Intelligent control system towards autonomous drilling in a subsea drilling platform

Karel van de Vrie
MSc Robotics, Faculty Mechanical Engineering
Delft, The Netherlands
Company Supervisor: Peter Looijen
University Supervisor: Luka Peternel

***Abstract*— Offshore geotechnical exploration presents unique challenges due to harsh environments, limited sensor data, and the need for high-fidelity soil characterization. This has led to the deployment of automated solutions, such as subsea platforms, that support exploration activities. These robotic systems are often coupled with intelligent systems capable of performing tasks like drilling autonomously. This work focuses on developing an intelligent advisory system for the Blue Dragon subsea platform to enable autonomous drilling operations. To address the challenges presented by this unique application, the research explores multi-task learning to enhance timeseries forecasting models that predict optimal drilling parameters including thrust, torque, and rotation speed.**

Four model architectures were evaluated: a baseline forecasting model, a forecasting model enriched with a separately trained lithology classifier, a jointly trained classifier-forecaster system, and an unsupervised embedding extractor approach. These models were tested on both synthetic vehicle dynamics data and real-world drilling data from Blue Dragon® operations. While the synthetic data demonstrated theoretical soundness of the approaches, evaluation on the drilling dataset revealed significant limitations due to insufficient data quality and quantity, resulting in severe overfitting that prevented conclusive model validation.

The results indicate that although lithology-informed forecasting shows theoretical promise, current data constraints prevent robust generalization to operational conditions. Future work should prioritize acquiring production-grade datasets and developing refined data preprocessing techniques to better isolate drilling-relevant signals. This research establishes foundational methodology for safe, efficient, and adaptive automation in offshore drilling, contributing to the broader objective of autonomous geotechnical exploration.

***Index Terms*—Autonomous drilling, Subsea platform, Intelligent control systems, Multi-task learning (MTL), Time series forecasting, Lithology classification, Offshore geotechnical exploration, Blue Dragon®, Soil characterization, Lithology Classification, Multi-Task Learning**

I. Introduction

Geotechnical exploration involves investigating subsurface conditions to understand the properties of soil and rock at a site. This process typically includes drilling boreholes, collecting soil samples, and conducting various tests to assess factors like soil composition, strength, and stability. The importance of geotechnical exploration lies in its ability to inform the design and construction of structures, ensuring they are safe and stable. It helps identify potential issues such as soil erosion, landslides, or groundwater problems, allowing engineers to mitigate risks and optimize construction methods.

Offshore geotechnical exploration is a critical process to understand the composition and characteristics of the seabed. This information is essential for various applications, including the construction of offshore structures such as oil rigs, wind farms, and underwater pipelines and cables. Traditional geotechnical exploration methods often involve multiple dives, manual operations, and the use of different equipment for testing, sampling, and coring. These methods can be time-consuming, labor-intensive, and prone to human error. Additionally, the harsh and unpredictable marine environment poses significant risks to personnel and equipment. To mitigate some of these issues, Fugro had developed the Blue Dragon®, a simplified and technical drawing of the system is visualized in Figure 1.

> "The Fugro Blue Dragon® revolutionizes offshore geotechnical exploration. This modular, fully-automated seafloor drill performs in situ testing, soil sampling, and rock coring in a single dive. Operating across various water depths and terrains, the Blue Dragon® minimizes risk, increases safety, contributes to sustainability, and ensures consistent data quality." [1]

In this thesis, we explore the automation of the drilling process for the Fugro Blue Dragon® system. This can bring the following benefits:

1) **Enhanced Safety**: Automation reduces the need for human operators to be directly involved in drilling decision process. This minimizes the risk of accidents and failures, ensuring a safer working environment.
2) **Increased Efficiency**: The Fugro Blue Dragon® can perform multiple tasks, such as in situ testing, soil sampling, and rock coring, in a single dive. Automation reduces the time spent on manual interactions and optimizes decision-making routines.
3) **Consistent Data Quality**: Automation ensures that the data collected is consistent and reliable. By minimizing human error and variability, we achieve more accurate and repeatable results, which are crucial for informed decision making in the planning and construction of offshore projects.
4) **Cost Savings**: Reducing the need for extensive manual labor lowers the overall cost of geotechnical exploration,

[1] Quote from Join us at OTC 2024 — Fugro.

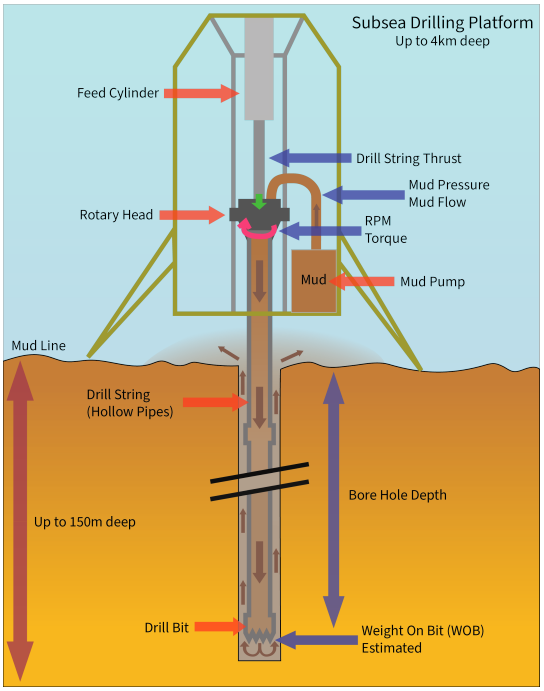


Fig. 1. A high-level overview of the drilling system with important components (in red) and measurement data sources (in blue). All parts, that are not directly connected to the drill systems are not shown, including the robotic manipulator which loads new pipes for the drill string. The red arrow shows the different hardware components. The blue arrow shows the sensor data sources, either directly measured, calculated, or estimated (uncertainty in the non-linearity of the formulas). The only measurement not show is the Rate of Penetration.

making the process more economically viable, especially for large-scale projects.

5) **Adaptability**: The modular design of the Blue Dragon® allows it to operate across various water depths and terrains. This versatility makes it suitable for a wide range of offshore projects, from shallow coastal areas to deep-sea environments. Automation simplifies the transition between different drilling contexts, making it easier for human operators to manage the system.

Drilling is inherently complex, often fraught with unexpected situations. The system's long and slender nature makes it prone to vibrations and extended transient periods. Adding to these challenges is the uneven distribution of measurements along the drilling system. Most real-time data is available only at the top of the system and drill string, with sparse information from downhole, making it difficult to apply standard control methods effectively. To address this, an intelligent control system must model the complete drilling process. This can be achieved through detailed models of both the drill and the process or by employing data-based algorithms such as neural networks.

The road to full automation takes time, trust, and proven reliability. For this we can look at the different autonomy levels as visualized in Figure 2. While the goal is to ultimately reach full autonomy, it is important to start from the very beginning and build this in steps. The first step, which is the focus on this work, is to focus on "Assistance". By creating and training an algorithm which can give recommended drilling parameters to the human operators (also called the "*drillers*"), both the higher level management as well as the operators directly working with the system will be able to build trust in the autonomous operations before it gets to the next level of "Partial Autonomy".

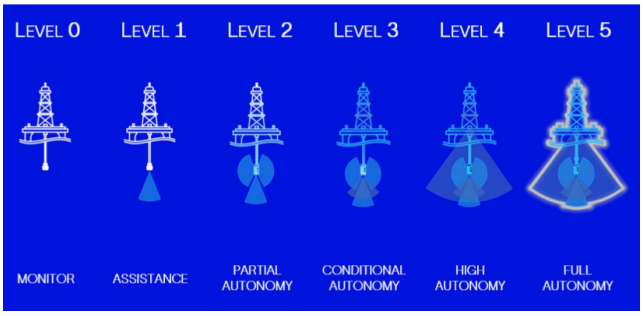


Fig. 2. The different levels of autonomy as specified by [1]. With at level 0 no autonomy and at level 5 full autonomy. In the context of drilling a full autonomous platform not just drills autonomously but also decides where and how to drill to the objective. In this work we try to achieve level 1 autonomy.

Taking inspiration from recent advances in multi-task learning (MTL), combining different models together to strengthen one or each other as described by S. Ruder [2], which have significantly enhanced time series modeling by enabling the joint optimization of related tasks such as forecasting, classification, and anomaly detection. The UniTS framework (Gao et al., 2024 [3]) exemplifies this trend by unifying predictive and generative time series tasks within a single transformer-based architecture, achieving superior performance across 38 datasets spanning healthcare, finance, and engineering. Similarly, Han et al. [4] propose a graph neural network-based MTL model that captures both global and local dynamic dependencies through cross-timestep feature sharing, improving generalization and accuracy in healthcare forecasting scenarios. More recently, Sun et al. [5] developed BiCA-LI, a dual-encoder model combining LSTM and Informer networks with bidirectional cross-attention and uncertainty-aware loss balancing, demonstrating robust performance in both regression and classification tasks for data center monitoring. The High-TS framework (Zhang et al., 2024 [6]), demonstrate the effectiveness of integrating multiple representations to enhance predictive performance. High-TS combines multiscale temporal embeddings with topological spatial embeddings and aligns them using contrastive learning, enabling richer and more generalizable representations. These approaches collectively support the integration of classification outputs into forecasting pipelines, as they highlight the benefits of shared representations and task-aware learning strategies in capturing complex temporal dependencies. This principle is reflected in applied systems such as those by Daireaux et al. [7], who developed an autonomous drilling system that continuously updates control parameters based on real-time wellbore conditions. Their system uses a set-point generation algorithm to optimize drilling performance while maintaining safety constraints. Cayeux et al. [8] extend this concept with a multi-layered protection system for autonomous drilling,

incorporating fault detection, mitigation, and safe transitions between manual and autonomous modes. Wu et al. [9] demonstrates the utility of classification-informed control by using lithology identification via a neural network to condition feed force parameters in a regression model, enabling adaptive and informed drilling operations. Xiufeng et al. [10] further demonstrates that intelligent systems can significantly improve drilling performance across varying geological conditions by classifying rock strength into three levels and applying that to artificial neural networks (ANNs) and an artificial bee colony (ABC) algorithm.

The current state of the art is focused on different domains, where the closest relative domain is the gas and oil industry. However in those, the platforms drill kilometers at a time in a fully destructive manner, whereas geological exploration drills at maximum a few hundred meters (most of the time tens of meters) at a time and a lot of holes in the same area. Geological exploration also requires the soil to stay as undisturbed as possible to measure the actual characteristics of the lithology. The state of the art uses relatively simple lithology classifiers (soft, medium, hard) compared to the level of detail in the lithology data Fugro has access to. Additionally, no research has directly and conclusively shown the use of a classifier or unsupervised embedding extractor within drilling (both for oil and gas, as well as geological exploration). To explore this, the current research focuses on those points and aims to answer the question, how to automate the drilling process on a subsea drilling platform. More concretely, the focus is to determine:

- Can an algorithm be trained to forecast similar drilling outputs as the current drillers do?
- Does a lithology classifier improve the forecasting algorithm when used as a data enrichment module? As researched by Wu et al. [9] and [10].

To demonstrate the effectiveness of the approaches developed, two distinct datasets were employed: a synthetic dataset representing a general 3rd order system representing vehicle dynamics, and the real world drilling data from the Blue Dragon.

This work is structured in six chapters. It begins with introduction, outlining context, background, current state of the art, and the reason for this work in Chapter I. Chapter II outlines the background theory required to understand the field. Chapter III outlines the developed methodology and corresponding approaches to be run. Chapter IV outlines the approaches that were run and the results of this. Chapter V discusses the results and the issues encountered along the way. And Chapter VI concludes the research and answers the research questions.

II. Background Theory

To be able to build a comprehensive understanding of the results of this work, first it is important to introduce the relevant background theory. This section is split in two parts. The first looks at the terminology related to offshore geotechnical exploration, focusing on the different processes. The second introduces the relevant technical terminology used throughout the rest of the work.

A. Offshore Geotechnical Exploration

Offshore geotechnical exploration involves investigating the seabed to understand its physical properties and composition. This information is crucial for the design and construction of offshore structures such as oil rigs, wind farms, and underwater pipelines. There are three important key components:

- **In Situ Testing**: This involves testing the soil and rock directly at the seabed without removing samples. Techniques like cone penetration testing (CPT) measure the resistance of the soil to penetration, providing valuable data on soil strength and composition.
- **Soil Sampling**: This process involves collecting samples of the seabed material for laboratory analysis. Different types of samplers, such as piston samplers or box corers, are used to retrieve undisturbed samples. These samples help determine the soil's physical and chemical properties.
- **Rock Coring**: This technique involves drilling into the seabed to extract cylindrical cores of rock. These cores are analyzed to understand the geological structure and properties of the seabed. Rock coring is essential for projects that require a detailed understanding of the subsurface conditions.

Together, these methods provide comprehensive data that is essential for safe and efficient offshore construction projects.

B. Technical Context

The drilling process is defined by multiple components, all of which are relevant in its definition. One should look at the type of control that can be used, the importance of mud, and the different parameters and measurements required.

1) Low level drill control: In the context of drilling, position and force control are crucial for effectively handling different types of materials:

- **Force Control**: This is used when drilling through hard materials like rock. The drill applies a controlled amount of force to penetrate the rock without causing damage to the equipment or the surrounding structure. Force control ensures that the drill can maintain steady pressure, allowing it to break through tough materials efficiently.
- **Position Control**: This is used when drilling through softer materials like dirt or mud. In these cases, precise positioning of the drill is more important than applying force. Position control ensures that the drill moves accurately to the desired location, preventing it from veering off course or causing unnecessary disturbance to the soft material.

By using force control for hard rock and position control for soft dirt or mud, the drilling process can be optimized for different conditions, ensuring an efficient and safe process.

2) Importance of Mud: A factor that is key in the drilling process is mud, as it can influence the performance of the system in multitude of ways. It can:

- make sure no debris can enter the drill string.
- carry away the debris from the drilling process.

- create a separating layer between the drill string and the borehole walls. This prevents a collapse which would cause the pipes to get stuck both rotationally and axially.
- provide cooling and lubrication to the drill bit.

3) Drilling parameters and measurements: The drilling process is influenced and controlled by a set of parameters and measurements. A simplified schematic of the system on the Blue Dragon® is shown in Figure 1, outlining the different components and measurements that are part of it. Those measurements can also be used as set point parameters:

- **Thrust**: Axial force applied to the drill bit. Provides both downward pressure during drilling or measuring with tools, as well as upward pressure when retracting the drill string or tools.
- **Mud Flow and Pressure**: Circulation of mud fluid and the force it exerts within the borehole, the importance is explained in II-B2.
- **Revolutions Per Minute (RPM)**: Number of rotations the drill bit makes per minute.
- **Torque**: Rotational force applied to the drill string to turn the drill bit.
- **Borehole Depth**: Vertical distance from the seabed (also called the mud line) to the bottom of the drilled hole.
- **Weight on Bit (WOB)**: Downward force exerted on the drill bit. This is calculated based on the feed cylinder thrust and the estimated weight of the drill string.
- **Rate of Penetration (ROP)**: The speed with which the drill bit moves into the drilling medium. Is a direct result of the speed with which the feed cylinder pushes the rotary head.

III. Methods

To implement an advisory/recommender system, which outputs different valve positions for outputs such as the RMP, Thrust, and Torque based on the measured drilling parameters, a forecasting model has to be used. This work uses a Many-to-One forecasting model which uses temporal information combined with long term data to predict the output values of the next time step.

The model/algorithm which provides the advisory values will be based on a timeseries forecasting model. The forecasting model has to be multivariate (able to process multiple variables for the same timestep) and able to process multiple series (due to different drilling sessions by different drillers in different physical locations). To answer the question about the use of a lithology classifier as a data enrichment module, an ablation study was run to compare effect of the classifier when used in different ways when combined with a forecasting model. In this chapter we first outline the approach setup, the data format, the synthetic data generated to test the approaches and the code base, and how the system could be implemented in the current setup of the Blue Dragon®.

A. Approaches

The following ablation study is build upon the theory of multi-task learning theory, however to test if multi-task learning provides a significant improvement in either training stability, training time, generalization to unseen data, and overall accuracy within the current domain has to be tested with the following approaches (unless otherwise specified the models are trained in a supervised manner):

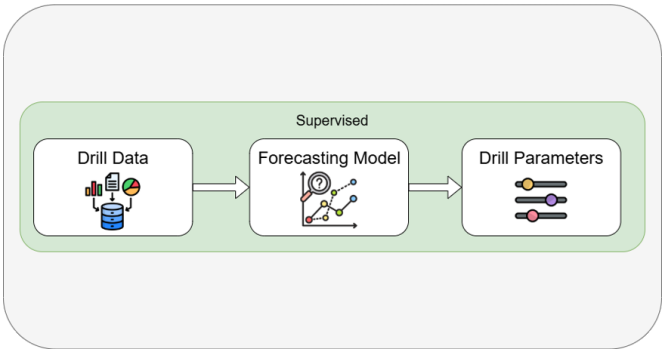
a) Approach 1: Baseline: This approach employs the timeseries model in isolation without augmentations or additional networks, as illustrated in Figure 3a. If this baseline model fails to converge or generalize to unseen data, it indicates fundamental limitations that will likely affect all subsequent approaches. This configuration establishes the performance baseline against which all other approaches are compared, particularly to quantify the classifier's impact on training dynamics and forecasting capabilities. Due to the absence of the classifier component and reduced input complexity, this approach is expected to demonstrate faster training and inference times while potentially underperforming relative to data-enriched approaches (both guided and unguided variants).

b) Approach 2: Separately Trained: This approach employs a two-stage training procedure where a classification network is first trained independently to predict lithology, with its output subsequently fed to the timeseries model, as depicted in Figure 3b. The sequential training strategy ensures that the classifier is fully optimized before providing input to the forecasting network, guaranteeing high-quality enriched data. While this serial training approach results in suboptimal computational resource utilization compared to joint optimization, it may achieve superior performance relative to the simultaneous training employed in Approach 3. This method is anticipated to outperform the baseline due to data enrichment; however, the critical questions remain regarding the magnitude of improvement and whether the performance gains justify the additional implementation complexity, maintenance overhead, and extended training time requirements.

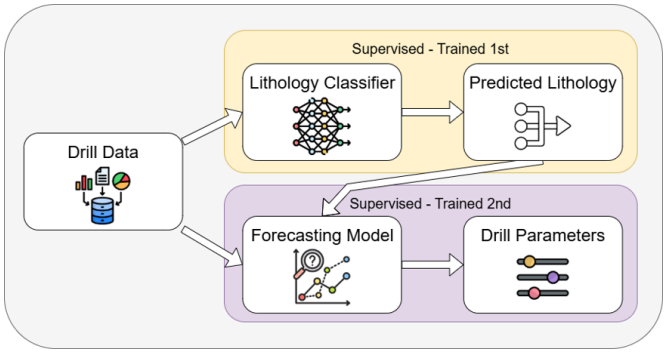
c) Approach 3: Jointly Trained: This approach trains both networks simultaneously, as illustrated in Figure 3c. The losses and gradients are coupled to enable bidirectional optimization: the forecasting model influences classifier training to focus attention on data regions most relevant for prediction, while the classifier acts as a regularizer for the forecasting model. Due to the classifier loss being an order of magnitude larger than the timeseries loss, the classifier loss is weighted at 0.05 to maintain training stability.

Joint training enables optimal computational resource utilization compared to the sequential approach. However, the mutual influence between randomly initialized networks is expected to produce less gradual training dynamics with increased volatility. While the classifier remains supervised, the extent to which the forecasting model can effectively guide classifier optimization remains uncertain, raising questions about the overall training efficiency of this coupled approach.

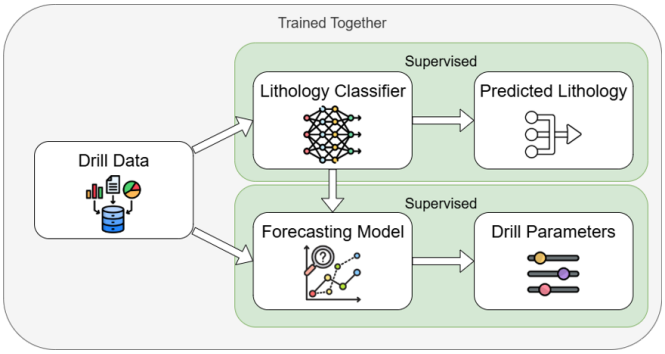
d) Approach 4: Embedding Model: The classifier functions as an unsupervised embedding extractor for the timeseries model by removing supervised outputs and coupling the loss function such that only the timeseries loss drives gradient optimization, as depicted in Figure 3d. This configuration allows the forecasting model to directly control the classifier's optimization process without competing against supervised classification objectives.



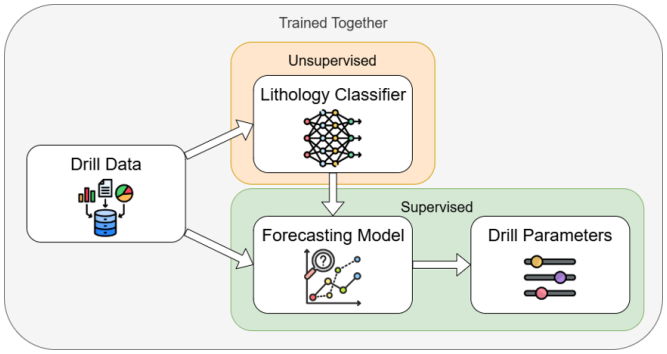
(a) Baseline approach with only a forecasting model.



(b) Lithology classifier is trained first, the output is used as an extra input for the forecasting model, both in a supervised manner.



(c) Trained at the same time in a supervised manner.



(d) Trained at the same time in a semi supervised manner, the forecasting model is supervised while the lithology classifier is unsupervised.

Fig. 3. The four approaches evaluated, visualizing their data and training flows.

This approach is expected to outperform Approach 3 since the forecasting model gains complete influence over the classifier's training rather than sharing control with supervised targets. The classifier can more effectively serve as both a regularizer and feature extractor, generating intermediate representations specifically optimized for forecasting performance. Given the relatively simple architecture of the forecasting model, the addition of an embedding extractor is anticipated to provide significant training improvements. However, this benefit may diminish with more complex forecasting models that possess greater representational capacity, potentially rendering the embedding extractor redundant and failing to exceed baseline performance.

B. Data Format

The data comprises a multivariate and multi-series timeseries structure, requiring the model to ingest and predict multiple variables simultaneously across different data sequences, such as distinct drilling sessions in our application, as illustrated in Figure 4. Critically, the forecasting task involves predicting output values that differ fundamentally from the input variables—unlike conventional stock price forecasting where future values of the same input series are predicted. Instead, the objective is to predict control values for various valves based on sensor input data, representing a cross-domain prediction challenge that requires the model to learn complex mappings between sensor measurements and control actions.

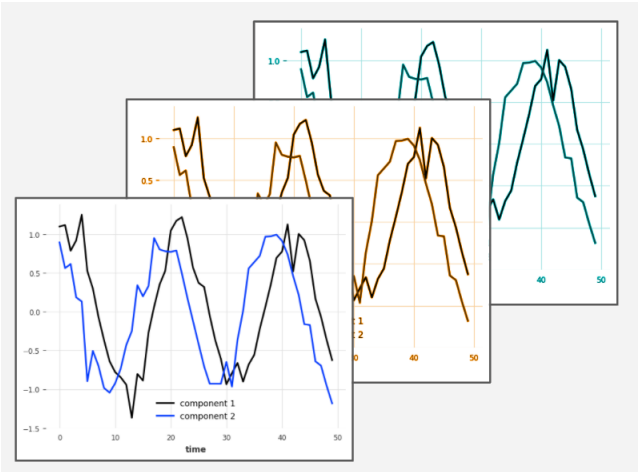


Fig. 4. An example of a multivariate and multiple timeseries data. From DARTS.

C. Synthetic data

To validate the general applicability and robustness of the proposed approaches in the section above, a synthetic dataset of a third order system, representing vehicle dynamics, was

generated. Third-order kinematic systems are particularly valuable for testing timeseries approaches because they naturally generate multiple correlated signals with varying temporal characteristics. The jerk-to-velocity integration creates timeseries with different smoothness properties and phase relationships, while the throttle and brake outputs introduce switching dynamics and control logic patterns. This complexity allows thorough evaluation of how well timeseries models can capture both the continuous kinematic relationships and the discrete control decisions, while the embedding extractor can learn meaningful representations from the temporal patterns in the input data.

The synthetic dataset was constructed by integrating jerk to distance of a vehicle departing from 0 and ending at 0 velocity after an x amount of time, this data is not fully representative of the actual process in real life. The dataset incorporates vehicle dynamics representing a complete driving cycle, startup, cruising, and arrival, with corresponding control outputs of throttle position and brake position. This multivariate timeseries structure provides multiple inputs (jerk, acceleration, velocity, speed) and outputs (acceleration, throttle position, brake position) that exhibit complex temporal dependencies and cross-correlations typical of real-world dynamic systems.

In Figure 5 a single instance of the randomly generated data is shown. To create a multi series dataset with distinct data the jerk time J was randomized, where x is the instance of the for loop creating a new dataset instance.

$$J = x * 10 + 10 \tag{1}$$

The total time in dataset instance x is computed by taking a random value q:

$$q \in [2 * (J * 2.5) + 100, 2 * (J * 2.5) + 300] \tag{2}$$

The acceleration a, speed v, and distance p values are directly integrated based on the jerk values. The throttle position $T(a, v)$ is linearly positive when the acceleration is positive, 100% when the $acceleration = 0$ and $speed > 0$, and linearly negative when the acceleration is negative.

$$T(a,v) = \begin{cases} \alpha\, a, & a > 0, \\ 100\%, & a = 0 \ \wedge \ v > 0, \\ \beta\, a, & a < 0, \end{cases}$$

The brake pressure $B(a)$ is 0 until the acceleration a is negative.

$$B(a) = \begin{cases} 0, & a \geq 0, \\ \gamma\, |a|, & a < 0. \end{cases}$$

For the specific input and output data, there are different parameters which define the data points.

- Input data: Time t (s), Jerk j (m/s^3), Acceleration a (m/s^2), Speed v (m/s), Distance p (m).
- Output data: Acceleration a (m/s^2), Throttle Position $T(a, v)$, Brake pressure $B(a)$.

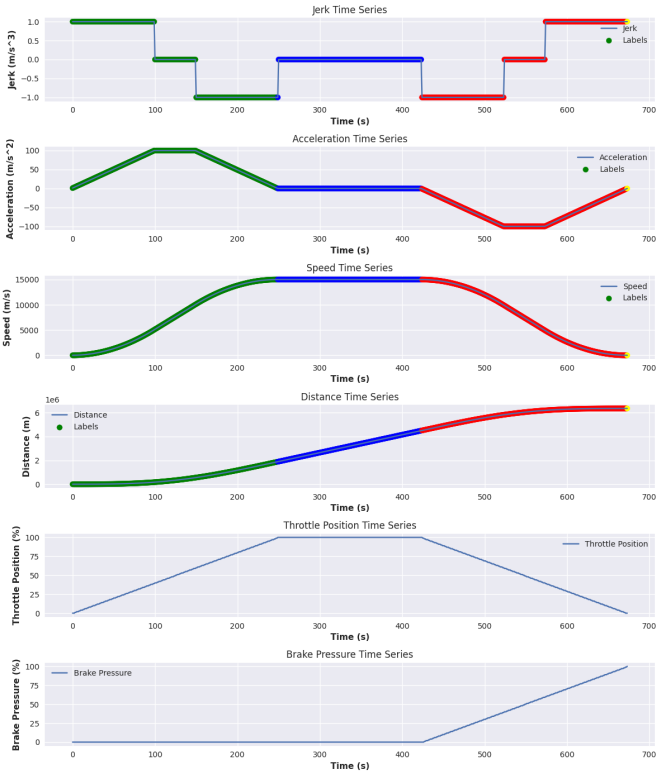


Fig. 5. An example of a single instance of the synthetic data, with in green the label "*Departing*", in blue "*Cruising*", in red "*Arriving*", and in yellow "*Arrived*". The inputs are the Jerk, Acceleration, Speed, and Distance, the outputs are Acceleration, Throttle Position, and Brake Pressure.

There are four classification labels: (1) Departing, (2) Cruising, (3) Arriving, (4) Arrived

D. Real data

Graphs representing the actual data for the 3 boreholes can be found in Appendix C. Appendix C-A shows the input data, both before and after filtering. The input data consists of:

- Time (s), Mud flow (l/min), Mud Pressure (bar), Torque (Nm), RPM (rpm), Thrust (ton), ROP (cm/s), Bit Depth (m), Borehole Depth (m)

To filter the data we remove all datapoints where the following 3 conditions hold:

1) `RPM`$(rpm < 20)$, drilling happens at a much higher rpm, low rpm's are used to connect and disconnect new drill pipe segments, do sensor measurements, and idle the system.
2) `Mudflow`$(l/min) > 0$, without mudflow we cannot drill, however the mud coupling isn't connected when connecting new drill pipe segments, and lowering tools.
3) `Thrust`$(ton) > 0$, when retracting the drill the thrust will be negative, we cannot drill in the negative direction thus we remove this data.

There are four different output values of interest, considered and shown in Appendix C-B:

- `ValveRateOfPenetration`
- `ValveRotationSpeed`

- `ValveTrust`
- `ValveTorque`

The different lithology classification values, shown in Appendix C-C, are plotted compared to the depth of the boreholes. The classification values have been computed using the Robertson 2009 method [11] based on the Cone Penetration Test (CPT) values during the hole drilling. Below are the unique labels present in the data, these labels are as specified by Robertson [12]:

- SAND mixtures - silty sand to sandy silt
- silt mixtures - clayey SILT to silty CLAY
- CLAYS - clay to silty clay
- SANDS - clean sand to silty sand
- gravelly sand to sand

E. System Implementation

To implement the advisory system within the current operating environment of the drillers a small study was conducted, for which experts and end users were consulted. This resulted in the addition of target indicators on the outside of the gauges within the existing dashboards used by the drillers. In Figure 6 the indicators are visualized. The algorithm can be enabled and disabled on demand.

Fig. 6. The dashboard as seen by the drillers during production. To indicate the values predicted by the algorithm indicators have been added above the gauges as shows with the arrow.

IV. Experiments

This section presents the experimental framework for evaluating the different approaches, defines the metrics employed to assess model performance and generalization capability, analyzes and discusses the results obtained from the experimental configuration, and examines the modifications attempted to enhance performance.

A. Design

To enable usage of multivariate and multi-series time series forecasting capabilities, the algorithms have been implemented by integrating DARTS [13] with PyTorch. To use the DARTS features a custom class was setup based on the `GlobalForecastingModel` class of DARTS, this class implemented a custom fit function, and a custom prediction function. This was required to implement the connections for the classifier model and the timeseries model from Approaches 3 and 4. This also required a custom dataloader that processed the input timeseries in the same way as DARTS does, this enabled the models to be multi variate, multi series, and have different inputs from the outputs.

For the direct approach comparison the timeseries model was implemented as a simple Gated Recurrent Unit (GRU) timeseries model, as can be seen in Appendix A-A. The classification model was implemented as a simple Long Term Short Term (LSTM) Classifier, as can be seen in Appendix A-B. Using an LSTM as a classifier model enabled the classifier to learn some temporal information when making lithology predictions, rather than just a single time step of input data as would be the case of a simple linear neural network. As can be seen in Section IV-F other model architectures were tested as well to see if a different network would result in better results.

B. Setup

To ensure a stable gradient flow and prevent features with larger numerical ranges from dominating the training process, we normalize the data to values between 0 and 1. To evaluate model performance during and after training, the datasets are split into training, validation, and test splits. This data splitting is performed prior to experimentation, ensuring that each approach uses identical training and validation data while being evaluated on the same test split for fair comparison. For the synthetic data the data split can be found in Appendix B, for the drilling data the split can be found in Appendix C-D.

The hyperparameters in Table I were used to train the approaches and the results shown in this document are based on those values.

Parameter	Synthetic Dataset	Drilling Dataset
Past Timesteps	50	500
Hidden Size	25	64
Epochs	50	50
Batch Size	128	128
Learning Rate	1×10^{-4}	1×10^{-4}
Optimizer	Adam	Adam
Dropout	0.3	0.3

TABLE I
HYPERPARAMETERS USED TO TRAIN THE MODELS ON THE SYNTHETIC AND DRILLING DATASETS.

C. Metrics

To evaluate the actual performance of the different approaches and thus the actual influence of the classifier/embedding extractor we can use different metrics. Because we fix all things which could influence the training process except for the specific model combinations, as explained in the different approaches in section III-A, we can measure the differences as explained below:

Training Performance Metrics:

- **Training time**: the time required to train an approach for 50 epochs, providing insight into computational efficiency and scalability.

- **Memory consumption**: peak GPU/CPU memory usage during training, indicating resource requirements and feasibility for deployment.
- **Model parameters count**: total number of trainable parameters, reflecting model complexity and potential for overfitting.
- **Convergence rate**: the number of epochs required to reach a stable loss plateau, indicating training efficiency.

Loss Analysis Metrics:

- **Overall loss curves**: used to track the training progress (decreasing training loss), training stability (absence of oscillations), overfitting (training loss decreases while validation loss increases), and underfitting (training loss quickly plateaus and no longer optimizes).
- **Training loss at epoch x**: quantitative assessment of model performance on training data at specific checkpoints.
- **Validation loss at epoch x**: quantitative assessment of model generalization capability at specific checkpoints.
- **Loss variance**: standard deviation of loss values over the final 10 epochs, measuring training stability.
- **Generalization gap**: relative difference between validation and training losses as shown in Equation 3, indicating overfitting severity. Thresholds: $\leq 15\%$ indicates healthy generalization, $15 - 50\%$ suggests moderate overfitting concerns, and $\geq 50\%$ indicates strong overfitting.

$$RGG = \frac{L_{val} - L_{train}}{L_{train}} \cdot 100 \tag{3}$$

Forecasting Accuracy Metrics:

- **Mean Squared Error (MSE) on forecasting**: the squared error between expected and predicted outputs, emphasizing larger errors, as shown in Equation 4.

$$MSE = \frac{1}{D}\sum_{i=1}^{D}(x_i - y_i)^2 \tag{4}$$

- **Mean Absolute Error (MAE) on forecasting**: the absolute error between expected and predicted outputs, providing robust error measurement, as shown in Equation 5.

$$MAE = \frac{1}{D}\sum_{i=1}^{D}|x_i - y_i| \tag{5}$$

- **Coefficient of Determination (R^2)**: proportion of variance explained by the model, indicating goodness of fit, as shown in Equation 6.

$$R^2 = 1 - \frac{\sum_{i=1}^{D}(x_i - y_i)^2}{\sum_{i=1}^{D}(x_i - \bar{x})^2} \tag{6}$$

Output-Specific Metrics:

- **Per-output MSE/MAE**: individual error metrics for the outputs, enabling component-wise performance analysis.
- **Peak error analysis**: maximum absolute error across all time steps, identifying worst-case performance scenarios.

D. Results Synthetic Data

First we look at the progress of the the training process of the models on the synthetic data. In the Figures 7a, 7b, 7c, and 7d the loss graphs for both the training and the validation of the different approaches trained on the synthetic dataset are visualized.

In Table II the loss values at epoch 50 for both the training and validation splits for the models trained on the drilling data are shown.

Approach	Training Loss	Validation Loss	Generalization Gap
1	0.0255	0.0056	-78.03%
2	0.0192	0.0038	-80.20%
3	0.0267	0.0150	-43.72%
4	0.0189	0.0039	-79.36%

TABLE II
FINAL TRAINING AND VALIDATION LOSS VALUES AT EPOCH 50 FOR EACH APPROACH TRAINED ON THE SYNTHETIC DATASET. THE FROZEN CLASSIFIER APPROACH (APPROACH 3) GENERALIZES BEST DESPITE THE UNSUPERVISED MODEL (APPROACH 4) ACHIEVING SLIGHTLY LOWER TRAINING LOSS. THE NEGATIVE GENERALIZATION GAP INDICATES THAT THE VALIDATION SET IS EASIER TO PREDICT THAN THE TRAINING SET, SUGGESTING POTENTIAL DATA DISTRIBUTION ISSUES OR OVERLY SIMPLISTIC SYNTHETIC DATA GENERATION.

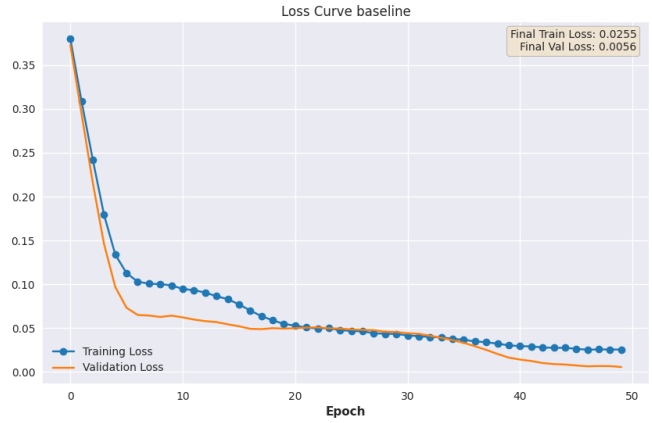
Overall the results of the synthetic data show that the different approaches each optimize towards an optimum and that each generalizes towards unseen data. This confirms that the idea behind the different models and that the multi-task learning can work. The fact that the generalization gap is negative indicates that the validation set is easier to predict than the training set. This can indicate issues with the data split, an unusual data distribution, or leakage from the training set to the validation set. In our case the data is very similar to each other with no noise thus it is logical the gap is negative. Due to the nature of the synthetic data no conclusions can be drawn from the results of the test split and thus will not be shown here, they can be found in Appendix D.

E. Results Drilling Data

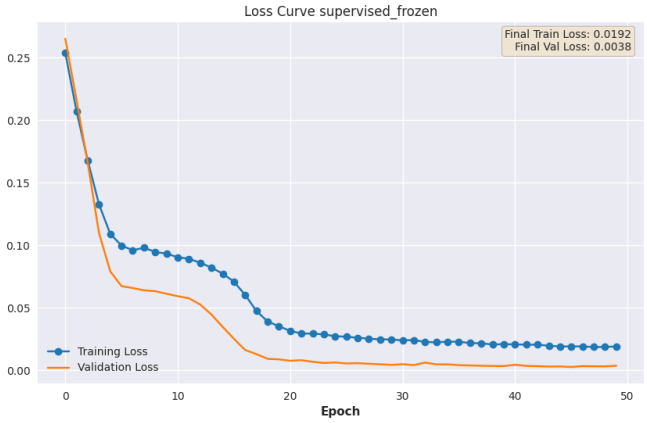
Now we look at the progress of the the training process of the models on the drilling data. In the Figures 8a, 8b, 8c, and 8d the loss graphs for both the training and the validation of the different approaches trained on the drilling dataset are visualized.

In Table III the loss values at epoch 50 for both the training and validation splits for the models trained on the drilling data are shown. As can be seen from the relative generalization gap the approaches overfit massively in all approaches. With approach 4, the unsupervised embedding extractor, performing best but still way outside nominal training parameters with a relative gap exceeding 200%, indicating severe memorization of the training data rather than learning generalizable patterns.

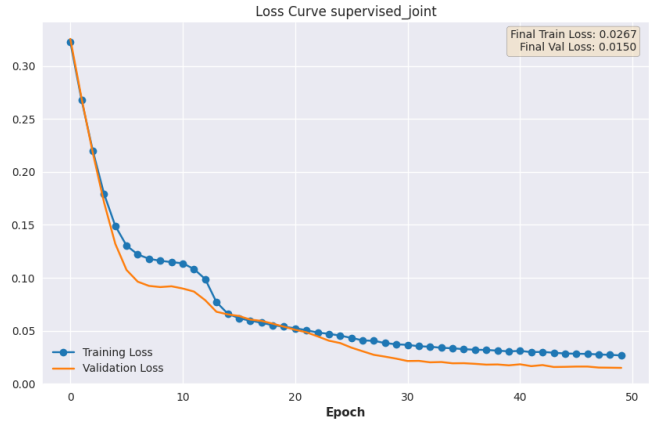
Using the data from the test split we can see how well the model performs on completely unseen data, for this we feed the model the input data up until a certain point and then let it predict the next output values, doing this in a loop over the test split we can forecast the entire timeseries.



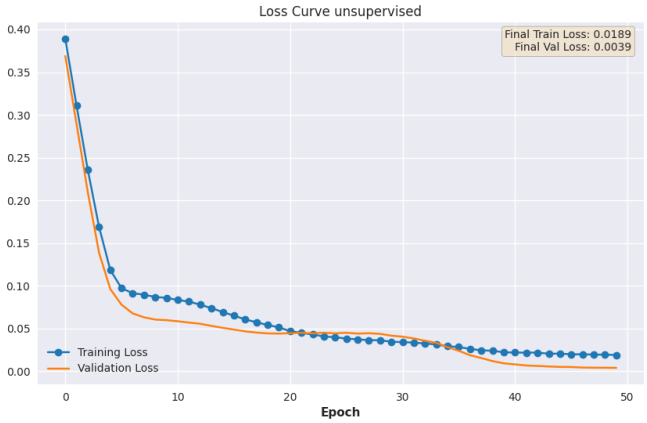
(a) The training and validation loss curves for the baseline approach from Figure 3a trained on the synthetic data. It shows that the model learns to predict on the training set and generalizes well to the validation set.



(b) The training and validation loss curves for the approach with the frozen classification network from Figure 3b trained on the synthetic data. It shows that the model learns to predict on the training set and generalizes well to the validation set but gets stuck in a local minima between epoch 6 and 13 before finding a better minima.



(c) The training and validation loss curves for the jointly trained approach from Figure 3c trained on the synthetic data. It shows that the model learns to predict on the training set and generalizes well to the validation set but gets stuck in a local minima between epoch 6 and 11 before finding a better minima.



(d) The training and validation curves for the approach with the unsupervised classification mode (embedding extractor) from Figure 3d trained on the synthetic data. It shows that the model learns to predict on the training set and generalizes well to the validation set.

Fig. 7. Training and validation loss curves for the different approaches on the synthetic vehicle data.

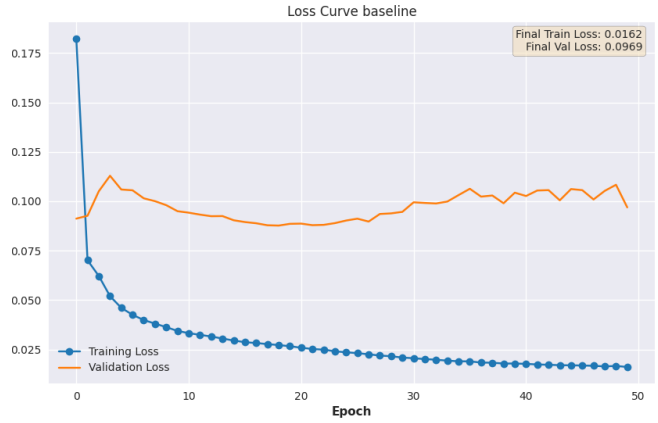
Approach	Training Loss	Validation Loss	Generalization Gap
1	0.0162	0.0969	498.15%
2	0.0196	0.1070	445.92%
3	0.0255	0.1279	401.57%
4	0.0165	0.0618	274.55%

TABLE III
FINAL TRAINING AND VALIDATION LOSS VALUES AT EPOCH 50 FOR EACH APPROACH TRAINED ON THE DRILLING DATASET. THE UNSUPERVISED EMBEDDING APPROACH (APPROACH 4) DEMONSTRATES THE BEST GENERALIZATION PERFORMANCE DESPITE THE BASELINE (APPROACH 1) ACHIEVING SLIGHTLY LOWER TRAINING LOSS. ALL APPROACHES EXHIBIT SEVERE OVERFITTING WITH GENERALIZATION GAPS EXCEEDING 250%, INDICATING FUNDAMENTAL CHALLENGES IN LEARNING GENERALIZABLE PATTERNS FROM THE REAL-WORLD DRILLING DATA.

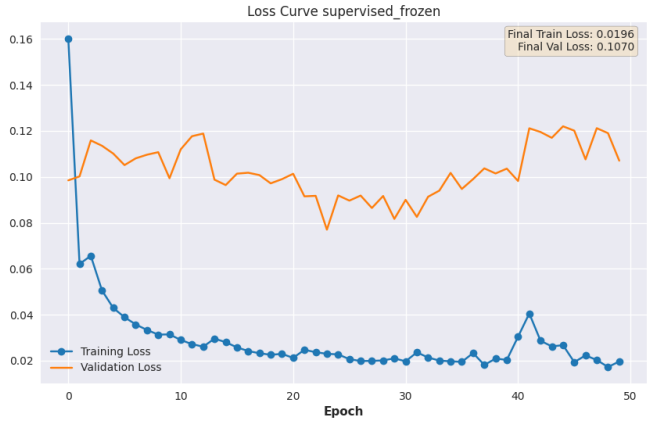
Approach	MSE	MAE
1	0.3883	0.8408
2	0.2487	0.7287
3	0.3117	0.7901
4	0.2643	0.7767

TABLE IV
AVERAGE MSE AND MAE OF FORECASTS ON THE TEST DATASET, CALCULATED BY SUMMING THE MSE AND MAE VALUES ACROSS THE 4 OUTPUTS (DETAILED RESULTS IN APPENDIX E). THE FROZEN CLASSIFIER APPROACH (APPROACH 2) DEMONSTRATES SUPERIOR TEST PERFORMANCE, OUTPERFORMING THE UNSUPERVISED EMBEDDING APPROACH (APPROACH 4) WHICH SHOWED THE BEST GENERALIZATION GAP DURING TRAINING.

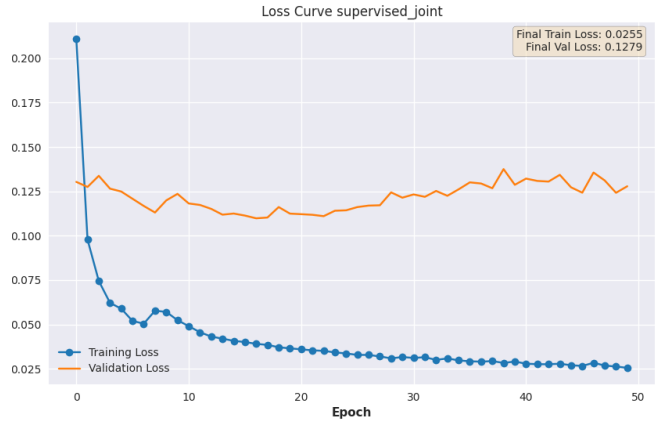
Specifically zooming in on the forecast of Approach 4 we can visualize the change in MSE during the training process, as seen in Figure 9, and we can visualize the expected output and the forecasted output of the model, as seen in Figure 10.



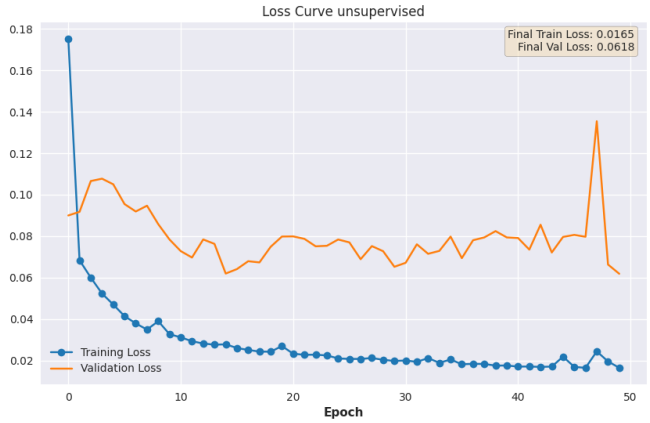
(a) The training and validation loss curves for the baseline approach from Figure 3a trained on the dilling data. It shows that the model learns to predict on the training set but this does not seem to carry over to the validation set, the validation loss is not getting worse so it isn't necessarily overfitting.



(b) The training and validation loss curves for the approach with the frozen classification network from Figure 3b trained on the dilling data. It shows that the model learns to predict on the training set, it has some more peaks than the baseline, indicating jumping gradients. Also here the training this does not seem to carry over to the validation set, the validation loss jumps more around and while for the first 30 epochs it is improving after this it starts overfitting on the training data.



(c) The training and validation loss curves for the jointly trained approach from Figure 3c trained on the dilling data. It shows that the model learns to predict on the training set, it has some more peaks than the baseline, indicating jumping gradients. Also here the training this does not seem to carry over to the validation set, after an initial dip for the validation loss it does look like the network starts overfitting on the training set.



(d) The training and validation curves for the approach with the unsupervised classification mode (embedding extractor) from Figure 3d trained on the dilling data. This model shows the most gains on the validation set of the different approaches, it still shows a disconnect between the different sets and no overfitting.

Fig. 8. Training and validation loss curves for the different approaches on the drilling data.

F. Attempted Fixes

As can be seen the approaches generalize as expected on the synthetic dataset but not on the drilling dataset with the expected data filtering and used models. The following steps have been taken to see if more data filtering (especially outlier filtering), different models, and different hyperparameter combinations would lead to a better model convergence. To see if a different combination of parameters, models, and filtering methods would result in better model convergence a randomized grid search was run.
Below the different networks which were tested with a short description:

- **Gated Recurrent Units (GRU) Timeseries Model**
  GRUs use gating mechanisms to control information flow through time, with reset and update gates that determine what information to forget or retain from previous time steps. They process sequences sequentially and are simpler than LSTMs, making them computationally efficient but with moderate bandwidth for complex temporal patterns.
- **Temporal Convolutional Network (TCN) Timeseries Model**
  TCNs apply dilated convolutions across time with causal padding to capture temporal dependencies without looking into the future. They can process sequences in parallel and use residual connections for training stability, offering high bandwidth for complex datasets due to their ability to capture long-range dependencies efficiently.

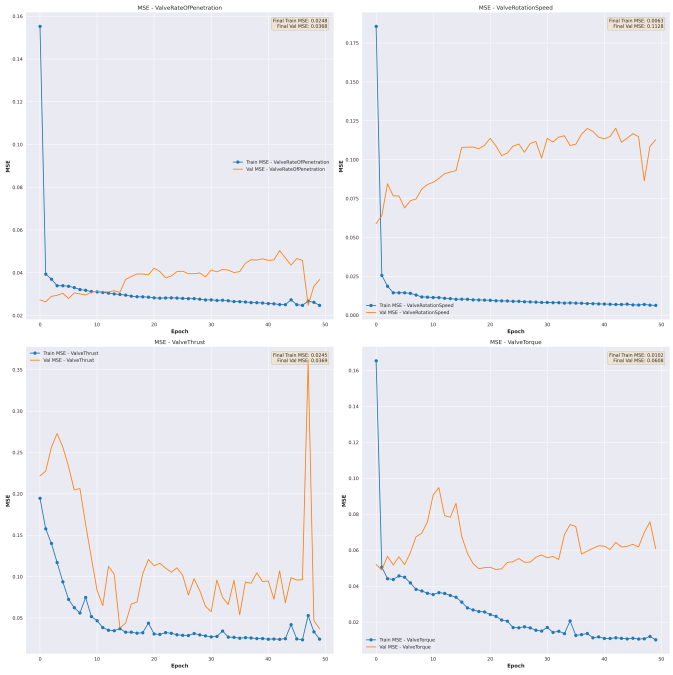


Fig. 9. MSE per output on the test data for Approach 4. For most of the outputs the the network overfits, only the `ValveThrust` output seems to generalize, even though that is the one with the biggest peaks and valleys. The results of the other approaches can be found in Appendix E .

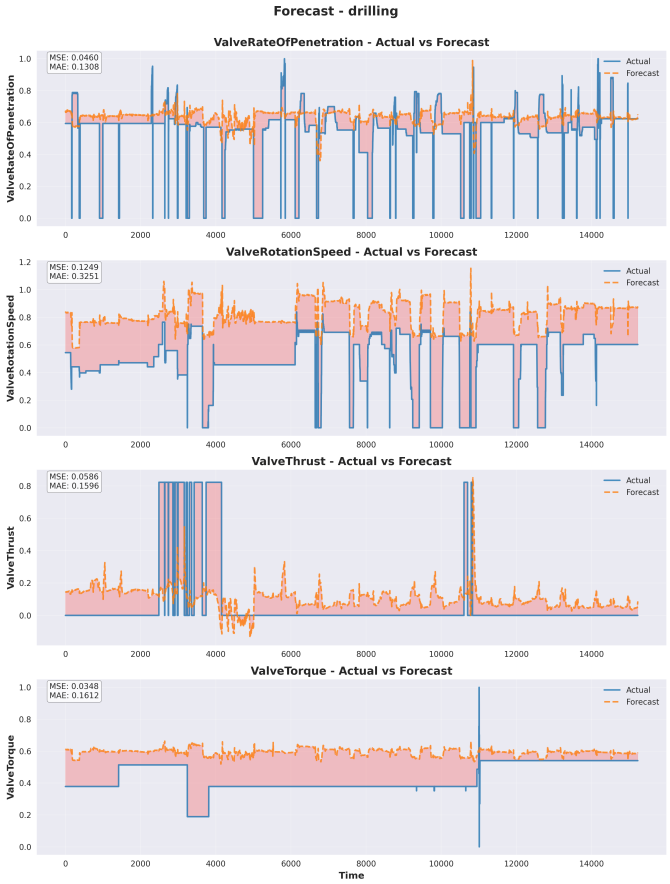


Fig. 10. Forecast on the test data, with calculated MSE and MAE for Approach 4. When looking at the forecast in most cases it seems to have a steady state offset, in some cases it doesn't respond at all as the actual output has, and in others it tries to follow but in none of the outputs does it perform as the actual test data indicates. The results of the other approaches can be found in Appendix E.

- **Long Short Term Memory (LSTM) Timeseries Model**
  LSTMs employ three gates (forget, input, and output) along with a cell state to selectively remember and forget information across long sequences. They are more complex than GRUs with higher computational overhead but provide greater bandwidth for modeling intricate temporal relationships and long-term dependencies.
- **Encoder Decoder (without attention) Timeseries Model**
  This architecture compresses the input sequence into a fixed-size context vector through an encoder, then uses a decoder to generate the output sequence based solely on this compressed representation. It is relatively simple but suffers from information bottleneck issues, limiting its bandwidth for complex, long sequences.
- **Encoder Decoder (with attention) Timeseries Model**
  This extends the basic encoder-decoder by adding attention mechanisms that allow the decoder to focus on different parts of the input sequence at each time step, rather than relying on a single context vector. It is the most complex architecture listed but offers the highest bandwidth for complicated datasets due to its ability to dynamically attend to relevant temporal information.

Below the different hyperparameters which were tuned with a short description, between the square brackets [] the parameters tested can be found:

- **Learning Rate** [1e-5, 5e-5, 1e-4, 5e-4, 1e-3]
  Controls the step size during gradient descent optimization, determining how quickly the model updates its weights, too high causes instability and overshooting, while too low leads to slow convergence or getting stuck in local minima.
- **Number of epochs** [25, 50, 100, 500]
  Defines how many complete passes through the training dataset the model makes, too few epochs result in underfitting, while too many can cause overfitting and waste computational resources.
- **Past Time Steps** [50, 150, 250, 500, 750]
  Specifies the length of the input sequence (lookback window) that the model uses to make predictions, longer sequences can capture more temporal dependencies but increase computational complexity and may introduce noise from irrelevant distant information.
- **Hidden size** [24, 48, 64, 128, 256]
  Determines the dimensionality of the internal representations in hidden layers, controlling the model's capacity to learn complex patterns, larger hidden sizes increase the model's expressiveness but also computational cost and risk of overfitting, especially with limited data.
- **Optimizer** [`Adam`, `Adagrad`, `SGD`, `RMSprop`]
  The algorithm used to update model parameters, each has different convergence properties, momentum handling, and adaptive learning rate capabilities that significantly

impact training stability and final performance.

- **Dropout** [0.0, 0.1, 0.2, 0.3, 0.4]
  Sets the probability of randomly zeroing neurons during training to prevent overfitting by forcing the model to not rely on specific neurons, higher dropout rates provide stronger regularization but may impair the model's ability to learn complex patterns if set too high.

Below the different data filtering methods which were tested with a short description, between the square brackets [] the parameters tested can be found:

- **Interquartile Range (IQR)** [multiplier: 0.7, 1.0, 1.2, 1.5, 2.0, 3.0]
  IQR identifies outliers by calculating the range between the 25th and 75th (Q2 to Q3) percentiles of the data, then flagging points that fall $< Q2 - multiplier \cdot IQR$ or $> Q3 + multiplier \cdot IQR$ as outliers. This method is robust to extreme values and works well for skewed distributions since it relies on quartiles rather than mean and standard deviation.
- **Zscore** [threshold: 2.0, 3.0, 4.0]
  Z-score detects outliers by measuring how many standard deviations each data point is from the mean, typically flagging points with absolute z-scores greater than 2 or 3 as outliers. This method assumes a normal distribution and is sensitive to extreme values since both the mean and standard deviation used in the calculation can be influenced by outliers themselves.
- **Rolling** [rolling window: 5, 10, 25, 50, 125; threshold_percentile: 50, 75, 90, 92, 95, 97, 99]
  Rolling outlier detection applies a moving window approach where outliers are identified based on percentile measure calculated within each window as it slides through the time series. This method is particularly effective for time series data as it can adapt to local trends and seasonal patterns, detecting points that are anomalous relative to their immediate temporal neighborhood rather than the entire dataset.

In the end `Zscore` created the best dataset which was visible in the fact that the training loss ended half as low as the unfiltered training loss, however the training loss became less stable and the validation loss showed only minor improvements.

V. Discussion

The developed solution in combination with the conducted experiments provides a base for drawing preliminary conclusions regarding the posed focus questions.

Based on the background theory and the actual results the following points have been concluded.

A. Overview

The synthetic data experiments demonstrate that the devised methodology, code implementation, and data processing pipeline function as intended. All approaches successfully optimize toward local minima and exhibit appropriate generalization to the validation split, with test set results meeting expectations. However, due to the highly similar and artificially crafted nature of this data, no meaningful conclusions can be drawn regarding the relative performance differences between the four approaches.

In contrast, the drilling data experiments reveal severe overfitting across all approaches, where models memorize training inputs but fail to generalize to unseen data. A more nuanced analysis of individual outputs shows that while some outputs demonstrate reasonable learning and generalization capabilities, others exhibit complete failure to converge.

Even with expanded datasets similar to the current collection (additional test boreholes from the same geographical area), the lithology classifier is not expected to provide significant benefits. This limitation stems from the homogeneous nature of the underlying geological data—predominantly sand and mud formations representing soft soil types that do not substantially alter drilling procedures. Consequently, even if algorithms performed well on current data, they would likely fail to generalize to geographically diverse areas with significantly more rocky formations.

B. Insights

Despite implementing enhanced data filtering techniques, more sophisticated models capable of capturing longer temporal dependencies, and exploring various hyperparameter combinations, all models continued to overfit. In some cases, performance degraded further, with no approach achieving significant improvement. This pattern strongly indicates that the current dataset lacks the quality and quantity necessary for training the employed models, preventing any definitive conclusions regarding the research questions.

This data quality issue is exemplified in Figure 13 (Appendix C-A), where the Rate of Penetration (ROP) remains zero for extended periods even after applying filtering methods to remove outliers. This phenomenon is physically impossible during active drilling operations, clearly indicating fundamental problems with the data collection process.

While established filtering protocols exist for removing non-drilling activities (drill string extension, sensor calibration periods, and other operational interruptions), these measures prove insufficient. The dataset contains excessive outliers beyond typical drilling interruptions such as retracts, blockages, and stuck drill strings. Applying aggressive outlier removal strategies eliminates these anomalies but simultaneously removes substantial amounts of potentially valid data essential for properly modeling the drilling process.

It is important to note that the current Blue Dragon iteration remains in active development and offshore testing phases. Consequently, high-quality production data does not yet exist. This developmental status is reflected in the data quality, as operators continue familiarizing themselves with system responses to both electronic inputs and environmental conditions. Real-time debugging, experimental input-output combinations, and inconsistent drilling practices characterize the current operational environment. The expectation is that once the system matures sufficiently to generate production-quality data, the resulting measurements will better represent stable drilling sequences, leading to significant improvements in algorithm performance.

VI. Conclusion

This work focused on developing an intelligent advisory system for autonomous drilling in offshore scenarios. Four different combinatorial approaches were evaluated, looking at different ways to train an underlying system to provide sound advice to an operator. The approaches were evaluated based on sampled data from real-life drilling scenarios.

Based on the results form the synthetic dataset experiments, the underlying theoretical approaches function correctly. However, due to fundamental deficiencies in the real drilling dataset, no definitive conclusions can be drawn regarding the performance of the individual approaches and their effectiveness in automating deep-sea drilling processes. To reach meaningful conclusions, the drilling dataset requires substantial expansion with production-quality drilling data, rather than test borehole data used for platform evaluation. This expansion should encompass a broader dataset featuring diverse lithological conditions rather than data limited to a single geographical area.

To answer the research questions

- *Can we train an algorithm to forecast similar drilling outputs as the current drillers do?*
  Currently, it is not possible to train an algorithm that accurately forecasts drilling outputs due to the identified dataset limitations.
- *Does a lithology classifier improve the forecasting algorithm when used as a data enrichment module?* This question remains unanswered due to the same dataset constraints that prevent reliable model evaluation.

A. Future Works

The first step for future research would be to acquire a more diverse dataset encompassing greater range and variety in sample characteristics. This expanded dataset would enable a comprehensive re-evaluation of the developed solution, determining whether improved data quality can yield significant performance differences between approaches. With higher-quality data representing multiple drilling sessions and stable time series, substantially improved results are theoretically expected, potentially providing definitive answers to the research questions.

Once sufficient data becomes available, another important step would be to look further into (advanced) data filtering techniques. Given the complexity of drilling operations, numerous measurement scenarios require precise classification and separation. It may be the case, that by creating a clearer discrimination between the different (active) drilling moments, operational pauses, and various other operator actions, the solution would be able to provide better predictions.

Another direction that the research could take is towards looking for alternative approaches. In this context the current work explored lithology classifiers build with particular structure and hyperparameters. It could be worth investing time into looking at different combinations, training approaches, model structures, all of which can influence the performance of the system. One could also consider looking into completely different modeling approaches, such as using reinforcement learning with a simulator to teach a system how to correctly drill, or other hybrid models.

References

[1] K. Khazali-Rosli, M. I. Amiruddin, M. N. Bohro, M. A. Mokhti, M. Zurhan, Q. Mohamed, M. Said, N. Mat Noh, M. Kereshanan, K. A. Rashid, S. Ting, C. Cheng, D. A. Wicaksana, V. K. Wibowo, V. Pincay, N. Panwar, and A. Armani, "Digital innovation – outstanding performance through ai-driven autonomous drilling operations," vol. SPE/IADC Asia Pacific Drilling Technology Conference and Exhibition, p. D021S006R005, 08 2024. [Online]. Available: https://doi.org/10.2118/219603-MS

[2] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1706.05098

[3] S. Gao, T. Koker, O. Queen, T. Hartvigsen, T. Tsiligkaridis, and M. Zitnik, "Units: A unified multi-task time series model," 2024. [Online]. Available: https://arxiv.org/abs/2403.00131

[4] X. Han, Y. Huang, Z. Pan, W. Li, Y. Hu, and G. Lin, "Multi-task time series forecasting based on graph neural networks," *Entropy*, vol. 25, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1099-4300/25/8/1136

[5] Z. Sun, Y. Zhou, Z. Gong, C. Wen, Z. Cai, and X. Zeng, "Bica-li: A cross-attention multi-task deep learning model for time series forecasting and anomaly detection in idc equipment," *Applied Sciences*, vol. 15, no. 13, 2025. [Online]. Available: https://www.mdpi.com/2076-3417/15/13/7168

[6] G. Lin, C. Shen, and A. Lin, "Higher-order cross-structural embedding model for time series analysis," 2024. [Online]. Available: https://arxiv.org/abs/2410.22984

[7] B. Daireaux, A. Ambrus, L. A. Carlsen, R. Mihai, K. Gjerstad, and M. Balov, "Development, testing and validation of an adaptive drilling optimization system," vol. SPE/IADC International Drilling Conference and Exhibition, p. D051S022R003, 03 2021. [Online]. Available: https://doi.org/10.2118/204083-MS

[8] E. Cayeux, B. Daireaux, A. Ambrus, R. Mihai, and L. Carlsen, "Autonomous decision-making while drilling," *Energies*, vol. 14, no. 4, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/4/969

[9] Z. Wu, J. Wang, K. Xi, Y. Guo, C. Yang, and M. Jiang, "Research on control system of small intelligent drilling rig based on lithology identification," *Journal of Physics: Conference Series*, vol. 2181, 2022.

[10] X. Liao, M. Khandelwal, H. Yang, M. Koopialipoor, and R. Bhatawdekar, "Effects of a proper feature selection on prediction and optimization of drilling rate using intelligent techniques," *Engineering with Computers*, vol. 36, 04 2020.

[11] P. Robertson, "Interpretation of cone penetration tests a unified approach," *Canadian Geotechnical Journal*, vol. 46, pp. 1337–1355, 11 2009.

[12] Robertson, "Cone penetration test (cpt)-based soil behaviour type (sbt) classification system - an update," *Canadian Geotechnical Journal*, vol. 53, pp. 1910–1927, 07 2016.

[13] J. Herzen, F. LÃ¤ssig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. KoÅcisz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch, "Darts: User-friendly modern machine learning for time series," *Journal of Machine Learning Research*, vol. 23, no. 124, pp. 1–6, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1177.html

APPENDIX A
APPROACHES

A. Timeseries Model Implementation

```python
import torch.nn as nn

class GRUTimeseriesModel(nn.Module):
    def __init__(self, input_size, hidden_size=64, output_size=1,
                 num_layers=1, dropout=0.2):
        super().__init__()
        self.rnn = nn.GRU(
            input_size,                # Features per timestep (e.g., sensor readings)
            hidden_size,               # Hidden state dimension
            num_layers=num_layers,     # Number of stacked GRU layers
            batch_first=True,          # Input shape: (batch, seq_len, features)
            dropout=dropout if num_layers > 1 else 0
            # GRU dropout only works with multiple layers (>1)
            # Applies dropout between GRU layers, not within a single layer
        )
        # Applied to the final hidden state to prevent overfitting
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # x shape: (batch_size, sequence_length, input_size)
        out, hidden = self.rnn(x)

        # Take only the last output (many-to-one)
        # out[:, -1, :] gets the last timestep for each batch
        last_output = out[:, -1, :]  # Shape: (batch_size, hidden_size)

        # Apply dropout regularization to prevent overfitting
        last_output = self.dropout(last_output)
        # Maps from hidden representation to actual output values
        output = self.fc(last_output)  # Shape: (batch_size, output_size)

        return output
```

B. Classification Model Implementation

```python
import torch
import torch.nn as nn

class LSTMClassifier(nn.Module):
    def __init__(self,
                 input_size, output_dim, hidden_size=64,
                 num_layers=1, dropout=0.3, bidirectional=False):
        super().__init__()
        self.bidirectional = bidirectional
        self.lstm = nn.LSTM(
            input_size=input_size,      # Features per timestep
            hidden_size=hidden_size,    # Hidden state dimension
            num_layers=num_layers,      # Number of stacked LSTM layers
            # Dropout between LSTM layers (only applies if num_layers > 1)
            dropout=dropout if num_layers > 1 else 0.0,
            batch_first=True,           # Input shape: (batch, seq_len, features)
            bidirectional=bidirectional # Process sequence in both directions
        )
        self.dropout = nn.Dropout(dropout)
        # Bidirectional LSTM concatenates forward and backward hidden states
        direction_factor = 2 if bidirectional else 1
        # Maps from hidden state to output classes/dimensions
        self.fc = nn.Linear(hidden_size * direction_factor, output_dim)

    def forward(self, x):
        _, (h_n, _) = self.lstm(x)
        # Extract the final hidden state for classification
```

```
        if self.bidirectional:
            # For bidirectional LSTM, concatenate forward and backward final states
            # h_n[-2]: final forward state, h_n[-1]: final backward state
            h_last = torch.cat((h_n[-2], h_n[-1]), dim=1)
        else:
            # For unidirectional LSTM, use the final hidden state
            h_last = h_n[-1]  # Shape: (batch_size, hidden_size)
        # Apply dropout for regularization and pass through final layer
        return self.fc(self.dropout(h_last))
```

APPENDIX B
MOCK DATA SPLIT

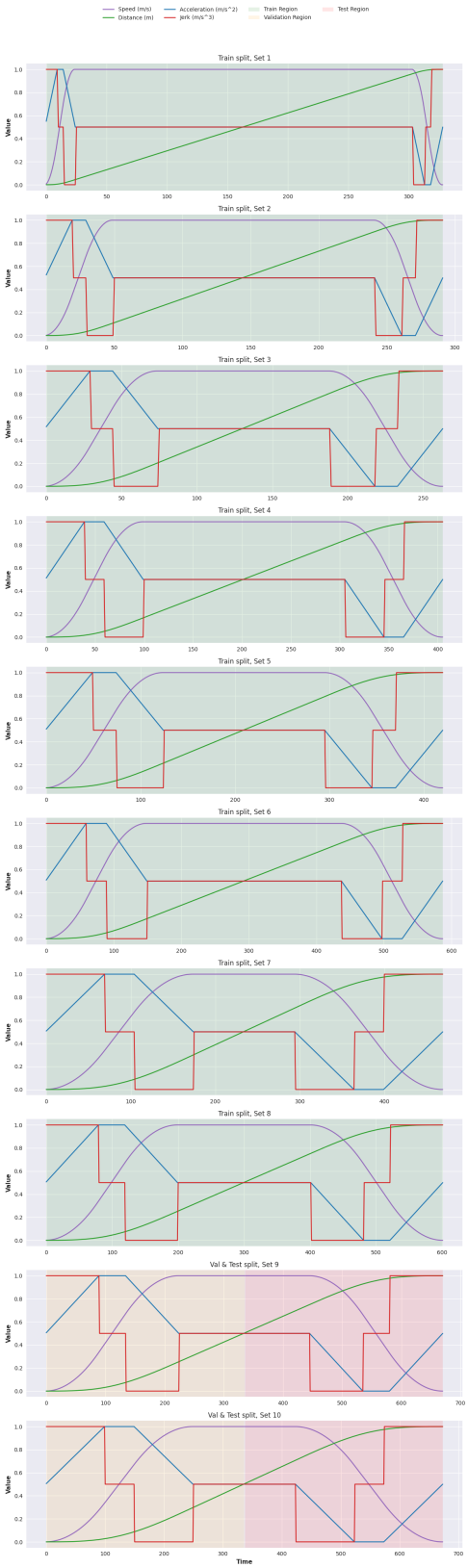


Fig. 11. Input data split, the 8 top graphs with the green background are used for training, the bottom two are split between the validation in yellow, and test in red.

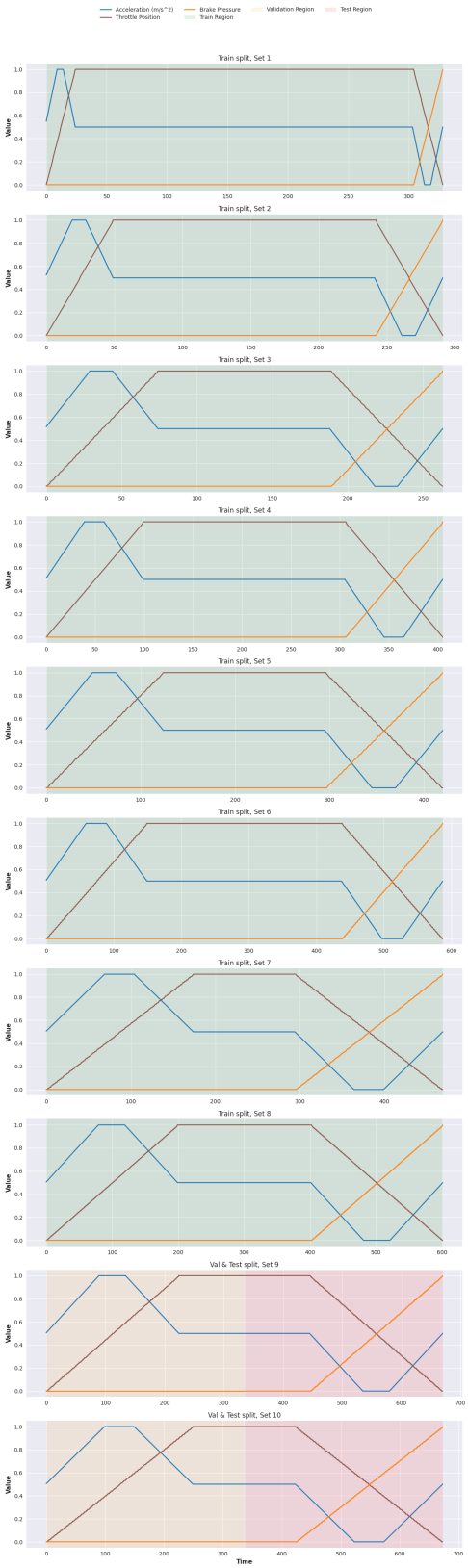


Fig. 12. Output data split, the 8 top graphs with the green background are used for training, the bottom two are split between the validation in yellow, and test in red.

APPENDIX C
DRILLING DATA

A. *Drilling Input Data*

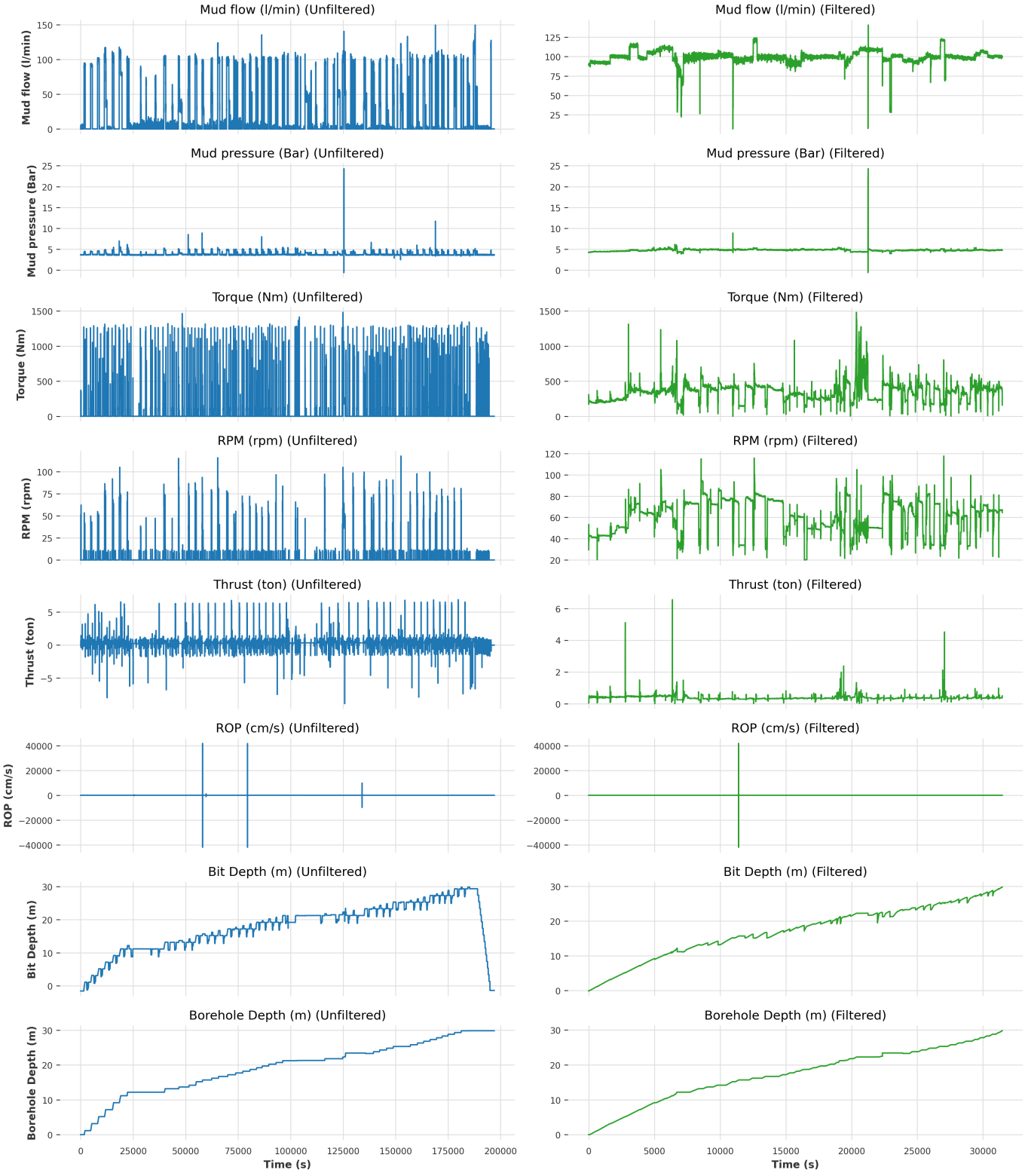


Fig. 13. Input data from borehole 1

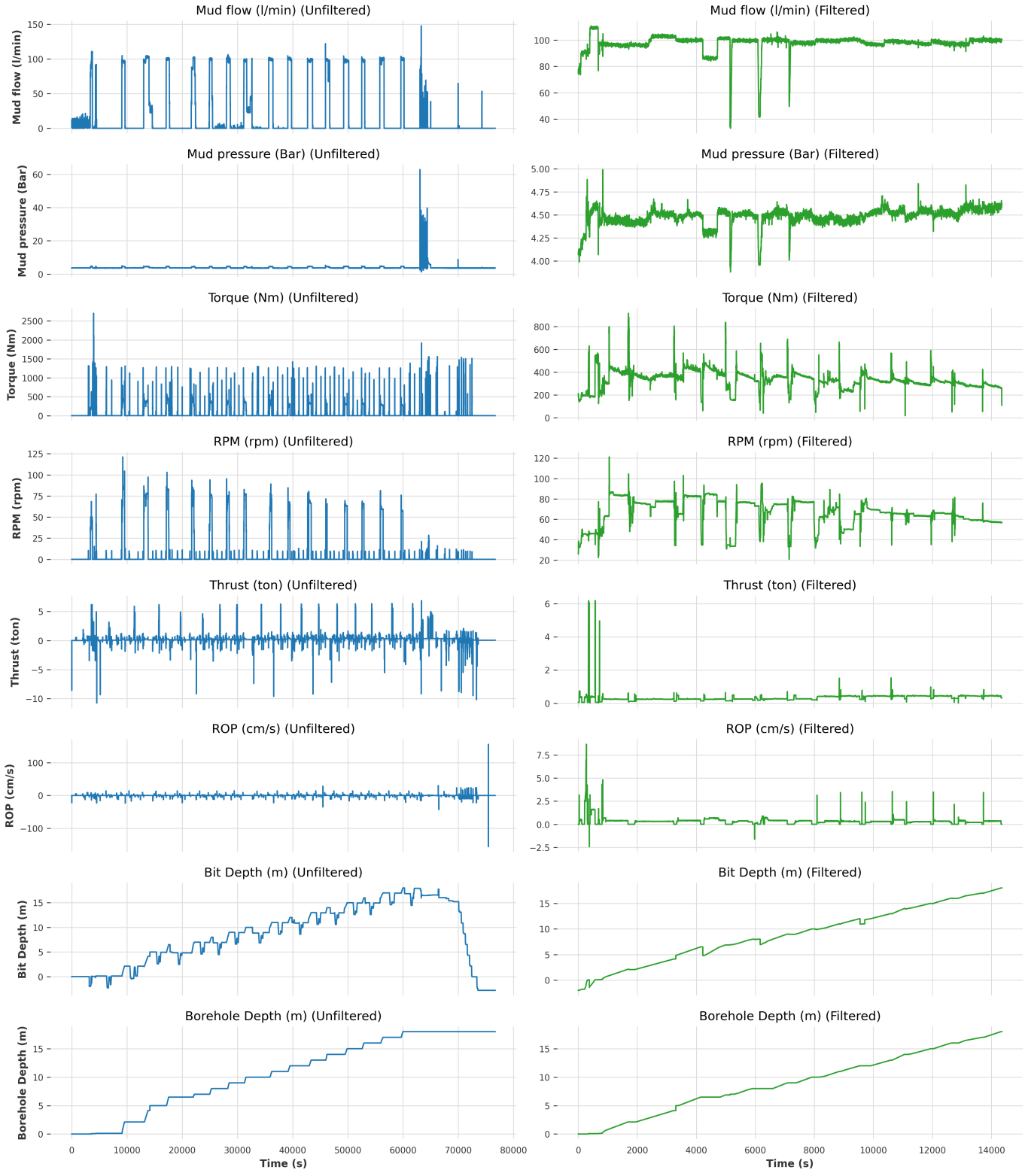


Fig. 14. Input data from borehole 2

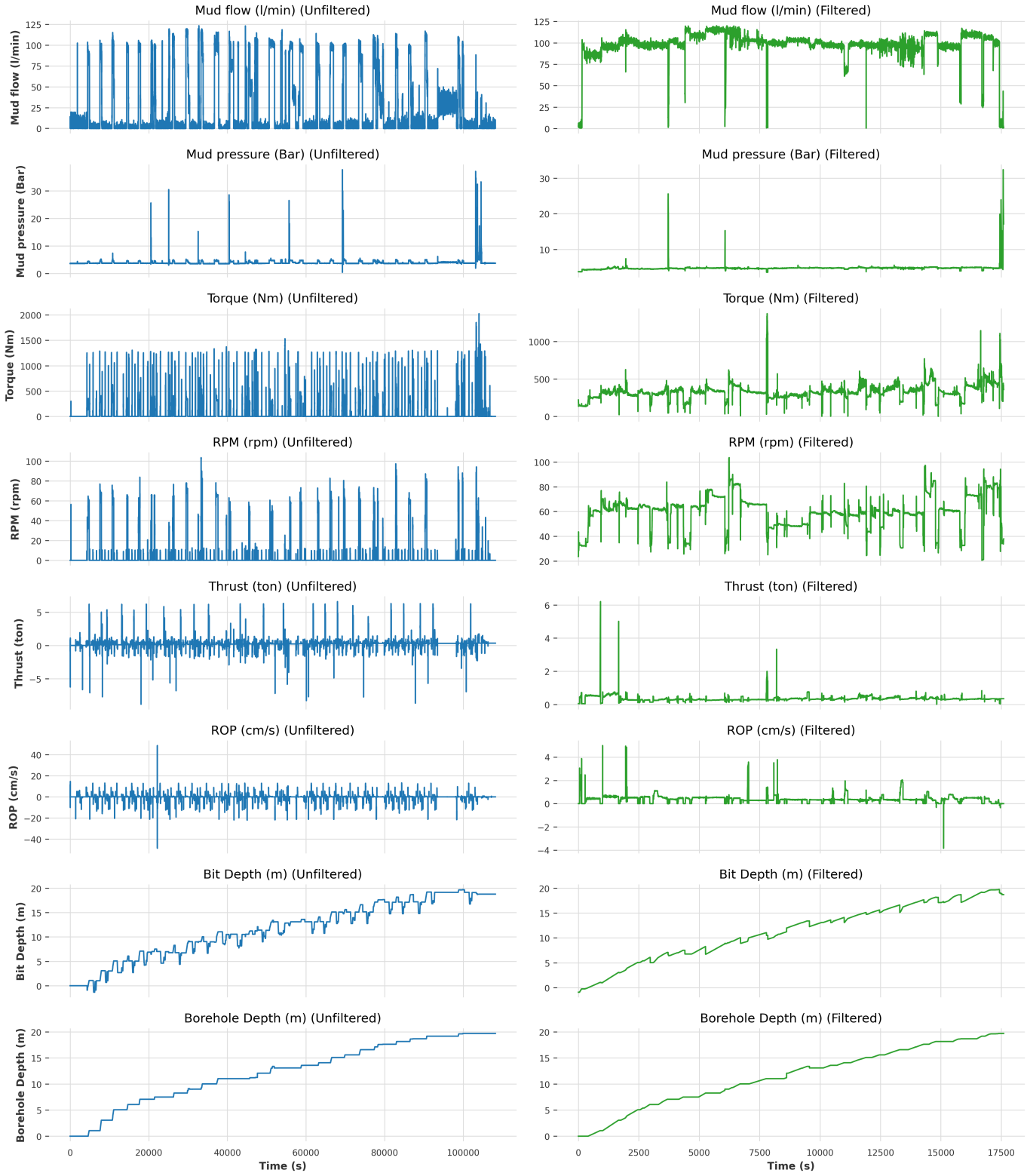


Fig. 15. Input data from borehole 3

B. Drilling Output Data

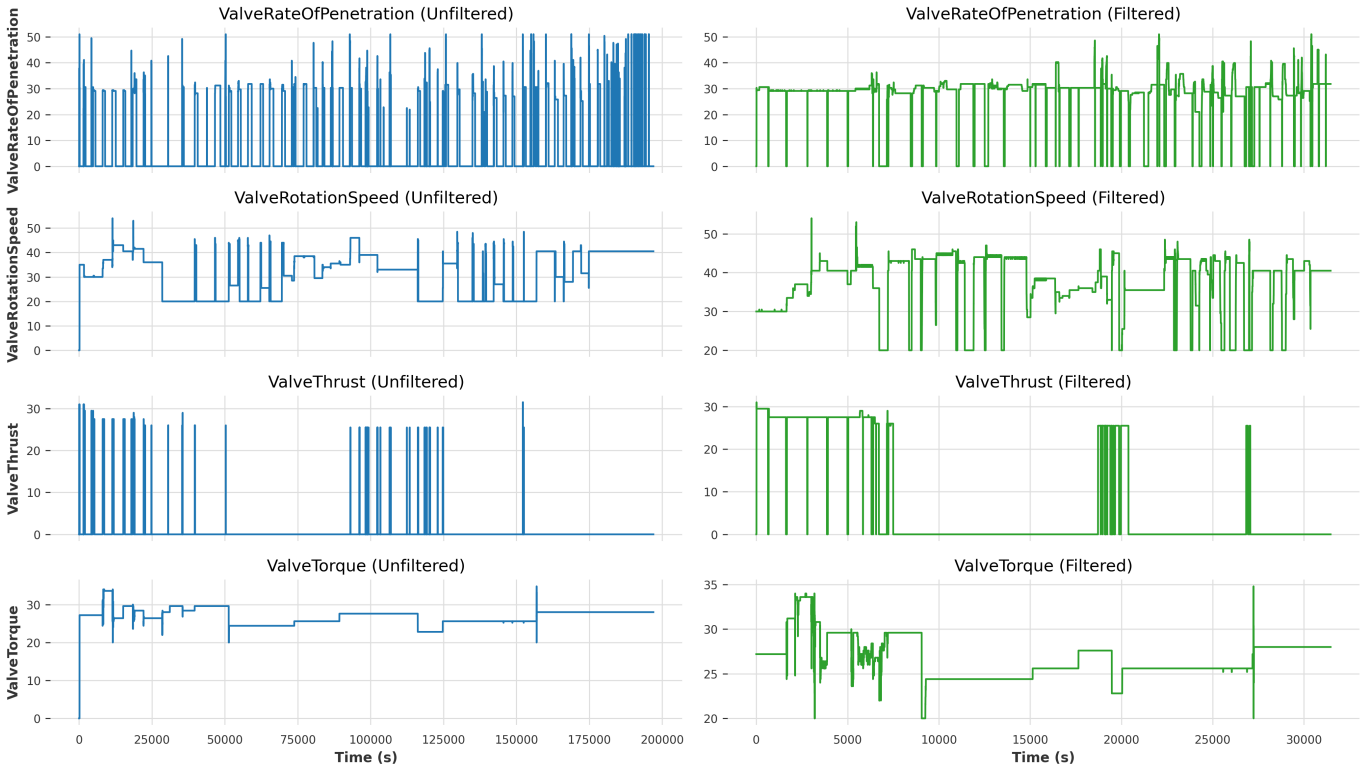


Fig. 16. Output data from borehole 1

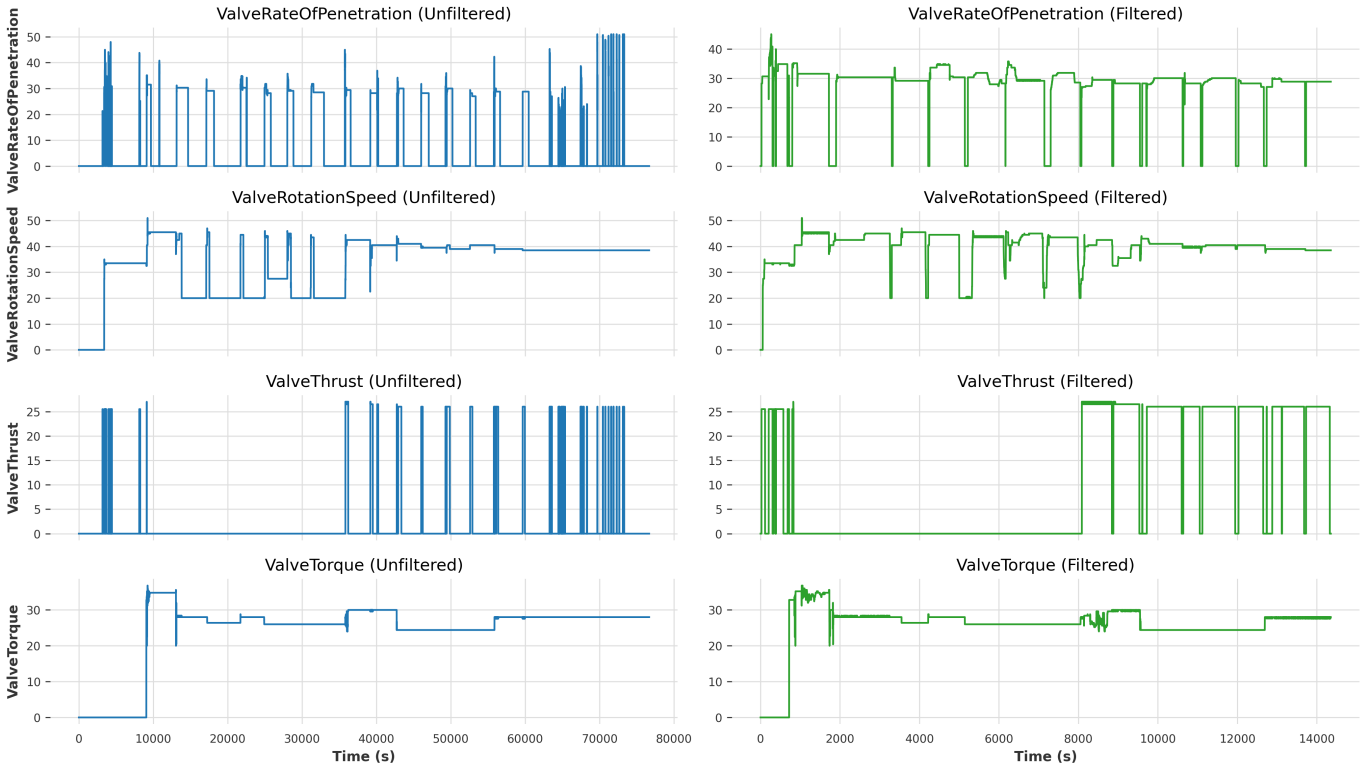


Fig. 17. Output data from borehole 2

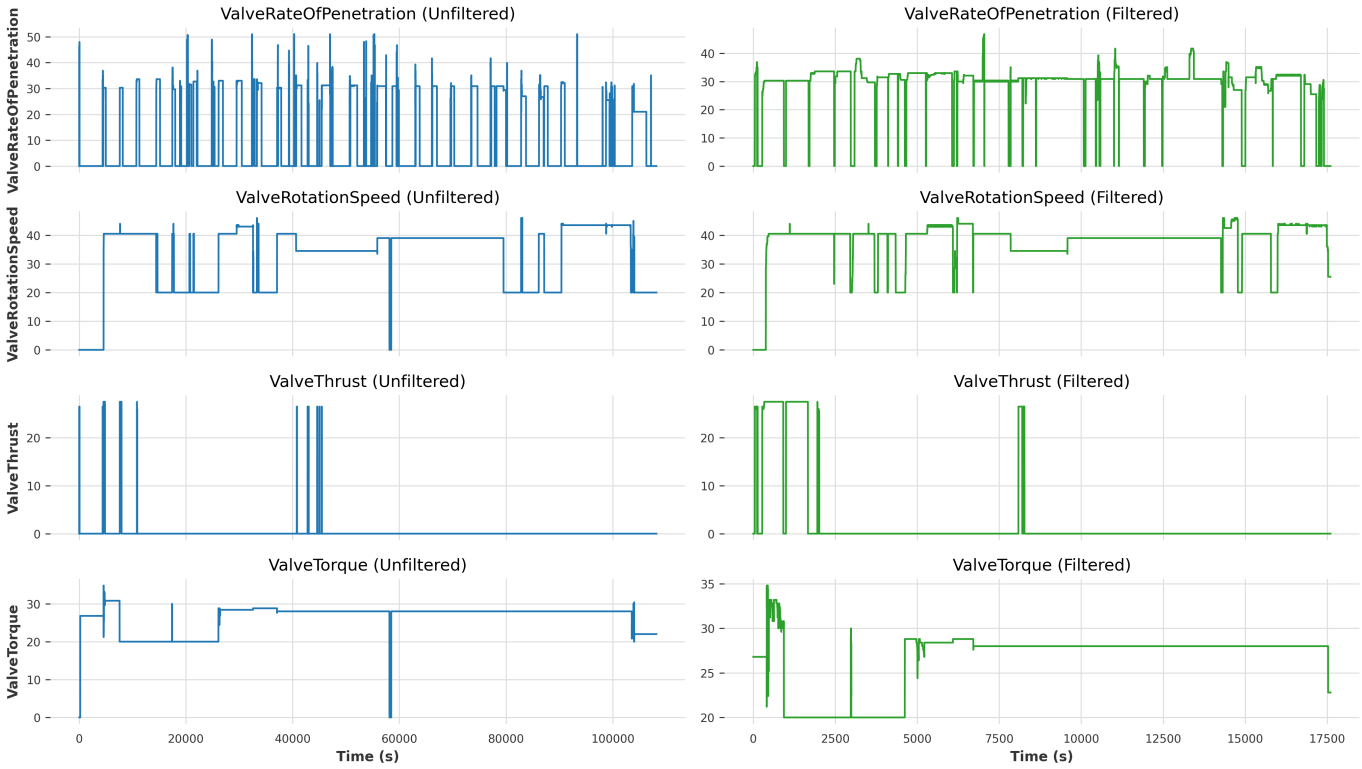


Fig. 18. Output data from borehole 3

C. Cone Penetration Test Classification

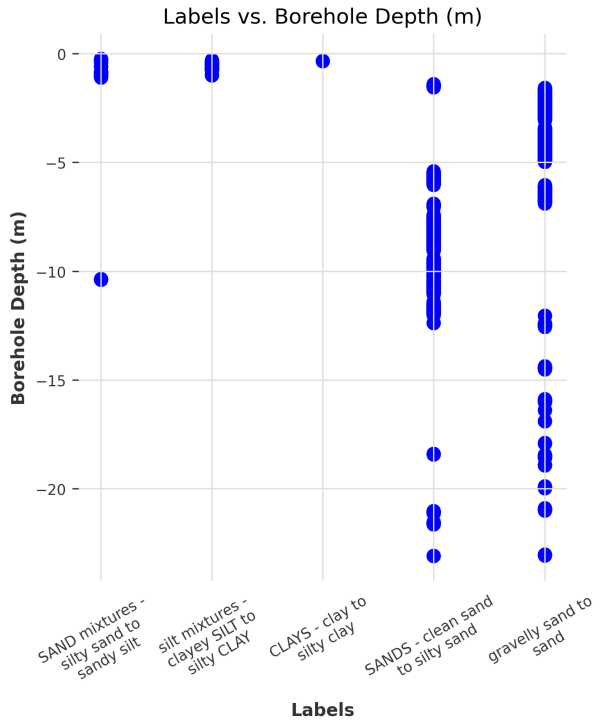


Fig. 19. CPT classification data from borehole 1 using the Robertson 2009 methoc

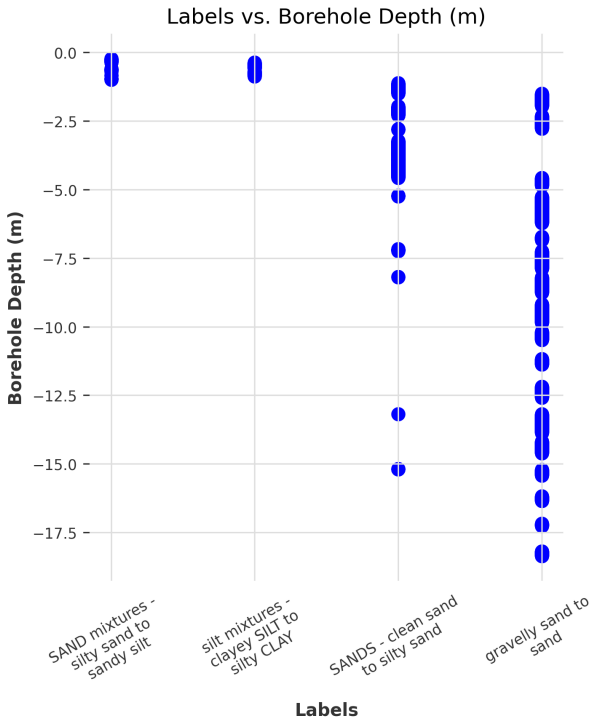


Fig. 20. CPT classification data from borehole 2 using the Robertson 2009 methoc

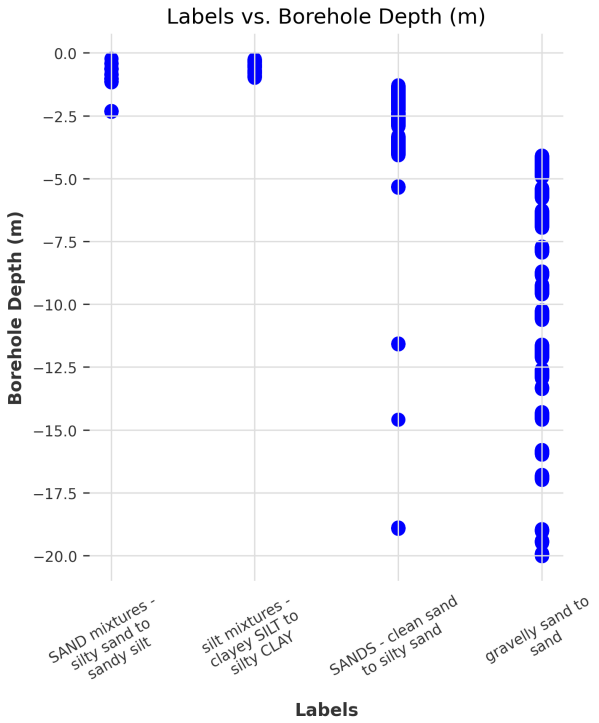


Fig. 21. CPT classification data from borehole 3 using the Robertson 2009 methoc

D. Data split

Fig. 22. Input data split, the two top graphs with the green background are used for training, the bottom one is split between the validation in yellow, and test in red.

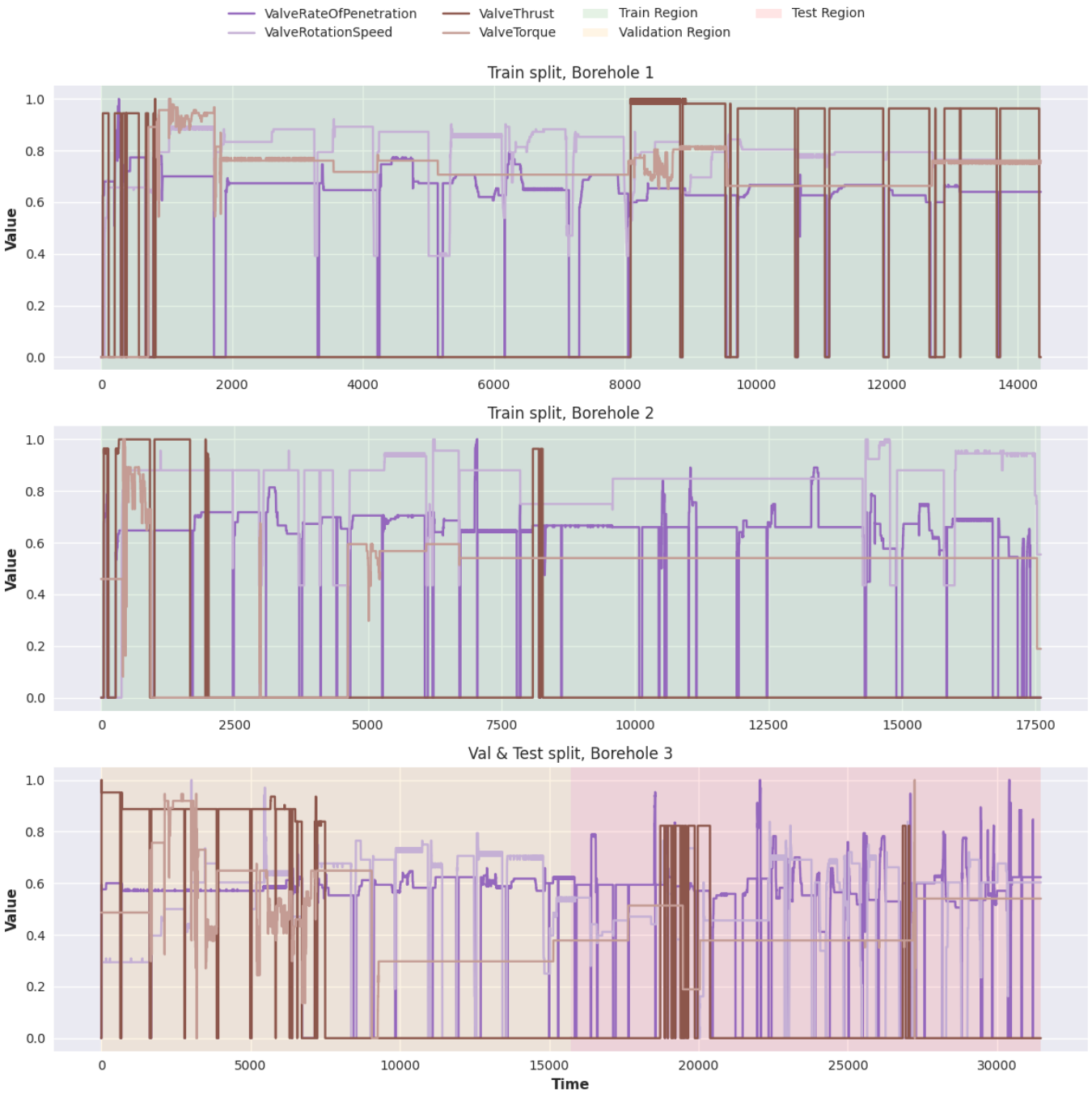


Fig. 23. Otput data split, the two top graphs with the green background are used for training, the bottom one is split between the validation in yellow, and test in red.

APPENDIX D
VEHICLE DATA APPROACH RESULTS

A. Mean Squared Error per Output

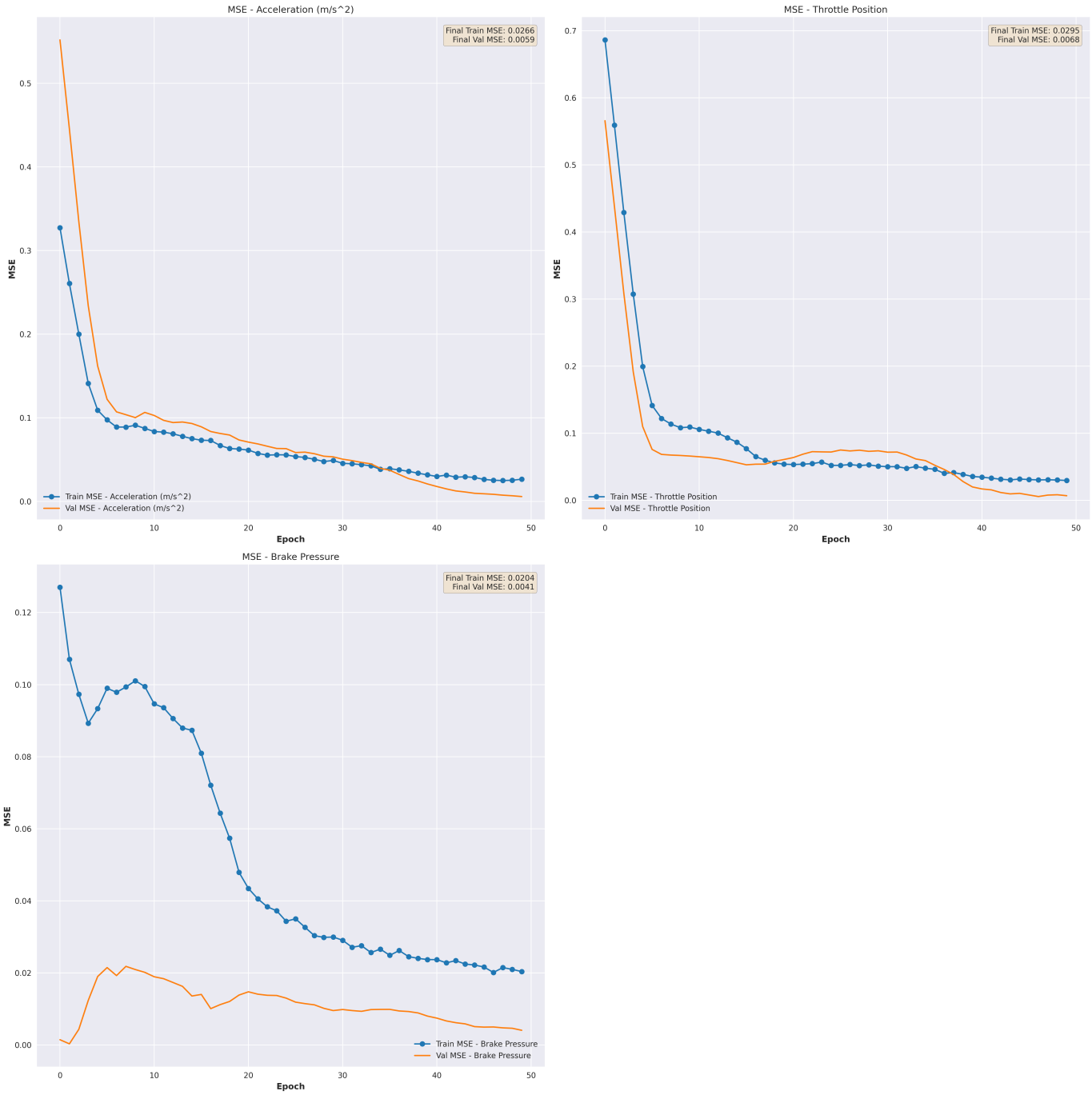


Fig. 24. MSE per output on the test data for Approach 1.

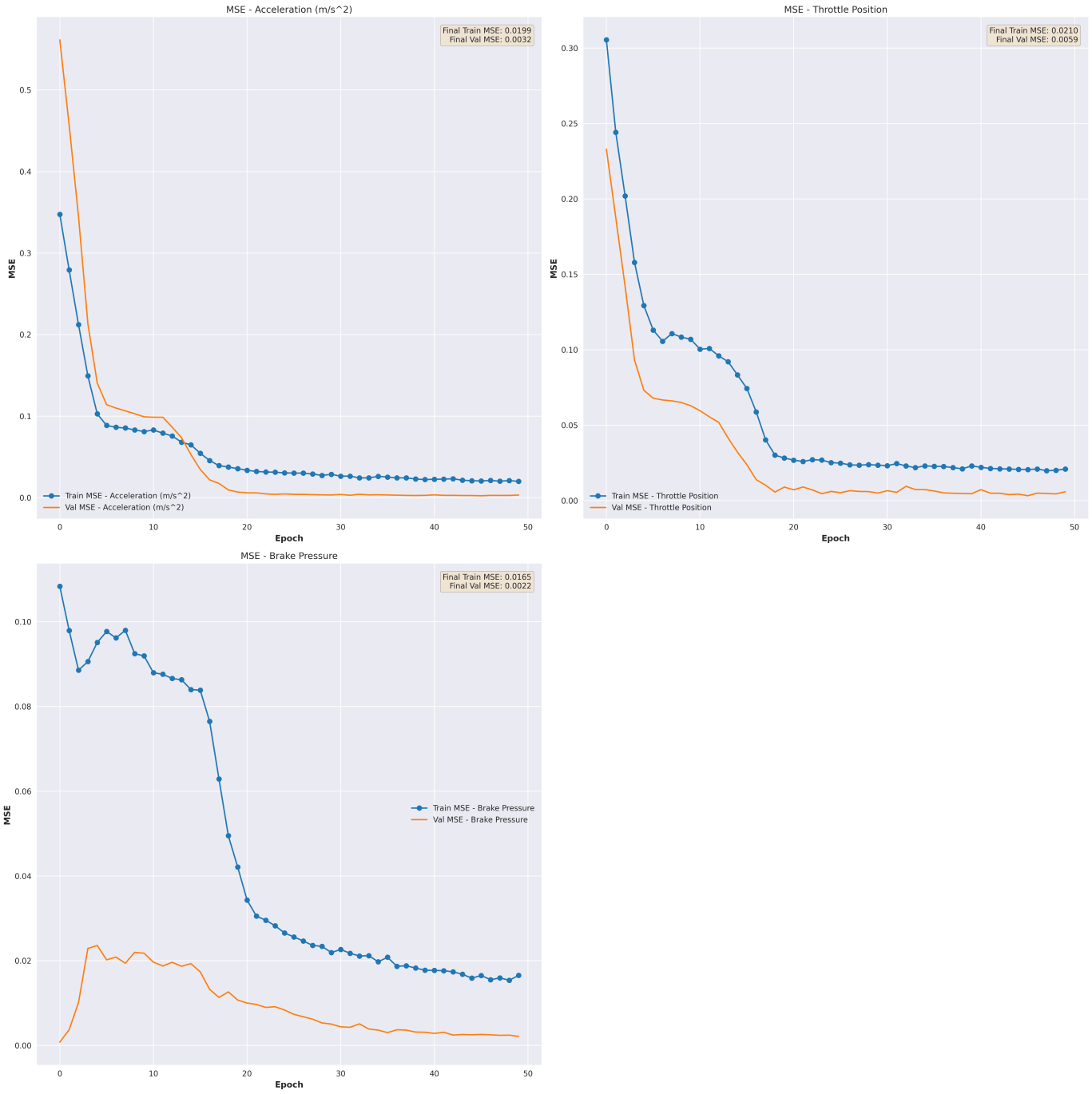


Fig. 25. MSE per output on the test data for Approach 2.

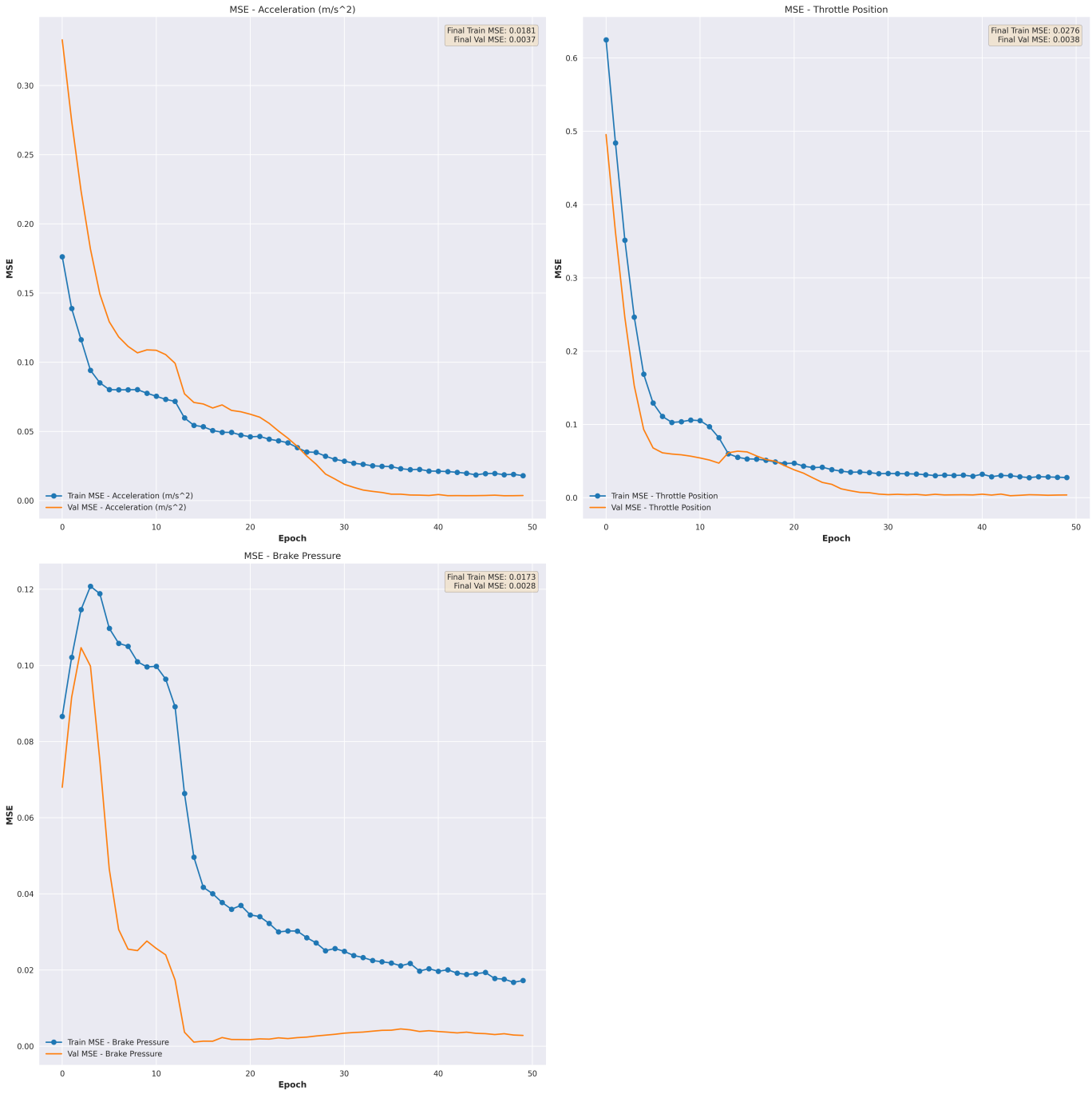


Fig. 26. MSE per output on the test data for Approach 3.

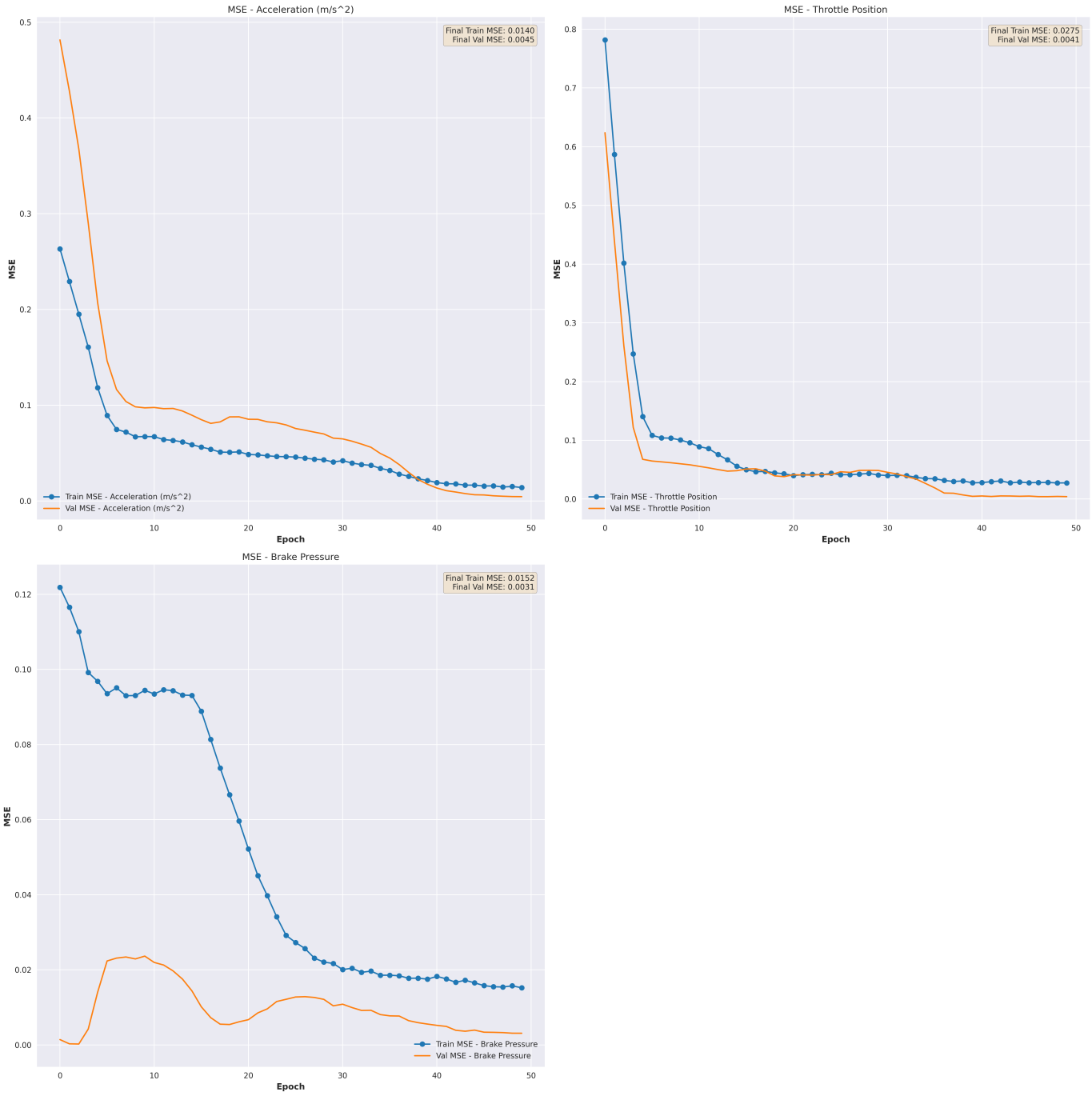


Fig. 27. MSE per output on the test data for Approach 4.

B. Forecast

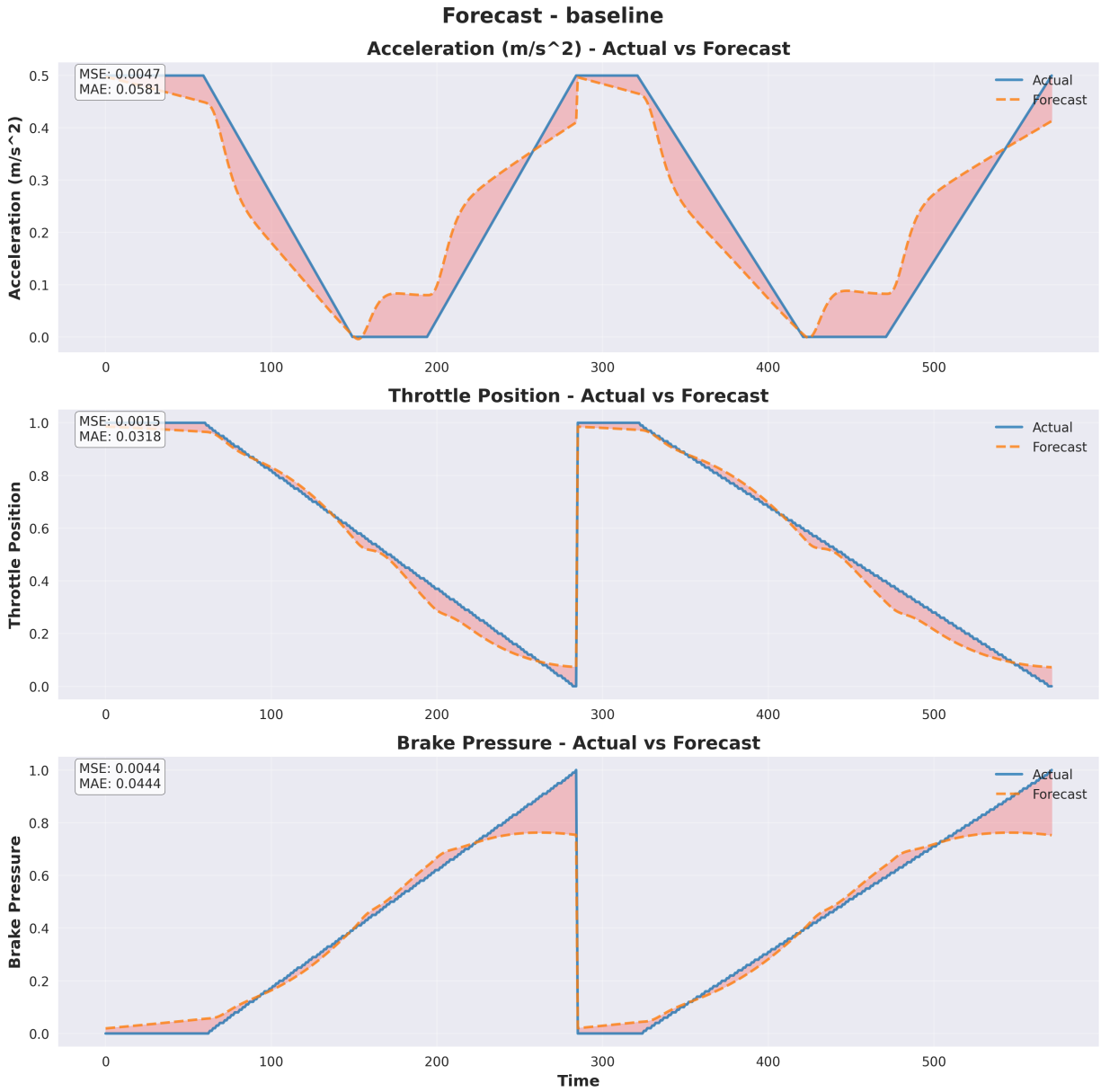


Fig. 28. Forecast on the test data, with calculated MSE and MAE for Approach 1.

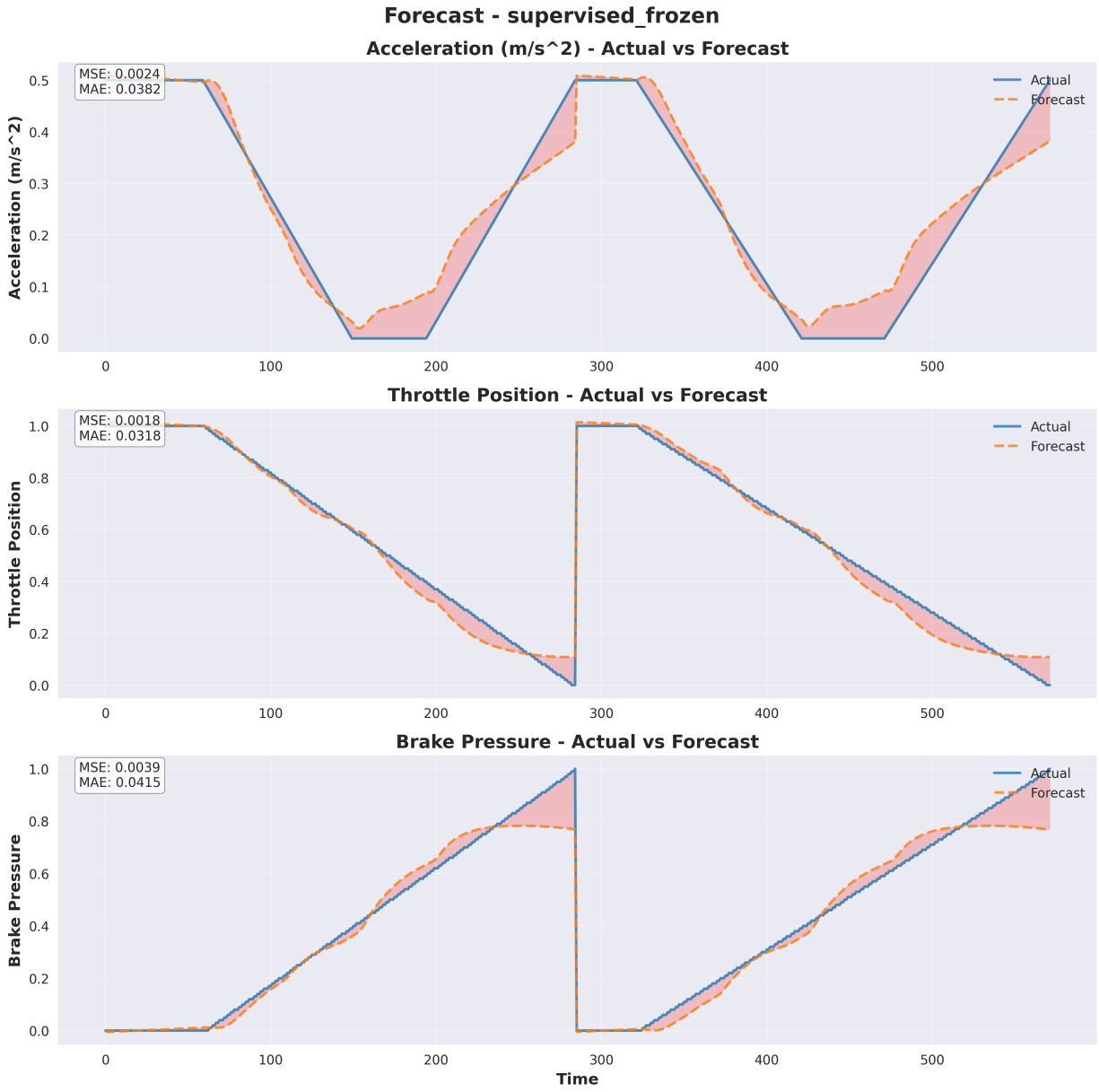


Fig. 29. Forecast on the test data, with calculated MSE and MAE for Approach 2.

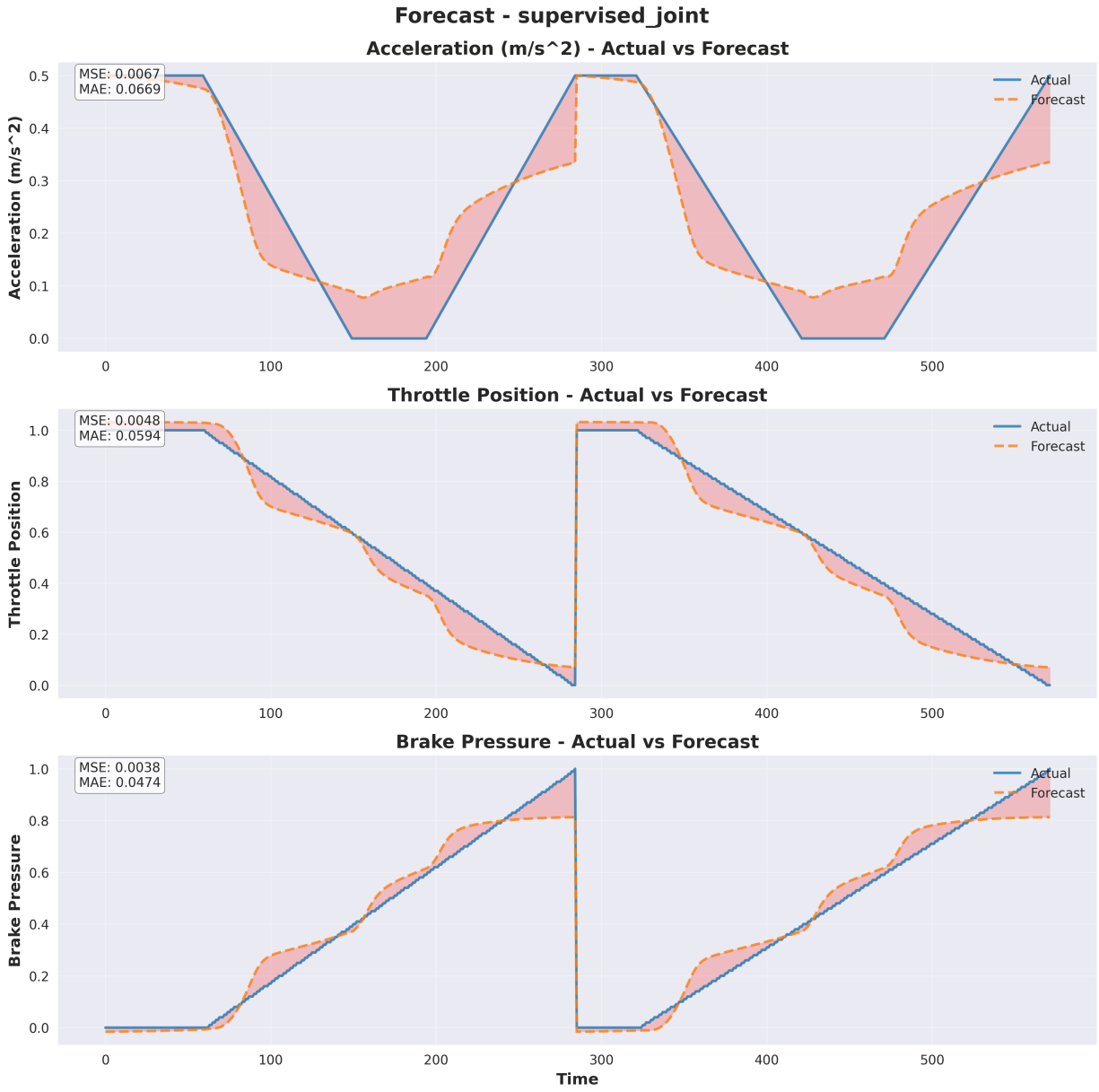


Fig. 30. Forecast on the test data, with calculated MSE and MAE for Approach 3.

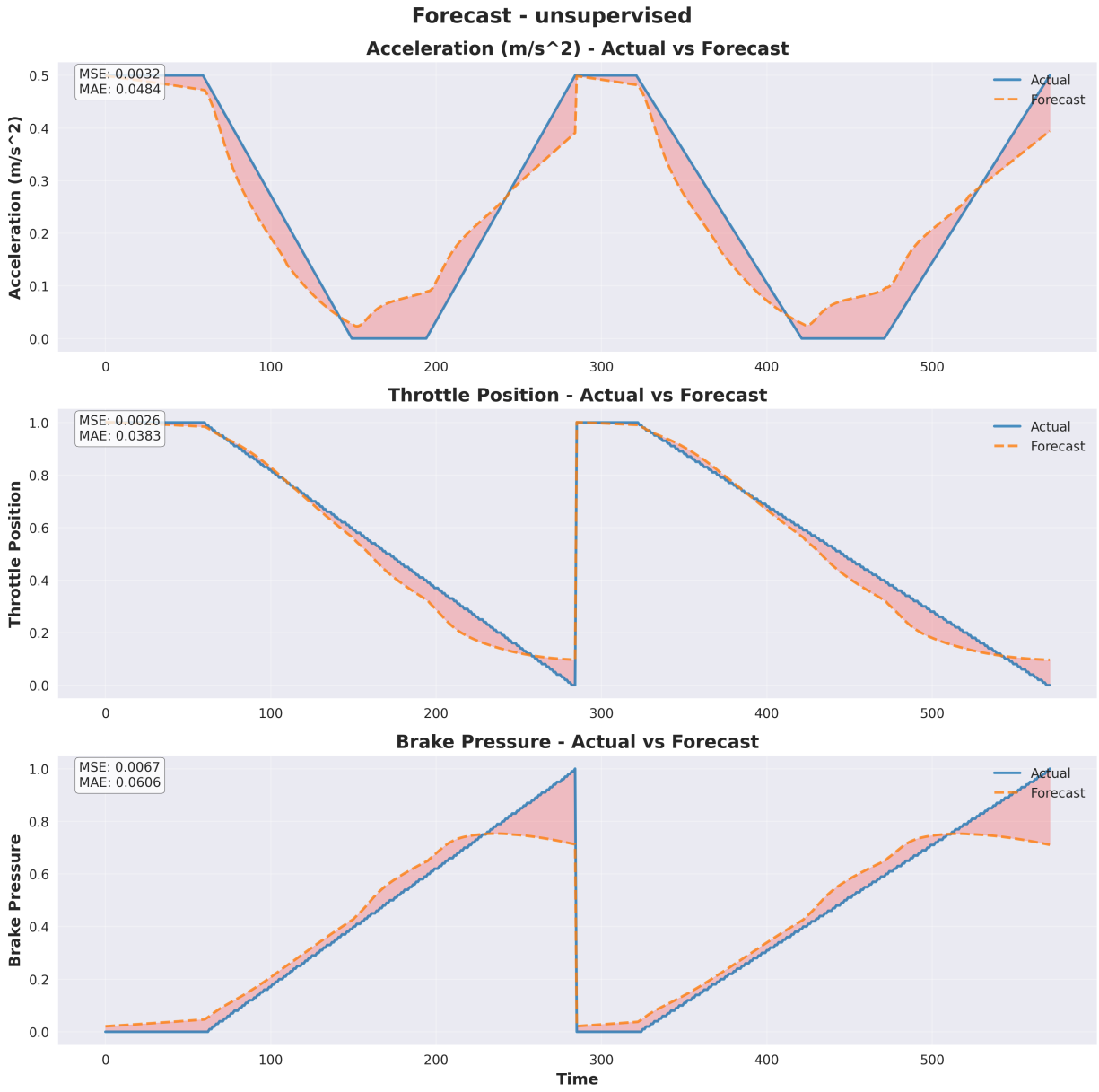


Fig. 31. Forecast on the test data, with calculated MSE and MAE for Approach 4.

APPENDIX E
DRILLING DATA APPROACH RESULTS

A. Mean Squared Error per Output

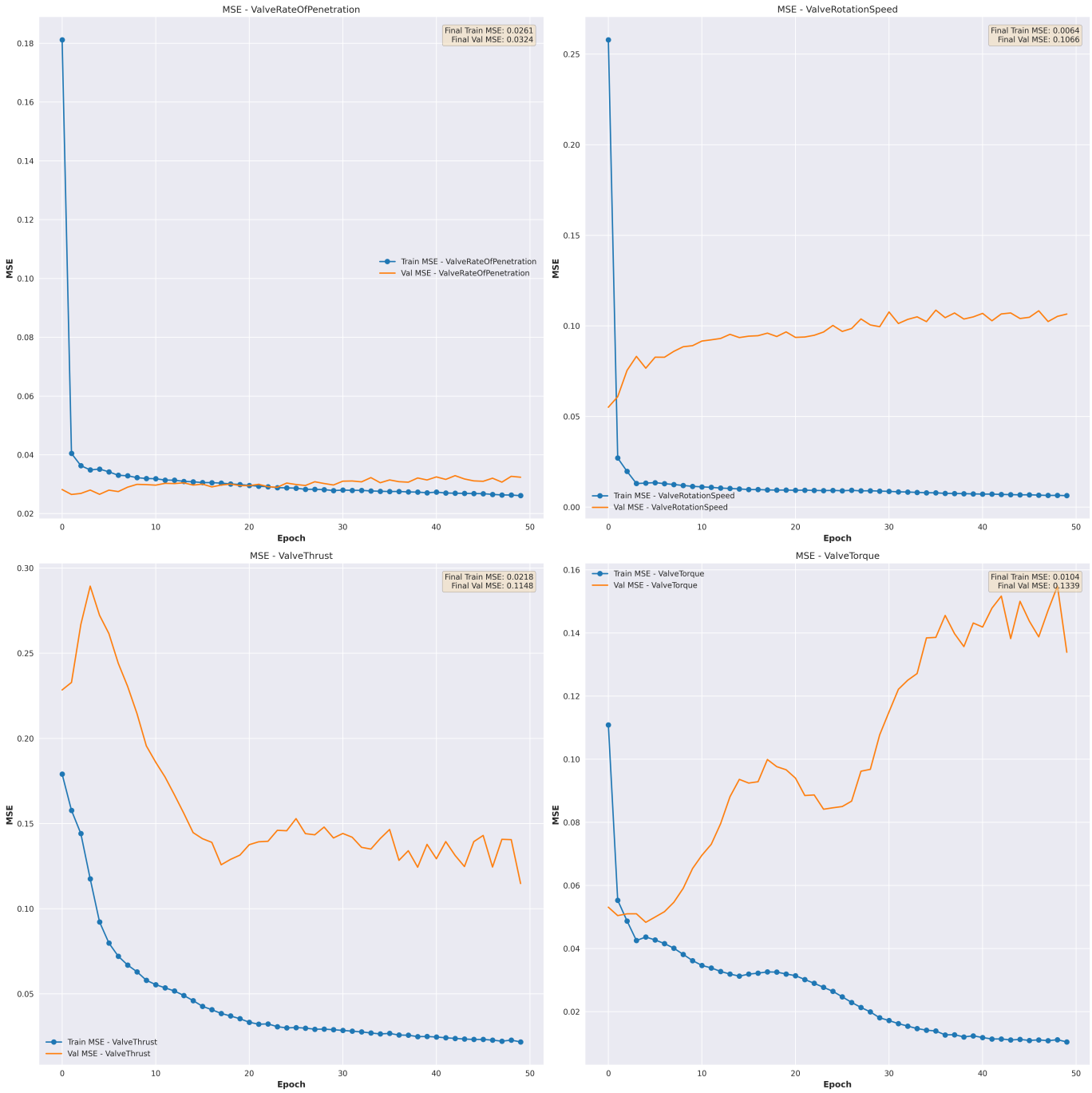


Fig. 32. MSE per output on the test data for Approach 1.

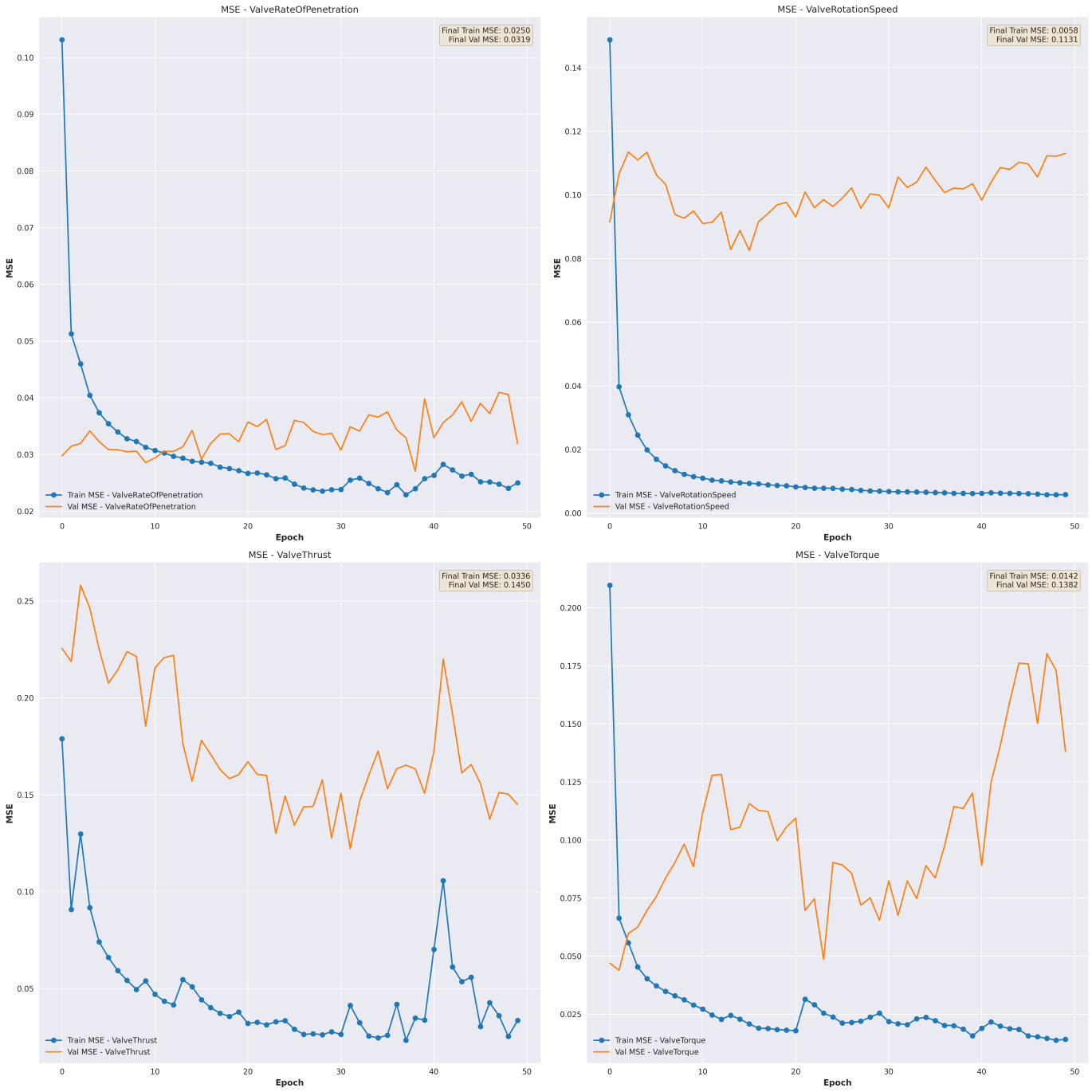


Fig. 33. MSE per output on the test data for Approach 2.

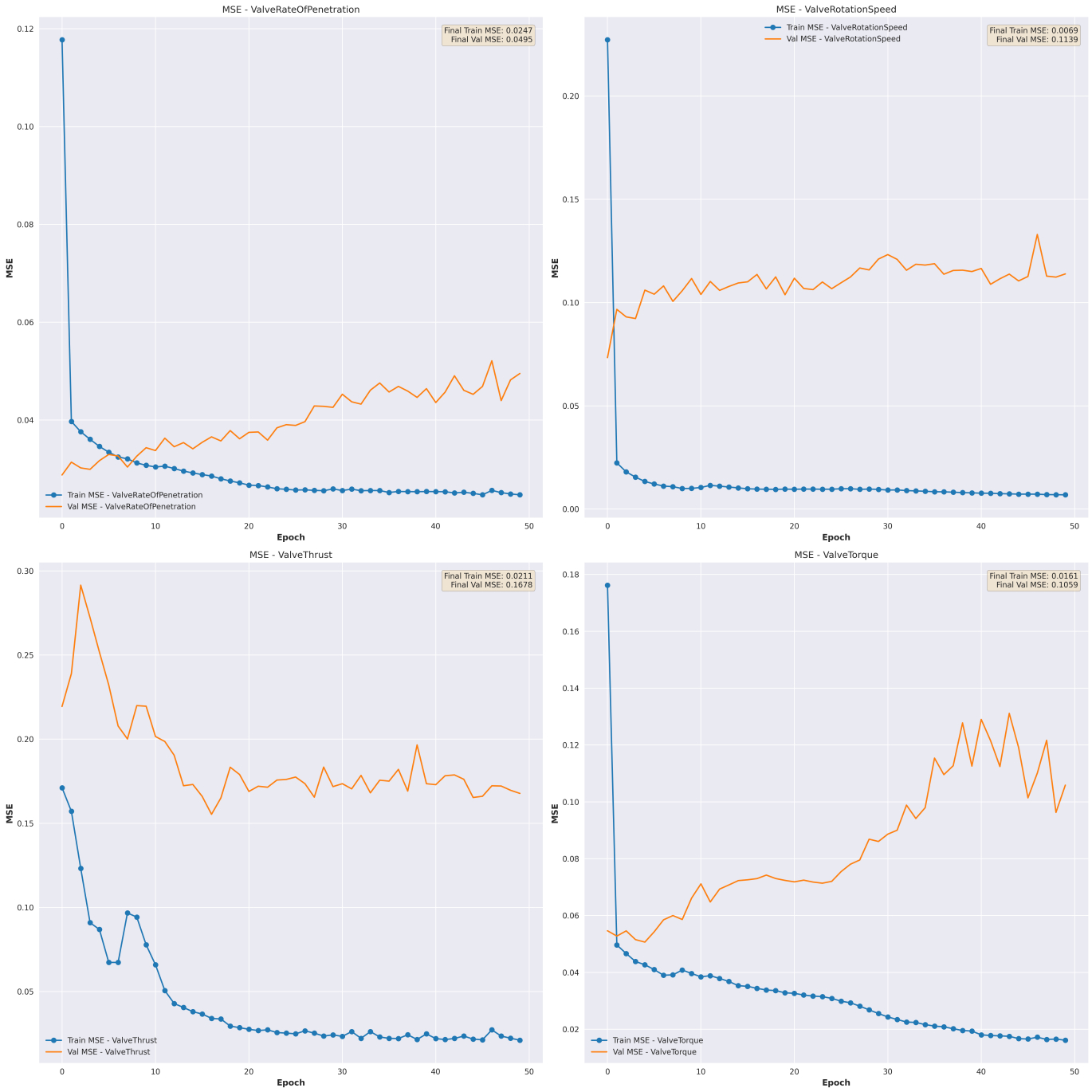


Fig. 34. MSE per output on the test data for Approach 3.

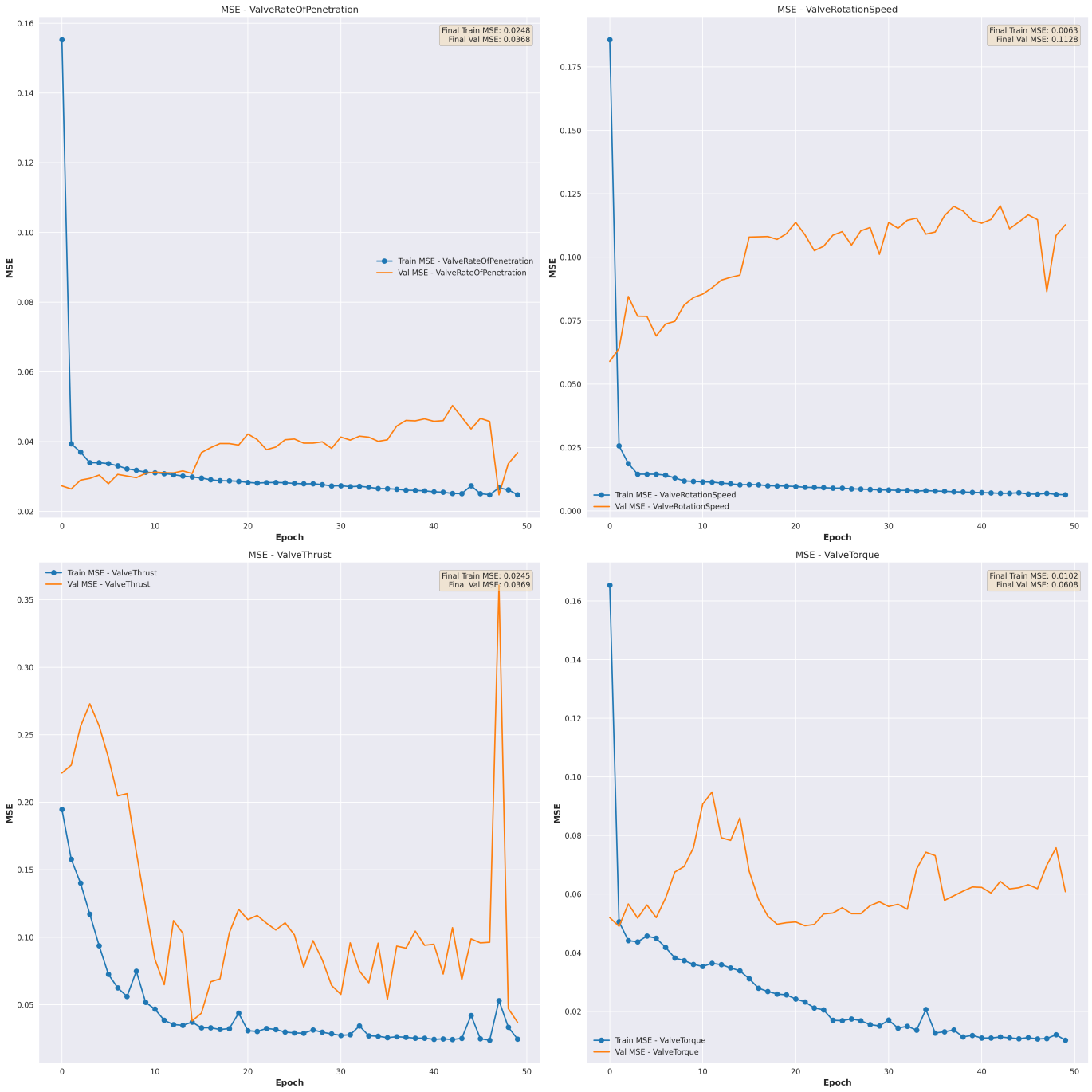


Fig. 35. MSE per output on the test data for Approach 4.

B. *Forecast*

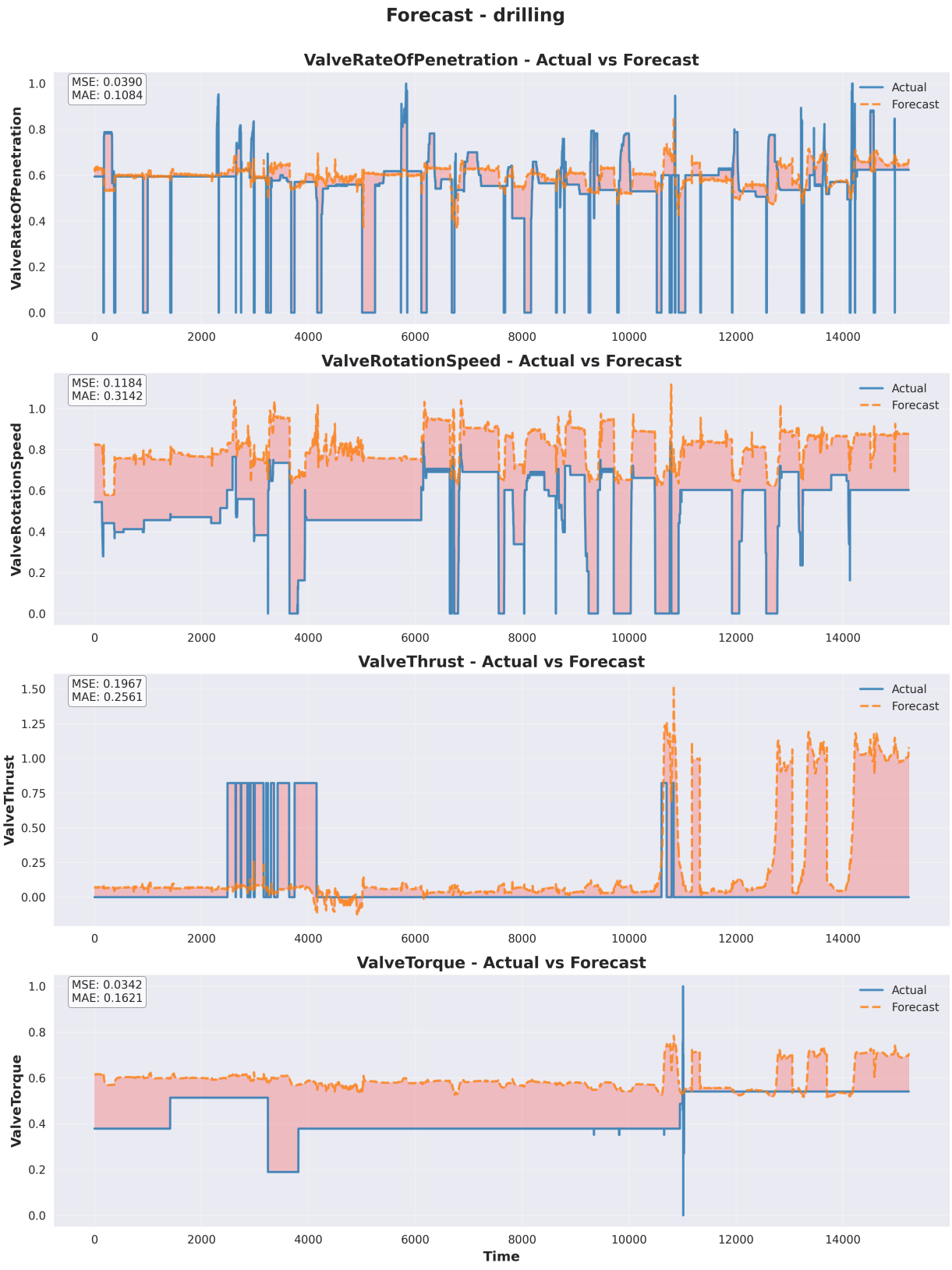


Fig. 36. Forecast on the test data, with calculated MSE and MAE for Approach 1.

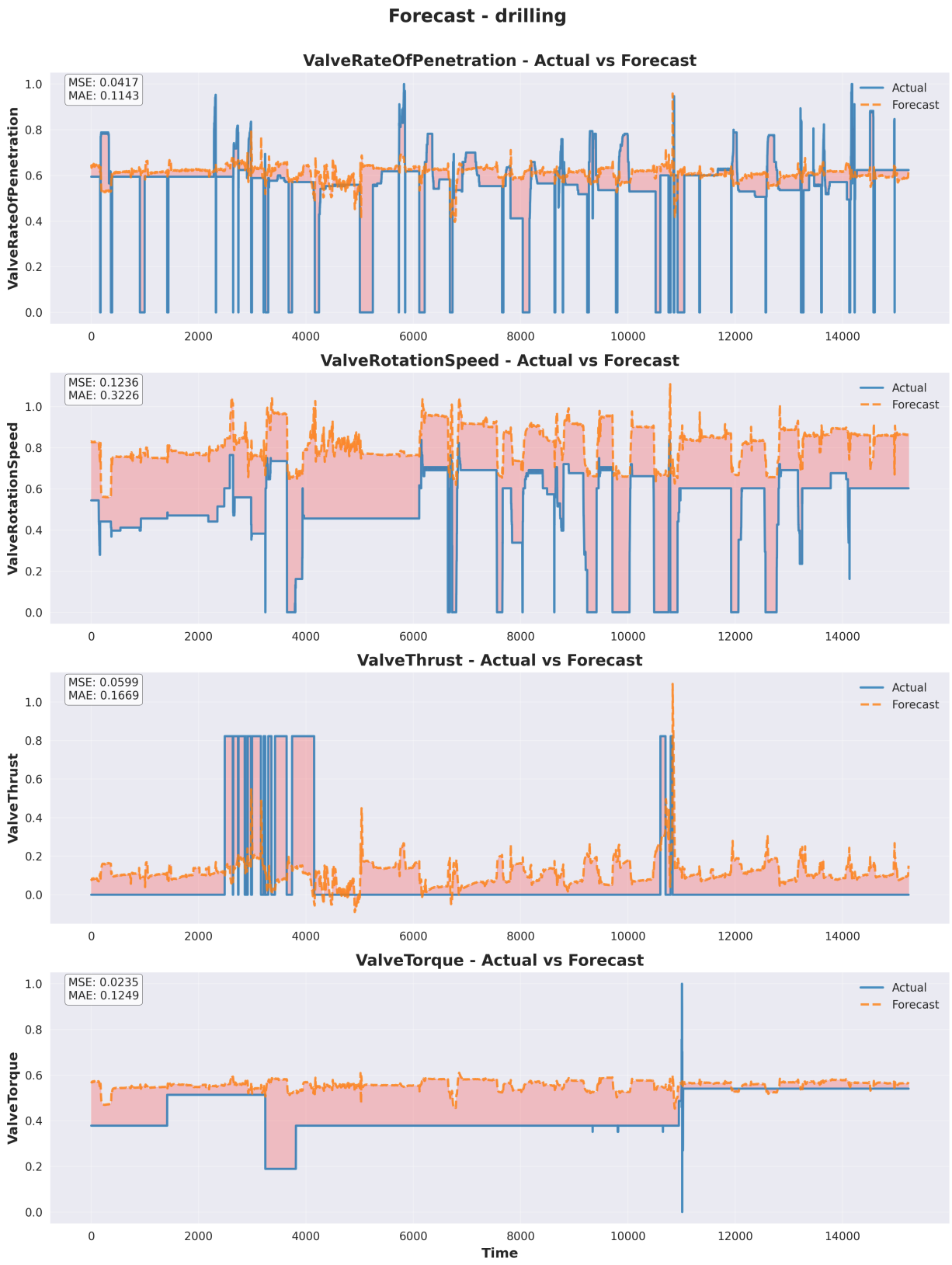


Fig. 37. Forecast on the test data, with calculated MSE and MAE for Approach 2.

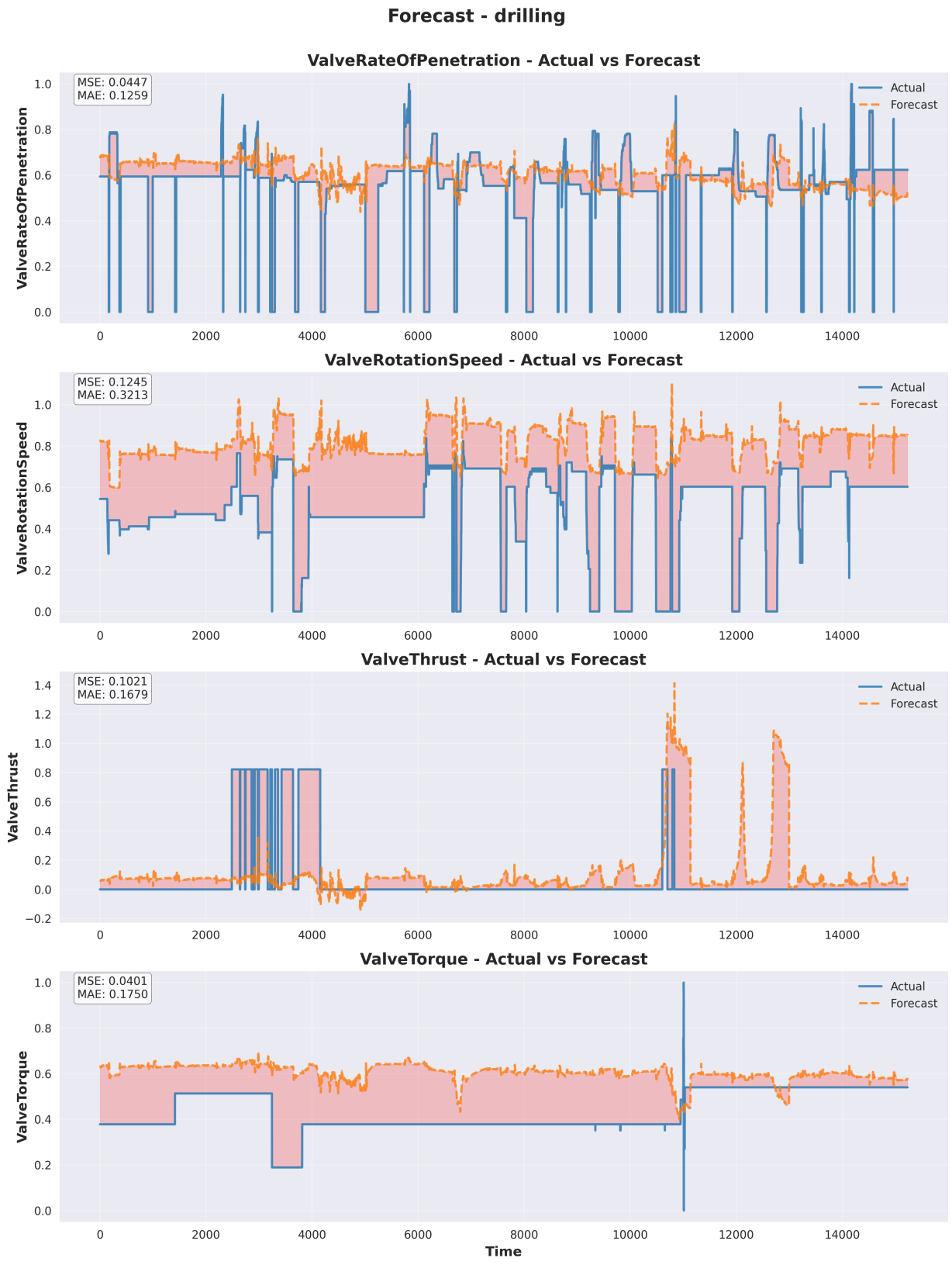


Fig. 38. Forecast on the test data, with calculated MSE and MAE for Approach 3.

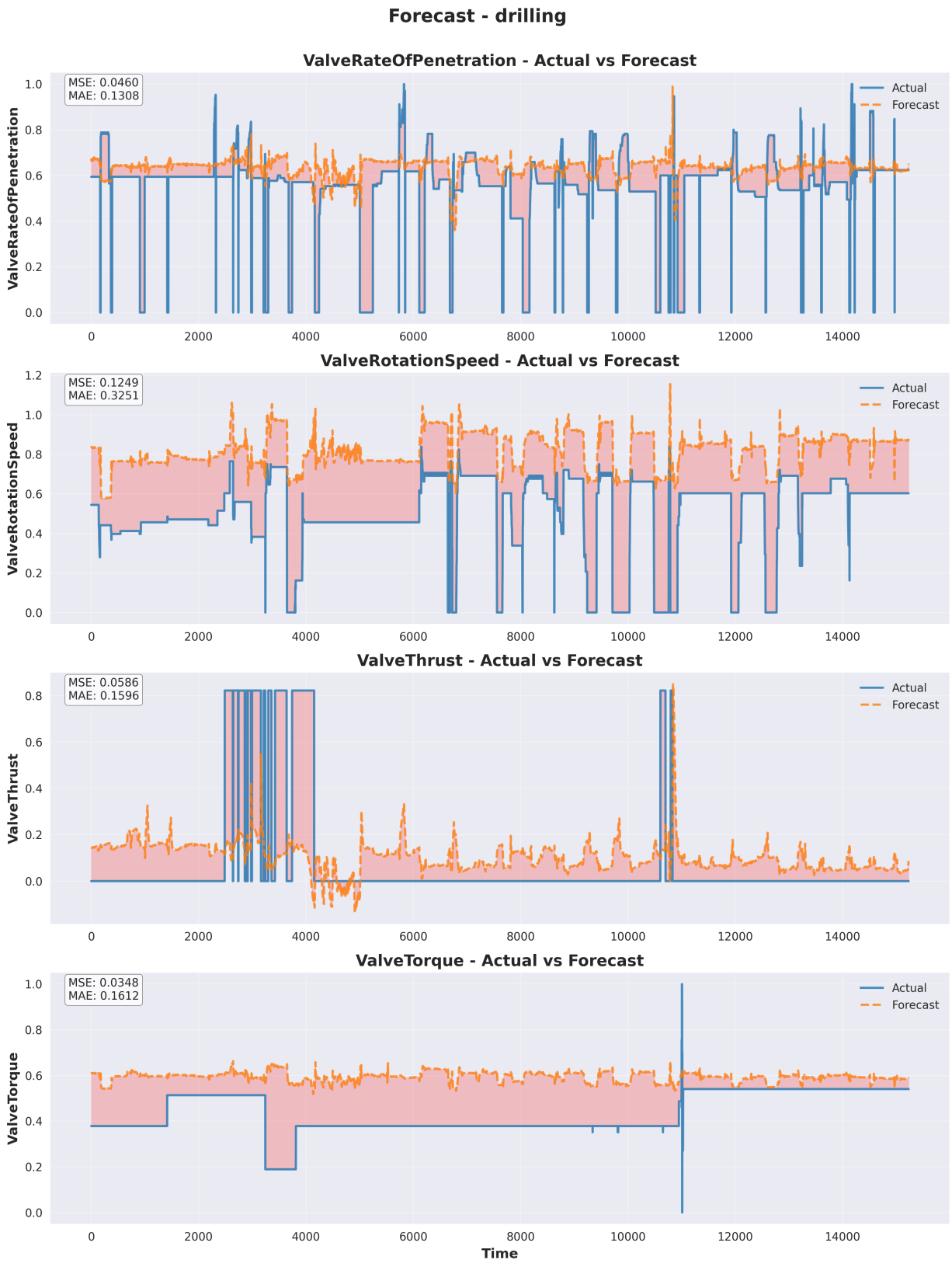


Fig. 39. Forecast on the test data, with calculated MSE and MAE for Approach 4.