Project work

Regression - Metal Working Fluid 1
Classification - FFT Classification

Muhammed Ali Çelikkol

September 2025

Contents

List of Figures

List of Tables

1 Introduction

In this paper two different Machine Learning Projects, both involve applying and evaluating modern data-driven methods on technical measurement datasets, are represented. Both projects aim to identify patterns in complex data and develop predictive models using Machine Learning techniques.

The first project focuses on predicting the distance between a sensor and a drilling location based on measured data. As features, numerous frequency coefficients are available. Various preprocessing techniques are applied to data-set to investigate whether model accuracy can be improved. Aside from that essential preprocessing steps are applied to be able to train Neural Networks.

The Second Project addresses the problem of classification of damage states in carbon fiber reinforced polymer (CFRP) materials, based on FFT data. The task includes five different labels.

In this work, two different approaches are implemented and compared: (i) a two-stage approach that first separates no damage and the damaged samples and then classifies the type of damage, and (ii) a single-stage classifier that directly assign labels to classes in one step. Different modern machine learning models such as multilayer perceptrons(MLPs) and convolutional neural networks (CNN) were investigated.

The objective of this comparison is to evaluate whether the two-stage approach provides a measurable benefit over a single-stage classifier in terms of classification metric such as accuracy. The results are discussed with respect to model complexity, computational cost, and generalization performance.

2 Metal Working Fluid 1

2.1 Dataset and Preprocessing

The dataset for the *Metal Working Fluid 1* project was structured in multiple folders corresponding to different subsets. In order to evaluate the final model, two folders were randomly selected as the validation and test sets, respectively. The remaining data is used for training the models. This random data split technique ensured that the validation and test sets represent unseen data and include samples from every distance, providing an unbiased measure for the final model evaluation.

Standardization of Features. As the first step of training the neural network models, the input features were standardized using the matlab's `zscore`. The mean (μ) and standard deviation (σ) were computed exclusively from the training set to avoid data leakage. Validation and test data were also standardized from these training data statistics.

$$X' = \frac{X - \mu}{\sigma} \tag{2.1}$$

Why it helps:

- Neural networks tend to converge faster when input features are centered and scaled [?]
- Standardization prevents exploding or vanishing gradients during backpropagation [?]
- It ensures that features on different scales contribute fairly to the weight updates.

Observation: The standardization of input features noticably improved convergence and training stability. In contrast, when the same procedure was applied to the response variables, performance tend to degrade, resulting in higher loss.

Principal Component Analysis. On top of previous feature pre-processing , Principal Component Analysis (PCA) was applied as a dimensionality reduction step. PCA transforms the high-dimensional feature space into a set of orthogonal principal components, ordered by the amount of variance they are explaining [Jol02].

Why it helps:

- PCA removes collinearity among features, making learning more efficient.
- It reduces input dimensionality, which accelerates model training and lowers computational load[Jol02].
- It emphasizes the most informative features, filtering out noise.

Observation: Using PCA prior to the Multilayer Perceptron (MLP) resulted in improved performance compared to the raw-feature MLP. the PCA + MLP combination achieved prediction quality on par with a convolutional neural network trained on raw features, while requiring substantially less computational load. There is also some features that have so little variance. These features can be eliminated safely.

Additional Preprocessing: Variance and Correlation Screening. To reduce noise and remove non-informative inputs prior to modeling, two lightweight filters were applied on the *training set only*:

(i) Near-constant features. Columns with (sample) variance below a small threshold were removed:

$$\mathrm{var}(X_{\mathrm{train},j}) < \tau_{\mathrm{var}}.$$

(ii) Univariate correlation with the target. Remaining features were screened by the absolute Pearson correlation with the response:

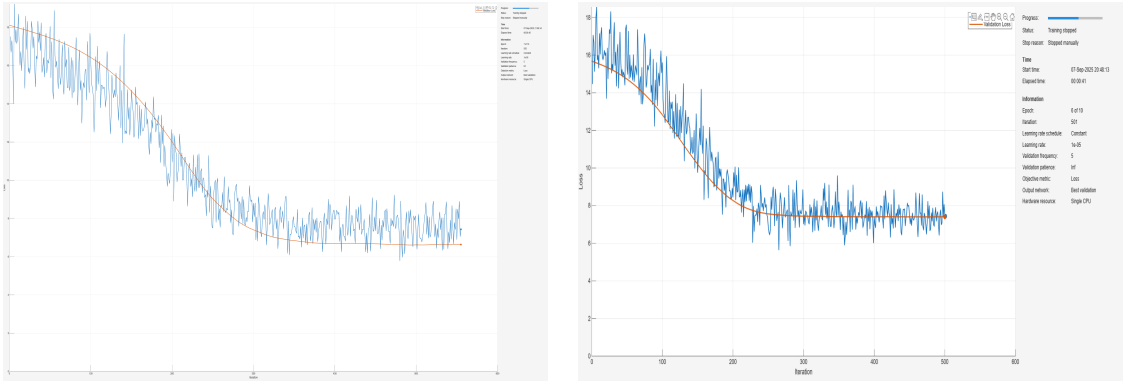
$$|\,\mathrm{corr}(X_{\mathrm{train},j},\, y_{\mathrm{train}})\,| \geq \tau_{\mathrm{corr}}.$$

This keeps features that show at least weak linear association with the target and discards clearly uninformative inputs.

Though, it should be noted that, these additional preprocessing steps only applied to see whether all features necessary or not. It is not used in training data set that has been feed to Neural Networks.

In addition, it was observed that not all subsets contributed equally to the training process. It has been seen samples from the `m21` and `m19` folders did not significantly improve training, suggesting that these subsets contain little useful information for training. Therefore, these subsets are suitable to be utilized as validation and test data. In contrast, samples

from the `m17` folder contributed significantly to the learning process, as reflected by a steeper decrease in the training loss curve. When these subsets are spared as training dataset, final prediction ability decreases strongly. This indicates that certain parts of the data contain more informative patterns.



(a) M19 - M21 (b) M15 - M17

Figure 2.1: Training loss curves for different subsets. Raw MLP benefits significantly from informative subsets like `m17` and `m15`.

2.2 Model Architectures

In order to evaluate the predictive potential of different Neural Network types, three distinct model architectures were implemented : A baseline multilayer Perceptron trained solely on raw features, an MLP trained on PCA transformed features, and a convolutional neural network(CNN) trained on raw features. In the following part, each model discussed.

Baseline MLP. The first model consist of a Deep MLP trained directly on the standardized raw input features. Fully connected layers stacked with gradually decreasing width. At each layer ReLU activation function and dropout layers were applied. The dropout rates were tuned per layer (ranging between 0.1 and 0.6) to prevent overfitting. Including this model, trainings were performed using the Adam optimizer with Huber loss and L2 regularization.

PCA + MLP. To reduce dimensionality and eliminate redundancy, after zscore, Principal component Analysis was applied to the whole dataset before spliting the data. The MLP architecture was same. Input dimension was substantially reduced (from 22507 to 5431) while preserving %95 variance in the dataset. The PCA + MLP method achieved

improved convergence behavior compared to the raw - feature MLP. Furthermore, due to lower input dimensionality, training was more computationally efficient. The empirical results showed that this combination reached performance comparable to Convolutional Neural Networks, while being faster to train.

Convolutional Neural Network (CNN). As comparison to baseline, a Convolutional Neural Network was trained directly on the raw input features. The CNN architecture consisted of several Convolutional layers with filter size of 7×1, each followed by an activation function (ReLU), max-pooling layers and a global average pooling layer. This architecture was designed to capture local correlations withing the feature vector. Despite its predictive performance, the CNN required more computational resources and longer training times compared to the PCA + MLP approach.

The use of convolutional architectures was inspired by related work in acoustic sensor networks, where CNNs were successfully applied for geometry calibration tasks [GSB+21].

Random Forest (RF). Aside from Neural Networks, a Random Forest regressor was trained as a classical Machine Learning baseline. The RF was trained on standarized input features, with MaxNumSplits 512, MinLeafSize of 10 and 1200 numberLearningCycles. Hyperparameters are tuned empirically. Model provided a strong non-linear baseline and required substantially less tuning compared to above given Neural Network methods.

2.3 Evaluation Metrics and Results

The performance of the proposed models were evaluated on both the validation and test sets using three standard regression metrics: root mean squared error(RMSE), mean absolute error(MAE), and the coefficient of determination (R^2). Metrics are defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.2}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{2.4}$$

Results Overview. The baseline MLP trained on raw features achieved low loss. Standardization of input features was crucial for stable training, but standardizing the response variable degraded performance visibly. Applying PCA prior to the MLP reduced dimensionality and improved convergence speed . The PCA + MLP model remarkably reached performance comparable to CNN, while being significantly faster to train. The Convolutional Neural Networks achieved also a strong predictive accuracy but has a higher computational cost. Seeing that Random Forest method beats all of the Deep Neural Networks proves that with limited data classical Machine Learning methods still performs better.

Quantitative Comparison. Table 2.1 summarizes the test results for all models. The PCA+MLP achieved a similar MAE to the CNN but required only a fraction of the training time. The raw-feature MLP performed strongest, between Neural Network architectures, but with limited data they are not good enough to explain variance. Classical Machine Learning methods are still better on it.

Table 2.1: Performance comparison of different model architectures. Best results per column are highlighted in bold. Results are obtained from test data.

Model	**RMSE**	**MAE**	R^2
Random Forest	**5.8445**	**5.0830**	**0.3880**
MLP (raw features)	7.8825	5.9983	0.0057
PCA + MLP	8.6979	6.9178	-0.3556
CNN (raw features)	8.3721	7.1748	-0.1216

3 FFT Classification

3.1 Dataset and Preprocessing

The dataset of Project-2 consists of FFT features extracted from CFRP measurements, labeled into five different damage classes. For the first approach (Two-stage classifier), the labels were first converted into a binary classification problem, where No damage samples(Noise) were labeled as 0 and all damaged states were labeled as 1. After samples are passed through first classifier, predicted samples are fed into second classifier but before feeding them to second classifier, the falsely as damage predicted Noise samples are deleted.

Mutual Information Feature Relevance. To assess the informativeness of each feature, the mutual information (MI) between features and the labels was computed, as recommended in information-theoretic feature selection frameworks [BPZL12], using Matlab's `fscmrmr` function. MI helps to capturing nonlinear relationships between individual features and target variable. All scores lie in a narrow range $(0.3 - 0.6)$, which indicates most features carry some amount of predictive signal but none of them can be considered as a "super-feature".

Variance Analysis. Feature variances were computed across the training set to identify near-constant features, which are known to degrade learning performance [PVG+11].

$$v_j = \mathrm{var}(X_{\mathrm{train},j}),$$

where j indexes are features. The histogram in the Figure 3.1 shows that most features have very small variance, suggesting that many features are nearly constant across samples and may require careful normalization to avoid exploding feature values.

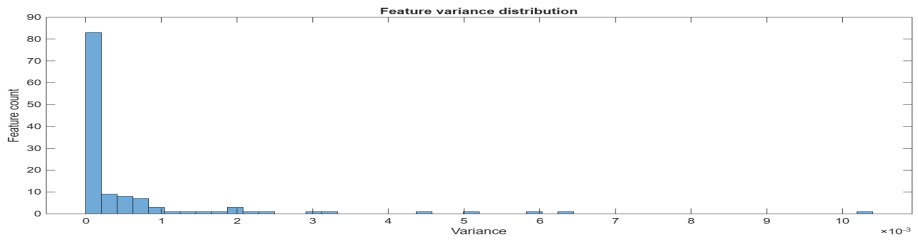


Figure 3.1: Distribution of feature variances across the training set. Most features exhibit very small variance.

Custom Feature Standardization. Instead of Matlab's built-in `zscore` function, a custom feature standardization was applied to prevent exploding values,a recommended practice to stabilize gradient-based optimization in neural networks [**?**], in near-constant features. For each feature j the mean μ_j and standard deviation σ_j were computed on the training data, and as precaution a minimum standard deviation floor $\sigma_{\min} = 10^{-3}$ was enforced:

$$X'_{ij} = \frac{X_{ij} - \mu_j}{\max(\sigma_j, \sigma_{\min})}.$$

With the help of clamping, division by extremely small $sigma_j$ values is avoided. Otherwise noise would be amplified and destabilize Neural Network Training. The same normalization procedure is applied to validation set to ensure a consistent data transformation.

3.2 Model Architectures

In order to classify CFRP damage states, two different Neural Network architectures were investigated: a Convolutional Neural Network and a Multilayer Perceptron approach. The same preprocessed data was fed into models. Their design and performance are described below.

Convolutional Neural Network(CNN). The CNN architecture was designed to catch local feature correlations in the input. The network consisted of three CNN blocks with filter size 7×1, each followed by batch normalization, ReLU activation, and dropout for regularization. The CNN stack was followed by a global average pooling layer, a dropout layer with $p = 0.4$, and a fully connected softmax output layer with five units(one per class).

Multilayer Perceptron (MLP). As an alternative approach, a deep feed-forward network is implemented. The network consisted of six fully connected layers with decreasing width ($1024 \to 512 \to 256 \to 128 \to 64 \to 32$), with after each layer ReLU activation and dropout layers with progressive smaller dropout rates applied. A final fully connected layer with four units and a softmax output performed the classification.

3.3 Evaluation Metrics and Results.

Training results: The CNN achieved a rapid convergence in the first epochs, reaching a training accuracy of 95.3% and a validation accuracy of 93.1% after five epochs. The validation loss initially decreased but started to fluctuate after several epochs and dropped to 20% and eventually stuck to that and training has gotten stall. Despite good performance for approach 1 step 1 of two stage approach, CNN accuracy dropped to approximately 60% in the second stage (classifying only damaged samples by damage type). Because of this degradation and the risk of overfitting, further experimentation with CNN architectures was discontinued.

Alternative approach (MLP) converged much more slowly than the CNN, requiring more epochs to reach high accuracy , but ultimately achieved much better classification performance on the validation set. The MLP proved a robust solution for the given problem, maintaining its performance when distinguishing between the individual classes.

Observation. Overall, the CNN was able to learn quickly and reached good first-stage classification accuracy but did not generalize well to the second-stage task. The MLP required substantially more training time but was more stable across both stages, making it the preferred architecture for this project. It is expected that with careful hyperparameter tuning, CNNs could potentially outperform MLPs, bu this was left for future work due to time constraints.

Table 3.1: Comparison of validation accuracies for the two classification approaches.

Approach	**Task**	**Validation Accuracy (%)**
Approach 2 (Single-stage)	5-class classification	94.06
Approach 1 (Two-stage)	Step 1: Damage vs. No damage	96.56
	Step 2: Damage type classification	88.96
	Total Accuracy	91.25

Comparison of Approaches. Table 3.1 compares the validation accuracies achieved by the two modeling strategies. The single-stage classifier (Approach 2) achieved an overall accuracy of 94.06% on the 5-class classification task. The two-stage classifier (Approach 1) obtained a higher accuracy in the first step (96.56%) but the second step reached only 88.96% resulting in total accuracy 91.25%. Also it is worth to be mentioned that Single-stage classifier was trained half of the time of Two-stage classifer.

These results suggest: even though two-stage approach provides excellent separation of damage and Noise samples, its total performance lags behind the single-stage approach.

This shows that end-to-end approaches are better for the given CFRP damage classification problem.

References

[BPZL12] Brown, Gavin ; Pocock, Adam ; Zhao, Ming-Jie ; Luján, Mikel: Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. In: *Journal of Machine Learning Research* 13 (2012), S. 27–66

[GSB+21] Gburrek, Tobias ; Schmalenstroeer, Joerg ; Brendel, Andreas ; Kellermann, Walter ; Haeb-Umbach, Reinhold: Deep Neural Network based Distance Estimation for Geometry Calibration in Acoustic Sensor Networks. In: *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, S. 196–200

[Jol02] Jolliffe, I. T.: *Principal Component Analysis*. New York : Springer, 2002

[PVG+11] Pedregosa, Fabian ; Varoquaux, Gaël ; Gramfort, Alexandre ; Michel, Vincent ; Thirion, Bertrand ; Grisel, Olivier ; Blondel, Mathieu ; Prettenhofer, Peter ; Weiss, Ron ; Dubourg, Vincent ; Vanderplas, Jake ; Passos, Alexandre ; Cournapeau, David ; Brucher, Matthieu ; Perrot, Matthieu ; Duchesnay, Édouard: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830