, Teresa Senserrick, Serigne Lo, Soufiane Boufous, Liz de Rome, and Rebecca Ivers. The motorcycle rider behavior questionnaire: Psychometric properties and application amongst novice riders in australia. *Transportation Research Part F: Traffic Psychology and Behaviour*, 22:126–139, 2014.

[SV25] Laxmi Kant Sahoo and Vijayakumar Varadarajan. Deep learning for autonomous driving systems: technological innovations, strategic implementations, and business implications - a comprehensive review. *Complex Engineering Systems*, 5(2), 2025.

[SXJS15] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. *arXiv e-prints*, page arXiv:1511.06452, nov 2015.

[SYL+25] Haifeng Sun, Junping Yao, Xiaojun Li, Yanfei Liu, and Hongyang Gu. Robust two stages federated learning for sensor based human activity recognition with label noise. *Scientific Reports*, 15(1), may 2025.

[SZL+25] Liyilei Su, Xumin Zuo, Rui Li, Xin Wang, Heng Zhao, and Bingding Huang. A systematic review for transformer-based long-term series forecasting. *Artificial Intelligence Review*, 58(3):80, jan 2025.

[TPPSM20] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, and Cecilia Mascolo. Exploring contrastive learning in human activity recognition for healthcare. *arXiv preprint arXiv:2011.11542*, 2020.

[TSK+24] Patara Trirat, Yooju Shin, Junhyeok Kang, Youngeun Nam, Jihye Na, Minyoung Bae, Joeun Kim, Byunghyun Kim, and Jae-Gil Lee. Universal Time-Series Representation Learning: A Survey. *arXiv e-prints*, page arXiv:2401.03717, jan 2024.

[TY24] Yixuan Tang and Yi Yang. Do We Need Domain-Specific Embedding Models? An Empirical Investigation. *arXiv e-prints*, page arXiv:2409.18511, sep 2024.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984.

[VC15] V. N. Vapnik and A. Ya. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, pages 11–30. Springer International Publishing, Cham, 2015.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(95):2837–2854, 2010.

[Vec25] Vector Informatik GmbH. Binary logging format (blf). Vector CANoe and CANalyzer file format specification, 2025. Industry-standard format for automotive bus data.

[VFCG25] Ranan Venancio, Vitor Filipe, Adelaide Cerveira, and Lio Gonçalves. Advanced driving assistance integration in electric motorcycles: road surface classification with a focus on gravel detection using deep learning. *Frontiers in Artificial Intelligence*, Volume 8 - 2025, 2025.

[vLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, page arXiv:1807.03748, jul 2018.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, page arXiv:1706.03762, jun 2017.

[WALB16] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On Deep Multi-View Representation Learning: Objectives and Optimization. *arXiv e-prints*, page arXiv:1602.01024, feb 2016.

[WDS+19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv e-prints*, page arXiv:1910.03771, oct 2019.

[WLK+24] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

[WLS+22] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting. *arXiv e-prints*, page arXiv:2202.01575, feb 2022.

[Wu12] Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

[WWL+24] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024.

[WX22] LIAO Xiaoling-JIANG Peiyu ZHANG Wei WANG Fang WANG Xu, MA Fei. Feature selection for recognition of driving styles based on multi-classification and supervised learning, 2022.

[WXCL17] Wenshuo Wang, Junqiang Xi, Alexandre Chong, and Lin Li. Driving style classification using a semisupervised support vector machine. *IEEE Transactions on Human-Machine Systems*, 47(5):650–660, 2017.

[WXWL21] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, 2021.

[WZZ+22] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in Time Series: A Survey. *arXiv e-prints*, page arXiv:2202.07125, feb 2022.

[WZZ+23] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. In *International Joint Conference on Artificial Intelligence(IJCAI)*, 2023.

[YCZB19] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 1358–1369. IEEE, 2019.

[YWD+21] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. TS2Vec: Towards Universal Representation of Time Series. *arXiv e-prints*, page arXiv:2106.10466, jun 2021.

[YZZ+17] Di Yao, Chao Zhang, Zhihua Zhu, Jian-Hui Huang, and Jingping Bi. Trajectory clustering via deep representation learning. In *2017*

International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017, pages 3880–3887. IEEE, 2017.

[YZZ+21] Jingbo Yang, Ruge Zhao, Meixian Zhu, David Hallac, Jaka Sodnik, and Jure Leskovec. Driver2vec: Driver Identification from Automotive Data. *arXiv e-prints*, page arXiv:2102.05234, feb 2021.

[ZGZ+24] Qiquan Zhang, Meng Ge, Hongxu Zhu, Eliathamby Ambikairajah, Qi Song, Zhaoheng Ni, and Haizhou Li. An Empirical Study on the Impact of Positional Encoding in Transformer-based Monaural Speech Enhancement. *arXiv e-prints*, page arXiv:2401.09686, jan 2024.

[Zho17] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.

[ZJM+21] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. *arXiv e-prints*, page arXiv:2103.03230, mar 2021.

[ZQZ+24] Chunfeng Zhang, Hao Qin, Yongjun Zhang, Chongying Jiang, Di Zhang, and Wenyang Deng. Augmenting time series data: An interpretable approach with metric learning and variational autoencoders. *International Journal of Electrical Power Energy Systems*, 161:110190, 2024.

[ZWZ+23] Kexin Zhang, Qingsong Wen, Chaoli Zhang, Rongyao Cai, Ming Jin, Yong Liu, James Zhang, Yuxuan Liang, Guansong Pang, Dongjin Song, and Shirui Pan. Self-Supervised Learning for Time Series Analysis: Taxonomy, Progress, and Prospects. *arXiv e-prints*, page arXiv:2306.10125, jun 2023.

[ZYW+25] Laura Zheng, Hamidreza Yaghoubi Araghi, Tony Wu, Sandeep Thalapanane, Tianyi Zhou, and Ming C. Lin. Quantifying and Modeling Driving Styles in Trajectory Forecasting. *arXiv e-prints*, page arXiv:2503.04994, mar 2025.

[ZZGZ19] Deli Zhao, Jiapeng Zhu, Zhenfang Guo, and Bo Zhang. Curriculum Learning for Deep Generative Models with Clustering. *arXiv e-prints*, page arXiv:1906.11594, jun 2019.

[ZZP+20] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *arXiv e-prints*, page arXiv:2012.07436, dec 2020.

[ZZTZ22] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. In *Proceedings of Neural Information Processing Systems, NeurIPS*, 2022.

Acknowledgements

Il dottorato è stata un'esperienza che mi ha permesso di crescere molto come persona, penso però che la mia crescita personale non sia solo legata alle esperienze fatte, ma anche alle persone che mi hanno circondato: in questi tre anni sono stato estremamente fortunato nell'avere attorno a me delle persone fantastiche.

La prima persona che sento di dover ringraziare è Maurizio Gabbrielli, il mio supervisor. Ho sviluppato con lui un rapporto personale che va oltre il semplice rapporto accademico. Allo stesso modo voglio ringraziare David Attisano e Davide Sette, i miei supervisor all'interno di Ducati Motor Holding. Siete stati preziosi nel guidarmi su come approciare i problemi in una realtà così importante, ve ne sarò sempre grato. Un ringraziamento altrettanto sentito va ai miei colleghi in Ducati e ai miei colleghi all'interno dell'Università di Bologna: siete stati degli amici prima che dei colleghi, ho sviluppato dei rapporti veramente forti e sinceri, grazie. Voglio anche ringraziare i miei amici storici e le amicizie che sono nate negli anni di dottorato. Avete reso le mie giornate più pesanti decisamente più leggere.

Un pensiero speciale va ai miei nonni, ai miei zii e ai miei cugini, che hanno sempre creduto in me. Siete stati fondamentali nel percorso che mi ha portato a essere la persona che sono oggi.

Concludo con le persone a me più vicine. Grazie a Isabella, la mia fidanzata, per la pazienza, il sostegno e per tutto il tempo condiviso. Infine, un grazie alle persone che mi sopportano da più tempo: i miei genitori e mio fratello. Siete sempre stati al mio fianco, supportandomi in ogni scelta. Grazie, vi voglio bene.

Funded by the European Union - Next Generation EU, Mission 4, Component 2, Investment 3.3 (Ministerial Decree 117/2023) CUP ***J33C22001400009***