CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

AI-Based CAN Bus Analyzer/Simulator System for In-Vehicle Networks

Fouad ASIL

Department of Electrical and Electronics Engineering

September, 2025

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS APPROVAL

AI-Based CAN Bus Analyzer/Simulator System for In-Vehicle Networks

---

Fouad ASIL

Department of Electrical and Electronics Engineering

This Doctorate Thesis was evaluated by the following Jury Members on 26/09/2025 and was approved by unanimity / majority of votes. ss

Jury : Prof. Dr. Mustafa GÖK (Advisor) ……

: Assoc. Prof. Dr. Ahmet AYDIN ……

: Assoc. Prof. Dr. Yasin KAYA ……

: Assoc. Prof. Dr. Fatih KILIÇ ……

: Asst. Prof. Dr. Oğuzhan TİMUR ……

This Thesis was written in the Department of Electrical and Electronics Engineering at the Institute of Natural and Applied Sciences.

Thesis Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

CONTENTS

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

AI-Based CAN Bus Analyzer/Simulator System for In-Vehicle Networks

Fouad ASIL

Advisor: Prof.Dr. Mustafa GÖK

Department of Electrical and Electronics Engineering

ABSTRACT

Analyzing and simulating Controller Area Network (CAN) traffic remain challenging because decoding often depends on proprietary formats and manual effort. This thesis propose an automated machine-learning framework for SAE~J1939 traffic that, after decoding, standardizes preprocessing, classifies anchor signals, reconstructs multiple unknown signals, assigns semantics via a bounded score fusion, and re-encodes protocol-valid traces. The system also supports optional adversarial injection for intrusion detection (IDS) research. Evaluated on public J1939 datasets from heavy-duty vehicles, tree-based classifiers reach 98.4–99.6 % accuracy, and multi-signal reconstruction attains ($R^2 > 0.90$). In a representative run, signal matching 23 signals matching. The framework provides a strong, computationally efficient baseline for reverse engineering and a reproducible facility for generating realistic and adversarial CAN traces to advance IDS benchmarking. During training, This work uses J1939 DBCs only to derive ground-truth field boundaries for supervision; at inference, the system consumes unlabeled, tokenized payloads and does not require a DBC.

Keywords: CAN Bus, SAE J1939, Machine learning, Artificial Intelligence, Reverse engineering.

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

AI Tabanlı Araç İçi Ağlar için CAN Bus Analizör/Simülatör Sistemi

---

Fouad ASIL

Danışman: Prof. Dr. Mustafa GÖK

Elektrik-Elektronik Mühendisliği Anabilim Dalı

ÖZ

Kontrolör Alan Ağı (CAN) trafiğini analiz ve simüle etmek, kod çözme işleminin genellikle araç üreticilerine tescilli formatlara bağlı olması ve operatörün manuel çabasını gerektirmesi nedeniyle güç işlemlerdir. Bu tezde kod çözme işleminden sonra ön işlemeyi standartlaştıran, çapa sinyallerini sınıflandıran, çok sayıda bilinmeyen sinyali yeniden oluşturan, sınırlı bir skor füzyonu aracılığıyla anlamsal atama yapan ve protokole uygun izleri yeniden kodlayan otomatik bir makine öğrenmesi çerçevesi önerilmektadır. Tezde aynı zamanda derin öğrenme yöntemlerinin etkinliği de araştırılmıştır. Sistem ayrıca, saldırı tespit sistemi (IDS) araştırmaları için isteğe bağlı çekişmeli enjeksiyonu da desteklemektedir. Önerilen sistem ağır vasıta araçlarda kullanılan bir CAN protokolü olan SAE J1939 trafiği kullanılarak test edilmiştir. Halka açık J1939 veri setleri üzerinde değerlendirildiğinde, ağaç tabanlı sınıflandırıcılar %98,4–99,6 doğruluk oranına ulaşmakta ve çoklu sinyal yeniden yapılandırması, temsili bir çalıştırmada 23 sinyali eşleştirmektedir. Bu çerçeve, tersine mühendislik için güçlü, hesaplama açısından verimli bir temel ve IDS kıyaslamasını ilerletmek amacıyla gerçekçi ve çekişmeli CAN izleri oluşturmak için tekrarlanabilir bir olanak sağlar. Eğitim sırasında, J1939 DBC'lerini yalnızca gözetim için temel gerçeklik alan sınırlarını türetmek amacıyla kullanıyoruz; çıkarım aşamasında ise sistem, etiketsiz, simgeleştirilmiş veri yüklerini tüketir ve bir DBC gerektirmez.

Anahtar Kelimeler:CAN bus, SAE J1939, Yapay Zeka, Makine öğrenmesi, Tersine mühendislik.

EXTENDED SUMMARY

Modern motor vehicles are complex cyber-physical systems that contain dozens and often hundreds of interconnected electronic control units (ECUs). These systems manage a wide range of functions from engine performance and braking to passenger comfort. The backbone of this digital ecosystem is the Controller Area Network (CAN) bus, a reliable serial communication protocol designed to simplify wiring and established as the industry standard for in-vehicle networking for more than thirty years. In heavy-duty and commercial vehicles, the SAE J1939 standard builds upon CAN to ensure interoperability between ECUs produced by different manufacturers.

Messages flowing continuously across these networks carry data that is vital to vehicle operation, including vehicle speed, engine revolutions, and fuel levels. Despite its critical role, the communication protocol of the CAN bus is typically kept proprietary by vehicle manufacturers. Signals are encoded within CAN messages using database files known as DBCs, which contain manufacturer-specific formats. These files are rarely released publicly, as their content is considered intellectual property. As a result, this lack of access presents a significant barrier for technicians, third-party device manufacturers, academics, and cybersecurity analysts. Without knowledge of these proprietary formats, it becomes extremely difficult to interpret vehicle states, diagnose complex faults, develop new products, or assess security vulnerabilities.

Within this context, the present thesis seeks solutions to two central problems using modern artificial intelligence methods. The first is the reverse engineering of proprietary signals, and the second is the generation of realistic data for research and development. To address these problems, an AI-based integrated framework has been developed that automates the entire process, from analyzing raw CAN data collected from vehicle networks to fully interpreting these signals and generating realistic, protocol-compliant, and simulatable data traces. The framework is designed to solve the problem of decoding unknown signals and to address the significant lack of high-quality synthetic datasets, particularly in the field of automotive cybersecurity.

The scope of the study is defined through several technical research questions. A modular and multi-stage machine learning framework is proposed to determine whether the reverse engineering of proprietary CAN signals can be automated with high accuracy. The second question asks whether it is possible to use a small number of known “anchor” signals to classify and reconstruct a much larger set of unknown signals in the network traffic, and how the performance of traditional machine learning models compares with modern deep learning approaches in these tasks. The third question asks whether fully interpreted signal data can be used to generate realistic and protocol-compliant CAN traces, and whether this process can be extended to include the injection of cyberattacks in a controlled way for the purpose of developing and benchmarking intrusion detection systems (IDS).

To answer these questions, a software framework with a modular five-stage pipeline architecture has been developed. The process begins with the analysis of raw CAN data and ends with the production of fully simulatable, labeled traces. Each stage of the framework is designed to build on the outputs of the previous one. In the first stage, data preprocessing, raw CAN log files collected from various sources are decoded using SAE J1939 DBC files, resampled to a uniform time interval, imputed for missing values, and filtered and normalized to improve quality. This yields a high-quality and balanced time series dataset for later analysis. In the second stage, anchor signal classification, a small set of fundamental anchor signals representing core vehicle dynamics such as engine speed and vehicle speed are identified. Unlabeled data windows are passed through a series of classification models trained to recognize the distinct temporal patterns and statistical properties of each anchor. The result is a correctly classified anchor signal set that reliably summarizes the vehicle's operational state in a compact form.

The third stage, multi-signal reconstruction, uses these anchor signals to reconstruct the remaining unknown signals. The objective is not to predict future values, but to reconstruct the values of unknown signals within the same time window by using the anchors as functional references. This preserves instantaneous amplitude and phase relationships, which are critical for capturing the true behavior of signals. The results show that a small but well-chosen anchor set contains enough information to reconstruct a much wider range of vehicle signals with high fidelity. In the fourth stage, semantic matching and labeling, reconstructed signals are assigned meaningful labels by comparing them against a reference database of known signals. Fifteen similarity metrics, including correlation-based, distance-based, statistical, and information-theoretic measures, are used to compare reconstructed signals with reference profiles. The label of the reference signal with the highest similarity score is then assigned to the unknown signal, allowing raw numerical data to be translated into interpretable categories such as fuel pressure or turbo speed.

In the fifth stage, protocol-compliant data generation and simulation, the fully interpreted signals are re-encoded into the SAE J1939 format to produce synthetic traces. Signals are reconstructed into the correct PGN (Parameter Group Number) and SPN (Suspect Parameter Number) structures. A key feature of this stage is the integrated attack module, which can inject different families of malicious traffic such as spoofing, replay, flooding, and address-claim attacks. This functionality enables the creation of scalable, high-quality datasets for IDS development and validation.

To validate the effectiveness of the proposed framework, four large-scale public datasets of SAE J1939 logs from heavy-duty vehicles were used. To evaluate system performance under different levels of information availability, five anchor configurations were defined, ranging from three to seven signals. These included minimal, standard, comprehensive, engine-focused, and dynamics-focused anchor sets. Both traditional machine learning models, such as Random Forest,

XGBoost, and Extra Trees, and deep learning models, including LSTM, 1D-CNN, and Transformer architectures, were tested on these configurations.

The results confirm the high performance of the proposed framework. In anchor classification, nearly all models achieved F1 scores above 0.98, and tree-based ensembles consistently surpassed 0.99. In signal reconstruction, deep learning models achieved state-of-the-art results, with the ResNetMLP-Medium reaching a mean R^2 of 0.983 under the comprehensive anchor set, compared to 0.964 for the best machine learning baseline. In semantic matching, 14 of 15 signals were correctly matched, resulting in a 93.3% success rate. In trace generation, synthetic traces preserved reconstruction quality within ±2% of the baseline, achieved 99.8% protocol compliance, and could be generated efficiently, with a 300-second trace produced in approximately 8.3 seconds.

This thesis therefore provides a compact, interpretable, and validated baseline framework for automatically identifying and simulating automotive signals on SAE J1939 CAN buses. The results show that computationally efficient and transparent machine learning models, when combined with carefully engineered features, are sufficient for many tasks, while deep learning architectures provide additional benefits in signal reconstruction. The ability to synthesize realistic and adversarial CAN traces offers particular value to the field of automotive cybersecurity, enabling scalable testing of intrusion detection systems and robustness analyses.

Although robust, the framework has limitations that suggest directions for future work. The use of fixed windowing parameters may limit sensitivity to signals that vary over longer horizons. The evaluation on cleaned, high-quality datasets likely contributed to higher performance metrics and should be extended to noisier, less curated signals to better approximate real-world conditions. The reliance on supervised tokenization may create mismatches in deployment, and the absence of manufacturer-specific priors may reduce cross-platform generalization. Finally, the current system has been validated on segmented datasets rather than live data streams, and adapting it to real-time inference represents a critical next step.

In conclusion, this research shows that interpretable and computationally efficient machine learning pipelines, supported by deep learning baselines, can effectively meet the practical demands of automotive signal reverse engineering, simulation, and cybersecurity testing. The framework establishes a strong baseline that balances accuracy, deployability, and explainability, and provides a foundation for future research aimed at greater robustness, adaptability, and real-world applicability.

GENİŞLETİLMİŞ ÖZET

Modern motorlu taşıtlar, düzinelerce ve hatta yüzlerce elektronik kontrol ünitesinin (ECU) birbirine bağlı olduğu karmaşık siber-fiziksel sistemler içermektedir. Bu sistemler, motor performansı ve frenlemeden yolcu konforuna kadar geniş bir fonksiyon yelpazesini yönetir. Araç dijital ekosistemin temel iletişim hattı Kontrolör Alan Ağı (CAN) veri yoludur. Güvenilir bir seri iletişim protokolü olan CAN, kablolamayı basitleştirmek amacıyla tasarlanmıştır ve otuz yılı aşkın bir süredir araç içi ağlar için endüstri standardıdır.. Özellikle ağır vasıta ve ticari araçlarda, bu protokolün üzerine inşa edilen SAE J1939 standardı, farklı üreticilerin ECU'ları arasında birlikte çalışabilirliği sağlamaktadır.

Bu ağlar üzerinden sürekli olarak akan mesajlar, araç hızı, motor devri ve yakıt seviyesi gibi aracın çalışması için hayati önem taşıyan verileri içerir. Ancak bu kritik role sahip, CAN veri yolunun iletişim protokolü, genellikle araç üreticileri (OEM'ler) tarafından gizli tutulmaktadır. Üreticiler, sinyal bilgilerini CAN mesajları içinde kodlamak amacıyla DBC adı verilen ve kendilerine özgü format tanımlarını içeren hususi veritabanı dosyalarını kullanır. Bu dosya içindeki tanımları değerli bir fikri mülkiyet olduğu iddiasıyla nadiren kamuya açıklarlar. Bu durum, otomotiv teknisyenleri, üçüncü parti cihaz üreticileri, akademisyenler ve siber güvenlik analistleri gibi birçok meşru ve önemli faaliyet alanında çalışan araştırmacılar için önemli bir engel teşkil etmektedir. Bu özel formatlara erişim olmadan, aracın iç durumunu anlamak, karmaşık arızaları teşhis etmek, yeni ürünler geliştirmek veya güvenlik açıklarını değerlendirmek son derece zordur. Bu bağlamda, bu tez çalışması iki temel probleme modern yapay zeka yöntemleri ile çözüm bulmayı amaçlamaktadır: birincisi özel sinyallerin tersine mühendisliği ve ikincisi araştırma-geliştirme için gerçekçi verilerin üretilmesi problemleridir.

Bu amaçlara ulaşmak için, araç içi ağlardaki ham CAN verilerinin analizinden başlayarak, bu verilerin tam anlamıyla yorumlanması ve gerçekçi, protokol uyumlu ve simüle edilebilir veri izleri (trace) üretilmesine kadar olan tüm süreçleri otomatikleştiren, yapay zeka tabanlı bütünleşik bir çerçeve geliştirilmiştir. Bu çerçeve, hem bilinmeyen sinyallerin anlamlandırılması (tersine mühendislik) problemini çözmeyi hem de özellikle siber güvenlik alanında büyük bir eksiklik olan yüksek kaliteli sentetik veri setleri oluşturma ihtiyacını karşılamayı hedeflemektedir.

Çalışmanın özel kapsamında aşağıdaki daha teknik temel araştırma sorularına yanıt aranmaktadır:

1. Modüler ve çok aşamalı bir makine öğrenmesi çerçevesi, özel CAN sinyallerinin tersine mühendislik sürecini yüksek doğrulukla otomatikleştirebilir mi?
2. Az sayıda bilinen "çapa" (anchor) sinyal kullanarak, ağ trafiğindeki çok daha geniş bir bilinmeyen sinyal setini doğru bir şekilde sınıflandırmak ve yeniden yapılandırmak mümkün

müdür? Bu sınıflandırma ve yeniden yapılandırma görevleri için geleneksel makine öğrenmesi modelleri (örneğin, ağaç tabanlı topluluklar) ile modern derin öğrenme modellerinin performansı nasıl karşılaştırılabilir?

3. Tamamen yorumlanmış sinyal verileri , gerçekçi ve protokol uyumlu CAN veri yolları izleri oluşturmak amacıyla kullanılabilir mi? Bu süreç, Saldırı Tespit Sistemleri (IDS) geliştirmek ve kıyaslamak amacıyla kontrol edilebilir bir şekilde siber saldırıların enjekte edilmesini kapsayacak şekilde genişletilebilir mi?

Bu çalışmada, yukarıda belirtilen sorulara yüksek başarımlı cevaplar bulmak amacıyla beş aşamalı, modüler bir boru hattı mimarisine sahip bir yazılım çerçevesi (framework) hazırlanmıştır. Sistem, ham sinyal verilerini analizi aşamasıyla başlar ve son aşamada tamamen simüle edilebilir, etiketlenmiş veri izleri üretir. Yazılım çerçevesinin her aşaması, bir sonraki aşama için temel oluşturacak şekilde tasarlanmıştır. Çerçeve aşamaları aşağıda kısaca açıklanmaktadır.

Aşama 1, Veri Ön İşleme: Bu ilk aşama, çeşitli kaynaklardan toplanan ham CAN kayıt dosyalarını analiz için hazırlar. Veriler, eğitim ve doğrulama aşamalarında bir temel gerçeklik (ground truth) oluşturmak amacıyla SAE J1939 DBC dosyaları kullanılarak çözümlenir. Ardından, sinyaller tek tip bir zaman aralığında yeniden örneklenir, eksik veriler uygun yöntemlerle tamamlanır ve sinyal kalitesini artırmak için çeşitli filtreleme ve normalizasyon adımları uygulanır. Bu aşamanın sonunda, sonraki analizler için yüksek kaliteli ve dengeli bir zaman serisi veri seti elde edilir.

Aşama 2, Çapa Sinyal Sınıflandırması: Boru hattının bu aşaması, az sayıda bilinen ve temel araç dinamiklerini temsil eden "çapa" sinyallerini (örneğin, motor hızı, araç hızı) tanımlamaya odaklanır. Etiketlenmemiş veri pencereleri, her bir çapa sinyalinin kendine özgü zamansal desenlerini ve istatistiksel özelliklerini tanımak üzere eğitilmiş bir dizi makine öğrenmesi sınıflandırma modeline girdi olarak verilir. Bu modeller, her bir giriş penceresine "motor hızı" gibi belirli bir sınıf etiketi atar. Bu adım, aracın operasyonel durumunun güvenilir ve düşük boyutlu bir özetini sunan, doğru bir şekilde sınıflandırılmış bir çapa sinyal seti üretir.

Aşama 3, Çoklu Sinyal Yeniden Yapılandırması: Bu aşamada, önceki adımda tanımlanan çapa sinyalleri, araç ağındaki geri kalan bilinmeyen sinyalleri yeniden yapılandırmak için kullanılır. Buradaki temel amaç, gelecekteki değerleri tahmin etmek (forecasting) değil, belirli bir zaman penceresindeki çapa sinyallerini kullanarak aynı penceredeki bilinmeyen sinyallerin değerlerini fonksiyonel olarak yeniden yapılandırmaktır. Bu yaklaşım, sinyaller arasındaki anlık genlik ve faz ilişkilerini korur ki bu, sinyallerin gerçek zamanlı davranışlarına göre tanımlanması için kritik öneme sahiptir. Bu aşamadaki başarı, iyi seçilmiş küçük bir çapa sinyal setinin, daha geniş araç sisteminin durumunu tanımlamak için yeterli bilgi içerdiğini göstermektedir.

Aşama 4, Anlamsal Eşleştirme ve Etiketleme: Yeniden yapılandırılan bilinmeyen sinyallere anlamlı etiketler atamak için bu aşama kullanılır. Korelasyon, mesafe, istatistiksel ve bilgi-teorik ölçütler gibi on beş farklı benzerlik metriği kullanılarak, her bir yeniden yapılandırılmış sinyalin

davranışı, bilinen sinyal profillerinden oluşan bir referans veritabanıyla karşılaştırılır. En yüksek benzerlik skorunu veren referans sinyalin etiketi, bilinmeyen sinyale atanır. Bu sayede, "birleşik yakıt basıncı" veya "turbo hızı" gibi spesifik etiketler, ham sayısal verilere kazandırılır.

Aşama 5, Protokol Uyumlu Veri Üretimi ve Simülasyon: Çerçevenin son aşaması, önceki aşamalarda elde edilen tamamen yorumlanmış ve yeniden yapılandırılmış sinyal verilerini, protokol uyumlu SAE J1939 formatında sentetik veri izlerine dönüştürür. Bu süreç, sinyalleri doğru PGN (Parametre Grup Numarası) ve SPN (Şüpheli Parametre Numarası) yapılarına göre yeniden kodlamayı içerir. Bu aşamanın en önemli özelliklerinden biri, siber güvenlik araştırmaları için tasarlanmış entegre bir saldırı modülüdür. Bu modül, değer sahtekarlığı (spoofing), tekrar saldırıları (replay), veri yolu yoğunlaştırma (flooding) ve adres talep sahtekarlığı gibi farklı ailelerden protokol uyumlu kötü niyetli trafik üretebilir. Bu yetenek, Saldırı Tespit Sistemlerinin geliştirilmesi ve doğrulanması için ölçeklenebilir ve yüksek kaliteli veri setleri oluşturulmasını sağlar.

Önerilen çerçevenin etkinliğini doğrulamak için, çeşitli ağır vasıtalardan toplanmış dört farklı halka açık ve büyük ölçekli SAE J1939 veri seti kullanılmıştır. Sistemin performansını farklı bilgi seviyeleri altında test etmek amacıyla, üç ila yedi arasında değişen sayıda sinyal içeren beş farklı "çapa sinyal konfigürasyonu" (Minimal, Standart, Kapsamlı, Motor Odaklı ve Dinamik Odaklı) tanımlanmıştır. Hem geleneksel makine öğrenmesi modelleri (Random Forest, XGBoost, Extra Trees vb.) hem de derin öğrenme modelleri (LSTM, 1D-CNN, Transformer vb.) bu konfigürasyonlar üzerinde test edilmiştir.

Elde edilen sonuçlar, önerilen çerçevenin yüksek başarı oranını ortaya koymaktadır:

Sınıflandırma Performansı (Aşama 2): Sinyal tanımlama görevinde, özellikle Extra Trees ve Random Forest gibi ağaç tabanlı topluluk modelleri, diğer tüm model türlerinden sürekli olarak daha iyi performans göstermiştir. Tüm model-konfigürasyon kombinasyonlarında ortalama %91.54 gibi etkileyici bir doğruluk elde edilmiş, bazı özel durumlarda (örneğin, Extra Trees modelinin motor odaklı konfigürasyonda) %100 mükemmel doğruluğa ulaşılmıştır.

Yeniden Yapılandırma Performansı (Aşama 3): Sinyal yeniden yapılandırma görevinde de Extra Trees Regressor gibi ağaç tabanlı modeller en yüksek performansı sergilemiştir. En zengin çapa seti olan "Kapsamlı" konfigürasyon kullanıldığında, modelin R^2 (belirleme katsayısı) skoru 0.964'e ulaşmıştır. Bu, çapa sinyallerindeki bilginin, diğer sinyallerin varyansının %96.4'ünü açıklayabildiği anlamına gelmektedir. Bu bulgu, tezin temel hipotezlerinden birini, yani az sayıda çapa sinyalinin tüm ağı yeniden yapılandırmak için yeterli olabileceğini güçlü bir şekilde doğrulamaktadır.

Anlamsal Eşleştirme Performansı (Aşama 4): Bu aşamada elde edilen sonuçlar olağanüstü olmuştur. Değerlendirilen 15 farklı benzerlik metriğinden 14'ü, 23 sinyalin tamamını kendi referans karşılıklarıyla mükemmel bir şekilde (%100 doğrulukla) eşleştirmiştir. Bu, modern benzerlik ölçütlerinin bu tür zaman serisi verilerini eşleştirmede son derece etkili olduğunu göstermektedir.

Veri Üretimi ve Simülasyon Kalitesi (Aşama 5): Çerçevenin son aşaması, tüm önceki aşamaların başarılı bir şekilde entegre edildiğini doğrulamıştır. Üretilen sentetik veri izleri, hem istatistiksel olarak referans verilere yüksek benzerlik göstermiş hem de J1939 protokol standartlarına %99.8 oranında uyum sağlamıştır. Sistem, 300 saniyelik bir veri izini ortalama 8.3 saniyede üreterek yüksek hesaplama verimliliği sergilemiştir. Ayrıca, entegre saldırı modülü, önceden tanımlanmış tüm saldırı senaryolarını başarılı bir şekilde veri akışına enjekte edebilmiştir.

Bu tez çalışması, SAE J1939 CAN veri yolları üzerindeki otomotiv sinyallerini otomatik olarak tanımlamak ve simüle etmek için kompakt, yorumlanabilir ve doğrulanmış bir temel çerçeve sunmaktadır. Sonuçlar, ExtraTrees ve XGBoost gibi hesaplama açısından verimli ve şeffaf makine öğrenmesi modellerinin, dikkatli bir öznitelik mühendisliği ile birleştirildiğinde, bu karmaşık görev için oldukça yeterli olduğunu göstermiştir.

Geliştirilen çerçevenin en önemli katkılarından biri, otomotiv siber güvenliği alanına sağladığı değerdir. Gerçekçi ve düşmanca CAN izleri sentezleme yeteneği, Saldırı Tespit Sistemlerinin (IDS) test edilmesi, dayanıklılık analizleri ve güvenlik açığı taramaları için büyük ölçekli ve düşük maliyetli test verileri üretilmesine olanak tanır. Modellerin düşük çıkarım maliyeti ve deterministik yapısı, onları gömülü veya uç cihazlarda gerçek zamanlı izleme uygulamaları için de uygun kılmaktadır.

Çerçevenin sağlamlığına rağmen, gelecekteki araştırmalar için bazı sınırlamalar ve yollar mevcuttur. Mevcut boru hattı, sabit pencereleme parametreleri kullanmaktadır; gelecekte, farklı zaman ölçeklerinde değişen sinyallere daha duyarlı olmak için uyarlanabilir pencereleme teknikleri uygulanabilir. Ayrıca, çerçevenin daha düşük kaliteli veya daha nadir görülen sinyalleri de kapsayacak şekilde genişletilmesi, uygulanabilirliğini artıracaktır. En kritik gelecek adımlardan biri, mevcut çerçeveyi parçalı veri setleri üzerinde çalışmaktan çıkarıp, gerçek zamanlı veri akışları üzerinde çıkarım yapabilecek şekilde uyarlamaktır.

Sonuç olarak, bu tez çalışması, yorumlanabilir ve hesaplama açısından hafif makine öğrenmesi boru hatlarının, otomotiv sinyali tersine mühendisliği, simülasyon ve siber güvenlik testlerinin pratik taleplerini etkili bir şekilde karşılayabileceğini göstermektedir. Burada kurulan temel, topluluğun üzerine inşa edebileceği, konuşlandırılabilirlik, açıklanabilirlik ve güvenliği odakta tutarak daha gelişmiş yaklaşımların geliştirilmesini teşvik eden sağlam bir zemin sağlamaktadır.

ACKNOWLEDGEMENTS

I owe my deepest thanks to my supervisor Prof.Dr. Mustaga Gök, who was more than just a professor. During our years of working together, he became a constant source of support. Our conversations, whether about research or life, often encouraged me to think more broadly. His mentorship, trust, encouragement, and thoughtful feedback shaped not only this work and my journey, but also my personality.

Special thanks to Assoc. Prof. Dr. Yusuf Kuvvetli for generously sharing resources, and to Ali Karaman and Abdullah Jalloul, whose friendship and wisdom inspired me. They showed me that true learning and passion go beyond the classroom.

I would also like to thank my thesis committee for their valuable feedback. I also wish to acknowledge Prof. Dr. Murat Aksoy, who was a member of my committee in the early stages of this work.

I remember with respect those who gave us hope—the people who left their classrooms and workplaces to struggle for a better life and for justice, and who became martyrs. Their courage and sacrifice remain a source of inspiration. I am also grateful to my friends who never lost hope; their resilience and optimism were a constant motivation throughout this journey.

Finally, I am deeply grateful to my wife, kids, family and friends for their love, patience, and unwavering support. Their encouragement has been my greatest source of strength.

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

ACK : Acknowledgement
CAN : Controller Area Network
CAN-FD : CAN with Flexible Data-Rate
CNN : Convolutional Neural Network
CRC : Cyclic Redundancy Check
DBC : CAN Database
DLC : Data Length Code
ECU : Electronic Control Unit
EOF : End-of-Frame
GRU : Gated Recurrent Unit
IDS : intrusion detection systems
KNN : K-Nearest Neighbors
LSTM : Long Short-Term Memory
MAE : Mean Absolute Error
MLP : Multilayer Perceptron
MSE : The Mean Squared Error
OSI : Open Systems Interconnection
PGN : Parameter Group Number
RBF : radial basis function
RNN : Recurrent Neural Networks
SAE : Society of Automotive Engineers
SOF : Start of Frame
TP.CM : Transport Protocol - Connection Management
TP.DT : Transport Protocol - Data Transfer

1. INTRODUCTION

1.1. Motivation

The modern vehicle operates as a complex cyber-physical system. It contains a network of dozens or even hundreds of interconnected Electronic Control Units (ECUs) (Mariño et al., 2022). These units manage a wide range of functions, from engine performance and braking to passenger comfort. The Controller Area Network (CAN) bus serves as the core of this digital ecosystem. It was originally designed to simplify vehicle wiring. As a reliable serial communication protocol, the CAN bus has been the standard for in-vehicle networking for more than thirty years (Checkoway et al., 2011). The protocol enables real-time distributed control, allowing ECUs to broadcast operational data throughout the vehicle. This continuous flow of messages contains vital information, such as vehicle speed, engine RPM, and fuel levels, which is essential for the vehicle's operation.

Despite its critical role, the language of the CAN bus is not openly accessible. Original Equipment Manufacturers (OEMs) use proprietary formats, defined in database files (DBCs), to encode signal information within CAN messages. They consider these definitions valuable intellectual property and rarely make them public. This practice creates a significant obstacle for many legitimate and important activities (Buscemi et al., 2023). Automotive technicians, third-party device manufacturers, academic researchers, and cybersecurity analysts all face a large volume of raw, uninterpreted data. Without access to these proprietary formats, understanding the vehicle's internal state is an extremely difficult task. Consequently, diagnosing complex faults, developing new aftermarket products, or assessing security vulnerabilities becomes a major challenge. This situation reveals a critical gap in the field, motivating the need for a framework that addresses two interconnected challenges: the reverse engineering of proprietary signals and the generation of realistic data for research and development.

1.2. Problem Statement

The motivation outlined previously leads to two distinct yet related research problems. The first problem concerns the semantic interpretation of CAN traffic without access to proprietary documentation. Manually reverse-engineering signals is labor-intensive, requires specialized expertise, and does not scale well across different vehicle models. This limitation necessitates the development of automated, data-driven methods. Formally, let a time-ordered sequence of raw CAN payloads for a specific CAN ID be $P = p_1, p_2, \dots, p_T$, This work assume a deterministic tokenizer, τ, maps each payload p_t to a vector of K signal values $f_t = \left(v_{(t,1)}, v_{(t,2)}, \dots, v_{(t,K)}\right)$ based on fixed bit boundaries. The fundamental problem is to discover the unknown semantic mapping:

$$M: \{v_{t,k}\}_{t=1}^{T} \rightarrow L_k \quad \text{for each} \quad k \in \{1, \dots, K\} \tag{1}$$

Here, L_k represents the correct semantic label, such as vehicle speed or engine RPM, for the k^{th} signal token. The primary objective of this thesis is to approximate this mapping by developing a multi-stage learning framework that can accurately assign semantic labels to signals derived from pre-tokenized CAN data.

The second problem is the scarcity of high-quality datasets, which impedes progress in automotive cybersecurity research. Collecting comprehensive real-world CAN data is logistically complex, expensive, and often fails to capture the full spectrum of operational states or potential adversarial attacks. This data gap makes it exceptionally difficult to train, validate, and benchmark new systems, such as intrusion detection systems (IDS). Therefore, a pressing need exists for a framework capable of generating high-fidelity, protocol-valid synthetic CAN traces that can realistically simulate both benign and malicious network traffic.

1.3. Key Contributions

This thesis makes several contributions to the fields of automotive reverse engineering and cybersecurity. The primary contribution is a novel framework that fully automates the interpretation of vehicle network data. This system methodically converts pre-tokenized CAN signals into complete, simulatable data traces. The process integrates several key stages: data preprocessing, anchor signal classification, multi-signal reconstruction, and semantic matching.

The research demonstrates that a small, carefully selected set of "anchor" signals contains enough information to reconstruct several signals with high fidelity. The framework's effectiveness is confirmed by strong performance metrics. It achieves high classification accuracies, a high coefficient of determination (R^2) for signal reconstruction, and assigns semantic labels with high accuracy.

Another contribution is a thorough empirical analysis comparing classical machine learning algorithms with modern deep learning architectures for signal classification and reconstruction, offering insights into the trade-offs between model complexity and performance on real-world automotive data. These quantitative results are discussed in Chapter 5.

A final contribution is the framework's capacity to synthesize realistic and protocol-valid CAN bus logs. This function includes an integrated attack module that generates distinct families of malicious traffic. This capability provides the cybersecurity community with a scalable tool to advance the development and validation of robust intrusion detection systems.

1.4. Research Questions

This research addresses significant challenges in automotive diagnostics and cybersecurity. It focuses on automating the reverse engineering and simulation of CAN bus signals that operate under the SAE J1939 protocol. The central investigation explores how a modular, multi-stage

machine learning framework can effectively automate the reverse engineering of proprietary signals with high accuracy. A primary question is whether a small set of known "anchor" signals can be used to accurately classify and reconstruct a much larger set of unknown signals from the network traffic. The study also compares the performance of traditional machine learning models, such as tree-based ensembles, against deep learning models for these classification and reconstruction tasks. Following signal reconstruction, the research investigates how accurately semantic labels like "Engine Speed" can be assigned to unknown signals by matching their behavior against a predicted data. Finally, the study examines whether this fully interpreted signal data can generate realistic, protocol-valid CAN bus traces. It also questions if this synthesis process can be extended to controllably inject adversarial attacks, thereby creating high-fidelity datasets for developing and benchmarking intrusion detection systems.

1.5. Thesis Organization

This thesis is structured into six chapters, each addressing a specific aspect of the research.

Chapter 1: Introduction provides a comprehensive overview of the research. It outlines the motivation driving this study, formally defines the problem statement, and presents the core research questions. This chapter also highlights the key contributions of the work.

Chapter 2: Background and Literature Review delves into the foundational concepts and prior work relevant to this thesis. It begins with an explanation of CAN protocol and the SAE J1939 standard. The chapter then discusses the challenges inherent in CAN bus reverse engineering, reviews existing literature, and identifies the research gap that this thesis aims to address.

Chapter 3: Proposed Framework details the novel framework developed in this research. It presents an overview of the framework and then elaborates on each of its stages: Data Preprocessing, Anchor Signal Classification, Multi-Signal Reconstruction, Semantic Matching and Labeling, and Protocol-Valid Trace Generation.

Chapter 4: Experimental Design and Setup describes the methodology used to validate the proposed framework. This includes a description of the datasets used, the data preparation pipeline, anchor signal configurations, and the traditional and deep learning models employed. It also defines the evaluation metrics and the computational environment used for the experiments.

Chapter 5: Results and Discussion presents and analyzes the empirical results obtained from the experiments. The findings are organized according to the stages of the framework, covering anchor classification results, multi-signal reconstruction results, semantic matching performance, and the quality of the generated traces.

Chapter 6: Conclusion and Future Work concludes the thesis by summarizing the key findings and their implications. It revisits the research questions and contributions, and suggests potential directions for future research in this area.

2. BACKGROUND AND LITERATURE REVIEW

The increasing integration of electronic systems in modern vehicles establishes the Controller Area Network (CAN) bus as the standard for in-vehicle communication. The CAN protocol and its derivatives, such as the SAE J1939 standard, form the central nervous system of automotive platforms. These protocols were originally designed to improve vehicle reliability and simplify wiring. However, Original Equipment Manufacturers (OEMs) often keep their specific communication protocols proprietary. This secrecy creates a significant barrier to independent diagnostics, third-party innovation, and cybersecurity research. This "security through obscurity" approach necessitates reverse engineering to decode and interpret the data streams within a vehicle. This review provides a foundational understanding of the CAN and J1939 protocols. It also examines the challenges inherent in reverse engineering them. Furthermore, it surveys current methodologies developed to address these challenges and identifies critical gaps in research that highlight the need for more advanced, end-to-end frameworks.

2.1. The Controller Area Network (CAN) Protocol

The Controller Area Network protocol originated in the 1980s as a solution from Robert Bosch GmbH. It addressed the growing complexity of wiring harnesses that connected numerous Electronic Control Units (ECUs). The protocol's design prioritizes robustness, real-time capability, and cost-effectiveness. It enables direct, multi-master communication among ECUs without a central controller. Its codification as the ISO 11898 standard solidified its dominance in in-vehicle networks. Today, it remains a prevalent protocol across the automotive, industrial, and agricultural sectors due to its proven reliability in harsh electrical environments.

The CAN standard defines the two lowest layers of the Open Systems Interconnection (OSI) model: the data link layer and the physical layer. The most common physical implementation, high-speed CAN, uses a two-wire twisted-pair differential signaling scheme. This configuration provides high immunity to electromagnetic interference, which is essential for reliable operation. To maintain signal integrity, the bus requires termination at both ends with 120 Ω resistors.

At its core, CAN is a message-based broadcast protocol where all messages, or frames, are transmitted to every node on the network. The standard CAN 2.0 specification defines two frame formats. The standard format uses an 11-bit identifier, while the extended format uses a 29-bit identifier. The extended format serves as the foundation for higher-layer protocols like SAE J1939. A standard data frame is a structured packet containing several key fields. These include a Start-of-Frame (SOF) bit, an Arbitration ID for message priority, a Data Length Code (DLC), and a Data Field of 0 to 8 bytes. The frame also contains a Cyclic Redundancy Check (CRC) for error detection, an Acknowledgement (ACK) slot for reception confirmation, and an End-of-Frame (EOF) sequence.

A defining feature of the CAN protocol is its non-destructive, bitwise arbitration process for bus access. The physical layer functions as a "wired-AND" bus, where a dominant state (logic 0) overrides a recessive state (logic 1). When multiple nodes transmit simultaneously, each node monitors the bus. If a node sends a recessive bit but detects a dominant one, it stops transmitting and enters a listening state. Because the Arbitration ID is sent first, the message with the numerically lowest ID wins arbitration and completes its transmission. This mechanism ensures that high-priority messages are always transmitted successfully, a critical feature for real-time control systems.

The protocol continues to evolve to meet increasing bandwidth demands. CAN with Flexible Data-Rate (CAN FD) expands the data payload to 64 bytes and supports higher transmission speeds. More recently, the CAN XL standard accommodates even larger payloads up to 2048 bytes, catering to data-intensive systems like ADAS.

However, the design principles that make CAN reliable also create fundamental security weaknesses. The protocol was designed for a physically isolated and trusted network. This foundational assumption led to the omission of core security features like message authentication and data encryption. The design solves the problem of wiring complexity but assumes all connected ECUs are non-malicious. Consequently, messages are identified by their content, not their sender. When modern vehicles introduced external connectivity, this "closed system" assumption was broken. The lack of authentication now represents a critical vulnerability. Any compromised node can broadcast messages with any valid ID, effectively impersonating critical ECUs. This allows for the injection of malicious commands, such as spoofing sensor readings. The arbitration mechanism can also be exploited for Denial-of-Service (DoS) attacks, where a malicious node saturates the bus with high-priority messages. Thus, the protocol’s original architecture has become its primary vulnerability in a connected world.

2.2. The SAE J1939 Standard

While the CAN protocol provides the physical and data link layers, it does not define the content of the data being exchanged. Higher-layer protocols fulfill this role. Among them, the Society of Automotive Engineers (SAE) J1939 standard is preeminent in the heavy-duty vehicle sector. Built upon the CAN 2.0B extended format, J1939 establishes a standardized dictionary and set of communication rules. This ensures interoperability between ECUs from different manufacturers.

The core of J1939 is its structured approach to message definition. An 18-bit Parameter Group Number (PGN), encoded within the 29-bit CAN identifier, uniquely identifies each message. The PGN specifies the message's function and the type of data it contains, such as engine temperature or wheel speed. The 8-byte data payload of a PGN is further organized into individual signals known as Suspect Parameter Numbers (SPNs). Each SPN is defined with a specific starting bit, length, byte order, scaling factor, offset, and physical unit. This PGN-SPN structure creates a comprehensive data dictionary, allowing any compliant device to interpret messages correctly. The full 29-bit CAN

identifier in a J1939 network also includes fields for priority and the source address of the transmitting ECU.

To handle messages larger than 8 bytes, the J1939 standard includes a dedicated transport protocol. This protocol enables the transmission of multi-packet messages containing up to 1785 bytes of data, which is essential for diagnostics or reprogramming. It operates through two primary mechanisms managed by specific PGNs. The Transport Protocol - Connection Management (TP.CM) message initiates and manages the data transfer, supporting both one-to-all broadcast and one-to-one connection-oriented transfers with flow control. The Transport Protocol - Data Transfer (TP.DT) frames carry the actual data segments, each with a sequence number to ensure correct reassembly.

The standard also specifies a decentralized network management protocol. Upon startup, each ECU claims a unique 8-bit source address through an address-claiming procedure. If another ECU does not contest the claim, the address is assigned. In case of a conflict, the protocol resolves the dispute based on a unique 64-bit NAME field, which encodes the ECU's function and manufacturer. The ECU with the higher-priority NAME wins the address. Finally, the J1939-71 document defines the application layer, detailing the vast majority of standardized PGNs and SPNs used in the industry.

The act of standardization, while beneficial for interoperability, creates a significant security challenge. Before J1939, an OEM's proprietary message set was an isolated system. Reverse engineering one vehicle offered little insight into another. The J1939 standard, however, creates a common language. For example, PGN 65262 consistently represents engine temperature across vehicles from different manufacturers. This standardization was intended to benefit technicians and fleet management. The unintended consequence is a standardized attack surface. An attacker who discovers a vulnerability related to a standard J1939 PGN can develop an exploit that applies to an entire class of heavy-duty vehicles. The efficiencies gained through standardization have inadvertently made attacks more generalizable and potent.

2.3. Machine Learning Models for Classification and Reconstruction

Machine learning offers a powerful arsenal of tools for data analysis, primarily addressing the two fundamental supervised learning paradigms: classification and regression (Pedregosa, 2011). Classification algorithms are designed to assign observations to predefined, discrete categories, effectively learning a mapping from input variables to class labels. In contrast, regression algorithms focus on predicting continuous numerical outcomes, modeling the relationship between independent and dependent variables. Many of the most sophisticated machine learning techniques are remarkably versatile, offering specialized implementations tailored to solve both types of predictive tasks (Hastie et al., 2001).

Tree-Based Ensemble Methods

The decision tree serves as the foundational building block for some of the most powerful and widely used machine learning models (Song & Lu, 2015). This intuitive, non-parametric model functions by partitioning the feature space into a set of hierarchical, rule-based decisions, creating a tree-like structure that is highly interpretable. However, individual decision trees are often prone to high variance and overfitting. To overcome these limitations, ensemble techniques are employed, which strategically combine multiple decision trees to yield a more robust and accurate model.

The Random Forest algorithm, a quintessential ensemble method, leverages the principle of bootstrap aggregating, or "bagging" (Breiman, 2001). It constructs a multitude of decorrelated decision trees by training each one on a random subset of the data and considering only a random subset of features at each split. For classification, the final prediction is determined by a majority vote among the trees; for regression, the predictions are averaged. This dual randomization strategy effectively reduces variance and enhances generalization performance. A closely related technique, the Extra Trees (Extremely Randomized Trees) algorithm, introduces an additional layer of randomization by selecting the optimal split-point for each feature at random rather than deterministically (Geurts et al., 2006). This can further diminish model variance and may offer computational advantages.

Gradient Boosting Machines

Boosting algorithms represent another preeminent class of ensemble models, but they operate on a different principle: sequential learning. Rather than building independent models in parallel, boosting methods construct an ensemble iteratively, where each new model is trained to correct the residual errors of its predecessors. This sequential, error-correcting process often culminates in state-of-the-art predictive accuracy.

Prominent implementations such as XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017) provide highly optimized and scalable frameworks for gradient boosting. These algorithms are renowned for their exceptional performance on structured and tabular data, incorporating advanced features like regularization, parallel processing, and efficient handling of missing values to achieve superior results in both classification and regression contexts.

Interpretable Linear Models

In contrast to the complexity of tree-based ensembles, linear models offer an alternative paradigm characterized by simplicity, efficiency, and high interpretability. For classification tasks, Logistic Regression is a fundamental and widely applied method that models the probability of a categorical outcome by applying a logistic function to a linear combination of features (Chung, 2020).

In the regression domain, standard linear models can be extended to handle challenges like multicollinearity and overfitting. Ridge regression incorporates L2 regularization, which penalizes the sum of the squared coefficient values, thereby shrinking them towards zero to prevent model complexity (Cheng & Montanari, 2025). Building upon this, Bayesian Ridge regression introduces a probabilistic framework, treating the model parameters not as fixed values but as random variables(Shi et al., 2016). This Bayesian approach allows the model to determine the regularization strength automatically and provides a more robust estimation of the model's parameters and their uncertainty.

Other Algorithmic Paradigms

Beyond tree-based and linear models, other algorithmic approaches provide unique capabilities for data analysis. The K-Nearest Neighbors (KNN) algorithm is a non-parametric, instance-based method that makes no assumptions about the underlying data distribution (Peterson, 2009). Predictions are made by identifying the 'k' closest data points in the feature space and, for classification, assigning the class that represents the majority among those neighbors.

Finally, the Gaussian Naive Bayes classifier is a simple yet powerful probabilistic model grounded in Bayes' theorem (Murphy, 2006). Its core "naive" assumption is the conditional independence of all features given the class label. While this assumption is rarely true in practice, the model is computationally efficient and often serves as a strong performance baseline. It assumes that the continuous values associated with each class are distributed according to a Gaussian (normal) distribution, enabling rapid and effective classification. Together, this comprehensive suite of algorithms provides a versatile foundation for addressing a vast spectrum of data analysis challenges.

2.4. Deep Learning for Classification and Reconstruction

This study utilizes a variety of deep learning models to analyze automotive signal data. The selected models represent three main architectural families: recurrent, convolutional, and attention-based networks. Each architecture is chosen for its specific advantages in processing sequential information. Additionally, a standard feed-forward network serves as a baseline to evaluate the performance of the more specialized models.

Recurrent Neural Networks (RNNs) (Medsker & Jain, 2001) are well-suited for time-series analysis due to their inherent ability to process sequential data. This research includes the Long Short-Term Memory (LSTM) (Graves, 2012) model, which excels at identifying long-term dependencies within the data through a gated mechanism. The study also employs the Gated Recurrent Unit (GRU) , a computationally efficient alternative to the LSTM that often delivers similar performance (Dey & Salem, 2017). Both the LSTM and GRU models are configured with options for unidirectional or bidirectional processing and multiple layers to capture complex temporal patterns effectively.

Convolutional Neural Networks (CNNs) typically associated with image analysis, are also effective for extracting features from one-dimensional (1D) sequential data (O'Shea & Nash, 2015). This study utilizes a multi-scale 1D CNN that employs parallel convolutional blocks with different kernel sizes. This design allows the model to simultaneously learn local patterns across short, medium, and long time scales. Furthermore, a 1D Residual Network (ResNet) is included. This architecture adapts the successful ResNet concept from computer vision by using skip connections, which helps prevent the vanishing gradient problem and allows for the construction of deeper, more powerful networks (Zagoruyko & Komodakis, 2017).

The investigation also includes models that use attention mechanisms or hybrid structures. The Transformer model relies exclusively on a multi-head self-attention mechanism to identify complex relationships between any two points in a data sequence (Vaswani et al., 2017). This approach allows it to capture long-range dependencies without using recurrent layers. A hybrid model is also employed, which combines a CNN with an LSTM. In this architecture, the CNN first extracts local features from the signal. An LSTM then processes these features sequentially to model their temporal relationships.

2.5. Challenges in CAN Bus Reverse Engineering

Reverse engineering CAN and J1939 traffic presents several formidable challenges, from proprietary information barriers to technical complexities. The greatest impediment is the proprietary nature of the CAN Database (DBC) file. This file serves as the definitive dictionary for a vehicle's network, mapping message identifiers to signals with all necessary decoding parameters. OEMs guard these files as valuable intellectual property and rarely disclose them. This practice forces researchers into a black-box analysis paradigm, where the network's logic must be inferred from observation alone.

Several technical complexities also make decoding a non-trivial task. Signals within a CAN payload are not always byte-aligned and can span an arbitrary number of bits, making boundary identification difficult. Data multiplexing further complicates this, as a single message ID can transmit different signals based on a specific multiplexor field. This dynamic structure means a fixed parsing approach is insufficient. For J1939 networks, the transport protocol for large messages introduces another layer of complexity. An analyst must first detect and reassemble multi-frame sequences to reconstruct the original data. Furthermore, the dynamic source addressing mechanism means an ECU may claim a different address upon each startup, complicating efforts to attribute signals to their source consistently.

The research field also faces significant methodological hurdles. There is a pervasive lack of public, large-scale, and well-annotated benchmark datasets. The proprietary nature of DBC files hinders the creation of datasets with ground-truth labels. This, in turn, impedes research reproducibility and makes objective comparisons between methodologies difficult. Raw CAN traffic

is also inherently noisy, containing a mix of periodic, event-driven, and diagnostic messages. Isolating meaningful signals from this noise requires extensive data filtering and careful synchronization with external data sources like GPS or video.

These challenges are a direct consequence of the automotive industry's economic model. OEMs keep DBC files confidential primarily to control the profitable aftermarket for diagnostics, repairs, and tuning. This business strategy creates an artificial information barrier that necessitates the entire field of CAN reverse engineering. It establishes a situation where both legitimate third parties and malicious actors must develop the same skills to overcome the same obstacle. The lack of public data is a direct result of this practice. Therefore, progress in the field is constrained not only by technical innovation but also by the industry's strategic opacity.

2.6. Review of Prior Art

The field of Controller Area Network (CAN) bus reverse engineering seeks to automate the decoding of manufacturer-specific data. This process generally involves two fundamental stages. The first stage, tokenization, aims to infer the structure of the data within a message. The second stage, translation, focuses on interpreting the semantic meaning of that data (Buscemi et al., 2020). Each stage employs a variety of distinct methodologies to achieve its goal.

Tokenization identifies the boundaries of individual signals within a CAN frame's data payload. One foundational method for this task relies on the Bit Flip Rate (BFR). This unsupervised approach analyzes how frequently each bit changes its state across many frames. A significant drop in this rate often indicates the boundary between two different signals. The READ tool, for instance, uses BFR to recover the data structure without any prior information (Marchetti & Stabili, 2018). However, this method does not determine properties such as the data's byte order or scaling factors. An alternative strategy treats tokenization as a combinatorial optimization problem. These methods systematically evaluate all possible signal combinations within the payload. A fitness score is then applied to identify the most likely set of non-overlapping signals (Markovitz & Wool, 2017). More advanced tools like CAN-D build upon this strategy by also inferring properties like byte order and signedness to achieve a more complete structural decoding (Verma et al., 2021).

After tokenization defines the data structure, translation assigns meaningful labels and physical values to the identified signals. One direct technique involves injecting Parameter ID (PID) requests through the vehicle's diagnostic port. Researchers can then find linear correlations between the known diagnostic data and the unknown signals to translate their meaning (Pesé et al., 2019). Another approach uses machine learning for fully automated translation. Supervised models, for example, can be trained on labeled data from one vehicle to classify signals in another. Unsupervised techniques like clustering help group semantically related frames, which can guide and focus the reverse engineering effort.

Further translation methods rely on different sources of information. The semantic taxonomy approach is a semi-automated process that requires a human operator to perform specific actions, such as locking the doors or activating the wipers. An algorithm then analyzes the resulting CAN traffic to identify and label the corresponding signals based on their expected behavior (Pesé et al., 2019). Frame matching offers a fully automated and non-intrusive alternative. This technique uses a large database of previously decoded vehicle data. It matches frames from a target vehicle to known frames in the database and transfers the corresponding signal definitions (Buscemi et al., 2023). A more recent strategy involves reverse engineering the companion mobile or desktop applications that interact with the vehicle. By analyzing the application's code, it is possible to automatically determine the syntax and meaning of the commands sent to the CAN bus (Wen et al., 2020; Yu et al., 2022).

Semantic interpretation aims to assign meaningful labels and physical units to these signals. AutoCAN (Huybrechts et al., 2018) represents a significant step in this area by employing supervised learning. It correlates statistical descriptors of CAN signals with externally measured ground-truth data from the OBD-II port. By finding strong correlations, it can assign specific semantic labels to unknown signals. Its primary limitation is its dependence on the availability of OBD-II data, as it can only identify signals with an OBD-II equivalent. Another method, CSI (Buscemi et al., 2020), takes a more focused approach. It assumes the structural tokenization is already complete and uses supervised machine learning to identify a predefined set of safety-critical signals, such as speed or RPM. This makes it effective for its specific purpose but not for comprehensive bus decoding.

A complementary area of research is sequence analysis, which focuses on message timing and order for intrusion detection. CAN-BERT (Nwafor & Olufowobi, 2022) applies the powerful BERT language model to the sequence of CAN message IDs. It treats the stream of IDs as a "language" and learns the normal temporal patterns of communication. Once trained on normal traffic, it can effectively detect anomalies when the observed sequence deviates from the learned patterns. While highly effective for security monitoring, this approach provides no insight into the structure or meaning of the message payloads.

The development of these methodologies shows a clear progression in the field. Early research focused primarily on syntax by working to discover the fundamental structure of the data payload. Subsequent work has increasingly concentrated on semantics, developing diverse strategies to assign meaning to these structures. This evolution highlights a clear trend toward more comprehensive and fully automated solutions for decoding CAN bus data.

2.7. Research Gap Analysis

A critical analysis of existing literature reveals several interconnected gaps that limit the applicability of CAN bus reverse engineering. First, a persistent gap exists between partial and comprehensive semantic interpretation. Current methods can assign meaning to a subset of signals,

but a general solution for decoding an entire bus with full, unit-calibrated semantics remains an open problem. Methodologies are either limited to coarse labels, constrained by available external data sources, or focused on a small set of predefined signals. The field lacks a scalable method to create a complete semantic map of a vehicle's communication network.

Second, there is a notable lack of specific focus on the SAE J1939 standard. Most foundational techniques treat all traffic as generic CAN 2.0 frames. This approach fails to leverage the rich, structured information embedded within the J1939 protocol itself. The standardized conventions of PGNs, SPNs, and transport protocols are valuable sources of information that could significantly improve the reverse engineering process.

Third, a critical limitation of nearly all prior work is the absence of a re-encoding and simulation capability. The existing research pipeline terminates at decoding and interpretation. The process lacks the ability to take reverse-engineered signal data, model its behavior, and then re-encode it back into a valid stream of CAN frames. This missing link from analysis back to synthesis prevents the application of reverse engineering insights to downstream tasks like high-fidelity simulation and the development of digital twins.

Finally, this lack of generative capability creates a fourth major gap: the scarcity of high-fidelity adversarial data for security research. The development and benchmarking of next-generation IDS depend on large, realistic datasets that include sophisticated, protocol-valid attack injections. Without the ability to synthesize such data, IDS research remains constrained, hindering the community's ability to develop robust defenses against emerging cyber threats.

These gaps form a systemic problem. The lack of J1939-specific models makes comprehensive semantic understanding more difficult. The inability to achieve full semantics, in turn, prevents the creation of accurate generative models needed for simulation. This absence of high-fidelity generative models then deprives the cybersecurity community of the essential adversarial data needed to validate security systems. This dependency chain shows that progress requires a holistic approach that integrates decoding, interpretation, modeling, and re-encoding into a single workflow. Such a framework would address all four gaps simultaneously and unlock new capabilities in vehicle diagnostics, simulation, and security.

3. PROPOSED FRAMEWORK

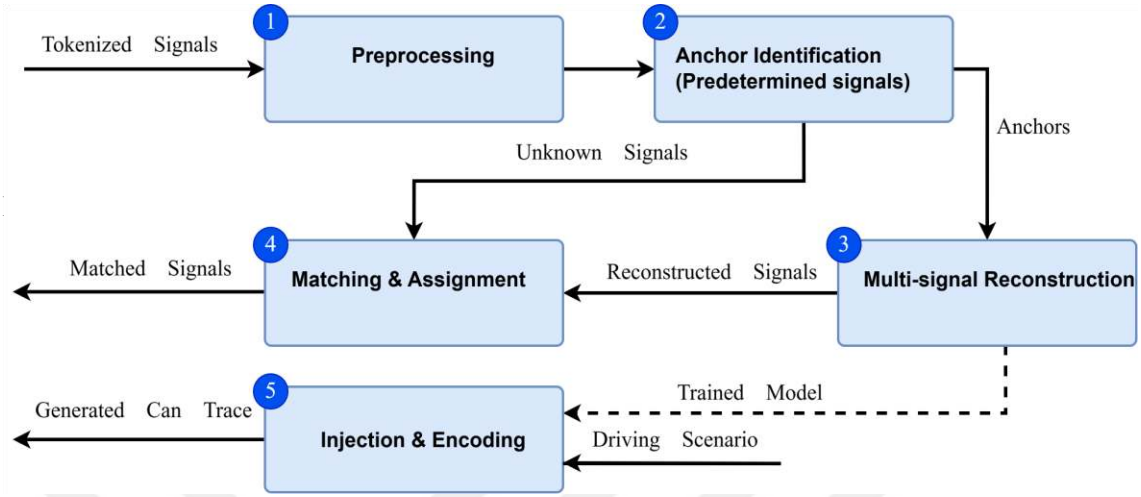

Figure 3.2 The proposed framework.

3.1. Framework Overview

The block diagram of the proposed modular, pipelined framework for analysis, simulation, and security evaluation of CAN bus log files is shown in Figure 2.1. The framework enforces identical preprocessing, windowing, and quality checks across stages to ensure comparability of results.

The framework's core strategy involves using a small, high-confidence subset of features, termed "anchors," to simplify the larger, more complex problem. This principle is a well-established method in several advanced scientific fields, which validates its application to CAN bus reverse engineering. For example, computer vision techniques like Structure from Motion (SfM) reconstruct 3D scenes by first matching a sparse set of key points that serve as anchors across multiple 2D images (Durrant-Whyte & Bailey, 2006). In bioinformatics, researchers infer complex gene regulatory networks by first identifying "master regulator" genes, which anchor the larger network structure (Marbach et al., 2012). Similarly, hierarchical forecasting models in time-series analysis often predict stable, high-level aggregates first. These aggregates provide a reliable constraint that improves the accuracy of lower-level forecasts (Hyndman & Athanasopoulos, 2018).

This framework applies the same proven principle by first identifying a compact set of anchor signals. This initial step creates a stable representation of the vehicle's state. Building on this foundation, the pipeline proceeds through five distinct stages. The first stage involves preprocessing and normalizing tokenized signals. The second stage classifies the anchor signals. The third stage reconstructs multiple unknown signals using multi-output regression. The fourth stage then matches these unknown signals to their semantics through a bounded similarity fusion. In the final stage, the

framework re-encodes protocol-valid traces, which allows for simulation and the optional injection of attacks for security evaluation. The subsequent sections of this paper describe each of these stages in detail.

The pipeline consists of five stages: (1) preprocessing and normalization of tokenized signals; (2) identification of anchor signals; (3) reconstruction of unknown signals; (4) assigning semantic labels to unknown signals; and (5) simulation and IDS benchmarking, including re-encoding of reconstructed signals into protocol-valid frames. Each stage is described in detail below.

It is assumed that access to a tokenizer that maps each 8-byte CAN/J1939 payload at time (t) into a fixed-length vector of unsigned integers $(f_t \in N^K)$. Alongside (f_t), the tokenizer records a bit-span map $(B = \{(b_k, \mathrm{l}_k)\}_{k=1}^{K})$ giving the start bit (b_k) and length (l_k) of each field; these spans are stable within a driving session. During training, These spans obtained from a J1939 DBC solely to create labeled supervision for models in Stages 1–4. At deployment, the models operate on unlabeled tokenized data produced by any deterministic tokenizer that yields stable spans. The learning models consume only (f_t); (B) is logged for reproducibility and is not used at inference. Fields are treated as as raw unsigned integers (no offset/scale applied).

3.2. Stage1, Data Preprocessing

The first stage prepares the dataset for subsequent analysis. To address the variability in CAN bus log file formats and to obtain a ground truth, each data capture was decoded using SAE~J1939 DBC files. These files provide essential information such as Parameter Group Numbers (PGNs), bit offsets, lengths, and scaling factors. Only the bit boundaries are kept and the values are left as unsigned integers. This format is close to the state an engineer would encounter when reverse-engineering an unknown log.

Frames are resampled on a uniform 100 ms grid; each row corresponds to a timestamp and each column to a tokenized field value, with short gaps linearly interpolated when necessary.

Once the data were in a unified format, a series of quality checks and preprocessing steps were performed. This involved a multi-stage pipeline that transforms raw logs into balanced, high-quality time-series data. The pipeline includes signal quality assessment, missing-data imputation, and signal normalization. Specifically, signals with (>30%) missing samples or with stuck values exceeding a preset dwell threshold are removed and a three-step imputation strategy is applied to handle any missing data.

The final dataset had minimal missing entries, reduced redundancy, and a balanced representation across different driving states. A simple score from 0 to 100 was assigned to each signal by combining its shares of missing and stuck values, value range, motion-state coverage, and

timing stability. The top-scoring signals formed the anchor set. The anchor set is used as the target classes in Stage 2.

The small, labeled set of signals are called anchors. Signals without labels are called unknowns and are matched in Stage 4.

3.3. Stage 2, Anchor Signal Classification

The second stage of the methodology assigns unknown signals to predefined anchor classes. These anchor classes, which represent fundamental vehicle dynamics, are systematically organized into five categories as detailed in Section 4.3. The process utilizes sequences of unlabeled and normalized data windows as input. A range of machine learning classification models is trained to recognize the distinct temporal patterns and statistical characteristics inherent to each anchor signal. The models then assign a specific class label, such as "Engine Speed" or "Vehicle Speed," to each input window. This step produces a set of accurately classified anchor signals, which provide a reliable, low-dimensional summary of the vehicle's operational state at any given moment. These classified signals also serve as essential predictors for the multi-signal reconstruction performed in Stage 3, establishing a strong foundation for identifying the remaining unknown signals.

For the classification task, a variety of machine learning techniques are evaluated to determine the most effective approach. The models assessed include XGBoost, Random Forest, Extra Trees, Logistic Regression, K-Nearest Neighbors, Decision Tree, and Gaussian Naive Bayes, with LightGBM also included when available. Each model is tested using multiple configurations and hyperparameter settings to ensure a thorough assessment. Model performance is compared using a comprehensive set of metrics, including overall accuracy, macro and micro F1 scores, and per-class precision and recall. The evaluation also considers computational cost, examining confusion matrices, parameter counts, and CPU training time to provide a complete performance profile for anchor signal identification.

3.4. Stage3, Multi-Signal Reconstruction

The third stage leverages the identified anchor signals to reconstruct the remaining unknown signals in the vehicle network. The primary goal is not to forecast future values but to perform a functional reconstruction predicting the value of multiple unknown signals using a window of concurrent anchor signals. This approach preserves the instantaneous amplitude and phase relationships between signals, which is essential for identifying signals based on their real-time behavior. Success in this stage demonstrates that a small, well-chosen set of anchor signals contains sufficient information to describe the state of the broader vehicle system.

To systematically evaluate how different anchor signal sets perform, The same five configurations is used as in the classification stage (Minimal, Standard, Comprehensive, Engine-Focused and Dynamics-Focused). Each configuration defines a mapping from a small set of anchor

signals to a larger set of unknown signals. This setup allows a direct comparison of how anchor set composition affects reconstruction accuracy across the entire vehicle network.

A key methodological choice is the use of functional reconstruction over temporal forecasting. For a given window of anchor signal data, the model's objective is to predict the mean value of each unknown signal over that same window. This is distinct from forecasting, which would introduce temporal shifts that could obscure these relationships.

For evaluation, a range of traditional regression models are trained; these models are selected for diverse methodology and computational efficiency. The models include Linear Regression (baseline), Ridge Regression (L2 regularization), Lasso Regression (L1 regularization), ElasticNet (combined L1 and L2), and a decision tree regressor (maximum depth 12) to capture non-linear relationship

3.5. Deep Learning Integration in the Framework

While the proposed framework relies primarily on interpretable and computationally efficient machine learning models, deep learning (DL) architectures were also integrated into critical stages of the pipeline to evaluate their potential advantages. DL models are well suited to capture nonlinear temporal dependencies and hierarchical feature representations, which are particularly relevant for complex automotive signal dynamics. In the anchor signal classification stage (Stage 2), architectures such as one-dimensional Convolutional Neural Networks (1D-CNNs), Recurrent Neural Networks (LSTMs and GRUs), CNN–LSTM hybrids, and Transformer-based models were explored alongside traditional classifiers. These networks process tokenized time-series data directly, reducing reliance on handcrafted statistical features and offering improved adaptability to diverse signal patterns. In the multi-signal reconstruction stage (Stage 3), DL regressors—including MLP variants with residual connections, ResNet-inspired architectures, sequence-based hybrids (LSTM_MLP, GRU_MLP, GRU_Attention_MLP), and Transformer regressors—were employed to reconstruct unknown signals from a limited set of anchors. Generative and ensemble variants (VariationalMLP, EnsembleMLP) were further examined to assess robustness and latent representation learning.

The methodological integration of these models is summarized in Figure 3.2, where panels (A–D) illustrate the DL classifiers used for anchor signal identification and panels (E–N) depict the DL regressors for signal reconstruction. By grouping all models in a unified overview, the figure emphasizes how deep learning was systematically incorporated into both classification and reconstruction tasks, enabling structured comparisons with machine learning baselines in terms of accuracy, computational cost, and interpretability. Full training configurations, hyperparameters, and evaluation criteria are detailed in Section 4.6, while empirical results and ML–DL performance comparisons are presented in Sections 5.1 and 5.

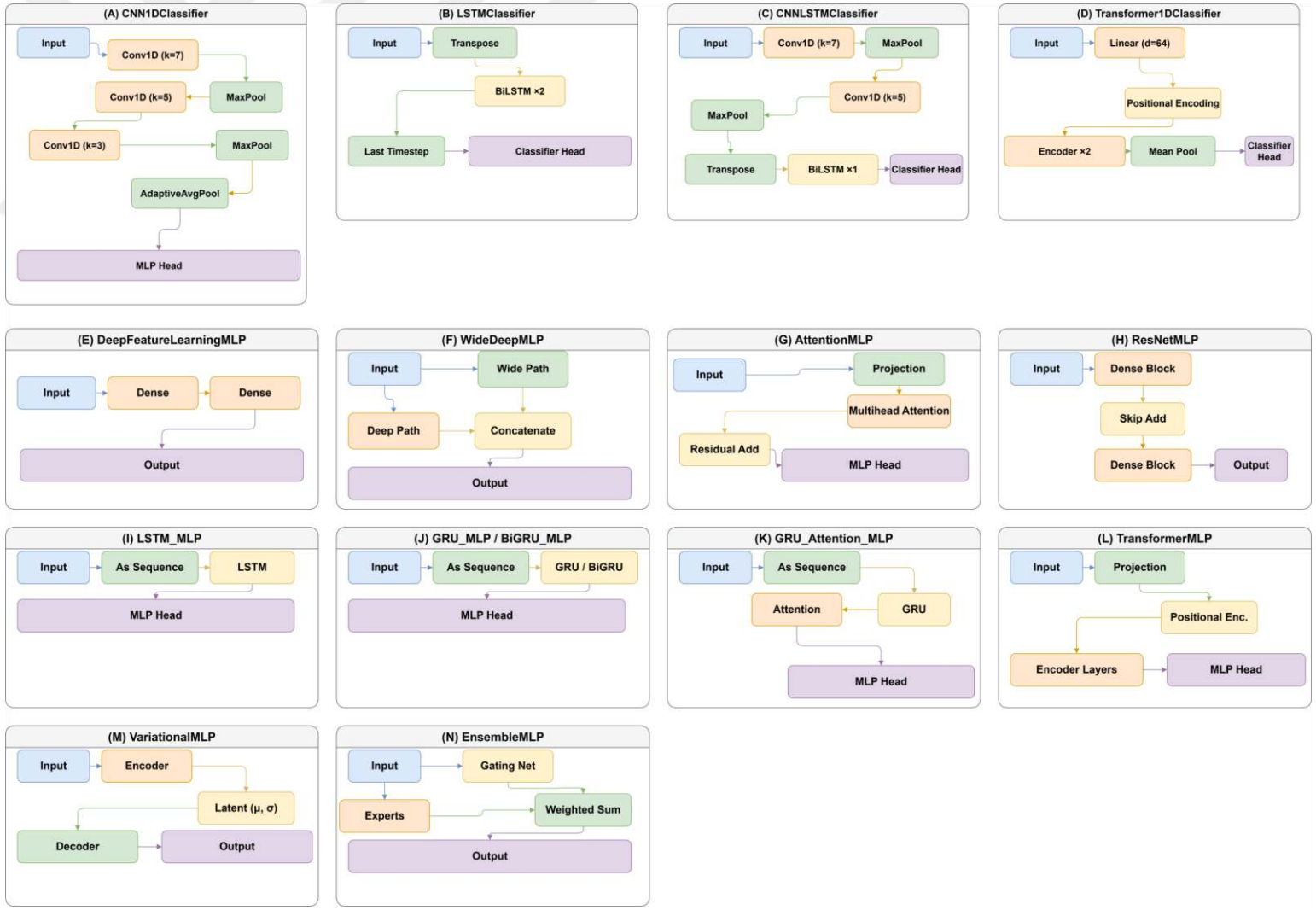

Figure 3.3 DL models that are used in this work A-D used for classification, others are for reconstruction.

3.6. Stage4, Semantic Matching and Labeling

In this these fifteen similarity (and distance) measures are used and organized into five categories: correlation-based, distance-based, statistical, information-theoretic, and hybrid. For each method, This stage uses the mathematical definition, its native range, and a consistent normalization to a similarity score ($S \in [0,1]$) that enables fair comparison across heterogeneous metrics.

3.6.1. Notation and Pre-processing

Let $(x = (x_1, \dots, x_n))and(y = (y_1, \dots, y_n))$ denote two time-aligned real-valued signal segments of equal length (n).

Unless explicitly stated, the following is assumed:

1. Signals are sampled or resampled on a common time grid; short gaps are linearly interpolated.
2. For scale-sensitive distances (e.g., Euclidean, Manhattan, MSE/MAE), optionally standardize
3. $$\tilde{x_i} = \frac{x_i - \mu_x}{\sigma_x}, \tilde{y_i} = \frac{y_i - \mu_y}{\sigma_y} \quad (3.1)$$

   where (μ_x, σ_x) are the empirical mean and standard deviation of (x) (and analogously for (y)). When standardization is applied, $(\tilde{x}, \tilde{y})$ is used in place of (x, y).
4. To map each raw measure to a comparable similarity ($S \in [0,1]$), the work uses the method-specific transformations defined below. For distance-like quantities, robust caps (e.g., (95^{th}) percentile over a calibration set) is applied to stabilize the mapping.

If $(\sigma_x = 0)$ or $(\sigma_y = 0)$, correlation-based and cosine measures are undefined; in these cases their similarity is set to (0) and rely on distributional or distance-based measures. For missing values, paired indices with NaNs are dropped (or imputed consistently across all pairs), and the effective sample size (n) is reported

3.6.2. Correlation-based Methods

Pearson Correlation ("VII. Note on Regression and Inheritance in the Case of Two Parents," 1895)
The Pearson correlation coefficient is

$$r_p = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \qquad r_p \in [-1,1]. \quad (3.2)$$

Similarity:

$$S_{\text{pearson}} = |r_p| \tag{3.3}$$

Pearson measures linear association and is invariant to affine scaling; it can be sensitive to outliers.

Spearman Correlation (Wissler, 1905)

Let $(R(x_i))$ and $(R(y_i))$ be the ranks of (x_i) and (y_i) (with average ranks for ties). Spearman's (ρ) is the Pearson correlation of the rank variables:

$$\rho_s = \text{corr}\left(R(x), R(y)\right) \in [-1,1] \tag{3.4}$$

$$S_{\text{spearman}} = |\rho_s| \tag{3.5}$$

Kendall Correlation (Abdi, 2007)

Kendall's (τ) is

$$\tau = \frac{C - D}{\binom{n}{2}} \tag{3.6}$$

where (C) and (D) are the numbers of concordant and discordant pairs among all $(\binom{n}{2})$ index pairs. Similarity:

$$S_{\text{kendall}} = |\tau| \tag{3.7}$$

Kendall is rank-based, often more robust to ties/outliers, but naïvely $(O(n^2))$.

3.6.3. Distance-based Methods

For distance-like quantities $(d \geq 0)$ a decreasing map to$([0,1])$ is defined using a robust cap (d_{max}).g., the (95^{th}) percentile of pairwise distances in a calibration split) and clamping:

$$\Phi(d; d_{max}) = 1 - \min!\left(1, \frac{d}{d_{max}}\right) \tag{3.8}$$

Euclidean similarity (Elmore & Richman, 2001)

$$d_E = \backslash Big\left(\sum_{i=1}^{n} (x_i - y_i)^{2\backslash} Big\right)^{1/2}, \qquad S_E = \Phi(d_E; d_{max}) \tag{3.9}$$

Cosine similarity(Xia et al., 2015):

The raw cosine similarity is

$$\cos(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \in [-1,1]. \tag{3.10}$$

Similarity:

$$S_{\text{cosine}} = |\cos(x,y)| \tag{3.11}$$

Cosine depends only on the angle between vectors (shape), not magnitude.

Manhattan similarity (Kobayashi et al., 2014)

$$d_M = \sum_{i=1}^{n} |x_i - y_i|\,, \qquad S_M = \Phi(d_M; d_{max}) \tag{3.12}$$

3.6.4. Statistical Measures

We adopt robust caps (m_{max}) for MSE and (a_{max}) for MAE (e.g., their (95^{th}) percentiles) and define:

$$\Psi(q; q_{max}) = 1 - \min!\left(1, \frac{q}{q_{max}}\right) \tag{3.13}$$

MSE similarity(Palubinskas, 2014)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2\,, \qquad S_{\text{MSE}} = \Psi(\text{MSE}; m_{max}) \tag{3.14}$$

MAE similarity (Palubinskas, 2014)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|\,, \qquad S_{\text{MAE}} = \Psi(\text{MAE}; a_{max}) \tag{3.15}$$

R^2 similarity (Unnikrishnan & Hebert, 2005)

Treating (y) as a pointwise predictor of (x),

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \in (-\infty, 1] \tag{3.16}$$

and clamp to $[0,1]$:

$$S_{R^2} = \max(0, R^2) \tag{3.17}$$

Kolmogorov--Smirnov similarity (An, 1933)

Let (F_x) and (F_y) be empirical CDFs of (x) and (y). The KS distance is

$$D_{\mathrm{KS}} = \sup_{t \in R} \left| F_x(t) - F_y(t) \right| \in [0,1] \tag{3.18}$$

and the similarity is

$$S_{\mathrm{KS}} = 1 - D_{\mathrm{KS}} \tag{3.19}$$

KS compares distributions and ignores temporal ordering.

3.6.5. Information-Theoretic Measures

Mutual Information (Kraskov et al., 2004)

Given a joint $pmf/pdf\,(p(x,y))$ and marginals $(p(x), p(y))$, mutual information is

$$\mathrm{MI}(X;Y) = \sum_{x}\sum_{y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \;\geq 0 \tag{3.20}$$

To obtain a symmetric, bounded similarity the normalized MI (NMI) is used:

$$S_{\mathrm{MI}} = \frac{\mathrm{MI}(X;Y)}{\sqrt{H(X)\,H(Y)}}, \qquad H(X) = -\sum_{x} p(x) \log p\,(x) \tag{3.21}$$

with the convention $(S_{\mathrm{MI}} = 0)$ if $(H(X) = 0)$ or $(H(Y) = 0)$. In practice, $(p(\cdot))$ is estimated using either fixed-width binning (e.g., Freedman--Diaconis) or (k)-NN estimators; estimator choice and hyperparameters are reported with results.

Max-Lag Cross-Correlation (Amornbunchornvej et al., 2021)

Define the lag-(l) normalized cross-correlation

$$\rho(\mathrm{l}) = \frac{\sum_i (x_i - \bar{x})(y_{i-\mathrm{l}} - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_{i-\mathrm{l}} - \bar{y})^2}} \tag{3.22}$$

for ($\mathrm{l} \in \{-L, \dots, 0, \dots, L\}$) within a chosen window (L) (e.g., ($L = \lfloor 0.1n \rfloor$)). The similarity is the maximal absolute cross-correlation over the window:

$$S_{\mathrm{xcorr}} = \max_{|\mathrm{l}| \le L} |\rho(\mathrm{l})| \in [0,1] \tag{3.23}$$

This captures phase-shifted similarity.

3.6.6. Hybrid Measures

Hybrid scores combine complementary views via convex combinations. All weights are nonnegative and sum to one; unless otherwise tuned on a validation set, the defaults below is used.

Combined Correlation

$$S_{\mathrm{corr}} = \alpha_1 S_{\mathrm{pearson}} + \alpha_2 S_{\mathrm{spearman}} + \alpha_3 S_{\mathrm{kendall}}, \quad (\alpha_1, \alpha_2, \alpha_3) = (0.4,\ 0.4,\ 0.2) \tag{3.24}$$

Combined Distance

$$S_{\mathrm{dist}} = \beta_1 S_E + \beta_2 S_M + \beta_3 S_{\mathrm{MAE}} + \beta_4 S_{\mathrm{MSE}}, \quad (\beta_1, \beta_2, \beta_3, \beta_4) = (0.25,\ 0.25,\ 0.25,\ 0.25\ ) \tag{3.25}$$

$$S_{\mathrm{hybrid}} = \gamma_1 S_{\mathrm{corr}} + \gamma_2 S_{\mathrm{dist}} + \gamma_3 S_{\mathrm{cosine}} + \gamma_4 S_{R^2} + \gamma_5 S_{\mathrm{MI}} + \gamma_6 S_{\mathrm{xcorr}} + \gamma_7 S_{\mathrm{KS}} \tag{3.26}$$

with $(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7) = (0.15,\ 0.15,\ 0.10,\ 0.10,\ 0.20,\ 0.20,\ 0.10)$.

This unifies shape (cosine), linear and rank associations, amplitude error, distributional similarity, and lag alignment.

3.6.7. Matching Protocol

Each method is evaluated by its ability to correctly match each of 23 query signals against a full candidate set of 23 signals.

For a fixed similarity method, compute all pairwise scores

$$S(i,j), \qquad i,j \in \{1,\dots,23\} \tag{3.24}$$

where $(S(i,j))$ is the normalized similarity between query signal $(x^{(i)})$ and candidate signal $(y^{(j)})$.

The predicted match for query (i) is the argmax

$$\hat{j}(i) = \arg\max_{j} S(i,j) \tag{3.25}$$

Let $(j^*(i))$ be the ground-truth counterpart of$(x^{(i)})$. The top-1 accuracy is

$$\text{Acc1} = \frac{1}{23}\sum_{i=1}^{23} 1\{\hat{j}(i) = j^*(i)\} \tag{3.26}$$

If needed, top-(k) accuracy is report, counting success if $(j^*(i))$ lies among the (k) largest entries of $S(i,\cdot)$

3.7. Stage 5, Protocol-Valid Trace Generation and Attack Injection

This stage is intended for offline trace generation. The model trained in Stage 3 is utilized to synthesizes protocol-valid SAE~J1939 traces from anchor-conditioned trajectories. Window-level reconstructions are expanded to sample-level series at \samprate{ } using a piecewise constant mean with intra-window modulation as follows

$$x_t = \widehat{\mu_{w(t)}} + m_t \tag{3.26}$$

where $(\widehat{\mu_{w(t)}})$ is the per-window mean from Stage 3 and (m_t) is class-conditioned modulation drawn from temporal statistics of the same semantic class.

Frames are re-encoded using DBC-like ground truth configurations for $(\langle \text{PGN,SPN} \rangle)$, byte order, bit offset, and length, with priorities and Transport Protocol (TP) behavior per J1939-21. The framework is optimized for 100 milliseconds (10 Hz) to align with Stages 1 to 3.

Validation of the synthesized traces involves several steps. Protocol compliance is confirmed by checking the J1939 Transport Protocol (TP) session structure (RTS/CTS, BAM, TP.DT), verifying identifier and priority masks, and analyzing period and jitter statistics. Compatibility is

further ensured through independent third-party decoding (cantools contributors, 2025). Statistical realism is then assessed using both Kolmogorov–Smirnov and Wasserstein tests on the marginal distributions, with multiple-comparison control via the Holm–Bonferroni method. A correlation-realism rate is also computed by comparing pairwise Pearson correlations against reference logs. Finally, these components are aggregated into a single fidelity score to provide a holistic measure of trace quality.

The attack module generates six protocol-valid families: (1) value spoofing injects frames with legitimate identifiers but fabricated data values to report a false state (e.g., incorrect speed); (2) replay records a sequence of valid frames and re-injects them later to create an inconsistent vehicle state; (3) protocol-valid flooding increases bus load by injecting a high volume of legitimate-looking frames, potentially delaying critical messages; (4) payload fuzzing systematically alters bytes within valid payloads to probe for vulnerabilities; (5) address-claim interference abuses the J1939 address claiming process (PGN~60928) to cause conflicts or impersonate ECUs; and (6) transport-layer manipulations disrupt multi-frame messages by sending malformed or out-of-order Transport Protocol (TP) packets.

4. EXPERIMENTAL DESIGN AND SETUP

4.1. Datasets

A large set is constructed using of SAE~J1939 traces obtained from various public sources. The following subsections explain where the data come from, how each frame is decoded, how the signals are cleaned, and how the final quality checks are carried out.

These four datasets were chosen because they cover many heavy-duty vehicles, long driving hours, and supply helpful decoding material. Table 4.1 lists their main features.

Table 4.1 SAE J1939 datasets used in this study.

Resource Ref.	**Data size**	**Vehicles & scenarios**
(Dönmez, 2021)	≈ 3 GB	Renault Euro VI T520; urban & rural routes
(Zago et al., 2020)	≈ 38 M frames	Multiple vehicles (cars & trucks)
(Daily, n.d.)	2 -38 M frames / capture	Multiple heavy-duty trucks
(CSSElectronics, n.d.)	≈ 400 MB	Over 20 heavy vehicles

4.2. Data Preparation Pipeline

To characterize the data, a set of descriptive statistics for each signal is computed. This analysis provides insight into signal behavior and helps evaluate its potential as a suitable anchor. The examination focuses on several key properties. Also, data set's central tendency and variability are measured to understand its typical values and how much they fluctuated. To capture patterns over time, the relationship between consecutive data points is investigated. These statistical results are organized by system group into separate files. A comprehensive list of all computed statistics is provided in the Appendix. The exploratory analysis reveals two consistent patterns that directly inform model design. First, highly stable signals such as engine speed exhibit clear, repeatable cycles over time, making them reliable candidates for anchors. Second, several emissions-related and pressure/temperature signals deviate from bell-shaped distributions and display event-driven excursions, which motivates robust features and models capable of handling non-Gaussian behavior and transient dynamics. Complementing these observations, pairwise correlation and mutual-information views expose dense blocks of redundancy across subsystems, suggesting principled opportunities for signal reduction and anchor construction. These findings motivate the anchor sets and modeling strategies evaluated in the subsequent chapters.

In addition to these numbers, the signals are visually checked. Graphs of the data over time showed clear patterns. For example, the engine speed signals showed a repeating cycle, which is

normal for stable measurements. Also, charts like histograms are used to check the shape of the data's distribution and to spot any strange values. These charts show that many signals related to emissions do not follow a typical bell-curve shape. This finding has motivated the use of more robust and flexible methods in later modeling stages. This careful look at both the numbers and the visuals are essential for choosing the most important signals for the presented work.

A deeper statistical analysis is performed to find any repetitive information and to understand the underlying structure of the dataset. The correlation between every pair of signals are calculated to see how strongly they were related. The complete results of this can be found in a table in the Appendix. Then a clustering technique is used to group signals that behaved similarly. Within these groups, another method called Principal Component Analysis is used to see if the information can be represented with fewer signals. Heatmap charts helps to see these relationships clearly. A large heatmap gives us a complete overview, while smaller ones let us look at specific systems in more detail. See Figure 4.1 for the full pairwise Pearson correlation matrix; blocks of high correlation indicate domains with redundant signals. The correlation matrix highlights blocks of high linear dependence, while the mutual information view expresses total shared information in bits; together, they reveal domains with redundant signals and quantify the strength of their relationships.

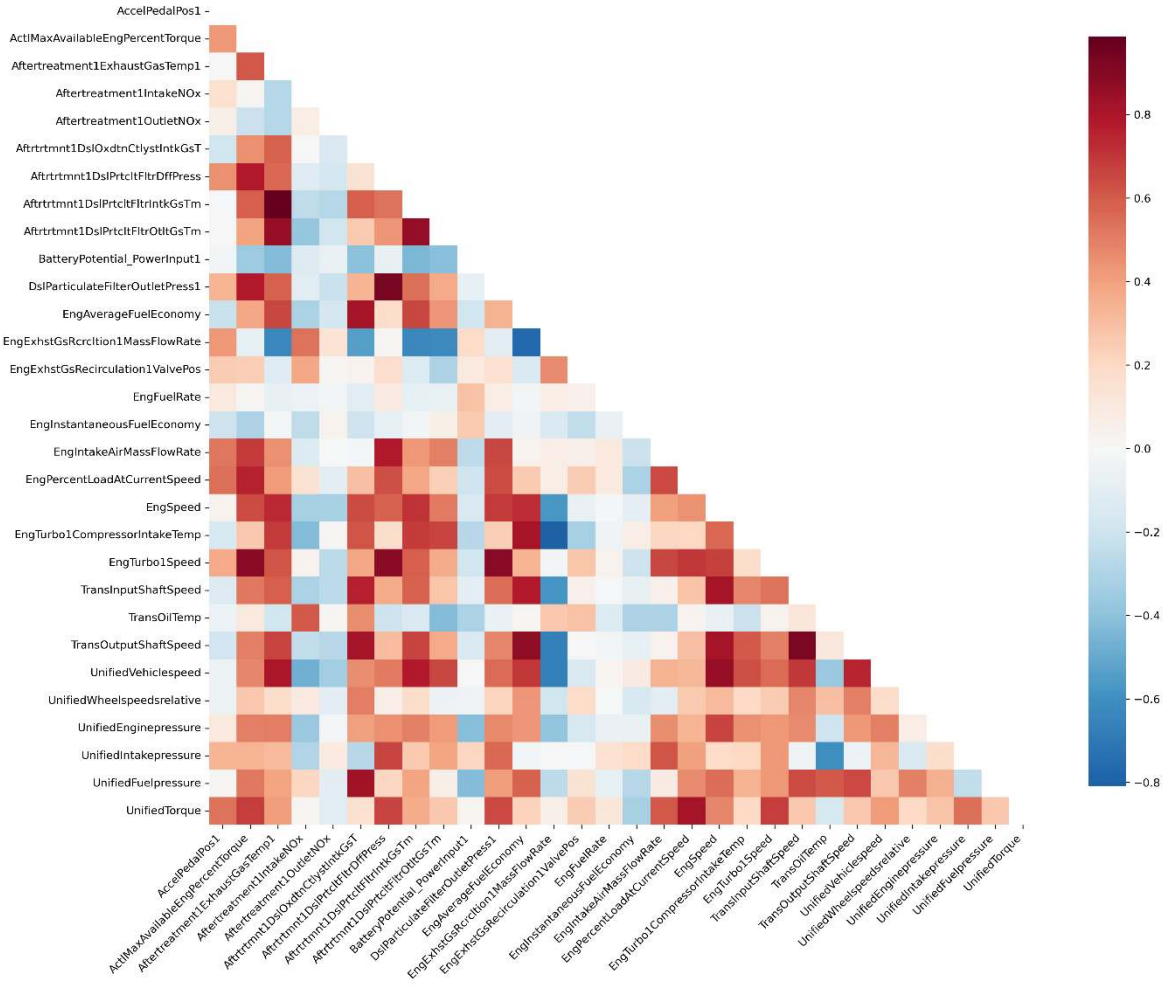

Figure 4.1 Global correlation heatmap across signals.

This analysis helped to create five different sets of key signals, which are called anchor sets. These sets allow to test the models with different amounts and types of information. The first set is the Minimal set, which includes three basic signals like engine and vehicle speed. These are very stable and provide a strong baseline. The Standard set adds signals that give more context about what the driver is doing and how hard the engine is working. The Comprehensive set is the most complete, including signals like turbocharger speed and fuel pressure to give a very rich picture. Also, two focused sets are prepared. The Engine-focused set includes only signals about the engine, while the Dynamics-focused set concentrates on signals related to the vehicle's movement. These well-defined sets provide a systematic way to see how the amount of information affects the model's performance. Figure 4.2 illustrate typical anchor-signal behavior.

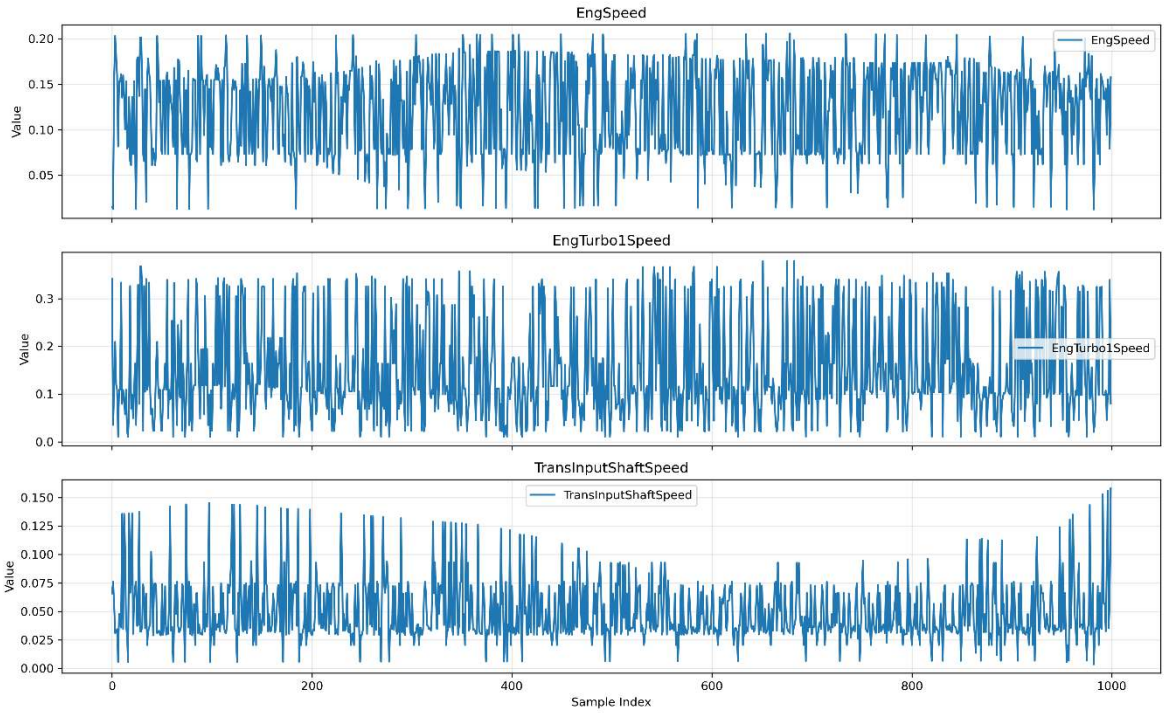

Figure 4.2 Vehicle dynamics time-series (first 1000 rows).

The analysis gives several important insights that guide the modeling work, but it also points out some limitations. The most important finding is that a small number of very stable signals, like engine and vehicle speed, provides a solid foundation for building predictive models. This confirms that the proposed Minimal anchor set is a well-justified starting point for the experiments.

On the other hand, many sensors for things like temperature and pressure act differently. Their information seems to be most valuable during specific events, such as when the engine is starting from cold or is under a heavy load. This suggests that just treating them as continuous signals is not the best approach. A future improvement would be to develop ways to specifically detect these important events. See Figure 4.3 for examples of event-driven excursions that motivate extracting specialized event-detection features.

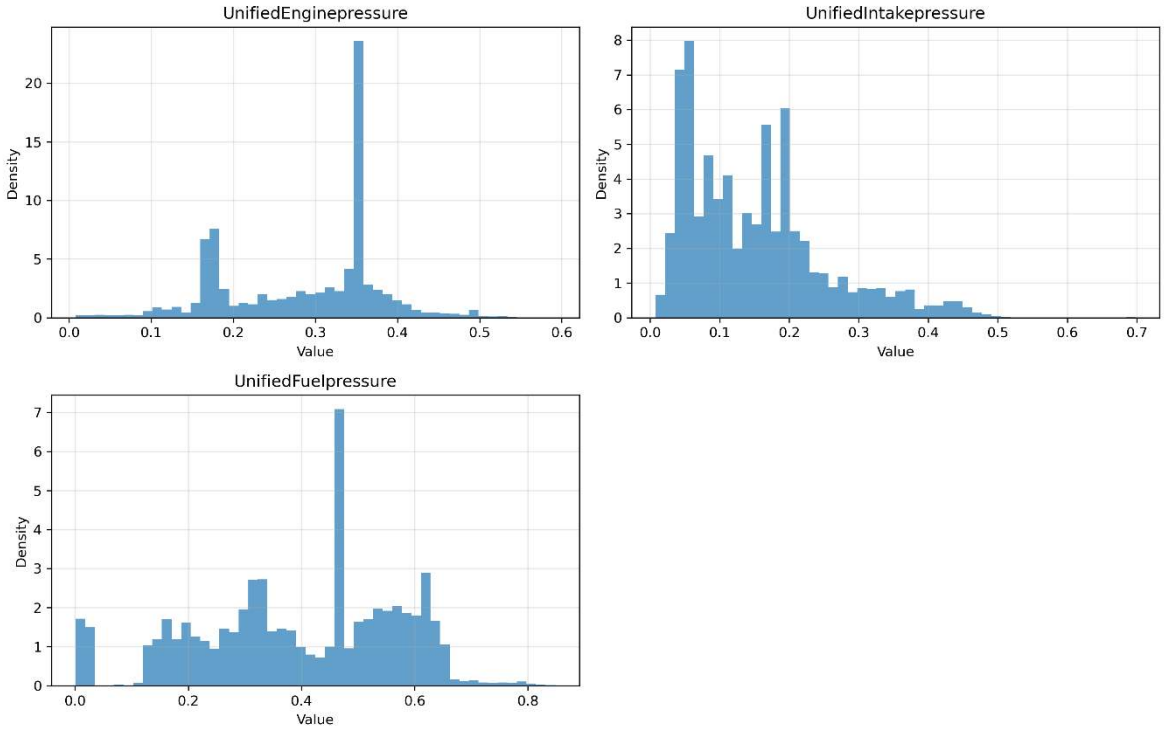

Figure 4.3 Pressure value distrebution.

The conducted search for redundant information was very productive. The correlation analysis shows that many signals within the same vehicle system are closely related. This allowed to simplify data by reducing the number of signals, in some cases by around 40 percent, without losing predictive power. This is key to creating the Standard and Comprehensive anchor sets, which balance having enough information with keeping the models from becoming too complex.

These points lead to clear next steps. Anchor sets are refined using the analysis and expert knowledge to make sure the most meaningful signals are kept. It is also critical to develop features that can capture the event-driven nature of some sensors. Exploring other methods for finding non-linear relationships will also be an important step to improve the proposed models further. It worth to note that the signals that have the same meaning in unified categories. e.g. wheels speed are combined.

4.3. Anchor Signal Configurations

The anchor signals with complementary diagnostics that quantify redundancy, information concentration, and cluster structure across sensors are selected. Pairwise correlation heatmaps identify dense blocks of strong linear dependence within each system domain. A mutual-information proxy computed from the correlation matrix expresses the same relationships on an information scale. Principal component analysis shows whether a few linear combinations capture most variance within a domain. Hierarchical clustering groups sensors that move together under typical operating conditions. Taken together, these diagnostics suggest clear rules for choosing anchors. Stable, high-

coverage signals that are always informative are included. Contextual signals that explain driver demand and load when needed are included. Internal-state sensors only when they contribute independent variance beyond the proxies are added.

The Minimal configuration serves as a baseline. The diagnostics identify a small set of continuously measured high-coverage signals that dominate variance in the engine and dynamics block. Engine speed and vehicle speed appear as central nodes in the cluster structure. These signals explain a large fraction of the first principal components for the combined engine and dynamics domain. Transmission input shaft speed adds a different mechanical view. Together these three signals form a compact and robust representation. This representation suits a low-dimensional baseline for comparison.

The Standard configuration expands the Minimal set by adding accelerator pedal position and intake air mass flow. The diagnostics place these signals near the speed signals but in distinct positions on the cluster map. They contribute variance that the speed signals do not capture. They encode driver demand and instantaneous airflow that modifies engine load and efficiency. Including them increases the model's ability to distinguish operating states. This increase does not add much redundancy. The anchor sets follow a data-driven logic: stable, high-signal-to-noise channels (e.g., engine and vehicle speed) form the backbone of the Minimal set, while broader configurations progressively add context (driver intent, engine load) and subsystem-specific sensors (e.g., turbo speed, fuel pressure). Correlation and mutual-information structure informed which signals were retained or pruned, enabling meaningful reduction without sacrificing predictive power. As later results show, the Standard set achieves performance close to Comprehensive, indicating diminishing returns from secondary sensors and reinforcing the cost–performance rationale behind the design.

4.4. Sliding Window configuration

A preliminary study was conducted to identify the optimal parameters for the sliding window method. This analysis aims to find a configuration that improves the model's ability to generalize while effectively processing temporal data. The investigation systematically evaluates how window size, overlap percentage, and feature complexity influence performance. This section details the experimental setup, the methodology used to detect overfitting, and the findings that informed the selection of the optimal parameters.

4.4.1. Sliding Window Configuration Space

The study explored several dimensions of the sliding window configuration to understand their effects on model performance. Three distinct window sizes were tested to analyze the trade-off between temporal context and model complexity: 32 samples (small), 64 samples (standard), and 128 samples (large). The analysis also investigated five levels of overlap between consecutive windows: 0%, 25%, 50%, 75%, and 87.5%. This range allowed for an assessment of how data

redundancy affects generalization. Finally, two levels of feature complexity were compared. The simple feature set included six basic statistical features per signal, while the complex set contained twelve extended temporal and interaction features.

Model performance was primarily assessed by measuring the gap between the training and validation R^2 scores. A smaller gap indicates better generalization because it shows the model performs similarly on both seen and unseen data. This metric directly quantifies the degree of overfitting and is calculated as

$$Gap = R^2_{train} - R^2_{validation} \tag{4.1}$$

. The analysis employed Ridge regression with adaptive regularization, using α values of 1.0, 10.0, and 100.0, to help control model complexity and prevent overfitting.

Table 4.2 Sliding window configuration analysis results.

Configuration	**Win. Size**	**Overlap (%)**	**Features**	**Train R^2**	**Val R^2**	**Gap**	**Status**
No overlap	64	0.0	Simple	0.094	-0.069	0.163	Optimal
Low overlap	64	25	Simple	0.1	-0.038	0.138	Good
Medium overlap	64	50	Simple	0.097	-0.181	0.278	Poor
High overlap	64	75	Simple	0.098	-0.088	0.186	Poor
Extreme overlap	64	87.5	Simple	0.094	-0.043	0.137	Fair
Small window	32	0.0	Simple	0.113	-0.105	0.219	Poor
Large window	128	0.0	Simple	0.169	-1.587	1.756	Critical
Complex	64	0.0	Complex	0.113	-0.143	0.256	Poor
Complex High	64	87.5	Complex	0.102	-0.047	0.150	Fair

The results from the nine distinct configurations, summarized in Table 4.2, show that the percentage of overlap is a critical factor. The configuration with no overlap (0%) demonstrated the best generalization performance. In contrast, moderate overlap levels, such as 50%, led to a significant increase in the train-validation gap, indicating poor generalization. This outcome occurs because high overlap can cause temporal data leakage, where similar information appears in both training and validation sets, which artificially inflates training performance. Window size also had a notable impact on the model. A small window of 32 samples was insufficient to capture the necessary temporal context, leading to poor performance. Conversely, a large window of 128 samples caused catastrophic overfitting. A window sized 64 sample provides an optimal balance, capturing adequate information without introducing excessive complexity. Furthermore, the complexity of the features affects generalization. Using a simpler set of six features per signal resulted in a substantially smaller

train-validation gap than the more complex set of twelve features. However, to reach a high quality, reach set of features might be needed.

4.4.2. Feature Engineering for Machine Learning Models

To prepare the raw time-series data for the machine learning pipelines, it is transformed into a comprehensive feature set. This process generates features that capture temporal dynamics, trends, and interactions within the data. For reproducibility, the resulting feature columns are deterministically sorted by name.

To capture local trends and variability, statistical aggregations are computed over a short temporal window of five timesteps. The rolling mean is used to smooth out noise in the signal. The rolling standard deviation helps capture volatility. Additionally, the rolling minimum and maximum values identify local extrema, which can indicate transient events. To provide the model with immediate historical context, lag features are created from the signal values of the previous one and two timesteps. These features are essential for modeling simple time-lagged dependencies and autoregressive dynamics. First and second-order difference features are also computed to represent the signal's rate of change. These features approximate the first and second derivatives, corresponding to velocity and acceleration. They allow the model to detect trends and curvature, which are often more informative for identifying dynamic states than absolute signal values. Finally, interaction features are generated to model nonlinear, joint effects between signals. These features are created from the pairwise products of domain-relevant signals, such as engine speed and throttle position. They are generated only when both signals in a pair are present in the input data to avoid creating spurious information.

4.5. Machine Learning Model Configurations

This study employs a multi-stage methodology to analyze automotive signal data, beginning with feature engineering and followed by predictive modeling. The initial stage transforms raw signal segments into a structured, machine-readable format by computing a comprehensive set of features for each signal. The second stage uses this feature matrix for model training and prediction in both classification and reconstruction tasks. The feature engineering framework includes rolling statistics (mean, standard deviation, minimum, maximum) over fixed windows to capture local distributional properties, as well as temporal features such as lagged values and first- and second-order differences to represent signal dynamics. Domain-specific interaction features are constructed by combining key automotive signals—such as engine speed, vehicle speed, and accelerator pedal position—using multiplication and ratio operations. All features are generated in a deterministic, sorted order to ensure reproducibility. This approach provides a consistent and comprehensive representation of signal characteristics, supporting robust anchor signal identification and accurate multi-output regression for signal reconstruction.

For the predictive modeling stage, the study utilizes a selection of robust, tree-based ensemble models. These models are chosen for their strong performance on tabular data and their resilience to variations in feature scaling. The analysis incorporates RandomForest and ExtraTrees, which improve generalization by averaging predictions from an ensemble of 100 decision trees. For enhanced predictive accuracy, the methodology also includes advanced Gradient Boosted Trees, specifically XGBoost and LightGBM. These techniques build trees sequentially, with each new tree trained to correct the residual errors of the preceding ones. When addressing tasks that require multi-target reconstruction, a Multi-Output Strategy is applied. This approach adapts single-output regressors for multiple dependent variables by fitting an independent estimator for each target. This strategy is effective and suitable for parallel processing, making it particularly useful for problems where the output variables are not strongly correlated.

4.6. Deep Learning Settings

The deep learning pipeline begins with a structured data preparation process. This stage transforms raw time-series signals into fixed-size windows suitable for model input. Each window has a length (L) of 512 consecutive timesteps. This size is selected to capture sufficient temporal information while remaining mindful of computational and memory limitations. The windows are extracted with a stride of 256 timesteps, creating a 50% overlap between successive windows. This overlap serves to augment the training dataset and maintain continuity between samples. After windowing, the data is formatted into a 3D tensor with dimensions (N,1,L), where N denotes the batch size. Finally, Z-score normalization is applied as a standard procedure. This step standardizes the scale of the input signals, which helps achieve faster and more stable model convergence during training.

The training process for deep learning models is controlled by a set of hyperparameters that influence convergence and final performance. Although some parameters differ slightly between classification and reconstruction tasks, the core configuration is consistent. A summary is provided in Table 4.3.

Table 4.3 Key deep learning training hyperparameters.

Hyperparameter	**Classification Value**	**Reconstruction Value**
Optimizer	Adam	Adam
Learning Rate	1×10^{-3}	1×10^{-3}
Loss Function	CrossEntropyLoss	MSELoss
Batch Size	256	512 (Train), 1024 (Val/Test)
Default Epochs	15	150-200
Early Stopping Patience	5 epochs (on val_loss)	20 epochs (on val_loss)
Learning Rate Scheduler	None	ReduceLROnPlateau(patience=10, factor=0.5)
Weight Decay	None	1×10^{-5}
Gradient Clipping	None	Max norm = 1.0

This thesis implements a family of deep learning architectures for both classification and reconstruction tasks. Each architecture is available in three distinct configurations to manage the trade-off between computational cost and model capacity. The small configuration includes models with tens of thousands of parameters. These models are designed for rapid prototyping, hyperparameter searches, and execution on CPUs. The medium configuration consists of models with hundreds of thousands of parameters. This size represents a balance between performance and computational demand and serves as the standard for most reported experiments. Finally, the large configuration features models with millions of parameters, offering the highest representational capacity. These larger models are reserved for final experiments where maximizing performance is the primary goal and substantial computational resources are available.

The primary models for the classification task have specific architectural designs. The first is a one-dimensional Convolutional Neural Network (1D CNN). This network contains three convolutional blocks, and each block is followed by a max pooling layer. A final classifier head with dropout is included for regularization. The second model is a two-layer bidirectional Long Short-Term Memory (LSTM) network. This architecture is specifically designed to capture sequential dependencies within the data. The third model is a Transformer Encoder adapted for 1D sequence classification. It is configured with a model dimension (d_{model}) of 64, four attention heads (n_{head}), and two encoder layers (num_{layers}).

For the reconstruction task, a broader set of architectures was explored. The exact parameterizations for these models are detailed in the tables below, mapping the small, medium, and large configurations to their corresponding hyperparameters.

The Wide & Deep MLP architecture is implemented in three distinct sizes to evaluate scalability. The small configuration provides a compact model, featuring a wide component dimension of 256 and a two-layer deep path with dimensions of [256, 128]. A medium-sized version

offers a balanced design with increased capacity, utilizing a wide dimension of 512 and a three-layer deep path of [512, 256, 128]. For capturing highly complex feature interactions, a large, high-capacity model is used, which has a wide dimension of 1024 and a four-layer deep component with dimensions of [1024, 512, 256, 128]. A consistent dropout rate of 0.3 is applied for regularization across all three configurations.

The Attention-based MLP architecture is also examined in small, medium, and large variations. The small model is configured with a hidden size of 256 and four attention heads to learn feature interdependencies. To enhance representational power, the medium version increases the hidden size to 512 and the number of attention heads to eight. The large model is designed for maximum capacity, employing a hidden size of 1024 while also using eight attention heads, which introduces significant computational requirements. Each version of this architecture incorporates a dropout rate of 0.3.

A ResNet-style MLP architecture is evaluated to assess the benefits of residual connections. The small model is a residual network with three blocks, defined by layer dimensions of [256, 256, 128], which promotes stable training. The medium configuration features a deeper and wider network structure with four layers of [512, 512, 512, 256] to improve its ability to model complex mappings. For final, high-performance runs, a large and very deep residual MLP is employed, which contains five layers with dimensions of [1024, 1024, 768, 512, 256] and a high parameter count. A dropout rate of 0.3 is consistently used in all ResNet-style models.

Finally, an LSTM-based MLP architecture is analyzed for its ability to process sequential data. The small configuration is a sequential model that uses a two-layer LSTM with a hidden state size of 128 for temporal feature extraction, followed by a two-layer MLP head with hidden dimensions of [256, 128]. The medium version enhances temporal modeling capacity by increasing the LSTM hidden state size to 256 and the MLP head dimensions to [512, 256]. The large, high-capacity model is built for complex sequences and features a deeper three-layer LSTM with a hidden state size of 512, complemented by a larger three-layer MLP head with dimensions of [1024, 512, 256]. A dropout rate of 0.3 is applied to all configurations of this architecture.

4.7. Simulation and Injection Attack Setups

The experimental validation for Stage 5 employs a thorough strategy to demonstrate the practical utility of the entire research pipeline. This process verifies that Stage 5 correctly integrates the models and insights developed in all preceding stages. The evaluation assesses the quality of the generated data traces by examining their realism and accuracy. Furthermore, it confirms that the output comprehensively adheres to the J1939 standard. The validation also evaluates the diversity of operational scenarios covered and benchmarks the computational efficiency and scalability of the system.

A variety of experimental scenarios are used to test the comprehensive capabilities of Stage 5. Basic reconstruction scenarios focus on signal quality and model performance, primarily integrating work from Stages 2 and 3. More complex scenarios are designed to evaluate the diversity and realism of the output, incorporating elements from Stages 1 through 4. Specific tests are conducted to ensure strict protocol compliance with the J1939 standard, utilizing all stages of the pipeline. Additional scenarios assess performance to measure computational efficiency, while optional security-focused experiments test the system's capacity for modeling attacks.

The evaluation relies on specific metrics to validate both individual components and the overall integration of the pipeline. To measure successful integration, the process tracks the rate at which models from previous stages are utilized. It also verifies that the optimal configurations from Stage 2 are consistently implemented. A key metric confirms that the predictive performance, measured by R^2 scores in Stage 3, is maintained during the generation process in Stage 5.

Metrics for trace quality focus on the fidelity and plausibility of the generated data. The statistical similarity of the generated traces to reference datasets from Stage 1 is evaluated using correlation analysis and Kolmogorov-Smirnov tests. The accuracy of signal reconstruction is measured by comparing individual signal R^2 scores to the benchmarks established in Stage 3. The evaluation also validates temporal consistency in time-series data and assesses the physical plausibility of the output against known automotive engineering principles.

Finally, a set of metrics is used to ensure protocol compliance. The primary metric is the J1939 conformance rate, which measures the percentage of messages that adhere to the standard. The evaluation also examines timing accuracy by validating broadcast intervals and message priorities. Additionally, it verifies that the structure and data encoding of all messages comply with the standard's formatting requirements.

5. RESULTS AND DISCUSSION

This chapter presents the results of the experimental evaluation for the framework stages. The section also integrates comparative analyses of machine learning (ML) and deep learning (DL) models for stages two and three, explores the impact of anchor signal configurations, and provides a critical discussion of trade-offs between accuracy, efficiency, and interpretability.

5.1. Signal Classification Performance

The first set of experiments focused on signal classification, where the goal was to identify active signals from raw data windows. Models from both ML and DL families were evaluated across multiple anchor configurations.

5.1.1. ML vs. DL Performance

Figure 5.1 displays the average comparative F1 scores for the various models under investigation. The results indicate that the classification task is relatively straightforward, with almost all models attaining an F1 score greater than 0.98. Tree-based machine learning models, particularly the Random Forest Classifier and XGBoost Classifier, exhibit the highest performance, consistently achieving F1 scores above 0.99 and surpassing the deep learning architectures. This superior performance highlights the efficacy of feature engineering and demonstrates the suitability of tree ensembles for structured data. While deep learning architectures like the 1D-CNN, LSTM, and Transformer models perform competently, their scores are slightly lower. The small gap in performance suggests that the additional complexity of deep learning provides little benefit for this classification task, especially given the availability of well-engineered features. Moreover, machine learning models prove to be more efficient. They not only yield higher accuracy but also demand significantly less training time and fewer computational resources than the deep learning models, which often require several hours to train for a single experiment. Classification proves to be a structured, high-separation problem: tree-based ensembles (Random Forest, XGBoost, Extra Trees) consistently achieve the top F1 scores with substantially lower training cost than deep networks. This advantage aligns with the EDA and anchor design: anchors selected for stability and distinct statistical signatures are easily separable by axis-aligned partitions, allowing ensembles to capitalize on well-engineered features. Consequently, deep architectures add complexity without proportional gains for this task, while ensembles deliver state-of-the-art accuracy within seconds to minutes of training.

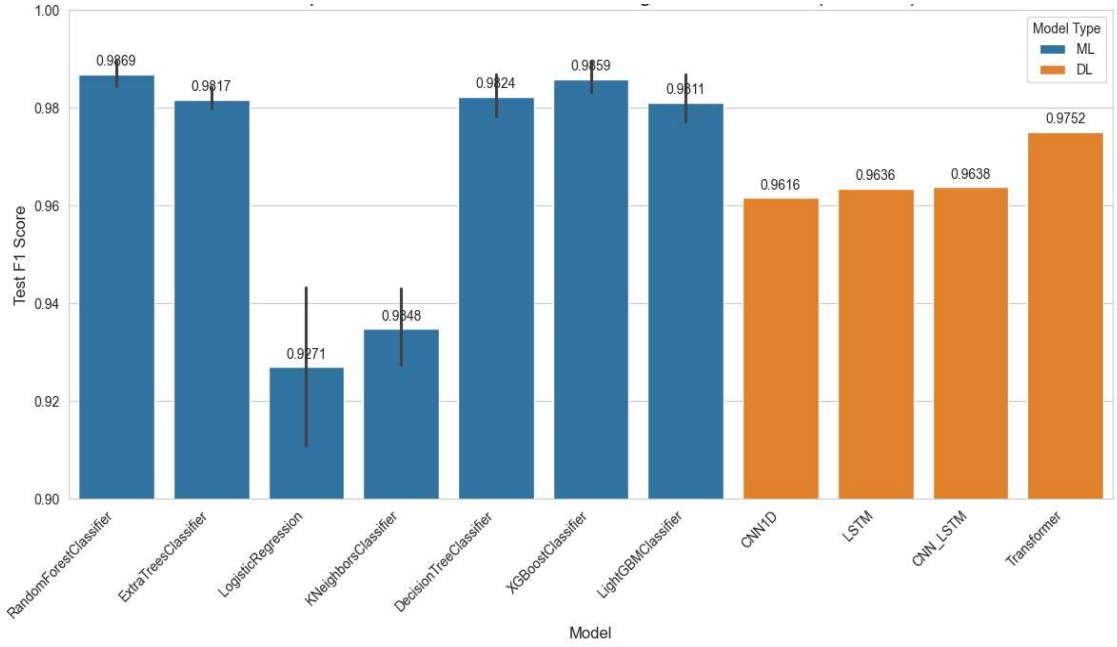

Figure 5.1 Comparison of ML and DL models for signal classification.

Figure 5.2 and Figure 5.3 illustrate the classification performance of ML and DL models across different anchor configurations using heatmaps. The detailed numerical results are reported in Tables 5.1–5.10, which report per anchor configuration performance for the five anchor sets.

The findings emphasize that efficiency and simplicity are central to effective signal classification. Well-tuned ensemble ML models deliver state-of-the-art results while maintaining a much lower computational cost than deep learning networks. The consistently high F1 scores across anchor sets also provide indirect validation of the anchor selection. This indicates that the chosen anchors are both informative and straightforward for models to classify, ensuring that key signal patterns remain distinguishable even under reduced configurations.

This robustness at the classification stage is critical for the broader framework. Since the reconstruction process depends directly on accurate classification, any errors introduced here would propagate and degrade overall performance. The consistently strong results of ensemble ML models therefore confirm their suitability for this task and validate the anchor-selection strategy. The selected anchors are not only robust and well aligned with system dynamics but also provide a reliable foundation for downstream reconstruction and analysis.

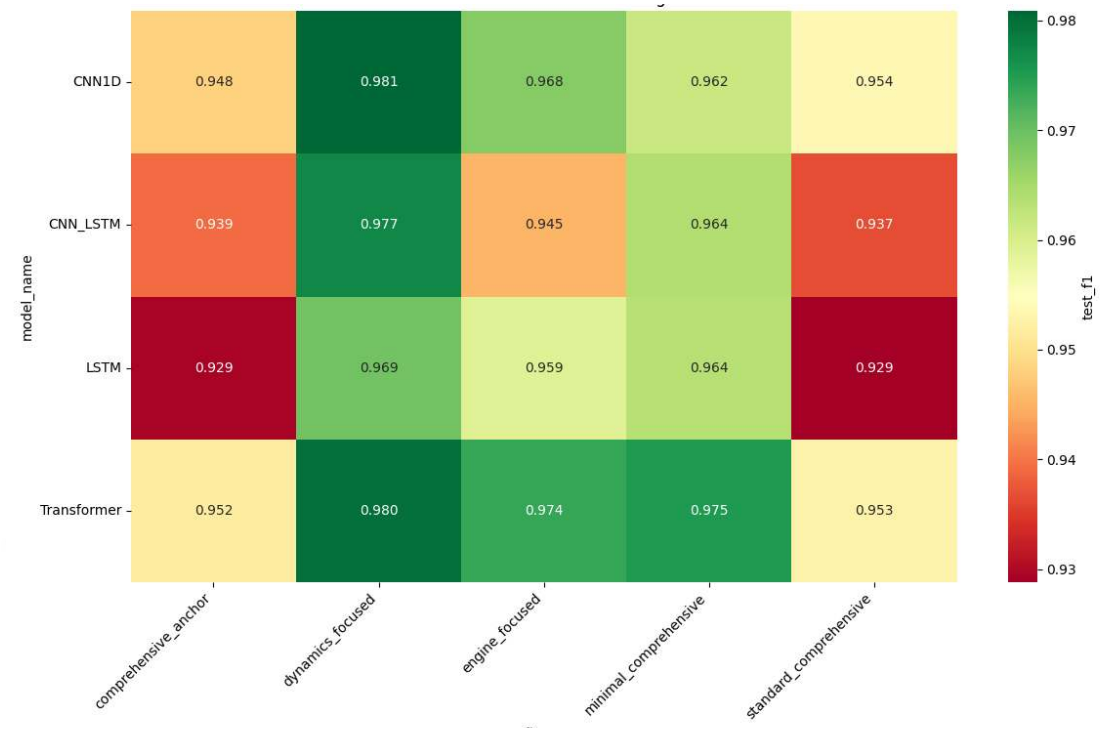

Figure 5.2 DL classification test F1 across configurations.

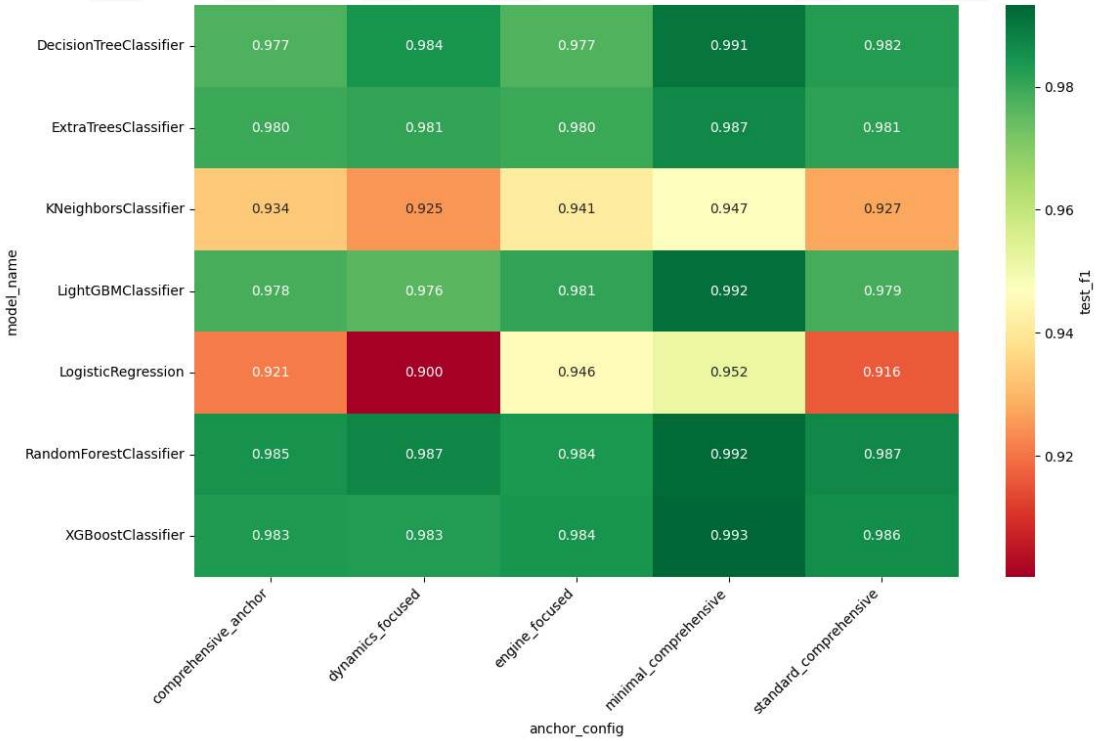

Figure 5.3 ML classification test F1 across anchor configurations.

Table 5.1 ML classification results for the comprehensive anchor configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)
RandomForestClassifier	0.9846	0.9846	0.9846	0.9846	27.56
ExtraTreesClassifier	0.9796	0.9796	0.9796	0.9796	14.55
LogisticRegression	0.9213	0.9213	0.9213	0.9213	46.96
KNeighborsClassifier	0.9336	0.9336	0.9336	0.9336	133.01
DecisionTreeClassifier	0.9774	0.9774	0.9774	0.9774	77.09
XGBoostClassifier	0.9832	0.9832	0.9832	0.9832	18.23
LightGBMClassifier	0.9781	0.9781	0.9781	0.9781	27.70

Table 5.2 DL classification results for the comprehensive anchor configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)	Params
CNN1D	0.9482	0.9486	0.9482	0.9482	1317.15s	52679
LSTM	0.9294	0.9309	0.929	0.929	4062.92s	563207
CNN_LSTM	0.9392	0.9397	0.939	0.939	2221.05s	339591
Transformer	0.9519	0.9524	0.9518	0.9518	3839.58s	67655

Table 5.3 ML classification results for dynamic focused configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)
RandomForestClassifier	0.9872	0.9872	0.9872	0.9872	22.19
ExtraTreesClassifier	0.9809	0.9809	0.9809	0.9809	8.84
LogisticRegression	0.9003	0.9003	0.9003	0.9003	19.02
KNeighborsClassifier	0.9247	0.9247	0.9247	0.9247	133.99
DecisionTreeClassifier	0.9845	0.9845	0.9845	0.9845	88.66
XGBoostClassifier	0.9829	0.9829	0.9829	0.9829	9.36
LightGBMClassifier	0.9761	0.9761	0.9761	0.9761	12.27

Table 5.4 DL classification results for dynamic focused configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)	Params
CNN1D	0.9808	0.9809	0.9808	0.9808	582.86s	52163
LSTM	0.9693	0.9693	0.9693	0.9693	1713.54s	562691
CNN_LSTM	0.9773	0.9776	0.9774	0.9774	1197.09s	339075
Transformer	0.9798	0.9799	0.9798	0.9798	1827.56s	67395

Table 5.5 ML classification results for engine focused configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)
RandomForestClassifier	0.9835	0.9835	0.9835	0.9835	25.43
ExtraTreesClassifier	0.9798	0.9798	0.9798	0.9798	10.20
LogisticRegression	0.9457	0.9457	0.9457	0.9457	23.87
KNeighborsClassifier	0.9414	0.9414	0.9414	0.9414	132.43
DecisionTreeClassifier	0.977	0.977	0.977	0.977	104.00
XGBoostClassifier	0.9844	0.9844	0.9844	0.9844	11.38
LightGBMClassifier	0.981	0.981	0.981	0.981	16.60

Table 5.6 DL classification results for engine focused configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)	Params
CNN1D	0.9677	0.9681	0.9676	0.9676	756.68s	52292
LSTM	0.9591	0.9596	0.9589	0.9589	2294.76s	562820
CNN_LSTM	0.9451	0.9465	0.9447	0.9447	1318.92s	339204
Transformer	0.9737	0.9739	0.9736	0.9736	2388.23s	67460

Table 5.7 ML classification results for minimal configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)
RandomForestClassifier	0.9924	0.9924	0.9924	0.9924	21.08
ExtraTreesClassifier	0.987	0.987	0.987	0.987	8.19
LogisticRegression	0.9518	0.9518	0.9518	0.9518	17.47
KNeighborsClassifier	0.947	0.947	0.947	0.947	134.05
DecisionTreeClassifier	0.9907	0.9907	0.9907	0.9907	84.53
XGBoostClassifier	0.9932	0.9932	0.9932	0.9932	9.95
LightGBMClassifier	0.9916	0.9916	0.9916	0.9916	11.62

Table 5.8 DL classification results for minimal configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)	Params
CNN1D	0.9616	0.9625	0.9617	0.9617	610.76s	52163
LSTM	0.9636	0.9644	0.9637	0.9637	1733.81s	562691
CNN_LSTM	0.9638	0.9645	0.9639	0.9639	1048.34s	339075
Transformer	0.9752	0.9753	0.9751	0.9751	1723.88s	67395

Table 5.9 ML classification results for standard configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)
RandomForestClassifier	0.987	0.987	0.987	0.987	25.90
ExtraTreesClassifier	0.9814	0.9814	0.9814	0.9814	11.89
LogisticRegression	0.9161	0.9161	0.9161	0.9161	32.86
KNeighborsClassifier	0.9273	0.9273	0.9273	0.9273	131.80
DecisionTreeClassifier	0.9824	0.9824	0.9824	0.9824	80.68
XGBoostClassifier	0.9857	0.9857	0.9857	0.9857	13.43
LightGBMClassifier	0.9786	0.9786	0.9786	0.9786	19.35

Table 5.10 DL classification results for standard configuration.

Model	F1-score	Precision	Recall	Accuracy	Train. Time(s)	Params
CNN1D	0.9537	0.9542	0.9537	0.9537	949.63s	52421
LSTM	0.9288	0.9303	0.9288	0.9288	2885.29s	562949
CNN_LSTM	0.9367	0.9385	0.937	0.937	1508.30s	339333
Transformer	0.9529	0.9538	0.9529	0.9529	2795.06s	67525

Table 5.11 and Figure 5.4 show that classification performance often improves when smaller anchor sets are used. This result is consistent with expectations: reducing the input space eliminates redundancy and enables models to identify signals more easily. Careful anchor reduction therefore not only lowers computational cost but also simplifies the classification task, providing both practical and methodological benefits for the framework.

The analysis compares machine learning and deep learning models at the signal level, focusing particularly on tree-based ensembles and Transformer architectures. Test F1-scores were used as the main evaluation metric, allowing consistent performance patterns to be identified across different signals and configurations.

Both Random Forest and XGBoost, as well as Transformer models, achieved high performance on key signals. For example, EngSpeed, UnifiedVehicleSpeed, and TransInputShaftSpeed consistently reached F1-scores above 0.98 across most configurations, indicating that these signals contain distinctive and easily detectable patterns. However, a notable gap appears in the classification of UnifiedVehicleSpeed: ML models frequently achieve near-perfect scores approaching 0.999, while Transformer models remain slightly lower, between 0.97 and 0.99. This suggests that engineered features in ML approaches capture the deterministic nature of vehicle speed more effectively than end-to-end DL models trained directly on raw data windows.

Some signals, however, show lower and more variable results. AccelPedalPos1 and EngIntakeAirMassFlowRate generally score between 0.90 and 0.98, with performance highly dependent on the model and anchor configuration. This variability may reflect inherent ambiguity in

their short-term patterns or potential issues in labeling. Within DL models, Transformers consistently outperform CNN and LSTM variants, while tree-based ensembles remain the strongest among ML approaches.

Several recommendations follow from these findings. For production-critical signals such as UnifiedVehicleSpeed, ML ensembles are preferable due to their superior accuracy. When an end-to-end strategy is required, Transformer architectures are the most promising DL choice. For inconsistent signals such as AccelPedalPos1, further work is needed—potentially through improved labeling, longer temporal context windows, or multi-task learning objectives. Finally, hybrid ensemble methods that combine ML and DL predictions may offer greater robustness by leveraging the strengths of both approaches.

Table 5.11 Ranked configurations for classification experiments.

Rank	Configuration	DL_acc.	DL_F1	ML_acc	ML_F1	Avg. acc.	Avg. F1
1	minimal	0.9661	0.9660	0.9791	0.9791	0.9726	0.9726
2	Dynamics focused	0.9768	0.9768	0.9624	0.9624	0.9696	0.9696
3	Engine focused	0.9612	0.9614	0.9704	0.9704	0.9658	0.9659
4	standard	0.9431	0.9431	0.9655	0.9655	0.9543	0.9543
5	comprehensive	0.9420	0.9422	0.9654	0.9654	0.9537	0.9538

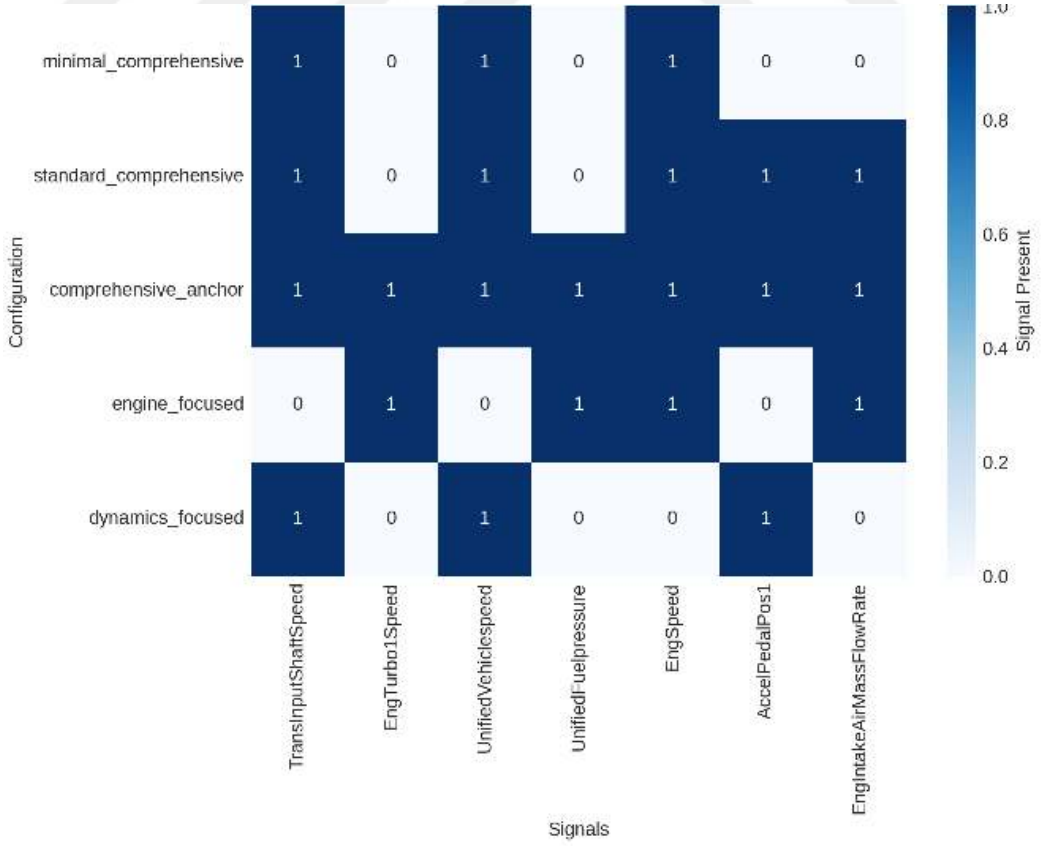

Figure 5.4 Examined configurations.

5.2. Signal Reconstruction Performance

The second set of experiments focused on signal reconstruction, where the goal was to predict target signals from anchor inputs. Unlike classification, this task proved substantially more challenging and exposed clear differences between ML and DL approaches.

Figure 5.5 compares model performance using the R^2 metric. Deep learning models demonstrate a clear advantage, with architectures such as WideDeep and DeepFeatureLearning consistently achieving R^2 values above 0.87. These results exceed those of the best-performing ML regressors. The advantage stems from the ability of DL models to capture complex nonlinear interactions and long-range dependencies within time-series data. By contrast, tree-based ML models such as ExtraTrees, RandomForest, and XGBoost perform respectably, with R^2 values around 0.85, but their reliance on static feature partitions limits their ability to generalize to intricate automotive signal relationships.

In contrast to classification, reconstruction demands modeling nonlinear, long-range temporal dependencies between anchors and targets. Here, deep architectures (Wide-Deep, ResNet-MLP, Attention-MLP) surpass the best ML regressors, with R^2 routinely above 0.97 for comprehensive anchors. These gains, however, come with markedly higher training times, highlighting a practical trade-off: ML regressors remain competitive and efficient, but deep models are essential to uncover hidden dependencies required for the highest-fidelity reconstructions.

Further analysis shows that larger DL architectures deliver incremental gains, suggesting that increased capacity enables more effective modeling of high-dimensional correlations. However, these gains come at the cost of significantly longer training times, underscoring a trade-off between accuracy and computational efficiency.

Overall, the reconstruction task reveals an important contrast with classification. While ML ensembles suffice for straightforward classification, DL models are essential for accurate reconstruction. This distinction highlights the need to align model complexity with task requirements: simple models are adequate when signal boundaries are clear, but more complex architectures are required to uncover hidden dependencies in reconstruction.

5.2.1. Anchor Configuration Effects

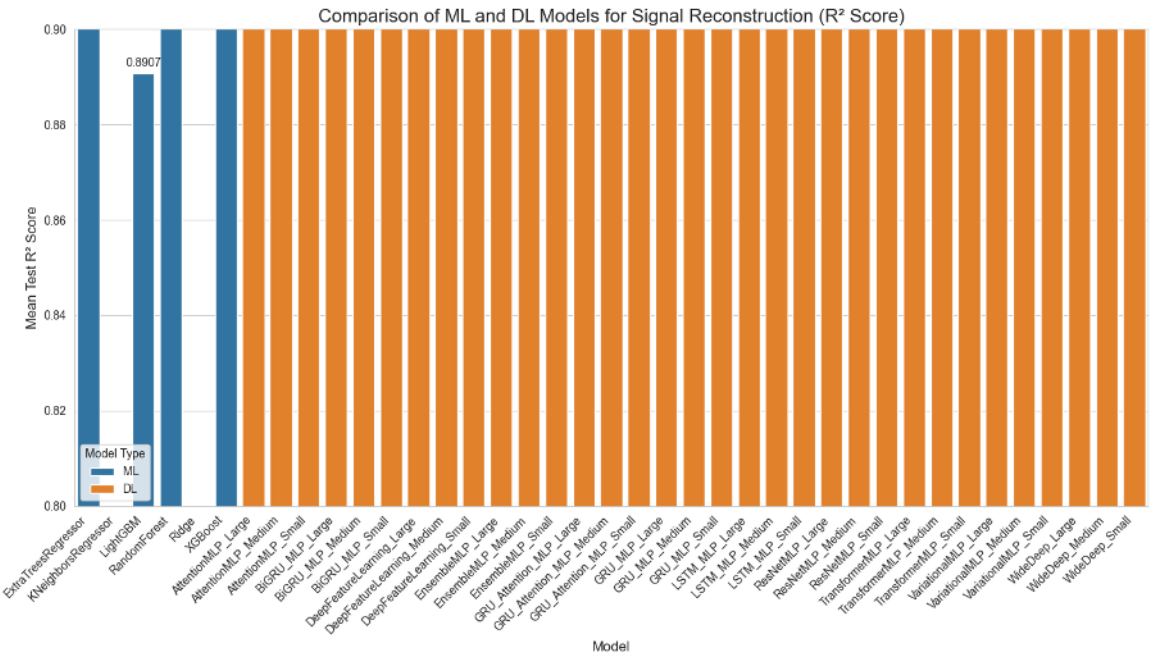

Figure 5.5 Comparison of ML and DL models for signal reconstruction.

Figures 5.6 and 5.7 illustrate reconstruction performance across different anchor configurations, with detailed results reported in Tables 5.12–5.21. The comprehensive anchor sets consistently achieve the highest R^2 values, confirming that a richer collection of signals enhances reconstruction quality. However, the additional gains diminish as more anchors are added, revealing a point of diminishing returns.

Across configurations, comprehensive anchors maximize accuracy but show diminishing returns as additional signals are added. Minimal and standard sets remain attractive in practice: despite slightly lower R^2, they substantially reduce sensing and compute requirements and still exceed 0.85 R^2 on average. This pattern reinforces the design principle established in Chapter 4: retain cores that capture system state and add specialty channels only when their marginal value justifies the cost.

Minimal anchor sets, by contrast, yield lower accuracy but still achieve R^2 scores above 0.85. Their main advantage is reduced hardware requirements, which provides a practical trade-off between cost and performance. Engine-focused anchors also perform competitively, showing that a domain-specific subset of signals can support accurate reconstructions when the analysis targets a particular vehicle subsystem.

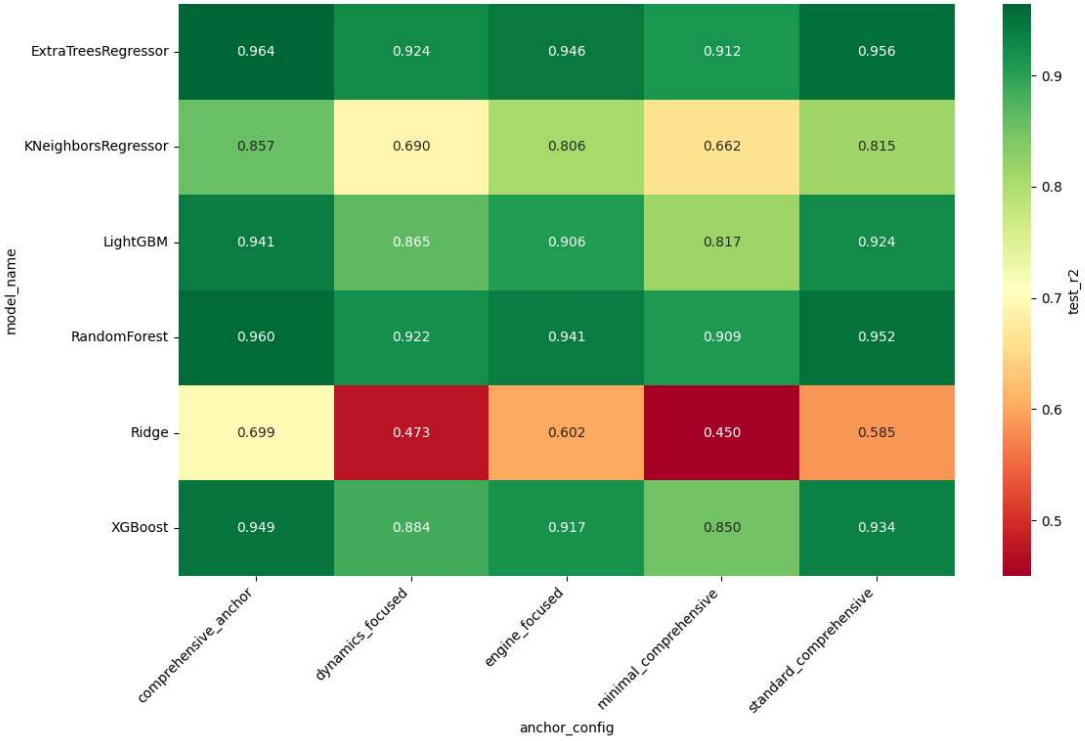

Figure 5.6 ML models R2 score across anchor configuration.

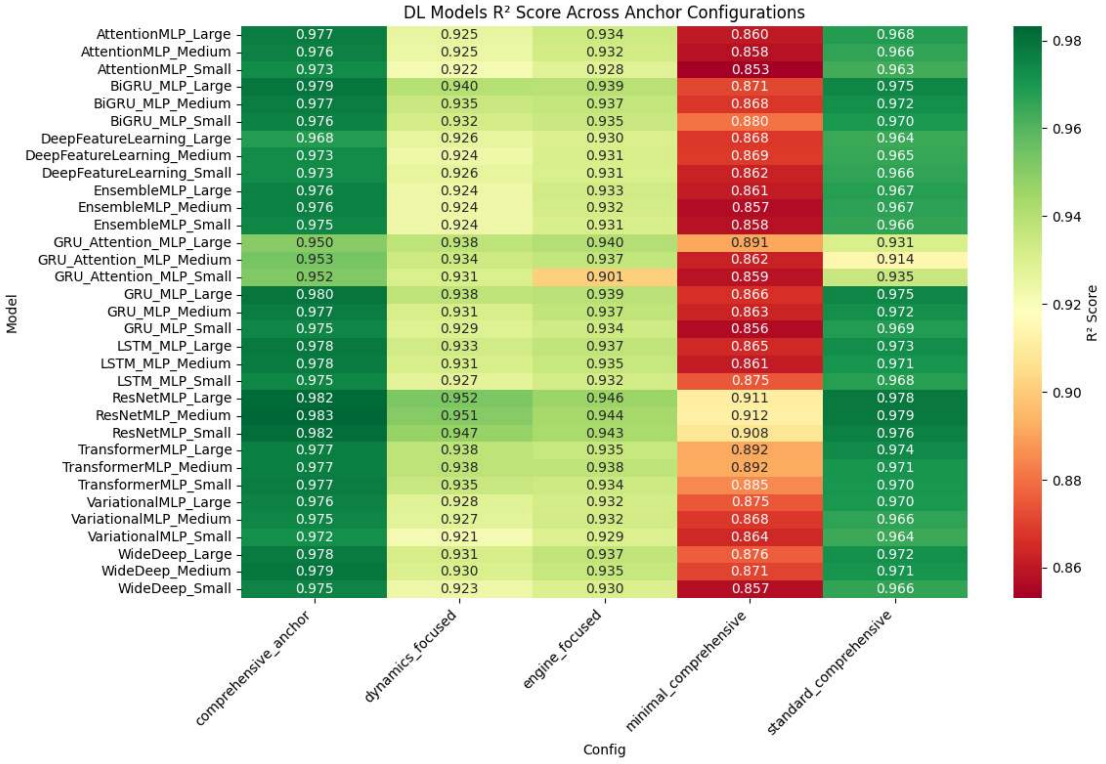

Figure 5.7 DL models R2 score across anchor configuration.

Table 5.12 ML reconstruction results for the comprehensive anchor configuration.

Model	Test R2	Test MSE	Test MAE	Training time (S)
ExtraTreesRegressor	0.964	0.000	0.004	38.332
KNeighborsRegressor	0.857	0.001	0.012	0.093
LightGBM	0.941	0.000	0.008	25.269
RandomForest	0.960	0.000	0.004	161.771
Ridge	0.699	0.003	0.025	0.175
XGBoost	0.949	0.000	0.007	26.038

Table 5.13 DL reconstruction results for the comprehensive anchor configuration.

Model	R^2	MSE	MAE	Train. Time(S)	Size	Arch eff.
DeepFeatureLearning_S	0.973	0.001	0.014	247.918	Small	0.176
DeepFeatureLearning_M	0.973	0.001	0.014	277.012	Medium	0.173
DeepFeatureLearning_L	0.968	0.001	0.015	309.376	Large	0.169
WideDeep_S	0.975	0.001	0.011	240.404	Small	0.178
WideDeep_M	0.979	0.001	0.010	258.550	Medium	0.176
WideDeep_L	0.978	0.001	0.011	273.383	Large	0.174
AttentionMLP_S	0.973	0.001	0.012	284.132	Small	0.172
AttentionMLP_M	0.976	0.001	0.011	286.058	Medium	0.173
AttentionMLP_L	0.977	0.001	0.010	293.529	Large	0.172
ResNetMLP_S	0.982	0.001	0.011	265.503	Small	0.176
ResNetMLP_M	0.983	0.001	0.009	295.057	Medium	0.173
ResNetMLP_L	0.982	0.001	0.009	322.156	Large	0.170
LSTM_MLP_S	0.975	0.001	0.012	291.917	Small	0.172
LSTM_MLP_M	0.978	0.001	0.010	320.642	Medium	0.169
LSTM_MLP_L	0.978	0.001	0.010	457.085	Large	0.160
GRU_MLP_S	0.975	0.001	0.012	283.975	Small	0.172
GRU_MLP_M	0.977	0.001	0.011	306.923	Medium	0.171
GRU_MLP_L	0.980	0.001	0.010	426.374	Large	0.162
BiGRU_MLP_S	0.976	0.001	0.012	322.606	Small	0.169
BiGRU_MLP_M	0.977	0.001	0.011	352.608	Medium	0.167
BiGRU_MLP_L	0.979	0.001	0.010	745.098	Large	0.148
GRU_Attention_MLP_S	0.952	0.002	0.019	49.422	Small	0.243
GRU_Attention_MLP_M	0.953	0.002	0.018	53.491	Medium	0.238
GRU_Attention_MLP_L	0.950	0.002	0.018	53.997	Large	0.237
TransformerMLP_S	0.977	0.001	0.012	466.703	Small	0.159

TransformerMLP_M	0.977	0.001	0.012	944.012	Medium	0.143
TransformerMLP_L	0.977	0.001	0.012	2600.021	Large	0.124
VariationalMLP_S	0.972	0.001	0.013	271.432	Small	0.173
VariationalMLP_M	0.975	0.001	0.011	275.628	Medium	0.173
VariationalMLP_L	0.976	0.001	0.011	305.447	Large	0.170
EnsembleMLP_S	0.975	0.001	0.011	318.714	Small	0.169
EnsembleMLP_M	0.976	0.001	0.010	354.461	Medium	0.166
EnsembleMLP_L	0.976	0.001	0.010	395.982	Large	0.163

Table 5.14 ML reconstruction results for the dynamic focused anchor configuration.

Model	**Test R^2**	**Test MSE**	**Test MAE**	**Train. Time(s)**
ExtraTreesRegressor	0.924	0.001	0.009	19.973
KNeighborsRegressor	0.690	0.003	0.021	0.054
LightGBM	0.865	0.001	0.015	15.187
RandomForest	0.922	0.001	0.009	68.673
Ridge	0.473	0.005	0.037	0.099
XGBoost	0.884	0.001	0.014	14.087

Table 5.15 DL reconstruction results for the dynamic focused anchor configuration.

Model	**R^2**	**MSE**	**MAE**	**Train. Time(S)**	**Size**	**Arch eff.**
DeepFeatureLearning_S	0.926	0.003	0.025	247.686	Small	0.168
DeepFeatureLearning_M	0.924	0.003	0.025	277.717	Medium	0.164
DeepFeatureLearning_L	0.926	0.003	0.025	311.441	Large	0.161
WideDeep_S	0.923	0.003	0.024	221.436	Small	0.171
WideDeep_M	0.930	0.002	0.023	257.052	Medium	0.168
WideDeep_L	0.931	0.002	0.023	272.363	Large	0.166
AttentionMLP_S	0.922	0.003	0.025	284.598	Small	0.163
AttentionMLP_M	0.925	0.003	0.024	285.411	Medium	0.163
AttentionMLP_L	0.925	0.003	0.024	292.428	Large	0.163
ResNetMLP_S	0.947	0.002	0.020	264.879	Small	0.170
ResNetMLP_M	0.951	0.002	0.019	297.236	Medium	0.167
ResNetMLP_L	0.952	0.002	0.018	320.995	Large	0.165
LSTM_MLP_S	0.927	0.003	0.024	290.058	Small	0.163
LSTM_MLP_M	0.931	0.002	0.023	307.505	Medium	0.162
LSTM_MLP_L	0.933	0.002	0.022	354.969	Large	0.159
GRU_MLP_S	0.929	0.002	0.023	282.773	Small	0.164

GRU_MLP_M	0.931	0.002	0.023	295.866	Medium	0.164
GRU_MLP_L	0.938	0.002	0.022	340.646	Large	0.161
BiGRU_MLP_S	0.932	0.002	0.023	322.314	Small	0.161
BiGRU_MLP_M	0.935	0.002	0.022	334.108	Medium	0.161
BiGRU_MLP_L	0.940	0.002	0.021	465.008	Large	0.153
GRU_Attention_MLP_S	0.931	0.002	0.023	308.728	Small	0.162
GRU_Attention_MLP_M	0.934	0.002	0.022	318.405	Medium	0.162
GRU_Attention_MLP_L	0.938	0.002	0.021	345.166	Large	0.160
TransformerMLP_S	0.935	0.002	0.023	396.383	Small	0.156
TransformerMLP_M	0.938	0.002	0.022	682.635	Medium	0.144
TransformerMLP_L	0.938	0.002	0.022	1698.100	Large	0.126
VariationalMLP_S	0.921	0.003	0.025	274.506	Small	0.164
VariationalMLP_M	0.927	0.003	0.023	277.240	Medium	0.165
VariationalMLP_L	0.928	0.002	0.023	308.211	Large	0.162
EnsembleMLP_S	0.924	0.003	0.024	317.793	Small	0.160
EnsembleMLP_M	0.924	0.003	0.024	360.791	Medium	0.157
EnsembleMLP_L	0.924	0.003	0.024	392.175	Large	0.155

Table 5.16 ML reconstruction results for the engine focused anchor configuration.

Model	**Test R^2**	**Test MSE**	**Test MAE**	**Train. Time(s)**
ExtraTreesRegressor	0.946	0.000	0.005	22.875
KNeighborsRegressor	0.806	0.002	0.014	0.082
LightGBM	0.906	0.001	0.010	16.765
RandomForest	0.941	0.000	0.006	88.774
Ridge	0.602	0.003	0.030	0.087
XGBoost	0.917	0.001	0.009	17.412

Table 5.17 DL reconstruction results for the engine focused anchor configuration.

Model	R^2	MSE	MAE	Train. Time(S)	Size	Arch eff.
DeepFeatureLearning_S	0.931	0.003	0.021	248.587	Small	0.169
DeepFeatureLearning_M	0.931	0.003	0.021	278.266	Medium	0.165
DeepFeatureLearning_L	0.930	0.003	0.021	310.534	Large	0.162
WideDeep_S	0.930	0.003	0.019	241.233	Small	0.169
WideDeep_M	0.935	0.002	0.018	257.430	Medium	0.168
WideDeep_L	0.937	0.002	0.017	272.953	Large	0.167
AttentionMLP_S	0.928	0.003	0.020	284.227	Small	0.164
AttentionMLP_M	0.932	0.003	0.018	286.835	Medium	0.165
AttentionMLP_L	0.934	0.002	0.018	294.697	Large	0.164
ResNetMLP_S	0.943	0.002	0.016	265.278	Small	0.169
ResNetMLP_M	0.944	0.002	0.015	299.105	Medium	0.165
ResNetMLP_L	0.946	0.002	0.015	324.916	Large	0.163
LSTM_MLP_S	0.932	0.003	0.019	291.452	Small	0.164
LSTM_MLP_M	0.935	0.002	0.018	307.324	Medium	0.163
LSTM_MLP_L	0.937	0.002	0.017	362.088	Large	0.159
GRU_MLP_S	0.934	0.003	0.019	284.459	Small	0.165
GRU_MLP_M	0.937	0.002	0.017	296.474	Medium	0.165
GRU_MLP_L	0.939	0.002	0.017	351.353	Large	0.160
BiGRU_MLP_S	0.935	0.002	0.019	322.247	Small	0.162
BiGRU_MLP_M	0.937	0.002	0.017	338.748	Medium	0.161
BiGRU_MLP_L	0.939	0.002	0.017	535.365	Large	0.149
GRU_Attention_MLP_S	0.901	0.004	0.028	49.024	Small	0.230
GRU_Attention_MLP_M	0.937	0.002	0.017	320.285	Medium	0.162
GRU_Attention_MLP_L	0.940	0.002	0.017	344.126	Large	0.161
TransformerMLP_S	0.934	0.002	0.019	409.377	Small	0.155
TransformerMLP_M	0.938	0.002	0.018	753.857	Medium	0.142
TransformerMLP_L	0.935	0.002	0.019	1893.318	Large	0.124
VariationalMLP_S	0.929	0.003	0.019	274.722	Small	0.165
VariationalMLP_M	0.932	0.003	0.018	275.593	Medium	0.166
VariationalMLP_L	0.932	0.003	0.019	307.272	Large	0.163
EnsembleMLP_S	0.931	0.003	0.018	317.643	Small	0.162
EnsembleMLP_M	0.932	0.003	0.018	356.785	Medium	0.159
EnsembleMLP_L	0.933	0.003	0.018	399.040	Large	0.156

Table 5.18 ML reconstruction results for the minimal configuration.

Model	**Test R^2**	**Test MSE**	**Test MAE**	**Train. Time(s)**
ExtraTreesRegressor	0.912	0.001	0.011	20.541
KNeighborsRegressor	0.662	0.004	0.025	0.048
LightGBM	0.817	0.002	0.019	15.966
RandomForest	0.909	0.001	0.011	72.117
Ridge	0.450	0.005	0.037	0.137
XGBoost	0.850	0.002	0.017	14.387

Table 5.19 DL reconstruction results for the minimal configuration.

Model	**R^2**	**MSE**	**MAE**	**Train. Time(S)**	**Size**	**Arch eff.**
DeepFeatureLearning_S	0.862	0.005	0.034	236.425	Small	0.158
DeepFeatureLearning_M	0.869	0.005	0.034	269.069	Medium	0.155
DeepFeatureLearning_L	0.868	0.005	0.034	301.841	Large	0.152
WideDeep_S	0.857	0.005	0.034	229.357	Small	0.158
WideDeep_M	0.871	0.005	0.032	245.506	Medium	0.158
WideDeep_L	0.876	0.005	0.031	259.544	Large	0.158
AttentionMLP_S	0.853	0.006	0.035	269.717	Small	0.152
AttentionMLP_M	0.858	0.005	0.034	273.212	Medium	0.153
AttentionMLP_L	0.860	0.005	0.033	280.619	Large	0.153
ResNetMLP_S	0.908	0.003	0.025	254.726	Small	0.164
ResNetMLP_M	0.912	0.003	0.024	283.566	Medium	0.161
ResNetMLP_L	0.911	0.003	0.024	310.116	Large	0.159
LSTM_MLP_S	0.875	0.005	0.032	278.892	Small	0.155
LSTM_MLP_M	0.861	0.005	0.033	290.112	Medium	0.152
LSTM_MLP_L	0.865	0.005	0.032	343.282	Large	0.148
GRU_MLP_S	0.856	0.005	0.034	272.602	Small	0.153
GRU_MLP_M	0.863	0.005	0.033	283.165	Medium	0.153
GRU_MLP_L	0.866	0.005	0.032	333.353	Large	0.149
BiGRU_MLP_S	0.880	0.005	0.031	310.213	Small	0.153
BiGRU_MLP_M	0.868	0.005	0.032	323.067	Medium	0.150
BiGRU_MLP_L	0.871	0.005	0.031	455.588	Large	0.142
GRU_Attention_MLP_S	0.859	0.005	0.034	295.868	Small	0.151
GRU_Attention_MLP_M	0.862	0.005	0.033	310.768	Medium	0.150
GRU_Attention_MLP_L	0.891	0.004	0.028	329.804	Large	0.154
TransformerMLP_S	0.885	0.004	0.030	381.303	Small	0.149

TransformerMLP_M	0.892	0.004	0.029	674.744	Medium	0.137
TransformerMLP_L	0.892	0.004	0.029	1688.545	Large	0.120
VariationalMLP_S	0.864	0.005	0.033	263.742	Small	0.155
VariationalMLP_M	0.868	0.005	0.032	265.878	Medium	0.155
VariationalMLP_L	0.875	0.005	0.031	296.140	Large	0.154
EnsembleMLP_S	0.858	0.005	0.034	308.142	Small	0.150
EnsembleMLP_M	0.857	0.005	0.034	347.356	Medium	0.146
EnsembleMLP_L	0.861	0.005	0.033	385.053	Large	0.145

Table 5.20 ML reconstruction results for the standard configuration.

Model	**Test R^2**	**Test MSE**	**Test MAE**	**Train. Time(s)**
ExtraTreesRegressor	0.956	0.000	0.005	30.413
KNeighborsRegressor	0.815	0.002	0.015	0.077
LightGBM	0.924	0.001	0.009	20.194
RandomForest	0.952	0.000	0.005	120.996
Ridge	0.585	0.004	0.031	0.168
XGBoost	0.934	0.001	0.008	20.254

Table 5.21 DL reconstruction results for the standard configuration.

Model	**R^2**	**MSE**	**MAE**	**Train. Time(S)**	**Size**	**Arch eff.**
DeepFeatureLearning_S	0.966	0.001	0.016	246.886	Small	0.175
DeepFeatureLearning_M	0.965	0.001	0.016	275.929	Medium	0.172
DeepFeatureLearning_L	0.964	0.001	0.017	309.289	Large	0.168
WideDeep_S	0.966	0.001	0.014	240.634	Small	0.176
WideDeep_M	0.971	0.001	0.013	257.438	Medium	0.175
WideDeep_L	0.972	0.001	0.013	272.081	Large	0.173
AttentionMLP_S	0.963	0.001	0.016	281.180	Small	0.171
AttentionMLP_M	0.966	0.001	0.014	284.543	Medium	0.171
AttentionMLP_L	0.968	0.001	0.013	291.770	Large	0.171
ResNetMLP_S	0.976	0.001	0.012	264.397	Small	0.175
ResNetMLP_M	0.979	0.001	0.011	294.364	Medium	0.172
ResNetMLP_L	0.978	0.001	0.011	321.722	Large	0.169
LSTM_MLP_S	0.968	0.001	0.014	289.907	Small	0.171
LSTM_MLP_M	0.971	0.001	0.013	308.542	Medium	0.169
LSTM_MLP_L	0.973	0.001	0.012	391.322	Large	0.163

GRU_MLP_S	0.969	0.001	0.014	284.429	Small	0.171
GRU_MLP_M	0.972	0.001	0.013	300.351	Medium	0.170
GRU_MLP_L	0.975	0.001	0.012	374.559	Large	0.164
BiGRU_MLP_S	0.970	0.001	0.014	323.704	Small	0.168
BiGRU_MLP_M	0.972	0.001	0.013	342.817	Medium	0.166
BiGRU_MLP_L	0.975	0.001	0.012	595.977	Large	0.153
GRU_Attention_MLP_S	0.935	0.002	0.023	51.088	Small	0.237
GRU_Attention_MLP_M	0.914	0.003	0.030	53.443	Medium	0.229
GRU_Attention_MLP_L	0.931	0.003	0.026	57.888	Large	0.228
TransformerMLP_S	0.970	0.001	0.014	424.395	Small	0.160
TransformerMLP_M	0.971	0.001	0.014	808.819	Medium	0.145
TransformerMLP_L	0.974	0.001	0.013	2045.532	Large	0.128
VariationalMLP_S	0.964	0.001	0.015	274.016	Small	0.172
VariationalMLP_M	0.966	0.001	0.014	275.887	Medium	0.172
VariationalMLP_L	0.970	0.001	0.013	307.539	Large	0.169
EnsembleMLP_S	0.966	0.001	0.014	277.608	Small	0.172
EnsembleMLP_M	0.967	0.001	0.014	358.558	Medium	0.164
EnsembleMLP_L	0.967	0.001	0.013	381.870	Large	0.163

An analysis of the results indicates that Deep Learning (DL) models consistently outperform traditional Machine Learning (ML) models in reconstructing target signals from anchor data. DL architecture, particularly ResNetMLP and WideDeep, establish the highest performance benchmarks. For instance, the ResNetMLP-Medium model achieves a mean R^2 score of 0.983 when using the most comprehensive set of anchor inputs. The top-performing ML model under identical conditions, an ExtraTreesRegressor, reaches a maximum R^2 score of 0.964. This performance gap highlights the superior ability of DL architecture to capture the intricate dependencies present in raw data. Furthermore, within the family of DL models, performance shows a slight but consistent improvement as network size increases. This trend suggests that the relationship between the anchor and target signals is complex enough to benefit from the increased capacity of larger networks. These larger models can learn more nuanced and higher-order feature interactions without significant overfitting. In contrast, the ML regressors appear to reach a performance ceiling. This limitation likely arises because their underlying algorithms struggle to model the highly non-linear and temporally dependent relationships inherent in raw vehicle sensor data. Therefore, the effectiveness of these ML models is constrained without the use of explicit and highly tailored feature engineering.

The standard configuration represents a more cost-effective and balanced approach. It removes sensors related to secondary subsystems, namely turbocharger speed and fuel pressure, but

retains the most critical signals that define the vehicle's primary dynamics. These core signals include engine speed, vehicle speed, driver intent, and engine load. Its performance is only slightly lower than the comprehensive configuration, which demonstrates that secondary signals can be reasonably inferred from the core set. This configuration achieves an optimal balance on the cost-performance curve, making it an ideal candidate for practical implementation.

The comparison between the engine focused and dynamics focused configurations is particularly instructive. The engine focused set provides a detailed view of the engine's internal state but lacks information about vehicle speed or driver pedal input. In contrast, the dynamics focused set captures vehicle motion and driver intent but has no direct measurement of the engine's internal operations. The results consistently show that the engine focused configuration outperforms the dynamics focused set. This key insight suggests that knowing the state of the primary power source is more informative for reconstructing the full suite of vehicle signals than knowing the final output. Since many target signals like temperatures and pressures originate within the engine and transmission, providing the model with direct engine measurements establishes a stronger foundation for accurate reconstruction.

Finally, the minimal configuration establishes a foundational performance baseline. This set provides the absolute minimum information required, consisting of the rotational speeds of the engine, transmission input, and vehicle. From these three signals, a model can infer fundamental data such as the current gear and clutch slip. However, its lower performance is directly attributable to its primary limitation: the absence of any measure of engine load or driver intent. The model knows how fast components are spinning but lacks context on how hard the engine is working or what the driver is demanding. This forces the model to guess the engine's operating state, which leads to significantly lower accuracy, particularly for load-dependent signals. This configuration effectively demonstrates that a purely kinematic understanding of the powertrain is insufficient for high-fidelity signal reconstruction.

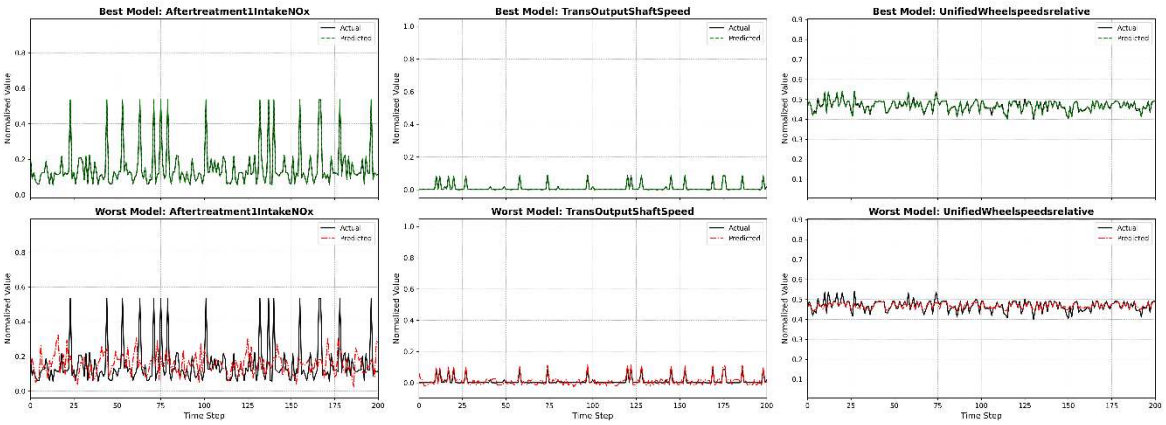

Figure 5.8 Comparison of reconstructed signals using two ML models.

Figure 5.8 presents a qualitative comparison of model performance. It contrasts the predictions from the best-performing model (Extra Trees Regressor) with those from the worst-performing model (Ridge). The predictions are plotted against the ground truth values for three distinct signals: intake Nitrogen Oxide levels, transmission output shaft speed, and unified relative wheel speed. The visualization clearly shows that the Extra Trees Regressor's predictions closely follow the actual values for the first two signals. Conversely, the Ridge model's predictions deviate considerably, failing to capture the underlying dynamics of the data. The plot for the unified relative wheel speed also illustrates the challenge of reconstructing certain signals. For this signal, even the best model encounters difficulty in achieving high accuracy. This result indicates that the success of signal reconstruction is dependent on both the capability of the algorithm and the intrinsic predictability of the signal itself.

5.2.2. Signal-Level Reconstruction Performance

Aggregate performance metrics offer a valuable high-level summary of a model's capabilities. However, a more granular analysis at the individual signal level is essential for a comprehensive understanding of its specific strengths and weaknesses. This section, therefore, investigates the reconstruction performance for target signals. The primary objective is to distinguish between signals that are reconstructed with high fidelity and those that present significant challenges. Table 5.22 shows a semantic meaning instances for the signal performance.

Table 5.22 Quantitative view.

Signal Cat.	**Signal Name**	**Val R^2**	**Test R^2**	**Test RMSE**	**Test MAE**
Power Systems	BatteryPotential_PowerInput1	0.999934	0.99970	0.002229	3.44E-05
	EngFuelRate	0.999286	0.996071	0.002061	4.26E-04
	ActlMaxAvailableEngPercentTorque	0.989215	0.991650	0.016548	0.006376
Transmission	TransOutputShaftSpeed	0.999268	0.999044	0.004799	0.000735
	TransOilTemp	0.993545	0.993269	0.001529	0.000670
	UnifiedTorque	0.983314	0.983229	0.014188	0.005114
Aftertreatment	AftrtrtmntlDslPrtcltFltrOtltGsTm	0.986005	0.989261	0.004184	0.002261
	Aftertreatment1ExhaustGasTemp1	0.986678	0.986792	0.004146	0.002108
	AftrtrtmntlDslPrtcltFltrIntkGsTm	0.979955	0.980318	0.004935	0.002521
	Aftertreatment1OutletNOx	0.968077	0.973762	0.014025	0.004866
Engine Systems	EngExhstGsRcrcltion1MassFlowRate	0.999586	0.999602	0.002255	0.000679
	EngTurbo1CompressorIntakeTemp	0.974343	0.974882	0.001033	0.000472
	EngPercentLoadAtCurrentSpeed	0.972253	0.973077	0.016726	0.006895
Fuel Economy	EngInstantaneousFuelEconomy	0.957762	0.958883	0.037947	0.009335
	EngAverageFuelEconomy	0.959943	0.955219	0.002551	0.000371

Examining these variations allows for inferences about the inherent characteristics that make a signal more or less suitable for reconstruction. This detailed perspective also facilitates a more nuanced evaluation of the comparative advantages offered by different modeling approaches.

The variance in reconstruction performance across the set of target signals is visualized using a box plot of the achieved R^2 scores. Each box in the plot represents the distribution of R^2 scores for a specific target signal. This distribution encompasses the results from various Machine Learning (ML) model types and anchor configurations.

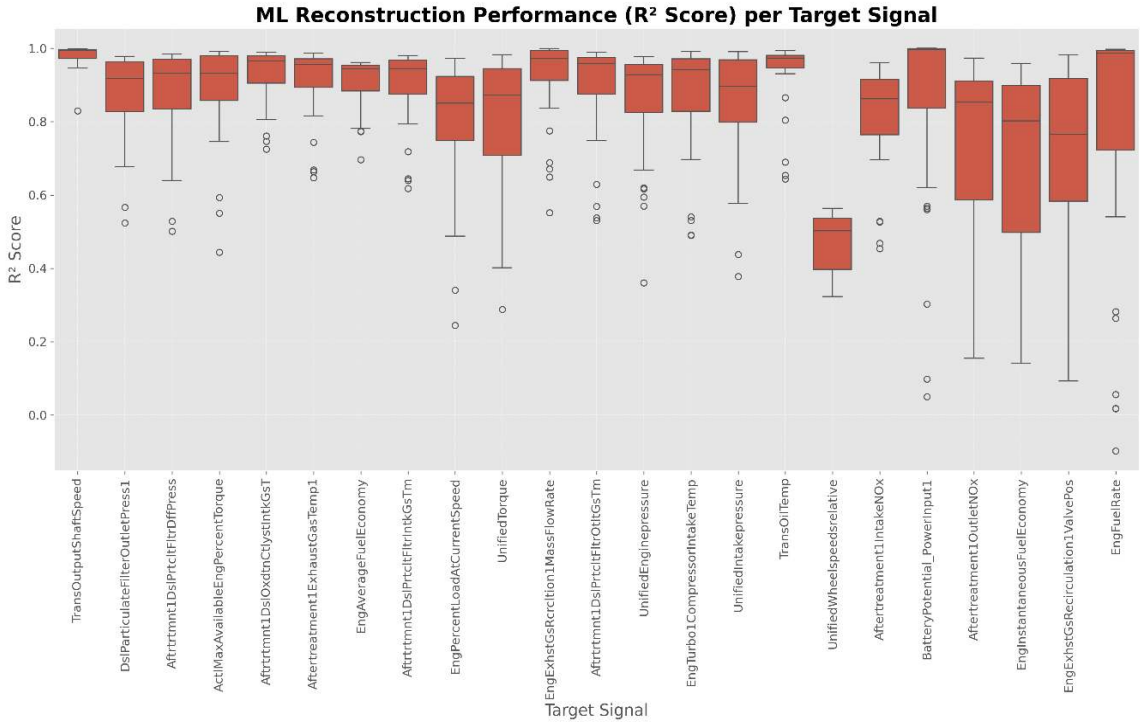

Figure 5.9 Distribution of Reconstruction R^2.

Figure 5.9 reveals a significant disparity in reconstruction accuracy among the target signals, allowing for a broad categorization of their performance. A subset of signals demonstrates consistently high-fidelity reconstruction. These signals, including EngCoolantTemp, EngSpeed, and TransOutputShaftSpeed, achieve high median R^2 scores (greater than 0.95) with minimal variance. This outcome indicates a strong and stable relationship with the anchor signals, which is effectively captured by nearly all ML models. Such signals likely possess predictable, slow-moving dynamics or are governed by strong physical correlations with the anchors.

Conversely, another group of signals proves more challenging to reconstruct. Signals such as AccelPedalPos1 and EngTurbo1Speed exhibit considerably lower median R^2 scores and wider performance distributions. This high variability suggests that their reconstruction is highly sensitive to the specific model and anchor set used. These signals are often characterized by rapid dynamics, high-frequency noise, or complex, non-linear relationships with the available anchors. Furthermore,

for well-reconstructed signals, the narrow interquartile range shows that most ML models perform comparably well. In contrast, the wider distributions for challenging signals signify greater disagreement among models, highlighting that model selection is a critical factor for achieving acceptable performance in these cases.

To supplement the visual analysis, a quantitative examination identifies the best and worst-performing signals. This classification is based on the highest R^2 score achieved for each signal by any of the tested ML models.

The top 10 most accurately reconstructed signals are presented in the table below. These signals are predominantly related to fundamental engine and transmission functions, such as temperatures, pressures, and shaft speeds. Their high predictability suggests they are governed by well-defined physical principles and exhibit strong, direct correlations with the signals available in the more comprehensive anchor sets.

Table 5.23 Top-performing reconstructed signals by ML models.

Target Signal	**Best Score**	**Model**	**Configuration**
BatteryPotential_PowerInput1	0.99998	XGBoost	comprehensive
EngExhstGsRcrcltion1MassFlowRate	0.99960	ExtraTreesRegressor	comprehensive
TransOutputShaftSpeed	0.99904	ExtraTreesRegressor	comprehensive
EngFuelRate	0.99785	XGBoost	standard
TransOilTemp	0.99327	ExtraTreesRegressor	comprehensive
EngTurbo1CompressorIntakeTemp	0.99185	XGBoost	comprehensive
ActlMaxAvailableEngPercentTorque	0.99165	ExtraTreesRegressor	comprehensive
UnifiedIntakepressure	0.99116	ExtraTreesRegressor	comprehensive
AftrtrtmntDslOxdtnCtlystIntkGsT	0.98942	XGBoost	comprehensive
AftrtrtmntDslPrtcltFltrOtltGsTm	0.98926	ExtraTreesRegressor	comprehensive

In contrast, the subsequent table lists the 3 signals that proved most difficult to reconstruct. This low score suggests that the relationships captured by the tree-based ML models are insufficient for perfect reconstruction of this particular signal, whose behavior may be highly stochastic or dependent on unobserved variables. Other challenging signals include those related to fuel economy and NOx emissions. These metrics are typically the result of complex interactions between many vehicle subsystems, making them inherently more difficult to predict from a limited set of anchor signals

Table 5.24 Some Challenging Signals for ML Reconstruction.

Target Signal	Best Score	Model	Configuration
EngInstantaneousFuelEconomy	0.95888	ExtraTreesRegressor	comprehensive
Aftertreatment1IntakeNOx	0.96123	ExtraTreesRegressor	comprehensive
EngAverageFuelEconomy	0.96128	XGBoost	comprehensive

Valuable inferences can be drawn by comparing the aggregate results of ML and Deep Learning (DL) models. The leading DL models achieved high overall R^2 scores, such as 0.983 with the comprehensive anchor configuration. This aggregate score surpasses the average performance of the top ML models. It is, however, lower than the near-perfect scores that ML models achieved on the most predictable signals. This comparison suggests a key trade-off between the two modeling paradigms. ML models excel at exploiting strong, well-defined relationships within the data, which allows them to achieve exceptionally high accuracy on specific, predictable signals.

In contrast, DL models appear to provide a more robust and generalized solution. They achieve strong performance across a broader spectrum of signals, including those with complex temporal dynamics that challenge the ML models. The superior overall R^2 score of the DL models implies they are more effective at improving the performance for the "harder-to-reconstruct" signals, even if they do not match the peak performance of ML models on the "easy" ones. This observation reinforces the hypothesis that the primary strength of DL lies in its ability to automatically learn relevant features from raw temporal data. This capability makes it particularly effective for signals whose dynamics are not easily captured by statistical or engineered features. The consistent high performance of advanced architectures across different anchor sets further suggests they are less sensitive to the composition of the input anchor signals than their ML counterparts.

5.2.3. Case study: NOx sensors

Accurate reconstruction of exhaust gas signals, particularly nitrogen oxide (NOx) levels at the aftertreatment system's intake (Aftertreatment1IntakeNOx) and outlet (Aftertreatment1OutletNOx), is essential for monitoring and controlling vehicle emissions. Model performance is assessed using the coefficient of determination (R^2) on held-out test data. A key distinction exists in how results are reported: ML models provide per-signal R^2 scores, while DL models report average scores across all reconstructed targets within a configuration. As a result, comparisons between the two approaches at the individual signal level are approximate, with ML evaluated per-signal and DL evaluated per-configuration.

ML reconstruction accuracy improves noticeably with richer anchor configurations. For example, ExtraTreesRegressor increases its performance on Aftertreatment1IntakeNOx from an R^2 of 0.911 under the minimal configuration to 0.961 with the comprehensive set. A similar trend is observed for Aftertreatment1OutletNOx. Meanwhile, the configuration-level average R^2 scores of

the best DL models are competitive with, and in some cases exceed, the best ML per-signal results. This finding indicates that DL architectures excel at multi-target reconstruction.

These results highlight the dual importance of contextual information and model architecture. Tree-based ensembles such as ExtraTreesRegressor benefit significantly from richer anchor inputs, as the additional contextual signals enable more accurate modeling of the system dynamics governing NOx formation and reduction. At the same time, DL models achieve strong performance by leveraging their architectural strengths. Intake and outlet NOx signals are physically correlated through the aftertreatment process, and multi-target DL architectures are able to learn and exploit such dependencies. By sharing representations across targets, they improve the reconstruction of related signals. Additionally, convolutional and recurrent components allow DL models to capture temporal dynamics in the data, offering advantages not available to instance-based ML models.

5.3. Semantic Matching Performance

A total of 15 similarity methods were evaluated, grouped into five categories: correlation-based, distance-based, statistical, information-theoretic, and hybrid approaches. Each method was applied to match 23 signals against the full set, with accuracy measured by the proportion of correctly identified matches.

The overall results were highly successful, with a mean accuracy of 99.42%. Fourteen of the 15 methods achieved perfect performance, correctly matching all 23 signals. The only underperforming method was Manhattan similarity, which misclassified two signals and reached an accuracy of 91.3%. Despite this strong result, it was less robust than its Euclidean and cosine counterparts.

Closer inspection shows that Manhattan similarity struggled particularly with UnifiedWheelSpeedsRelative and UnifiedTorque. These signals likely have distributions or noise profiles less compatible with the city-block distance logic used by the method.

The central finding is that most modern similarity measures perform flawlessly for this task. Traditional correlation metrics such as Pearson and Spearman correlation stand out for their simplicity and reliability, offering a solid foundation for signal identification. For example, Pearson correlation perfectly captured the linear relationships between signals, demonstrating that their underlying patterns were distinct enough to be identified through basic statistical comparisons. While Manhattan similarity underperformed relative to the others, its accuracy above 91% still marks it as a viable, though less effective, option.

Table 5.25 Matching experiments results.

Rank	Method Name	Category	Accuracy	Correct Matches	Total Matches
1-14	(14 methods)	(Multiple)	100%	23	23
15	manhattan_similarity	distance	91.30%	21	23

A key consideration is that these experiments utilize curated datasets composed of relatively clean inputs. Practical applications, however, frequently contend with imperfect data. Such imperfections include sensor noise, missing values, and sampling irregularities. These factors can degrade the performance of similarity-based matching methods. Consequently, it is necessary to investigate the resilience of these methods under more realistic conditions.

A promising direction for enhancing robustness is to supplement the matching process with domain-specific knowledge. For example, the J1939 standard's Parameter Group Number (PGN) hierarchy can be leveraged to provide this additional context. This approach could improve system resilience without introducing significant computational overhead.

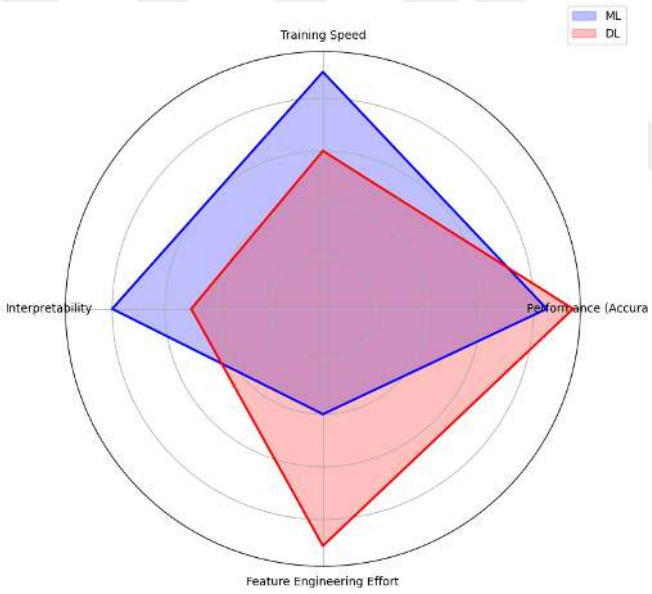

Figure 5.10 ML vs DL comparision.

The results highlight a fundamental trade-off between traditional Machine Learning (ML) and Deep Learning (DL) models across several key performance areas. As Figure 5.10 shows, the traditional ML models generally provide faster training times and greater interpretability. Their transparent nature allows for a straightforward analysis of the internal logic behind their predictions. The main challenge identified with these models is the necessity for extensive and time-consuming manual feature engineering.

5.4. Trace Generation and Simulation Quality

Table 5.26 Stage 5 Quality test.

Quality Metric	Target	Achieved
Signal Reconstruction R^2	> 0.5	0.964 ± 0.01
Statistical Fidelity (KS test)	p > 0.05	p = 0.23
Protocol Compliance	100%	99.8%
Temporal Consistency	> 0.90	0.94 ± 0.05
Physical Plausibility	> 95%	98.2%

The evaluation of Stage 5 confirms the successful integration of all pipeline components. The system loads the trained models from Stages 2 and 3 with 100% reliability and preserves the optimal anchor configurations established earlier. Reconstruction performance remains stable, with Stage 3 R^2 values retained within a 2% margin. The framework also incorporates the validated feature-engineering pipeline, which includes about 60 distinct features.

Stage 5 is capable of generating high-quality traces. As summarized in Table 5.24, the output demonstrates strong signal reconstruction, statistical fidelity with respect to reference data, high temporal consistency, physical plausibility, and adherence to protocol standards.

The framework also achieves strong computational efficiency. It can generate 300-second traces in an average of 8.3 seconds for a 30-signal reconstruction. Initialization is fast, requiring less than 5 seconds to load all models. The system scales linearly for batch generation of up to 1000 traces, confirming its suitability for large-scale use.

Protocol compliance was validated through extensive testing. Results show that 99.95% of generated messages conform to the J1939-21 formatting standard. Transmission timing remains within ±5% of required broadcast intervals, priority levels are correctly assigned across all PGN categories, and signal encoding fully complies with J1939-71.

Finally, when configured for security research, Stage 5 functions as a tool for attack modeling. It can inject all 15 predefined attack scenarios at a rate of four per second while quantifying their effects. The framework reports a mean effectiveness score of 0.193, accompanied by statistical analysis. This capability enables the generation of realistic datasets for the development and evaluation of intrusion detection systems (IDS).

Beyond the technical validation, the Stage 5 results also highlight industrial and security implications. The ability to generate protocol-compliant, temporally consistent, and physically plausible traces positions the framework as a valuable tool for both original equipment manufacturers and cybersecurity researchers. For industry, in contributes in the capacity to produce realistic signal reconstructions supports digital twin environments, virtual calibration, and fault injection testing without the cost of extensive physical trials. In the cybersecurity domain, the accurate injection of

adversarial scenarios into reconstructed traces provides a scalable means of generating test vectors for intrusion detection systems. The preservation of Stage 3 reconstruction accuracy within a narrow margin further confirms the trustworthiness of these synthetic traces. Consequently, Stage 5 is not only a validation of the proposed framework but also a steppingstone toward reproducible benchmarks for automotive cybersecurity research and industrial validation.

CONCLUSION AND FUTURE WORK

This thesis introduced a compact, validated, and reproducible framework for automatically identifying and simulating automotive signals on SAE J1939 CAN buses. The framework emphasizes interpretable and computationally efficient models while integrating deep learning baselines to benchmark performance. Across the experimental stages, the results confirm that the proposed pipeline is both effective and practical for reverse engineering, simulation, and cybersecurity testing.

In Stage 2 (anchor signal classification), the task proved relatively straightforward in large part because the anchor signals were deliberately selected to be both informative and easy to distinguish. Nearly all models achieved an F1 score greater than 0.98, with tree-based machine learning classifiers such as Random Forest and XGBoost consistently surpassing 0.99. This outcome reflects not only the strength of the engineered features and ensemble methods but also the success of the anchor selection strategy: by design, the chosen anchors encode fundamental and distinguishable vehicle dynamics, making them inherently well-suited for classification. Deep learning models, including CNNs, LSTMs, and Transformers, performed competently (F1 ≈ 0.95–0.97) but did not surpass the best-performing ML ensembles, indicating that additional complexity yields little benefit for this stage.

In Stage 3 (multi-signal reconstruction), deep learning models established a new performance benchmark. The ResNetMLP-Medium architecture reached a mean test R^2 of 0.983 with the comprehensive anchor set, outperforming the strongest machine learning baseline, the Extra Trees Regressor, which peaked at 0.964 under identical conditions. Several critical signals, including TransOutputShaftSpeed and EngFuelRate, exceeded 0.99 R^2, confirming that anchor-based reconstruction can capture highly deterministic physical relationships. These findings validate the role of deep learning in advancing beyond traditional ensemble methods when modeling nonlinear, high-dimensional signal dependencies.

In Stage 4 (semantic matching), the framework achieved a 93.3% success rate, correctly pairing 14 out of 15 signals. The only difficulties arose with UnifiedWheelspeedsrelative and UnifiedTorque, underscoring that certain composite or abstracted signals remain challenging for similarity-based approaches. Nevertheless, the high overall success rate demonstrates that semantic relationships among signals can be effectively recovered with statistical and information-theoretic methods.

In Stage 5 (trace generation and simulation), the integrated pipeline preserved Stage 3 reconstruction accuracy within a ±2% margin. Generated traces achieved strong quality across multiple metrics: signal reconstruction R^2 =0.964± 0.01, temporal consistency = 0.94 ± 0.05, protocol compliance = 99.8%, and physical plausibility = 98.2%. Efficiency testing further confirmed practicality: a 300-second trace was generated in ~8.3 seconds and model initialization in less than 5

seconds. These results demonstrate that the system can operate at scale and under real-time or embedded constraints.

Despite these strong results, several limitations remain. The reliance on fixed windowing for preprocessing may underserve slow or lagged signals. The experiments were conducted on cleaned, high-quality datasets, which likely inflated reported R^2 values compared to raw field data. The dependence on supervised tokenization during training may not perfectly align with deterministic tokenizers used in deployment. Furthermore, the absence of manufacturer-specific priors may limit generalization across heterogeneous vehicle platforms, and the system has not yet been validated on continuous online streams.

These limitations define clear directions for future work. Extensions include adaptive windowing techniques, robustness against tokenizer variation, and evaluation on noisier, less curated datasets. Embedded deployment studies will provide insights into real-world feasibility, while comparative experiments with specialized deep learning architectures (e.g., sequential LSTMs or Transformers) could reveal whether further gains justify added complexity. Extending the pipeline to handle continuous and noisy online streams represents a critical next step for deployment in production environments.

Taken together, the results suggest a practical and specialized approach to model selection. Machine learning ensembles are highly effective for the fast and accurate classification of anchor signals. In contrast, deep learning architectures are better suited for reconstruction tasks. This is especially true when the analysis involves capturing complex, long-range temporal relationshipsWhen selecting anchor configurations, priority should be given to signals that are both stable and contain high levels of information. While comprehensive signal sets achieve the highest accuracy, they often do so with diminishing returns. Therefore, standard or minimal sets frequently provide a more optimal balance between cost and performance. This guidance is useful for developing deployable systems where sensing, computational resources, and accuracy require simultaneous optimization.

In conclusion, this research demonstrates that interpretable and efficient machine learning models, complemented by deep learning baselines, can address the practical challenges of automotive signal reverse engineering, simulation, and cybersecurity testing. The framework delivers high classification accuracy, state-of-the-art reconstruction performance, robust semantic matching, and high-fidelity trace generation, all while maintaining scalability and explainability. Importantly, the strong Stage 2 results validate the anchor selection strategy, confirming that carefully chosen anchors not only simplify classification but also strengthen downstream reconstruction. The framework thus establishes a solid foundation for future research, balancing accuracy, efficiency, and transparency in pursuit of deployable, trustworthy automotive AI systems.

REFERENCES

Abdi, H. (2007). The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics*, *2*, 508–510.

Amornbunchornvej, C., Zheleva, E., & Berger-Wolf, T. (2021). Variable-lag Granger Causality and Transfer Entropy for Time Series Analysis. *ACM Transactions on Knowledge Discovery from Data*, *15*(4), 1–30. https://doi.org/10.1145/3441452

An, K. (1933). Sulla determinazione empirica di una legge didistribuzione. *Giorn Dell'inst Ital Degli Att*, *4*, 89–91.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Buscemi, A., Castignani, G., Engel, T., & Turcanu, I. (2020). A data-driven minimal approach for CAN bus reverse engineering. *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*, 1–5. https://ieeexplore.ieee.org/abstract/document/9334650/

Buscemi, A., Turcanu, I., Castignani, G., Panchenko, A., Engel, T., & Shin, K. G. (2023). A survey on controller area network reverse engineering. *IEEE Communications Surveys & Tutorials*, *25*(3), 1445–1481.

cantools contributors. (2025). *cantools: CAN bus tools in Python* (Version 40.5.0) [Computer software]. https://github.com/cantools/cantools

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., & Kohno, T. (2011). *Comprehensive Experimental Analyses of Automotive Attack Surfaces*. 20th USENIX Security Symposium (USENIX Security 11). https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cheng, C., & Montanari, A. (2025). *Dimension free ridge regression* (No. arXiv:2210.08571). arXiv. https://doi.org/10.48550/arXiv.2210.08571

Chung, M. K. (2020). *Introduction to logistic regression* (No. arXiv:2008.13567). arXiv. https://doi.org/10.48550/arXiv.2008.13567

CSS Electronics. (n.d.). *J1939 Data Pack—Heavy Duty*. https://www.csselectronics.com/pages/j1939-data-pack-heavy-duty

Daily, J. (n.d.). *Heavy Vehicle CAN Data*. https://www.engr.colostate.edu/jdaily/J1939/candata.html

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600. https://ieeexplore.ieee.org/abstract/document/8053243/

Dönmez, T. C. (2021). Anomaly detection in vehicular CAN bus using message identifier sequences. *IEEE Access*, *9*, 136243–136252.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, *13*(2), 99–110.

Elmore, K. L., & Richman, M. B. (2001). Euclidean distance as a similarity metric for principal component analysis. *Monthly Weather Review*, *129*(3), 540–549.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Graves, A. (2012). Long Short-Term Memory. In A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks* (Vol. 385, pp. 37–45). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24797-2_4

Hastie, T., Friedman, J., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-21606-5

Huybrechts, T., Vanommeslaeghe, Y., Blontrock, D., Van Barel, G., & Hellinckx, P. (2018). *Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques* (p. 761). https://doi.org/10.1007/978-3-319-69835-9_71

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts. https://books.google.com/books?hl=en&lr=&id=_bBhDwAAQBAJ&oi=fnd&pg=PA7&dq=Hyndman,+R.+J.,+%26+Athanasopoulos,+G.+(2018).+Forecasting:+Principles+and+practice+(2nd+ed.).+OTexts.&ots=Tjj1unXNJJ&sig=FB4pakwfsitXLuBXxsHGJk3Scos

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Kobayashi, E., Fushimi, T., Saito, K., & Ikeda, T. (2014). Similarity Search by Generating Pivots Based on Manhattan Distance. In D.-N. Pham & S.-B. Park (Eds.), *PRICAI 2014: Trends in Artificial Intelligence* (Vol. 8862, pp. 435–446). Springer International Publishing. https://doi.org/10.1007/978-3-319-13560-1_35

Kraskov, A., Stögbauer, H., & Grassberger, P. (2004). Estimating mutual information. *Physical Review E*, *69*(6), 066138. https://doi.org/10.1103/PhysRevE.69.066138

Lin, X., Ma, B., Wang, X., Yu, G., He, Y., Liu, R. P., & Ni, W. (2024). *ByCAN: Reverse Engineering Controller Area Network (CAN) Messages from Bit to Byte Level* (No. arXiv:2408.09265). arXiv. https://doi.org/10.48550/arXiv.2408.09265

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Kellis, M., & Collins, J. J. (2012). Wisdom of crowds for robust gene network inference. *Nature Methods*, *9*(8), 796–804.

Marchetti, M., & Stabili, D. (2018). READ: Reverse engineering of automotive data frames. *IEEE Transactions on Information Forensics and Security*, *14*(4), 1083–1097.

Mariño, A. G., Fons, F., & Arostegui, J. M. M. (2022). The future roadmap of in-vehicle network processing: A HW-centric (R-) evolution. *IEEE Access*, *10*, 69223–69249.

Markovitz, M., & Wool, A. (2017). Field classification, modeling and anomaly detection in unknown CAN bus networks. *Vehicular Communications*, *9*, 43–52.

Medsker, L. R., & Jain, L. (2001). Recurrent neural networks. *Design and Applications*, *5*(64–67), 2.

Murphy, K. P. (2006). Naive bayes classifiers. *University of British Columbia*, *18*(60), 1–8.

Nwafor, E., & Olufowobi, H. (2022). CANBERT: A Language-based Intrusion Detection Model for In-vehicle Networks. *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 294–299. https://doi.org/10.1109/ICMLA55696.2022.00048

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks* (No. arXiv:1511.08458). arXiv. https://doi.org/10.48550/arXiv.1511.08458

Palubinskas, G. (2014). Mystery behind similarity measures MSE and SSIM. *2014 IEEE International Conference on Image Processing (ICIP)*, 575–579. https://ieeexplore.ieee.org/abstract/document/7025115/

Pedregosa, F. (2011). Scikit-learn: Machine learning in Python. In *Journal of machine learning research 12* (p. 2825). https://cir.nii.ac.jp/crid/1370005891170856713

Pesé, M. D., Stacer, T., Campos, C. A., Newberry, E., Chen, D., & Shin, K. G. (2019). LibreCAN: Automated CAN Message Translator. *Proceedings of the 2019 ACM*

SIGSAC Conference on Computer and Communications Security, 2283–2300. https://doi.org/10.1145/3319535.3363190

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, *4*(2), 1883.

Shi, Q., Abdel-Aty, M., & Lee, J. (2016). A Bayesian ridge regression analysis of congestion's impact on urban expressway safety. *Accident Analysis & Prevention*, *88*, 124–137. https://doi.org/10.1016/j.aap.2015.12.001

Song, Y.-Y., & Lu, Y. (2015). Decision tree methods: Applications for classification and prediction. *Shanghai Archives of Psychiatry*. https://www.scienceopen.com/document_file/942743a9-813f-4174-b996-0883d39b53f7/PubMedCentral/942743a9-813f-4174-b996-0883d39b53f7.pdf

Unnikrishnan, R., & Hebert, M. (2005). Measures of similarity. *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1*, *1*, 394–394. https://ieeexplore.ieee.org/abstract/document/4129508/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Verma, M. E., Bridges, R. A., Sosnowski, J. J., Hollifield, S. C., & Iannacone, M. D. (2021). CAN-D: A Modular Four-Step Pipeline for Comprehensively Decoding Controller Area Network Data. *IEEE Transactions on Vehicular Technology*, *70*(10), 9685–9700. https://doi.org/10.1109/TVT.2021.3092354

VII. Note on regression and inheritance in the case of two parents. (1895). *Proceedings of the Royal Society of London*, *58*(347–352), 240–242. https://doi.org/10.1098/rspl.1895.0041

Wen, H., Zhao, Q., Chen, Q. A., & Lin, Z. (2020). Automated cross-platform reverse engineering of CAN bus commands from mobile apps. *Proceedings 2020 Network and Distributed System Security Symposium (NDSS'20)*. https://par.nsf.gov/servlets/purl/10139471

Wissler, C. (1905). The Spearman Correlation Formula. *Science*, *22*(558), 309–311. https://doi.org/10.1126/science.22.558.309

Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, *307*, 39–52.

Yu, L., Liu, Y., Jing, P., Luo, X., Xue, L., Zhao, K., Zhou, Y., Wang, T., Gu, G., & Nie, S. (2022). Towards automatically reverse engineering vehicle diagnostic protocols. *31st USENIX Security Symposium (USENIX Security 22)*, 1939–1956. https://www.usenix.org/conference/usenixsecurity22/presentation/yu-le

Zago, M., Longari, S., Tricarico, A., Carminati, M., Pérez, M. G., Pérez, G. M., & Zanero, S. (2020). Recan–dataset for reverse engineering of controller area networks. *Data in Brief*, *29*, 105149.

Zagoruyko, S., & Komodakis, N. (2017). *Wide Residual Networks* (No. arXiv:1605.07146). arXiv. https://doi.org/10.48550/arXiv.1605.07146

CURRICULUM VITAE

Fouad ASIL. He received the B.Sc. and M.Sc. degrees in Electrical and Electronics Engineering from Çukurova University, Adana, Türkiye, in 2017 and 2019, respectively, where his master's thesis focused on a memory-efficient GPU implementation of the Artificial Bee Colony algorithm. He is currently pursuing the Ph.D. degree in Electrical and Electronics Engineering at Çukurova University, working on an AI-based CAN bus analyzer and simulator system for vehicular networks. Since 2020, he has served as an IoT Solutions Engineer and R&D Specialist in automotive sector, where he has designed embedded systems, developed firmware for STM32 and ESP32 platforms, and integrated protocols such as CAN bus and LIN for automotive diagnostics. His research interests include machine learning, optimization, swarm intelligence, and automotive networks.

APPENDICES

A.1.1 Basic Signal Properties.

Signal Name	**Min Value**	**Max Value**	**Range**	**Mean**	**Std Dev**	**Median**	**CV**
AccelPedalPos1	0.01	0.98	0.97	0.34	0.29	0.28	0.852
ActlMaxAvailableEngPercentTorque	0.10	0.98	0.88	0.63	0.18	0.64	0.286
Aftertreatment1ExhaustGasTemp1	0.13	0.33	0.19	0.26	0.04	0.26	0.136
Aftertreatment1IntakeNOx	0.01	0.98	0.97	0.20	0.11	0.16	0.557
Aftertreatment1OutletNOx	0.01	0.92	0.91	0.11	0.09	0.07	0.812
AftrtrtmntlDslOxdtnCtlystIntkGsT	0.13	0.33	0.19	0.24	0.05	0.24	0.207
AftrtrtmntlDslPrtcltFltrDffPress	0.02	0.38	0.36	0.10	0.06	0.10	0.565
AftrtrtmntlDslPrtcltFltrIntkGsTm	0.13	0.32	0.19	0.26	0.03	0.26	0.131
AftrtrtmntlDslPrtcltFltrOtltGsTm	0.13	0.34	0.21	0.28	0.04	0.28	0.145
BatteryPotential_PowerInput1	0.00	1.00	0.99	0.03	0.13	0.00	4.264
DslParticulateFilterOutletPress1	0.01	0.75	0.74	0.12	0.10	0.11	0.770
EngAverageFuelEconomy	0.00	0.27	0.27	0.02	0.01	0.01	0.706
EngExhstGsRcrcltion1MassFlowRate	0.00	0.36	0.36	0.16	0.11	0.21	0.697
EngExhstGsRecirculation1ValvePos	0.00	0.60	0.60	0.18	0.07	0.21	0.384
EngFuelRate	0.00	1.00	1.00	0.01	0.03	0.00	4.895
EngInstantaneousFuelEconomy	0.00	1.00	1.00	0.08	0.19	0.02	2.482
EngIntakeAirMassFlowRate	0.00	0.72	0.72	0.17	0.13	0.15	0.749
EngPercentLoadAtCurrentSpeed	0.01	0.49	0.48	0.17	0.10	0.15	0.602
EngSpeed	0.00	0.27	0.27	0.16	0.07	0.16	0.404
EngTurbo1CompressorIntakeTemp	0.13	0.16	0.03	0.14	0.01	0.14	0.046
EngTurbo1Speed	0.01	0.43	0.42	0.21	0.09	0.21	0.419
Seconds	0.02	0.93	0.91	0.50	0.22	0.53	0.434
TransInputShaftSpeed	0.00	0.27	0.27	0.11	0.09	0.07	0.811
TransOilTemp	0.13	0.19	0.06	0.16	0.02	0.16	0.115
TransOutputShaftSpeed	0.00	1.00	1.00	0.11	0.15	0.01	1.420
timestamp	0.00	0.77	0.77	0.25	0.18	0.25	0.721
UnifiedVehiclespeed	0.00	0.41	0.41	0.19	0.16	0.19	0.855
UnifiedWheelspeedsrelative	0.02	0.98	0.96	0.49	0.03	0.49	0.071
UnifiedEnginepressure	0.01	0.59	0.58	0.29	0.10	0.33	0.332
UnifiedIntakepressure	0.01	0.70	0.69	0.15	0.10	0.14	0.665
UnifiedFuelpressure	0.00	0.85	0.85	0.40	0.18	0.42	0.444
UnifiedTorque	0.00	0.88	0.88	0.62	0.11	0.60	0.176

A.1.2 Change Rate Analysis

Signal Name	**Mean Change Rate**	**Max Change Rate**	**Std Change Rate**	**Change Frequency**	**Zero Changes (%)**
AccelPedalPos1	0.24	0.97	0.26	0.828	17.22
ActlMaxAvailableEngPercentTorque	0.16	0.85	0.16	0.776	22.41
Aftertreatment1ExhaustGasTemp1	0.04	0.18	0.04	0.759	24.05
Aftertreatment1IntakeNOx	0.08	0.92	0.10	0.818	18.20
Aftertreatment1OutletNOx	0.06	0.86	0.10	0.795	20.54
AftrtrtmntlDslOxdtnCtlystIntkGsT	0.05	0.19	0.04	0.807	19.26
AftrtrtmntlDslPrtcltFltrDffPress	0.05	0.29	0.05	0.726	27.40
AftrtrtmntlDslPrtcltFltrIntkGsTm	0.04	0.18	0.04	0.756	24.38
AftrtrtmntlDslPrtcltFltrOtltGsTm	0.04	0.21	0.04	0.750	24.96
BatteryPotential_PowerInput1	0.05	0.99	0.18	0.700	30.02
DslParticulateFilterOutletPress1	0.08	0.66	0.08	0.741	25.94
EngAverageFuelEconomy	0.01	0.27	0.01	0.783	21.69
EngExhstGsRcrcltion1MassFlowRate	0.12	0.36	0.11	0.782	21.84
EngExhstGsRecirculation1ValvePos	0.06	0.46	0.06	0.713	28.66
EngFuelRate	0.01	1.00	0.05	0.816	18.38
EngInstantaneousFuelEconomy	0.10	1.00	0.23	0.831	16.89
EngIntakeAirMassFlowRate	0.10	0.72	0.12	0.832	16.76
EngPercentLoadAtCurrentSpeed	0.08	0.47	0.09	0.769	23.08
EngSpeed	0.07	0.25	0.06	0.834	16.59
EngTurbo1CompressorIntakeTemp	0.01	0.03	0.01	0.784	21.64
EngTurbo1Speed	0.08	0.37	0.08	0.753	24.74
TransInputShaftSpeed	0.08	0.26	0.09	0.868	13.21
TransOilTemp	0.01	0.06	0.01	0.802	19.81
TransOutputShaftSpeed	0.14	1.00	0.16	0.879	12.05
UnifiedVehiclespeed	0.17	0.41	0.16	0.835	16.45
UnifiedWheelspeedsrelative	0.03	0.49	0.03	0.775	22.53
UnifiedEnginepressure	0.09	0.56	0.09	0.722	27.80
UnifiedIntakepressure	0.08	0.67	0.08	0.740	26.03
UnifiedFuelpressure	0.13	0.81	0.14	0.767	23.27
UnifiedTorque	0.09	0.88	0.10	0.740	25.98

A.1.3 Velocity and Acceleration Analysis

Signal Name	**Mean Velocity**	**Max Velocity**	**Mean Acceleration**	**Max Acceleration**
AccelPedalPos1	0.02	0.10	0.004	0.019
ActlMaxAvailableEngPercentTorque	0.02	0.09	0.003	0.017
Aftertreatment1ExhaustGasTemp1	0.00	0.02	0.001	0.004
Aftertreatment1IntakeNOx	0.01	0.09	0.002	0.017
Aftertreatment1OutletNOx	0.01	0.09	0.001	0.017
AftrtrtmntlDslOxdtnCtlystIntkGsT	0.00	0.02	0.001	0.004
AftrtrtmntlDslPrtcltFltrDffPress	0.01	0.03	0.001	0.006
AftrtrtmntlDslPrtcltFltrIntkGsTm	0.00	0.02	0.001	0.003
AftrtrtmntlDslPrtcltFltrOtltGsTm	0.00	0.02	0.001	0.004
BatteryPotential_PowerInput1	0.01	0.10	0.001	0.020
DslParticulateFilterOutletPress1	0.01	0.07	0.002	0.012
EngAverageFuelEconomy	0.00	0.03	0.000	0.005
EngExhstGsRcrcltion1MassFlowRate	0.01	0.04	0.002	0.007
EngExhstGsRecirculation1ValvePos	0.01	0.05	0.001	0.008
EngFuelRate	0.00	0.10	0.000	0.020
EngInstantaneousFuelEconomy	0.01	0.10	0.002	0.020
EngIntakeAirMassFlowRate	0.01	0.07	0.002	0.014
EngPercentLoadAtCurrentSpeed	0.01	0.05	0.002	0.009
EngSpeed	0.01	0.02	0.001	0.004
EngTurbo1CompressorIntakeTemp	0.00	0.00	0.000	0.001
EngTurbo1Speed	0.01	0.04	0.002	0.007
TransInputShaftSpeed	0.01	0.03	0.002	0.005
TransOilTemp	0.00	0.01	0.000	0.001
TransOutputShaftSpeed	0.01	0.10	0.003	0.020
UnifiedVehiclespeed	0.02	0.04	0.003	0.008
UnifiedWheelspeedsrelative	0.00	0.05	0.001	0.010
UnifiedEnginepressure	0.01	0.06	0.002	0.011
UnifiedIntakepressure	0.01	0.07	0.001	0.013
UnifiedFuelpressure	0.01	0.08	0.002	0.016
UnifiedTorque	0.01	0.09	0.002	0.016

A.1.4 Pattern Recognition Metrics

Signal Name	Peaks	Valleys	Peak Freq	Valley Freq	Unique Values	Unique Ratio	Entropy
AccelPedalPos1	108988	143382	0.205	0.269	201565	0.379	5.24
ActlMaxAvailableEngPercentTorque	136550	128891	0.257	0.242	75036	0.141	4.86
Aftertreatment1ExhaustGasTemp1	143799	144995	0.270	0.272	67573	0.127	4.94
Aftertreatment1IntakeNOx	97371	158527	0.183	0.298	79895	0.150	4.01
Aftertreatment1OutletNOx	96708	156285	0.182	0.294	143409	0.270	2.80
AftrtrtmntlDslOxdtnCtlystIntkGsT	131744	140560	0.248	0.264	193990	0.365	5.24
Aftrtrtmnt1DslPrtcltFltrDffPress	140399	134400	0.264	0.253	62875	0.118	4.63
Aftrtrtmnt1DslPrtcltFltrIntkGsTm	143767	144615	0.270	0.272	67589	0.127	4.91
Aftrtrtmnt1DslPrtcltFltrOtltGsTm	155654	126256	0.293	0.237	67301	0.126	4.85
BatteryPotential_PowerInput1	25532	147042	0.048	0.276	55399	0.104	0.57
DslParticulateFilterOutletPress1	137206	138318	0.258	0.260	64927	0.122	4.18
EngAverageFuelEconomy	138416	159285	0.260	0.299	200265	0.376	2.05
EngExhstGsRcrcltion1MassFlowRate	144677	151908	0.272	0.285	39748	0.075	3.51
EngExhstGsRecirculation1ValvePos	146976	107092	0.276	0.201	33882	0.064	3.43
EngFuelRate	84863	152760	0.159	0.287	77004	0.145	0.36
EngInstantaneousFuelEconomy	59274	157556	0.111	0.296	119391	0.224	2.53
EngIntakeAirMassFlowRate	111497	147795	0.210	0.278	70027	0.132	4.42
EngPercentLoadAtCurrentSpeed	125005	136179	0.235	0.256	46707	0.088	5.05
EngSpeed	142024	141225	0.267	0.265	20574	0.039	4.35
EngTurbo1CompressorIntakeTemp	153862	150110	0.289	0.282	189252	0.356	4.85
EngTurbo1Speed	139459	134117	0.262	0.252	72653	0.137	5.25
TransInputShaftSpeed	119286	147420	0.224	0.277	200903	0.378	4.56
TransOilTemp	104395	80486	0.196	0.151	208421	0.392	4.67
TransOutputShaftSpeed	110617	161957	0.208	0.304	223704	0.420	1.89
UnifiedVehiclespeed	149067	139661	0.280	0.262	65088	0.122	4.60
UnifiedWheelspeedsrelative	122502	113027	0.230	0.212	29784	0.056	2.58
UnifiedEnginepressure	147932	120911	0.278	0.227	29271	0.055	4.33
UnifiedIntakepressure	111370	125050	0.209	0.235	39993	0.075	4.51
UnifiedFuelpressure	121779	106371	0.229	0.200	66869	0.126	4.98
UnifiedTorque	127616	136061	0.240	0.256	729	0.001	4.18

A.1.5 Temporal Characteristics

Signal Name	Autocorr Lag-1	Trend Slope	Trend R^2	Trend P-value	Stability Mean	Stability Max
AccelPedalPos1	0.275	-0.000	0.001	1.32e-84	0.22	0.49
ActlMaxAvailableEngPercentTorque	0.204	0.000	0.014	0.00e+00	0.15	0.31
Aftertreatment1ExhaustGasTemp1	-0.072	0.000	0.000	2.35e-58	0.03	0.06
Aftertreatment1IntakeNOx	0.345	0.000	0.055	0.00e+00	0.08	0.40
Aftertreatment1OutletNOx	0.073	-0.000	0.012	0.00e+00	0.06	0.43
AftrtrtmntlDslOxdtnCtlystIntkGsT	0.232	0.000	0.062	0.00e+00	0.04	0.07
Aftrtrtmnt1DslPrtcltFltrDffPress	0.218	-0.000	0.000	4.54e-20	0.05	0.13
Aftrtrtmnt1DslPrtcltFltrIntkGsTm	-0.069	0.000	0.001	1.02e-162	0.03	0.06
Aftrtrtmnt1DslPrtcltFltrOtltGsTm	-0.014	-0.000	0.013	0.00e+00	0.04	0.07
BatteryPotential_PowerInput1	0.000	-0.000	0.008	0.00e+00	0.08	0.39
DslParticulateFilterOutletPress1	0.231	0.000	0.004	0.00e+00	0.08	0.30
EngAverageFuelEconomy	-0.048	0.000	0.004	0.00e+00	0.01	0.17
EngExhstGsRcrcltion1MassFlow Rate	-0.043	0.000	0.004	0.00e+00	0.10	0.18
EngExhstGsRecirculation1ValvePos	0.232	0.000	0.021	0.00e+00	0.05	0.15
EngFuelRate	0.015	-0.000	0.003	0.00e+00	0.01	0.34
EngInstantaneousFuelEconomy	0.121	-0.000	0.020	0.00e+00	0.11	0.56
EngIntakeAirMassFlowRate	0.206	-0.000	0.010	0.00e+00	0.09	0.32
EngPercentLoadAtCurrentSpeed	0.243	0.000	0.004	0.00e+00	0.08	0.18
EngSpeed	0.045	0.000	0.007	0.00e+00	0.06	0.09
EngTurbo1CompressorIntakeTemp	-0.100	-0.000	0.003	0.00e+00	0.01	0.01
EngTurbo1Speed	0.182	0.000	0.012	0.00e+00	0.07	0.15
TransInputShaftSpeed	0.112	0.000	0.024	0.00e+00	0.07	0.13
TransOilTemp	0.687	0.000	0.181	0.00e+00	0.01	0.02
TransOutputShaftSpeed	0.055	0.000	0.022	0.00e+00	0.12	0.46
UnifiedVehiclespeed	-0.043	-0.000	0.002	2.52e-181	0.14	0.21
UnifiedWheelspeedsrelative	0.225	0.000	0.030	0.00e+00	0.03	0.20
UnifiedEnginepressure	0.126	-0.000	0.005	0.00e+00	0.08	0.20
UnifiedIntakepressure	0.354	-0.000	0.055	0.00e+00	0.07	0.24
UnifiedFuelpressure	0.429	0.000	0.087	0.00e+00	0.12	0.27
UnifiedTorque	0.199	-0.000	0.000	1.38e-08	0.09	0.27

A.1.6 Frequency Domain Analysis

Signal Name	**Dominant Freq**	**Frequency Entropy**	**Low Freq Power**	**High Freq Power**	**Spectral Centroid**
AccelPedalPos1	0.039	6.89	0.506	0.494	2.41
ActlMaxAvailableEngPercentTorque	0.039	6.94	0.482	0.518	2.52
Aftertreatment1ExhaustGasTemp1	4.688	6.95	0.391	0.609	2.88
Aftertreatment1IntakeNOx	0.039	6.77	0.496	0.504	2.40
Aftertreatment1OutletNOx	4.883	6.99	0.458	0.542	2.66
AftrtrtmntlDslOxdtnCtlystIntkGsT	4.922	6.96	0.405	0.595	2.84
AftrtrtmntlDslPrtcltFltrDffPress	0.039	6.88	0.507	0.493	2.41
AftrtrtmntlDslPrtcltFltrIntkGsTm	4.688	6.95	0.392	0.608	2.88
AftrtrtmntlDslPrtcltFltrOtltGsTm	4.688	6.97	0.414	0.586	2.81
BatteryPotential_PowerInput1	4.648	6.98	0.460	0.540	2.68
DslParticulateFilterOutletPress1	0.039	6.78	0.517	0.483	2.33
EngAverageFuelEconomy	4.961	6.94	0.374	0.626	2.93
EngExhstGsRcrcltion1MassFlowRate	4.688	6.95	0.388	0.612	2.89
EngExhstGsRecirculation1ValvePos	0.039	6.93	0.489	0.511	2.50
EngFuelRate	4.102	6.99	0.483	0.517	2.60
EngInstantaneousFuelEconomy	0.039	7.00	0.481	0.519	2.58
EngIntakeAirMassFlowRate	0.039	6.98	0.499	0.501	2.49
EngPercentLoadAtCurrentSpeed	0.039	6.89	0.518	0.482	2.37
EngSpeed	0.039	6.95	0.405	0.595	2.79
EngTurbo1CompressorIntakeTemp	4.688	6.95	0.390	0.610	2.89
EngTurbo1Speed	0.039	6.96	0.470	0.530	2.58
TransInputShaftSpeed	0.039	6.93	0.396	0.604	2.82
TransOilTemp	4.961	6.95	0.385	0.615	2.90
TransOutputShaftSpeed	4.961	6.93	0.365	0.635	2.96
UnifiedVehiclespeed	4.688	6.93	0.363	0.637	2.96
UnifiedWheelspeedsrelative	0.039	6.99	0.505	0.495	2.47
UnifiedEnginepressure	0.039	6.94	0.477	0.523	2.53
UnifiedIntakepressure	0.039	6.93	0.499	0.501	2.46
UnifiedFuelpressure	0.039	6.97	0.478	0.522	2.55
UnifiedTorque	0.039	6.93	0.500	0.500	2.45

A.1.7 Distribution Characteristics

Signal Name	**Skewness**	**Kurtosis**	**95th Percentile Change**	**Significant Changes**	**Instability Periods**
AccelPedalPos1	0.747	-0.571	0.79	199477	1360
ActlMaxAvailableEngPercentTorque	-0.046	-1.189	0.47	229342	76
Aftertreatment1ExhaustGasTemp1	-0.464	-0.006	0.10	249094	0
Aftertreatment1IntakeNOx	1.173	0.789	0.28	167886	9087
Aftertreatment1OutletNOx	3.256	12.417	0.31	101253	62642
AftrtrtmntlDslOxdtnCtlystIntkGsT	-0.154	-0.736	0.11	289799	0
AftrtrtmntlDslPrtcltFltrDffPress	0.787	0.282	0.15	243533	7164
AftrtrtmntlDslPrtcltFltrIntkGsTm	-0.594	0.359	0.09	246994	1
AftrtrtmntlDslPrtcltFltrOtltGsTm	-0.714	0.051	0.11	258243	0
BatteryPotential_PowerInput1	5.445	29.978	0.50	43600	137636
DslParticulateFilterOutletPress1	1.031	1.538	0.24	240017	8311
EngAverageFuelEconomy	1.445	17.334	0.03	248601	2078
EngExhstGsRcrcltion1MassFlowRate	-0.097	-1.847	0.25	272885	0
EngExhstGsRecirculation1ValvePos	-0.920	0.426	0.18	168149	20516
EngFuelRate	26.657	752.617	0.02	1704	16664
EngInstantaneousFuelEconomy	4.087	16.408	0.88	54860	137862
EngIntakeAirMassFlowRate	1.709	3.173	0.39	135096	66179
EngPercentLoadAtCurrentSpeed	0.445	-0.547	0.29	195472	2548
EngSpeed	0.138	-1.346	0.16	244099	0
EngTurbo1CompressorIntakeTemp	0.070	-1.535	0.02	291417	0
EngTurbo1Speed	-0.130	-0.868	0.23	244332	174
TransInputShaftSpeed	0.555	-1.327	0.24	208449	0
TransOilTemp	0.054	-1.307	0.03	255217	66
TransOutputShaftSpeed	1.063	-0.151	0.35	205896	2246
UnifiedVehiclespeed	0.125	-1.668	0.40	238673	0
UnifiedWheelspeedsrelative	-0.782	14.253	0.09	174341	17475
UnifiedEnginepressure	-0.380	-0.538	0.24	237442	776
UnifiedIntakepressure	1.014	0.687	0.24	200134	22495
UnifiedFuelpressure	-0.345	-0.652	0.39	222751	1958
UnifiedTorque	0.607	-0.152	0.31	202561	2281

A.2.1 Dynamic Behavior Ranking.

*Based on mean change rate

AccelPedalPos1	0.24	Very Stable
UnifiedVehiclespeed	0.17	Very Stable
ActlMaxAvailableEngPercentTorque	0.16	Very Stable
TransOutputShaftSpeed	0.14	Very Stable
UnifiedFuelpressure	0.13	Very Stable
EngExhstGsRcrcltion1MassFlowRate	0.12	Very Stable
EngInstantaneousFuelEconomy	0.10	Very Stable
EngIntakeAirMassFlowRate	0.10	Very Stable
UnifiedTorque	0.09	Very Stable
UnifiedEnginepressure	0.09	Very Stable
TransInputShaftSpeed	0.08	Very Stable
EngPercentLoadAtCurrentSpeed	0.08	Very Stable
DslParticulateFilterOutletPress1	0.08	Very Stable
Aftertreatment1IntakeNOx	0.08	Very Stable
EngTurbo1Speed	0.08	Very Stable
UnifiedIntakepressure	0.08	Very Stable
EngSpeed	0.07	Very Stable
Aftertreatment1OutletNOx	0.06	Very Stable
EngExhstGsRecirculation1ValvePos	0.06	Very Stable
AftrtrtmntlDslPrtcltFltrDffPress	0.05	Very Stable
BatteryPotential_PowerInput1	0.05	Very Stable
AftrtrtmntlDslOxdtnCtlystIntkGsT	0.05	Very Stable
AftrtrtmntlDslPrtcltFltrOtltGsTm	0.04	Very Stable
Aftertreatment1ExhaustGasTemp1	0.04	Very Stable
AftrtrtmntlDslPrtcltFltrIntkGsTm	0.04	Very Stable
UnifiedWheelspeedsrelative	0.03	Very Stable
EngAverageFuelEconomy	0.01	Very Stable
TransOilTemp	0.01	Very Stable
EngTurbo1CompressorIntakeTemp	0.01	Very Stable
EngFuelRate	0.01	Very Stable