Cross-Architecture Evaluation and Proposing an Ensemble Method for Jet Engine Remaining Life Prediction Using NASA's C-MAPSS

49308, 49580, 45733, 50250

Abstract

Remaining Useful Life (RUL) prediction is a central task in Prognostics and Health Management (PHM), with recent literature increasingly exploring deep learning approaches. While prior work focuses on individual architectures and their specific modeling benefits, we present a systematic comparative study using the NASA C-MAPSS dataset. Our contribution is threefold: (1) we apply rigorous preprocessing including standardization, data augmentation, and piece-wise linear RUL transformation; (2) we evaluate four distinct architectures—Hierarchical Deep Neural Network (HDNN), Autoencoder-BiLSTM, Residual CNN (ResCNN), and Transformers —across all four C-MAPSS subdatasets; (3) we propose an ensemble "supermodel" that integrates variations of the previous architectures, leveraging complementary model characteristics via shared feature extraction. Experimental results demonstrate that optimal base model performance varies across subdatasets, validating our ensemble strategy. Our supermodel achieves competitive results with an RMSE of 12.7 and score of 257 on FD001. While not always outperforming individual models, the ensemble consistently improves generalization and performance stability across all datasets.

1. Introduction

Unplanned equipment failures in industrial systems lead to costly downtime and safety risks. While preventive maintenance mitigates some of these issues, it can cause premature part replacement and increased operational costs. Predictive maintenance offers a more efficient alternative by leveraging sensor data to estimate Remaining Useful Life (RUL) and optimize intervention timing.

Recent research demonstrates that deep learning has strong potential for RUL prediction, yet individual architectures often exhibit dataset-specific performance characteristics. Prior work typically benchmarks single model types—such as CNNs for local feature extraction or LSTMs for temporal dependencies—without addressing their complementarity.

In this paper, we evaluate four diverse deep learning architectures on the NASA C-MAPSS dataset: (1) a Hierarchical Deep Neural Network (HDNN), (2) a residual Convolutional Neural Network (ResCNN), (3) an Autoencoder-BiLSTM hybrid, and (4) a Transformer. Each model is trained following consistent preprocessing, including standardization, data augmentation, and transformation of RUL targets to a piece-wise linear form. Our contributions are threefold:

- We provide a comparative analysis of model performance across all four C-MAPSS subdatasets, identifying cases where each architecture excels.
- We propose an ensemble framework combining Autoencoder-based HDNN, BiLSTM, and ResCNN models to exploit complementary strengths.
- We show that the ensemble improves generalization and robustness, achieving competitive RUL prediction performance with reduced variance across datasets.

These findings advance the development of more adaptable and accurate deep learning pipelines for predictive maintenance in industrial settings.

2. Literature Review

In their comprehensive survey, Li et al. (2024) review 106 studies on different deep learning approaches and model architectures for predictive maintenance. They find that a wide range of deep learning architectures can achieve strong performance in RUL prediction, with their effectiveness largely depending on the specific characteristics of the dataset, task requirements, and model design choices. The most common architectures for RUL prediction are LSTM networks and CNNs due to their robust performance in discovering knowledge about potential failures and anomalies from long-term sequential data. As outlined in Section 1, this paper focuses on the in-depth investigation and evaluation of four specific architectures. The following literature review outlines the key strengths of each of the four selected architectures for

RUL prediction, supported by representative examples from recent state-of-the-art research.

2.1. Hybrid Deep Neural Networks

Hybrid deep neural networks (HDNNs) combine different deep learning architectures to jointly leverage their strengths in capturing temporal and spatial features from sensor data. These models are particularly effective for RUL prediction tasks where degradation signals are both temporally dependent and noisy.

Zhao et al. (2017) propose a Convolutional Bi-directional LSTM (CBLSTM) model for tool wear prediction from raw sensor data. A CNN first extracts local features to reduce noise and enhance robustness, which are then processed by stacked bi-directional LSTMs to capture temporal dependencies. The model, concluding with dense and regression layers, achieves improved performance, illustrating the effectiveness of local feature extraction followed by deep temporal encoding for accurate RUL prediction.

Al-Dulaimi et al. (2019) propose a HDNN model that, unlike prior work that processes sensor data in sequence, extracts temporal features via LSTM and spatial features via CNN simultaneously, combining them through dense layers. The model is evaluated on all four NASA's C-MAPSS datasets. Results show that the proposed HDNN consistently outperforms other methods, achieving over 20% lower RMSE on FD002 and 13% lower RMSE on FD004 compared to DCNN. This indicates that the HDNN framework offers generalization across varying operational conditions.

2.2. Transformers

The benefit of applying transformer models to RUL prediction is mainly rooted in three characteristics: (I) Self-attention effectively captures relationships between distant time steps, (II) the attention mechanism allows the model to focus on the most informative parts of the sequence for RUL estimation and (III) transformers allow faster training through parallel computation.

Zhang et al. (2022) propose DAST (Dual Aspect Self-Attention Transformer), a transformer architecture for RUL prediction that leverages parallel self-attention encoders to separately model sensor-wise and temporal dependencies. The dual-encoder design enables the model to capture long-term dependencies and selectively focus on the most relevant sensors and time steps. Evaluated on the C-MAPSS dataset, DAST significantly outperforms state-of-the-art baselines such as BiLSTM, DCNN, and ELSTMNN, particularly on the more challenging subsets FD002 and FD004, achieving up to 10.02% reduction in RMSE and 35.67% in Score.

Wahid et al. (2023) propose a self-attention Transformer-based architecture for RUL estimation. The model integrates parallel sensor and time-stamp encoder layers using multi-head self-attention, allowing simultaneous extraction of relevant spatial and temporal features from variable-length input sequences. Evaluated on the newer version of the C-MAPSS dataset (DS03), the model achieves high prediction accuracy and shows robust generalization across units.

2.3. Hybrid Autoencoders

Hybrid autoencoder architectures can significantly improve standalone CNN or RNN architectures by performing feature extraction and reducing high-dimensional sensor data into compact, informative latent representations. This reduces overfitting and can improve model interpretability by providing insight into key degradation patterns.

İnce & Genc (2023) present a joint autoencoder-regressor DNN for RUL prediction, combining a CNN-based autoencoder with an LSTM regressor. Evaluated on the C-MAPSS dataset, their proposed nine-layer architecture demonstrates competitive performance with regards to the RMSE and score of other state-of-the-art methods. Notably, non-linear transformations and piecewise RUL labeling significantly enhance prediction accuracy, achieving up to 50% improvement in scenarios with single-sensor faults. Next to the introduction of a novel model architecture, the study highlights the importance of appropriate RUL labeling strategies and data pre-processing in boosting model performance.

Lu et al. (2020) propose an Autoencoder-Gated Recurrent Unit (AE-GRU) architecture for RUL prediction, where the autoencoder compresses raw sensor data into informative latent features, and the GRU models temporal dependencies to forecast RUL. Experimental results demonstrate that AE-GRU outperforms standard recurrent models such as LSTM and GRU, achieving lower RMSE and faster convergence. Notably, reducing the input dimensionality from 24 to 15 features further improves prediction accuracy, confirming the effectiveness of the autoencoder's feature extraction. The model also benefits from shorter training time.

2.4. Residual Convolutional Neural Networks

Residual Convolutional Neural Networks (ResCNNs) address degradation in deep CNNs by using shortcut connections that stabilize gradient flow, enabling deeper and more efficient training. For RUL prediction, this allows the model to learn complex degradation patterns and robust feature hierarchies from noisy, multivariate sensor data.

Wen et al. (2019) propose an ensemble ResCNN model for RUL prediction, integrating the residual CNN architecture with a k-fold ensemble technique to improve generalization and accuracy. The residual building blocks in their network bypass the convolutional layers using learned transformations, allowing the model to capture complex degradation features in time-series data. Additionally, the ensemble strategy combines multiple models trained on different data splits, reducing overfitting and improving robustness.

Evaluated on all four subsets of the C-MAPSS dataset, the proposed ensemble ResCNN shows strong performance compared to existing deep learning methods. It achieves the lowest RMSE on FD001, FD002, and FD003, and the lowest PHM score on all four subsets. In particular, the RMSE on FD001 is reduced to 12.16 and the score on FD002 to 2087.77, outperforming methods such as LSTM, DBN, and DCNN. These results provide empirical evidence of the benefits of combining residual learning with ensemble strategies for accurate and reliable RUL estimation.

3. Dataset: NASA's C-MAPSS

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset was created by NASA due to the difficulty of obtaining observational data on engine degradation and consists of four separate datasets distinguished by the number of fault modes and operating conditions (OC).

Dataset	Train Shape	Engines	OC	Faults
FD001	(20631, 26)	100	1	1
FD002	(53759, 26)	260	6	1
FD003	(24720, 26)	100	1	2
FD004	(61249, 26)	249	6	2

Table 1. Key characteristics of the four CMAPSS sub-datasets.

FD002 and 004 are generally considered more difficult due to an additional layer of complexity from different OCs. Intuitively, different OCs represent sensor readings taken when the engine is flying at different altitudes, angles or speeds leading to different sensor measurements. Each engine has 21 sensors that provide readings on metrics such as HPC outlet pressure or Physical Fan Speed.

4. Pre-Processing and Data Augmentation

We prepare the data in several steps. Firstly, we de-duplicate highly correlated features, delete constant features and discard other degenerate features based on visual inspection.

Secondly, we standardise the data as the features have different ranges. For datasets FD001 and FD003 with only one set of operating conditions, we standardise each feature across the whole dataset. However, for FD002 and FD004, this fails to equalize the variables' ranges and introduces extreme oscillation as can be observed in Figure 1. Instead, we follow Chadha et al. (2021) and conduct clustering to identify the set of six operating conditions and perform standardisation within each of the clusters. The clustering is necessary since the operating conditions are slightly distorted by additional noise introduced during the simulation process.

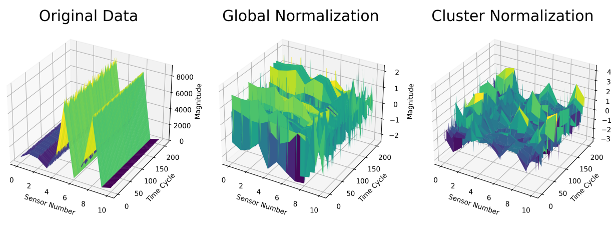

Figure 1. Visualization of original, global-normalized, and cluster-normalized sensor data for FD002.

Thirdly, we introduce data augmentation to improve the variety of samples contained in the training data. In line with the literature on time series data augmentation for deep learning, in particular Iglesias et al. (2023) and Wen et al. (2021), we focus on three established methods: adding random noise, permuting the time series by dropping and interpolating some readings, and time warping (Figure 7).

Random Gaussian Noise adds a random value drawn from the normal distribution to sensor readings. Our variation of *Permutation* replaces some sensor readings with missing values and fills them based on preceding values. *Time Warping* underlays the time series with a warping curve that determines the parts of the sequence to be sped up or slowed down. Subsequently, it interpolates the adjusted sections back to the original time scale, compressing or stretching sections of the time series.

Moreover, we use a sliding window approach to generate samples from different parts of the engines' lifespans. While this also serves as a data augmentation, it importantly ensures input data to the models have a fixed length for all engines and datasets. This approach is standard in the literature on RUL prediction, see for instance Zhang et al. (2022) and Al-Dulaimi et al. (2019).

We test whether the augmentation improves RUL-prediction by comparing RMSE and PHM scores on all four datasets (Figure 6). As can be observed, the procedure substantially improves performance on FD001 and FD003, and offers modest improvement on FD002. On FD004, the augmentation appears to degrade the performance. This is a well-documented issue in the literature; for instance, Iglesias et al. (2023) argue that data augmentation does not fully preserve time dependencies and can therefore introduce invalid samples. Since designing separate augmentation procedures for all four datasets is beyond the scope of the project and the augmentation produces important enhancements for three of four datasets, we take note of this limitation and continue to use the approach for the remainder of our project.

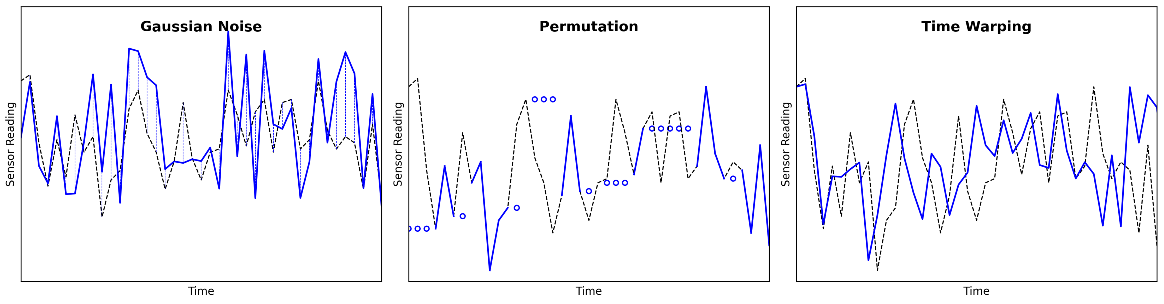

Figure 2. Illustration of data augmentation techniques on sensor `s_11` for engine 1.

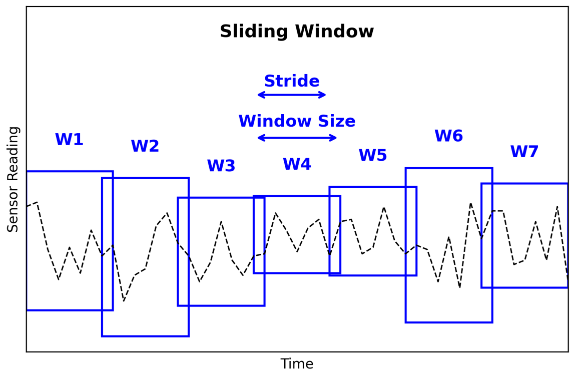

Figure 3. Illustration of sliding window technique.

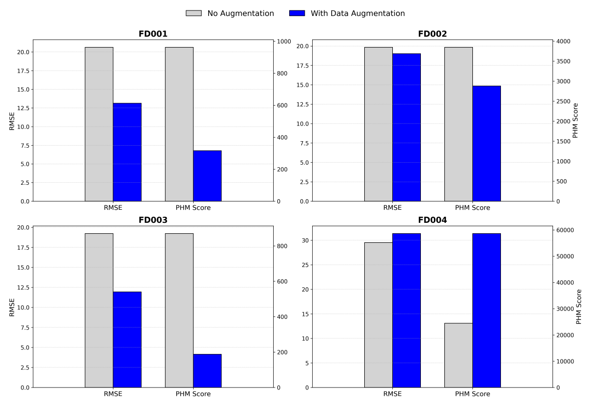

Figure 4. Comparison with no augmentation (grey) and our augmentation (blue) on the four datasets using the Transformer model.

5. Evaluation of Literature-Inspired Models

5.1. Model Architecture

We initially consider 4 models for the RUL prediction problem: (1) Hybrid DNN (2) ResCNN (3) Autoencoder - BiLSTM (4) Transformers. The exact model architecture is shown in diagrams in the Appendix A. These models are in part inspired by the literature review as we wish to use current state-of-the-art methods as an initial attempt before considering improvements. This also grounds our work in the context of current research in this field. We first explain our models and then briefly discuss training parameters.

5.1.1. Hybrid Autoencoder-BiLSTM

The hybrid Autoencoder-BiLSTM architecture is inspired by Lu et al. (2020), combining the strengths of feature extraction and temporal modeling. The model first uses an autoencoder (AE) to reduce the dimensionality of the input features and then passes the encoded representations to a Bidirectional Long Short-Term Memory (BiLSTM) network. The BiLSTM is specifically chosen over simpler recurrent units like GRUs to leverage its ability to capture complex, long-term dependencies in both forward and backward temporal directions, which is critical for degradation sequence modeling.

The model starts with an encoder made of two Conv1D layers. The first convolutional layer consists of 64 filters with a kernel size of 5 and applies ReLU activation, followed by batch normalization and dropout with a rate of 0.3. The second convolutional layer reduces the feature dimension by using fewer filters (latent_dim) and a smaller kernel size of 3, again followed by batch normalization. After the encoder, the decoder reconstructs the input through a Conv1D layer with 64 filters and a second Conv1D layer that restores the original feature size, using a sigmoid activation at the output. Based on the compressed features from the encoder, a Bidirectional LSTM is applied, consisting of two stacked BiLSTM layers with 64 units each, with dropout layers between them to prevent overfitting. The RUL prediction is generated by a final Dense layer with linear activation. The model is trained jointly on predicting the RUL and reconstructing the input, aiming to learn both meaningful feature representations and accurate RUL estimates.

5.1.2. HDNN

The design of the HDNN model draws inspiration from hybrid approaches discussed in the literature. Recognizing the strengths and limitations of both approaches, this model integrates the two ideas: maintaining parallel feature extraction paths while strengthening the temporal modeling through stacked bidirectional LSTMs.

The proposed HDNN architecture consists of two parallel branches. The first branch applies 2D convolutions with small kernel sizes ((3,1) and (5,1)) along the sensor feature dimension. This design enables localized feature extraction across sensors while maintaining temporal context. Batch normalization and tanh activations are used to stabilize training and introduce non-linearity, followed by max-pooling and flattening to downsample the learned representations. The second branch processes the raw input directly through two stacked bidirectional LSTM layers with 64 units each. The use of bidirectional LSTMs enables the model to capture both forward and backward temporal dependencies.

The outputs from the CNN and BiLSTM branches are concatenated, combining local spatial patterns and long-term temporal features. This merged vector is passed through two fully connected layers (64 and 32 neurons) and a final dense layer with a ReLU activation generates RUL prediction. Compared to previous hybrid architectures, this model enhances the temporal modeling capacity through the use of stacked Bidirectional LSTMs. The parallel architecture allows the HDNN to model engines operating under diverse conditions and varying degradation behaviors.

5.1.3. ResCNN

The model uses 1D convolutional layers to extract temporal patterns and local dependencies within the sensor data sequences. Residual connections are incorporated to allow deeper network training and mitigate vanishing gradient issues. These residual blocks preserve important features from earlier layers, improving learning stability and performance. The architecture includes batch normalization and dropout layers to enhance generalization and reduce overfitting. As the signal flows through the network, it is downsampled and transformed into high-level features. A global average pooling layer is applied to condense these features into a compact representation. In each residual block, the main path applies multiple convolutions and non-linear transformations, while the shortcut path allows the input to bypass these operations and be added directly to the output. This ensures that the original signal characteristics are preserved and helps the network learn identity mappings when deeper transformations are unnecessary.

5.1.4. Transformer Architecture

The architecture follows a DAST-inspired two-branch structure (Zhang et al., 2022), with a time- and sensor-dimension. Both branches apply sinusoidal positional encoding (Irani & Metsis, 2025) to enable the transformer model to perceive feature values in sequential context, allowing for long-term dependencies and interactions between different sensors (e.g. temperature and pressure). The core model uses multi-head attention layers, residual connections to prevent gradient vanishing and dropout layers against overfitting. Moreover, we use longer windows of size 50 due to the transformers' better capacity to handle long sequences (Li et al., 2022).

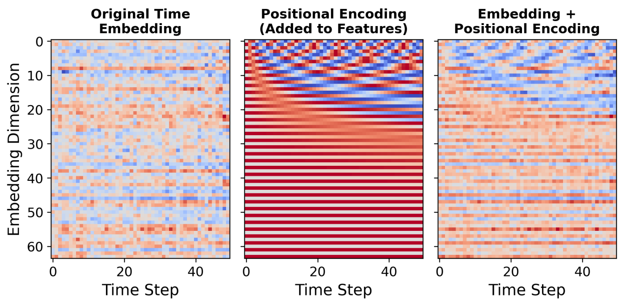

Figure 5. Effect of sinusoidal positional encoding.

Moreover, the model employs cross-domain attention to link the two branches. This mechanism enables interactions between temporal and sensor-based patterns, e.g. distinguishing between signals or anomalies. For example, a spike in the readings of sensor 13 at time period 5 might be an anomalous malfunction, but that same spike in time period 50 might signal the imminent engine breakdown.

The model then applies global average and maximum pooling to summarize overall trends and the most dominant signal from a sequence. The combined feature vector is passed through a prediction block consisting of two dense layers, each followed by batch normalization and dropout to stabilize training and prevent overfitting. A single dense layer neuron outputs the model's RUL prediction.

5.2. Training Methodology

5.2.1. Tracked Metrics

We train with RMSE as it is the standard practice in RUL prediction problems.

The RMSE: This metric is used to evaluate prediction accuracy of the RUL. It is commonly used as a performance measure since it gives equal weights for both early and late predictions. The formulation of the RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{M_{\text{te}}}\sum_{i=1}^{M_{\text{te}}} h_i^2}. \quad (1)$$

We also report the PHM Score, which is the canonical function employed by the International Conference on Prognostics and Health Management (PHM08) Data Challenge:

$$S = \sum_{i=1}^{M_{\text{te}}} s_i, \quad (2)$$

$$s_i = \begin{cases} \exp\left(\frac{h_i}{13}\right) - 1, & \text{for } h_i < 0, \\ \exp\left(\frac{h_i}{10}\right) - 1, & \text{for } h_i \geq 0, \end{cases} \quad (3)$$

where S is the computed score, M_{te} is the total number of testing data samples, and $h_i = \widehat{\text{RUL}}_i - \text{RUL}_i$ (estimated RUL - true RUL, with respect to the i-th data point). The characteristic of this scoring function leans towards early predictions (i.e., the estimated RUL value is smaller than the actual RUL value) more than late predictions (i.e., the estimated RUL is larger than the actual RUL value) as delayed predictions and therefore delayed maintenance poses the risk of severe outcomes such as engine or system failure.

5.2.2. Piecewise-Linear RULs

A critical aspect of the training strategy involves the adoption of a piecewise-linear representation for the RUL target function. During the initial phase of system operation, the degradation process is assumed to be negligible, and the RUL is held constant to reflect the system's healthy condition. Once a predefined "hinge point" is reached, the RUL decreases linearly to capture the progressive deterioration of the system. This modeling choice reduces the tendency of predictive models to overestimate RUL during the early operational phase. Furthermore, the piecewise-linear formulation offers a pragmatic and robust framework in scenarios where detailed knowledge of the system's degradation dynamics is either unavailable or difficult to model accurately.

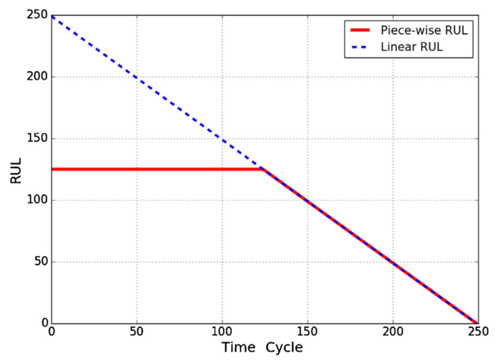

Figure 6. Illustration of piece-wise linear RUL target function.

5.2.3. Mitigating Overfitting

A learning rate scheduling mechanism was applied through the *ReduceLROnPlateau* callback, which monitored the validation loss and reduced the learning rate by a factor of 0.5 if no improvement was observed for 5 consecutive epochs, with a minimum threshold set at 10^{-6}. In addition, an early stopping criterion was adopted, halting training if the validation loss did not improve for 10 epochs and restoring the model weights corresponding to the best validation performance. Dropout layers with rates between 0.2 and 0.5 were incorporated into the model architectures to further regularize training. Certain layers were also equipped with L_2 regularization to penalize large weights and encourage generalization. Finally, each model was trained using 5-fold cross-validation, and predictions across folds were averaged, thereby reducing variance and improving robustness.

5.2.4. Different Optimizers

We experimented with five different optimizers (Adam, SGD, RMSprop, Adagrad, and Adadelta) to evaluate their impact on the ResCNN model using the FD001 CMAPSS dataset. The comparative analysis revealed that Adam outperformed the rest. Adam probably did better due to its combination of momentum and adaptive learning rate properties, incorporating both first and second moments of the gradients. This approach is particularly effective for RUL prediction tasks in the NASA CMAPSS dataset where the error surface may contain both steep and flat regions.

5.3. Model Results

		HDNN	ResCNN	BiLSTM	Trans
FD001	RMSE	13.1	14.6	13.8	**12.7**
	Score	307	411	300	**226**
FD002	RMSE	22.1	**21.2**	22.4	21.7
	Score	8149	**1861**	8500	5164
FD003	RMSE	11.9	16.0	**11.23**	12.2
	Score	189	708	**170**	199
FD004	RMSE	**28.7**	29.3	29.78	33.5
	Score	27919	**6354**	34500	95000

Table 2. Performance of initial models

The results in table 2 show that no single model performs best across all four FD datasets. On FD001, the Transformer model performs best, indicating strong accuracy and early failure prediction in a low-complexity setting. On FD002, ResCNN performs best, suggesting that its local feature extraction ability handles the multiple operating conditions effectively. For FD003, BiLSTM outperforms all other models, likely due to its ability to capture long-range dependencies in the presence of dual fault modes. On FD004, which combines multiple operating conditions and fault types, HDNN achieves the lowest RMSE while ResCNN obtains the lowest score. This suggests that hybrid models like HDNN can generalize well under complexity, but ResCNN's predictions may align more closely with early warnings.

Overall, the results confirm that each architecture has strengths in specific settings, driven by fault type and operational variability. This motivates the use of an ensemble method to leverage these complementary capabilities in the next section.

6. Proposed Ensemble Method

6.1. Model Architecture

Three model architectures capturing different patterns or relationships are trained: (1) Autoencoder + BiLSTM (2) Autoencoder + ResCNN and (3) Autoencoder + HDNN.

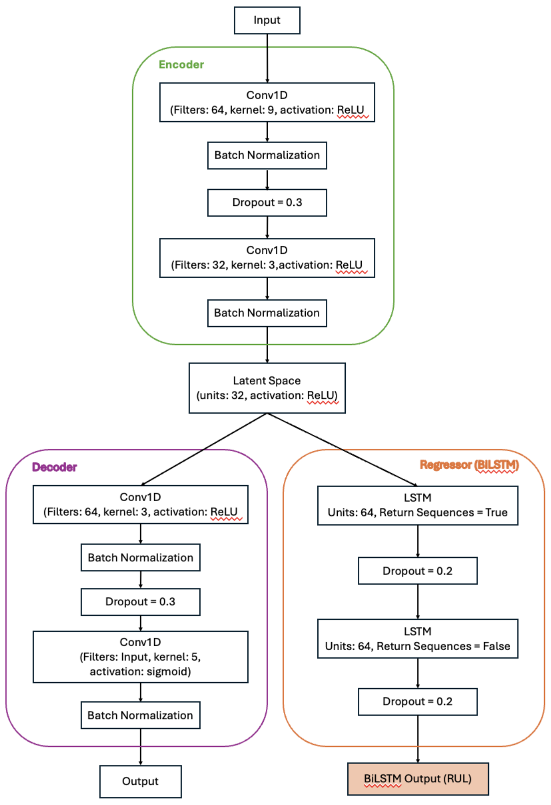
Input
Encoder
Conv1D
(Filters: 64, kernel: 9, activation: ReLU
Batch Normalization
Dropout = 0.3
Conv1D
(Filters: 32, kernel: 3,activation: ReLU
Batch Normalization
Latent Space
(units: 32, activation: ReLU)
Decoder
Conv1D
(Filters: 64, kernel: 3, activation: ReLU
Batch Normalization
Dropout = 0.3
Conv1D
(Filters: Input, kernel: 5, activation: sigmoid)
Batch Normalization
Output
Regressor (BILSTM)
LSTM
Units: 64, Return Sequences = True
Dropout = 0.2
LSTM
Units: 64, Return Sequences = False
Dropout = 0.2
BiLSTM Output (RUL)

Figure 7. Model Architecture for one of the 3 methods

We present here a visualisation of the architecture of the Autoencoder-BiLSTM. The part of the architecture which is encircled as "Regressor" is replaced with the HDNN or the ResCNN for the other two parts of the ensemble. We also present model architectures for the ResCNN and HDNN models in the Appendix A.

The ensemble is constructed via three major steps:

- Each model architecture is trained using 5-fold cross-validation (to prevent overfitting) and 5 separate models are trained (with different train/test splits).
- For each architecture, the 5 models' predictions are averaged to get one single prediction per architecture.
- The three architecture-level predictions are combined using a weighted average with weights inversely proportional to each individual model's prediction RMSE.

We combine the predictions from different models as such:

$$w_i = \frac{1/\text{RMSE}_i}{\frac{1}{\text{RMSE}_1} + \frac{1}{\text{RMSE}_2} + \frac{1}{\text{RMSE}_3}} \quad \text{for } i \in \{1, 2, 3\}$$

$$\begin{aligned}\text{Ensemble Predict} = {} & w_1 \times \text{BiLSTM prediction} \\ & + w_2 \times \text{ResCNN prediction} \\ & + w_3 \times \text{HDNN prediction}\end{aligned}$$

Our rationale is that BiLSTMs are good for sequential dependencies, while ResCNNs are good for local feature extraction and the HDNN captures hierarchical relations across features and by combining different models in an ensemble we should improve the robustness and improve generalisation capabilities.

We do not include the Transformer model in the ensemble for two reasons. Firstly, it is already an encoder-decoder architecture which would therefore duplicate the ensemble setup. Secondly, our transformer model is relatively complex and expensive to train, which would increase the computational cost of the ensemble further.

6.2. Model Results

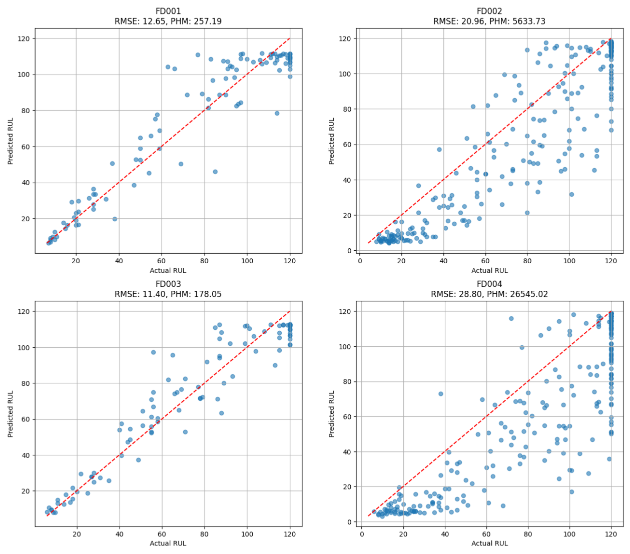

Figure 8. True test RUL compared against predicted RUL. Each dot represents one engine in the respective test set.

Figure 8 shows that the ensemble model achieves the lowest RMSE on FD001 (12.65) and FD002 (20.96), outperforming the best individual models—Transformer (12.7) and ResCNN (21.2), respectively. On FD003 and FD004, while not the top performer, the ensemble stays within 1–5% of the best RMSE, indicating strong consistency across varying fault modes and operating conditions. However, while the RMSE remains low on FD004 (28.8), the corresponding PHM score (26,545) indicates that a few overestimates significantly impact the score due to the exponential penalty. This highlights the importance of loss functions that penalize late predictions more explicitly during training.

We also highlight its tendency to slightly underestimate RUL as can be seen by more predictions below the red dotted line in Figure 8. This makes it better aligned with safety-critical applications where early maintenance is preferred over late interventions and models that err on the side of caution are likely better.

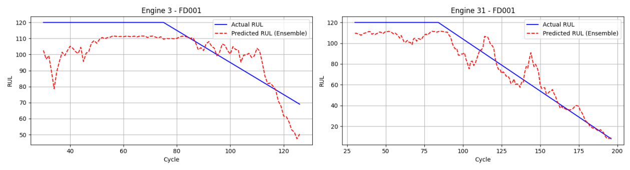

Figure 9. True piece-wise vs predicted RUL of test engines.

Additionally we contextualise our ensemble model alongside other state-of-the-art architectures in Table 3 and notice that our proposed architecture is an outperformer in datasets with less operational settings. Furthermore to sense-check our models and improve interpretability we have output a feature importance plot in appendix B.

7. Future Extensions

Based on the model's performance and observed behaviors across datasets, the following limitations of our approach have been identified. These also highlight opportunities for future work to further improve RUL prediction models.

Incorporating Asymmetric Loss Functions: The current models are trained using the symmetric RMSE loss, which penalizes overestimation and underestimation of RUL equally. In real-world prognostics, overestimating the RUL is typically more critical, as it increases the risk of unexpected failures. Adopting an asymmetric loss function, such as an asymmetric mean squared error (MSE) that applies a higher penalty to overestimations, could improve model reliability by favoring more conservative RUL predictions aligned with practical safety requirements.

Hyperparameter Tuning for Piecewise-Linear RUL Curves and Window Size: The current preprocessing uses a fixed hinge point for generating piecewise-linear RUL targets and a fixed window size for the input sequences. However, the optimal hinge point and window size may vary depending on the operational conditions. Systematic tuning of these hyperparameters could better capture the degradation process and improve model learning.

Ensemble Weighting Based on Operational Complexity: The ensemble method currently combines model predictions through a weighted average based on validation RMSE. However, our results suggest that different models excel under different operational conditions. Future work could focus on developing a complexity-aware dynamic weighting mechanism. For instance, clustering engines by operating conditions (e.g., load, speed) and assigning higher weights to the models that historically perform better in similar conditions can improve accuracy.

8. Conclusion

This paper presents a comprehensive evaluation of four deep learning architectures for Remaining Useful Life (RUL) prediction on the NASA C-MAPSS dataset, followed by the development of an ensemble model integrating their complementary strengths. Our results demonstrate that no single model consistently outperforms the others across all subdatasets: the Transformer performs best on FD001, ResCNN on FD002, BiLSTM on FD003, and HDNN on FD004. These findings support our ensemble approach, which combines Autoencoder-based variants of HDNN, BiLSTM, and ResCNN to enhance robustness and generalization.

While the ensemble does not always surpass the top-performing individual model on each subdataset, it consistently reduces performance variance across datasets and achieves competitive results overall. This supports the practical utility of ensemble learning in predictive maintenance scenarios with heterogeneous operating conditions.

Table 3. Performance comparison of 7 methods based on C-MAPSS datasets.

		Proposed Model	DCNN (Li et al., 2018)	LSTMBS (Liao et al., 2018)	MODBNE (Zhang et al., 2017)	HDNN (Al-Dulaimi et al., 2019)	GB (Zhang et al., 2017)	D-LSTM (Zheng et al., 2017)
FD001	RMSE	**12.6**	12.6	13.3	15.0	13.0	15.7	16.1
	Score	257	274	**216**	334	245	474	338
FD002	RMSE	21.0	22.4	22.9	25.1	**15.2**	29.1	24.5
	Score	5634	10412	2796	5585	**1282**	87280	4450
FD003	RMSE	**11.4**	12.6	16.0	12.5	12.2	16.8	16.2
	Score	**178**	284	317	422	1596	577	852
FD004	RMSE	28.8	29.3	24.3	28.7	**18.2**	29.0	28.2
	Score	26545	12466	**3132**	6558	1527	17818	5550

References

Al-Dulaimi, A., Zabihi, S., A., A., and Mohammadi, A. Hybrid deep neural network model for remaining useful life estimation. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3872–3876. IEEE, May 2019. doi: 10.1109/ICASSP.2019.8683763.

Al-Dulaimi, A., Zabihi, S., Asif, A., and Mohammadi, A. A multimodal and hybrid deep neural network model for remaining useful life estimation. *Computers in Industry*, 108:186–196, 2019. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2019.02.004. URL https://www.sciencedirect.com/science/article/pii/S0166361518304925.

Chadha, G. S., Panara, U., Schwung, A., and Ding, S. X. Generalized dilation convolutional neural networks for remaining useful lifetime estimation. *Neurocomputing*, 452:182–199, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.04.109. URL https://www.sciencedirect.com/science/article/pii/S0925231221006676.

Iglesias, G., Talavera, E., Ángel González-Prieto, Mozo, A., and Gómez-Canaval, S. Data augmentation techniques in time series domain: a survey and taxonomy. *Neural Computing and Applications*, 35(14):10123–10145, 2023. ISSN 1433-3058. doi: 10.1007/s00521-023-08459-3. URL https://doi.org/10.1007/s00521-023-08459-3.

İnce, K. and Genc, Y. Joint autoencoder-regressor deep neural network for remaining useful life prediction. *Engineering Science and Technology, an International Journal*, 41:101409, 2023.

Irani, H. and Metsis, V. Positional encoding in transformer-based time series models: A survey, 2025. URL https://arxiv.org/abs/2502.12370.

Li, X., Ding, Q., and Sun, J. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018. ISSN 0951-8320. doi: 10.1016/j.ress.2017.11.021.

Li, X., Li, J., Zuo, L., Zhu, L., and Shen, H. T. Domain adaptive remaining useful life prediction with transformer. *IEEE Transactions on Instrumentation and Measurement*, 71:1–13, 2022. doi: 10.1109/TIM.2022.3200667.

Li, Z., He, Q., and Li, J. A survey of deep learning-driven architecture for predictive maintenance. *Engineering Applications of Artificial Intelligence*, 133:108285, 2024.

Liao, Y., Zhang, L., and Liu, C. Uncertainty prediction of remaining useful life using long short-term memory network based on bootstrap method. In *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–8, 2018. doi: 10.1109/ICPHM.2018.8448818.

Lu, Y.-W., Hsu, C.-Y., and Huang, K.-C. An autoencoder gated recurrent unit for remaining useful life prediction. *Processes*, 8(9):1155, 2020.

Wahid, A., Yahya, M., Breslin, J. G., and Intizar, M. A. Self-attention transformer-based architecture for remaining useful life estimation of complex machines. *Procedia Computer Science*, 217:456–464, 2023.

Wen, L., Dong, Y., and Gao, L. A new ensemble residual convolutional neural network for remaining useful life estimation. *Mathematical Biosciences and Engineering*, 16(2):862–880, 2019. doi: 10.3934/mbe.2019040.

Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., and Xu, H. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, IJCAI-2021, pp. 4653–4660. International Joint Conferences on Artificial Intelligence Organization, August 2021. doi: 10.24963/ijcai.2021/631. URL http://dx.doi.org/10.24963/ijcai.2021/631.

Zhang, C., Lim, P., Qin, A., and Tan, K. C. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2306–2318, 2017. doi: 10.1109/TNNLS.2016.2582798.

Zhang, Z., Song, W., and Li, Q. Dual-aspect self-attention based on transformer for remaining useful life prediction. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022.

Zhao, R., Yan, R., Wang, J., and Mao, K. Learning to monitor machine health with convolutional bi-directional lstm networks. *Sensors*, 17(2):273, 2017. doi: 10.3390/s17020273.

Zheng, S., Ristovski, K., Farahat, A., and Gupta, C. Long short-term memory network for remaining useful life estimation. In *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95, 2017. doi: 10.1109/ICPHM.2017.7998311.

A. Model Architectures

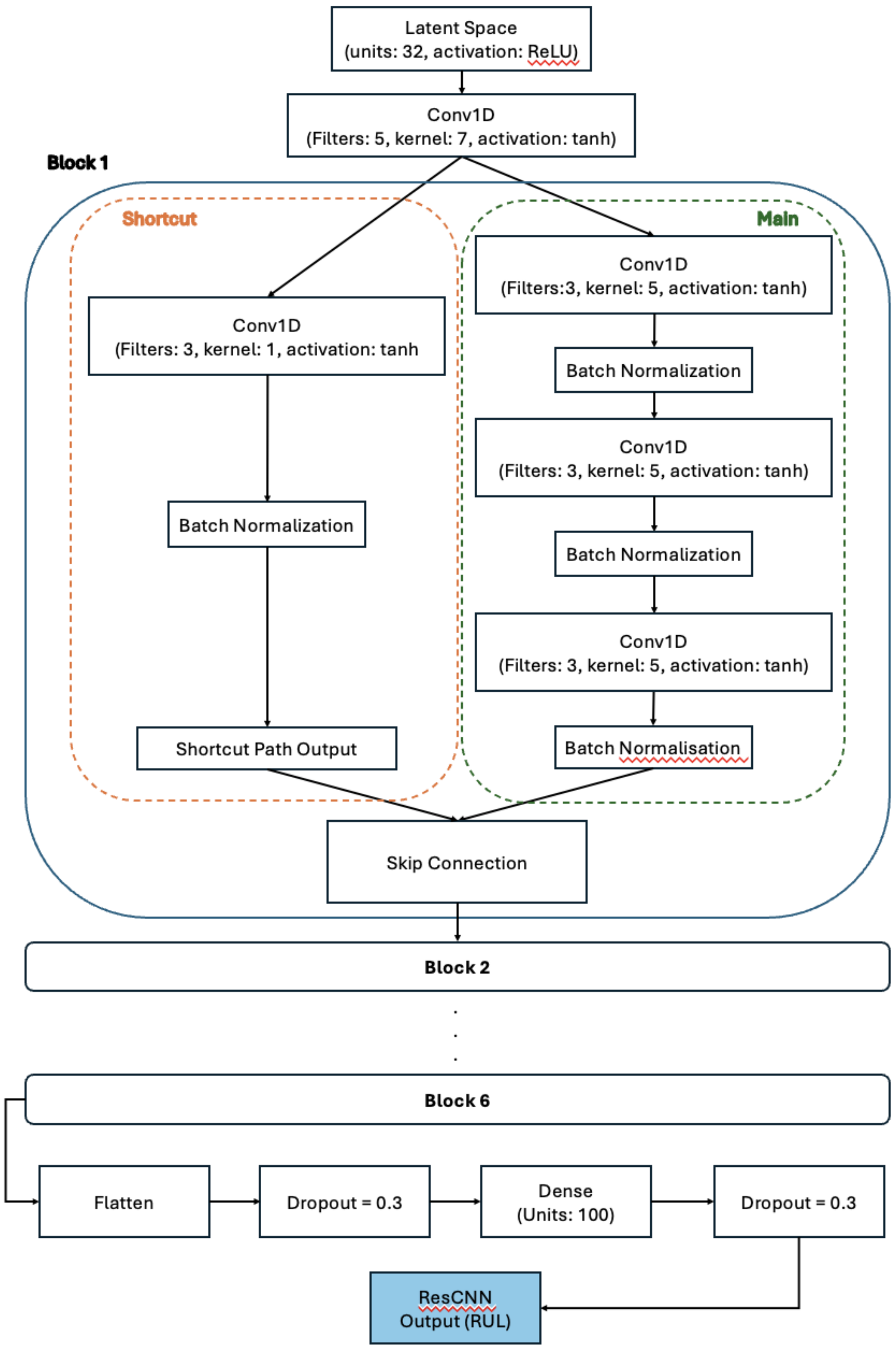

Figure 10. Illustration of a ResCNN block

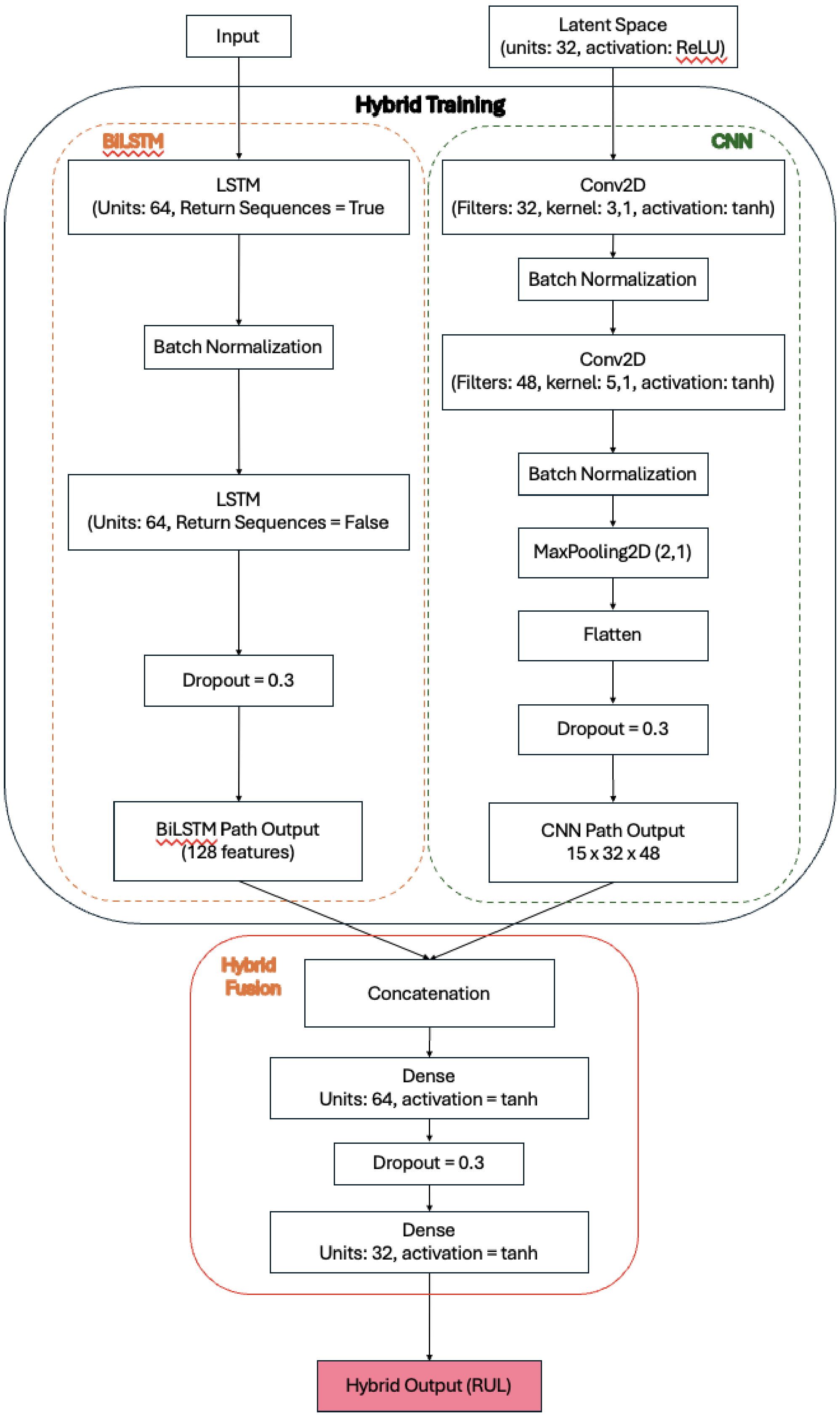

Figure 11. Illustration of the HDNN model

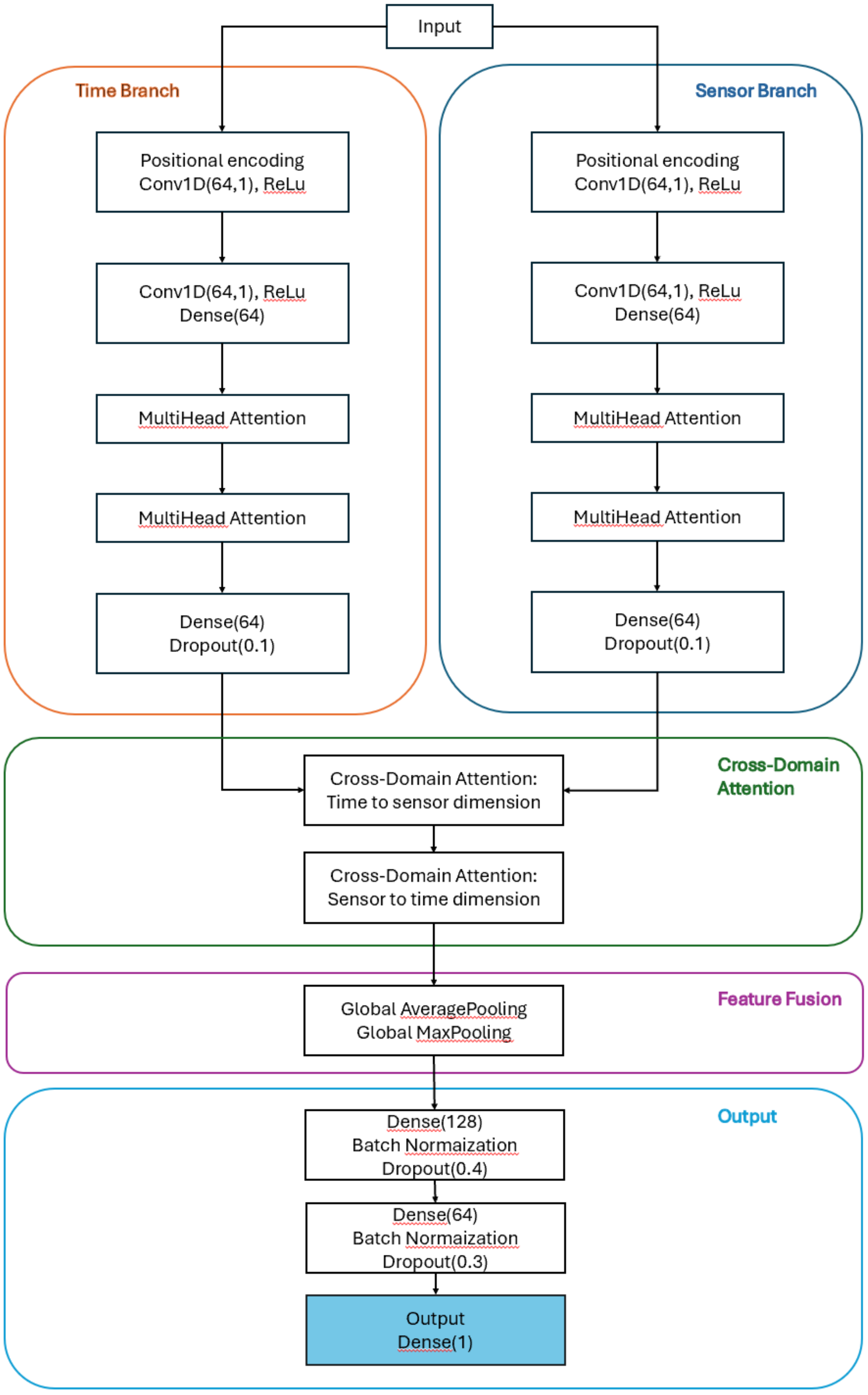
Input
Time Branch
Sensor Branch
Positional encoding Conv1D(64,1), ReLu
Conv1D(64,1), ReLu Dense(64)
MultiHead Attention
MultiHead Attention
Dense(64) Dropout(0.1)
Positional encoding Conv1D(64,1), ReLu
Conv1D(64,1), ReLu Dense(64)
MultiHead Attention
MultiHead Attention
Dense(64) Dropout(0.1)
Cross-Domain Attention
Cross-Domain Attention: Time to sensor dimension
Cross-Domain Attention: Sensor to time dimension
Feature Fusion
Global AveragePooling Global MaxPooling
Output
Dense(128) Batch Normaization Dropout(0.4)
Dense(64) Batch Normaization Dropout(0.3)
Output Dense(1)

Figure 12. Illustration of the Transformer model

B. Feature Importance Plot

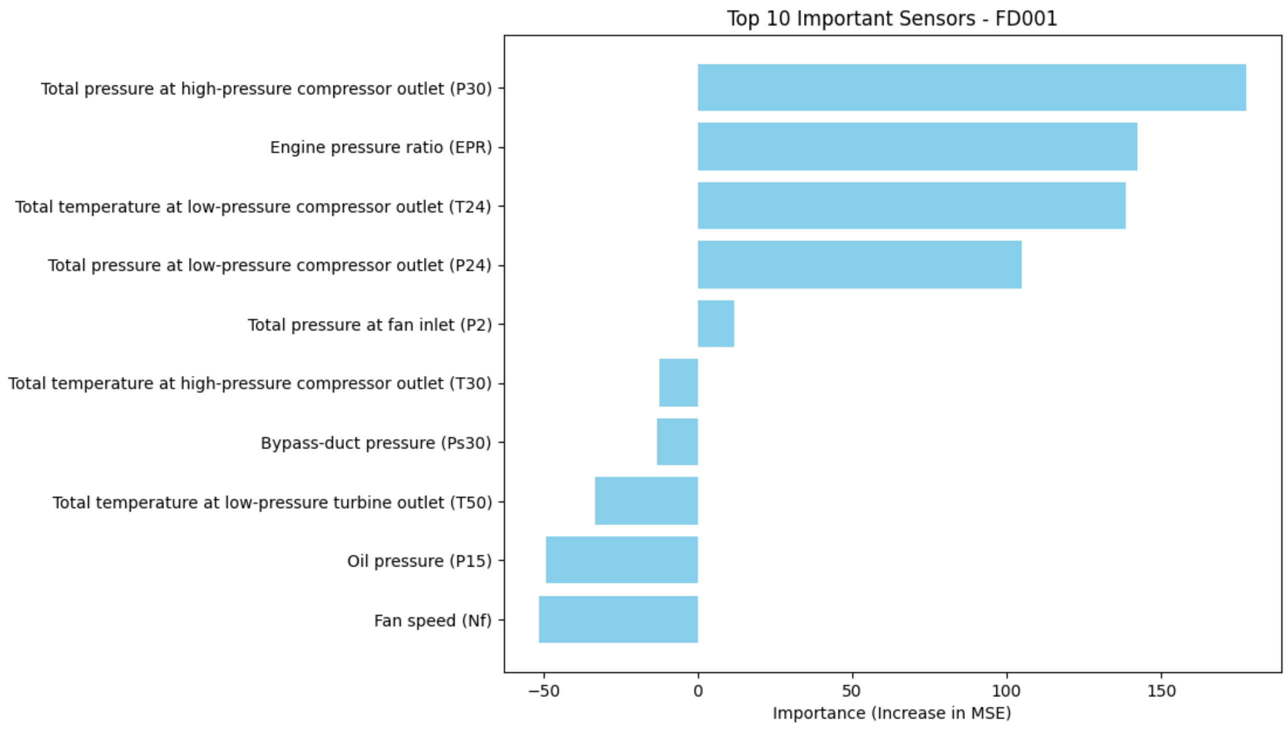

Figure 13. This chart provides intuition to engineers, not to us deep learning enthusiasts!

C. Statement of Individual Technical Contributions

The work for this paper was divided equally between all group members. Each group member was assigned the following technical tasks specifically:

- **Candidate 50250:**
  - Researched background literature extensively and decided which models we should pursue
  - Build, train and evaluate HDNN model
- **Candidate 49308:**
  - Build, train and evaluate Hybrid Autoencoder-BiLSTM model
  - Build, train and evaluate Ensemble model
- **Candidate 45733:**
  - Build, train and evaluate Transformer model
  - Preprocessing and Data augmentation
- **Candidate 49580:**
  - Build, train and evaluate ResCNN model
  - Preprocessing and Data augmentation