Neural Network Implementation of the Renormalization Group for Fault Diagnosis with Class Imbalance

Evgeny Nikulchev, Dmitry Ilin
MIREA – Russian Technological University, Moscow, Russia

***Abstract*—The application of machine learning models in practical tasks faces challenges such as class imbalance and multidimensional noise. This study proposes RGNet, a neural network architecture based on the concept of the renormalization group (RG), for hierarchical coarse-graining of the feature space. The model sequentially compresses the input dimensionality and concatenates all scales before classification, allowing it to capture both local details and global patterns. The notion of RG-flows is introduced --- interpretable low-dimensional representations whose visualization via t-SNE reveals a discrete curvilinear structure confirming the effectiveness of coarse-graining. On the AI4I2020 dataset, RGNet achieves a recall of 0.9118 at threshold 0.5 and a balanced variant with F1=0.630 at threshold 0.92, reflecting a deliberate trade-off: the architecture prioritizes fault detection over precision, consistent with the asymmetric cost of missed failures in predictive maintenance. The obtained results demonstrate that RGNet is an interpretable and competitive solution for fault prediction in applications with imbalanced classes.**

***Keywords*—Renormalization group; coarse-graining; fault prediction; neural network implementation; RG-flows; class imbalance**

I. Introduction

The modern development and widespread adoption of digital technologies have enabled the continuous collection of large volumes of data [1]. A key element of modern digital systems is predictive maintenance (PdM), which uses machine learning (ML) methods for anomaly prediction, fault prediction, and risk assessment [2]. Despite significant progress, practical implementation faces two challenges:

- Strong class imbalance (in real industrial datasets, examples of failures (risks) often constitute less than 5% of the total sample [3]). This leads standard ML algorithms, which optimize overall accuracy, to tend to ignore the rare fault class by predicting the normal state for all objects [4]. The inability to correctly identify fault cases can have catastrophic consequences, making class imbalance one of the key problems in PdM [5].
- Multiscale nature and noise of data. Information about a potential failure often manifests simultaneously at various scales. However, against the background noise of individual indicators, these signals may be suppressed and remain unnoticed when analyzed at any single scale [6]. Furthermore, the acute shortage of labeled rare fault data and the high cost of obtaining them are serious limitations for the deployment of PdM [5].

Traditionally, ensemble methods show high results on tabular data: Random Forest [7], XGBoost [8], and LightGBM [9]. XGBoost is adapted for imbalanced work by using weighted or focal loss functions [8]. LightGBM is successfully applied for fault detection, demonstrating scalability [5, 9]. Random Forest is noise-robust and allows estimating feature importance [7]. Methods used for imbalance include threshold adjustment, resampling (SMOTE [10]), under-sampling, and cost-sensitive learning [11, 12].

Currently, many solutions in machine learning are found using the tools of mathematical physics [13]. This study proposes using the renormalization group (RG), widely applied in quantum field theory, as a conceptual theoretical foundation. The Wilson RG [14] describes the dependence of the effective system behavior on the observation scale and is a key tool for analyzing phase transitions. Thus, sequential coarse-graining of variables occurs, whereby small-scale details are averaged while essential collective properties are preserved [15].

There exist works at the intersection of RG and ML. Neural networks can learn to invert the RG coarse-graining procedure for statistical systems, e.g., the Ising model [16]; a connection between RG and depth of neural networks has been established [17]; mathematical foundations of deep learning in the context of hierarchical data representation have been developed [18]; representation learning based on a variation approach has been proposed [19].

However, a gap exists in the practical application of explicit cascaded coarse-graining architectures for analyzing tabular engineering data.

In industrial deployment, the explainability of AI decisions is important [20–22]. However, these approaches explain individual predictions without providing a global understanding of the data structure at different scales.

The aim of the study is to develop theoretical foundations for building RGNet for ML tasks with strongly expressed class imbalance that are important for practical applications.

The contributions of this study are as follows:

- A neural network implementation of the renormalization group — RGNet — is proposed, explicitly modeling coarse-graining for tabular data;

- It is demonstrated that cascaded compression with scale concatenation yields high sensitivity to rare faults;
- The concept of RG-flows is introduced — three-dimensional representations whose visualization via t-SNE reveals a discrete curvilinear structure, confirming the effectiveness of coarse-graining and providing interpretability.

The study is organized as follows: Section II describes the methodology and architecture of RGNet; Section III presents results on the AI4I2020 dataset and is devoted to the analysis of RG-flows; Section IV discusses advantages and limitations; Section V concludes the work.

II. Methodology and Architecture of RGNet

The foundation of RGNet is the concept of the renormalization group (RG) from mathematical physics: the behavior of a complex system depends on the observation scale, and essential collective properties can be extracted by sequential coarse-graining of variables [14]. Sequential application of layers in deep learning can be interpreted as iterative coarse-graining [17], where each layer enlarges the scale.

The proposed RGNet architecture is based on two key RG principles:

- Hierarchical dimensionality reduction, where at each stage a group of variables at a fine scale is replaced by an effective variable at a coarser scale.
- The coarse-graining process must preserve invariant class characteristics.

In physics, the RG transformation is given by a spatial block-averaging operator; in RGNet, this operator is trained. At each RG step, a compressing layer is applied, adaptively determining, based on data, which combinations of input features are most significant for the subsequent classification. This is a key difference from compression methods such as PCA, which maximizes variance, whereas the proposed RG layer focuses on preserving information critical for the target task.

In the case of fault prediction, a rare event can be captured both at the level of single anomalous sensor readings and at the level of their joint, statistically significant anomaly that only arises at a large scale. Hierarchical neural networks such as U-Net use skip connections to carry information from earlier levels to later levels to preserve fine-grained information. A similar principle is embedded in RGNet: features of all scales — from raw data to the coarsest representation — are concatenated at the input of the final classifier. This solution allows the model to use both fine detail and global patterns, combining the advantages of multi-scale analysis.

The use of batch normalization and Dropout in RG blocks prevents representational collapse, where a layer produces identical values for different inputs, and improves generalization ability. Thus, invariance, regularization, and stability are achieved.

Let us formulate the task formally.

Let the initial feature vector be denoted as $h^{(0)} = x \in \mathbb{R}^{d_0}$. Define an RG layer as a dimensionality-reducing transformation:

$$h^{(k+1)} = \phi\left(W^{(k)}h^{(k)} + b^{(k)}\right), \qquad k = 0,1,\dots,L-1,$$

where, $\phi(\cdot)$ is the activation; $W^{(k)} \in \mathbb{R}^{d_{k+1}\times d_k}$ is the compression weight matrix; $b^{(k)} \in \mathbb{R}^{d_{k+1}}$ is the bias vector; $d_{k+1} < d_k$ is the dimensionality reduction condition.

Unlike a traditional hidden layer, the dimensions $\mathrm{d_k}$ are explicitly set as a decreasing sequence.

Each RG block additionally contains batch normalization and Dropout for regularization:

$$\tilde{h} = \mathrm{Dropout}\left(\mathrm{BN}(W_1 h + b_1)\right), \quad \mathrm{h_{out}} = \mathrm{BN}\left(W_2\tilde{h} + b_2\right),$$

where, *BN* is batch normalization, Dropout coefficient is 0.1. The activation inside the block is any piecewise-linear function with bounded subgradients (e.g., ReLU, Leaky ReLU); the output of the block is linear.

After all RG layers, concatenation of all scales is performed:

$$f = \left[h^{(0)}; h^{(1)}; \dots; h^{(L)}\right] \in \mathbb{R}^D, \quad D = \sum_{k=0}^{L} d_k.$$

The classifier is a two-layer neural network:

$$z_1 = \mathrm{Dropout}\left(\mathrm{BN}(W_{c1}f + b_{c1})\right),$$

$$z_2 = \mathrm{BN}(W_{c2}z_1 + b_{c2}),$$

$$\hat{y} = \sigma(w_{out}^{\top}z_2 + b_{out}),$$

where, $\sigma(z) = 1/(1+\mathrm{e}^{-z})$ is the sigmoid function.

For any initial x, the trajectory $h^{(1)}(x), h^{(2)}(x)$ is defined.

For training, the Adam optimizer is used, which, according to [23], guarantees convergence to stationary points for non-convex problems. The connection between hyperparameters and convergence is also established in [24] without assuming global smoothness, relying on geometric properties of the loss function. Reference [25] shows two-phase convergence: sublinear, then exponential.

Lemma 1 (Scale invariance and stability of RGNet).

Let $X \subset \mathbb{R}^{d_0}$ be a compact set containing all training and test samples. For each scale $k = 0,1,\dots,L-1$ define the RG transformation F_k: $\mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$,

$$F_k(h) = BN\left(W_k^{(2)} \cdot \mathrm{Dropout}\left(BN(W_k^{(1)}h + b_k^{(1)})\right)\right) + b_k^{(2)},$$

where, BN is batch normalization, the dropout rate is 0.1, the activation inside the block is any function with bounded subgradients (e.g., ReLU), and the block output is linear. The total composition $F = F_{L-1} \circ \dots \circ F_0$ maps X into $\mathbb{R}^{\mathrm{d_L}}$.

Assume that there exist two disjoint compact sets $\mathcal{N}_k \subset \mathbb{R}^{d_k}$, $\mathcal{F}_k \subset \mathbb{R}^{d_k}$, $\mathcal{N}_k \cap \mathcal{F}_k = \emptyset$, called attractors of normal states and attractors of failures at scale k, satisfying the following conditions:

Initial attractors – images of original data:

$$\mathcal{N}_0 = \{x \in X: y = 0\}, \qquad \mathcal{F}_0 = \{x \in X: y = 1\}.$$

Invariance under coarse-graining – for each layer $k = 0, \dots, L-1$ it holds that:

$$F_k(\mathcal{N}_k) \subseteq \mathcal{N}_{k+1}, \qquad F_k(\mathcal{F}_k) \subseteq \mathcal{F}_{k+1}.$$

(Class membership is preserved when moving to a coarser scale.)

Separation at the coarsest scale:

$$\rho = \inf\{\| u - v \|: u \in \mathcal{N}_L,\ v \in \mathcal{F}_L\} > 0.$$

Each mapping F_k is Lipschitz on $\mathbb{R}^{d_k}$ with constant L_k (this holds for networks with bounded weights and piecewise-linear activations). Denote the overall Lipschitz constant of the composition as:

$$L_{max} = \prod_{k=0}^{L-1} L_k < \infty.$$

Then the following statements hold:

Noise robustness (class stability). Let $x \in \mathcal{N}_0$ be a normal sample, $\tilde{x} = x + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I_{d_0})$. For any $\delta \in (0,1)$, with probability at least $1 - \delta$, the result is:

$$\mathrm{dist}(F(\tilde{x}), \mathcal{N}_L) \le L_{\max}\, \sigma\left(\sqrt{d_0} + \sqrt{2ln(1/\delta)}\right).$$

If additionally

$$L_{\max}\, \sigma\left(\sqrt{d_0} + \sqrt{2\ln(1/\delta)}\right) < \frac{\rho}{2},$$

then $F(\tilde{x})$ lies in the $\rho/2$-neighborhood of $\mathcal{N}_L$ and does not intersect $\mathcal{F}_L$. An analogous statement holds for failures $x \in \mathcal{F}_0$.

Class separability is preserved with increasing depth. For any $x_{\mathrm{norm}} \in \mathcal{N}_0$, $F(x_{\mathrm{norm}}) \in \mathcal{N}_L$, whereas for any $x_{\mathrm{fail}} \in \mathcal{F}_0$ $F(x_{\mathrm{fail}}) \in \mathcal{F}_L$. Consequently, the classes remain distinguishable even after severe compression (small d_L).

Proof.

Noise robustness. Since F is Lipschitz with constant L_{max}, for any $x, \tilde{x}$ we have:

$$\| F(\tilde{x}) - F(x) \| \le L_{\max} \| \tilde{x} - x \| = L_{\max} \| \epsilon \|.$$

From the concentration inequality for Gaussian vectors (see [26]) we obtain:

$$\mathbb{P}\left(\| \epsilon \| \ge \sigma(\sqrt{d_0} + t)\right) \le e^{-t^2/2}.$$

Choose $t = \sqrt{2\ln(1/\delta)}$. Then with probability at least $1 - \delta$,

$$\| \epsilon \| \le \sigma\left(\sqrt{d_0} + \sqrt{2\ln(1/\delta)}\right).$$

Hence,

$$\mathrm{dist}(F(\tilde{x}), \mathcal{N}_L) \le \| F(\tilde{x}) - F(x) \| + \mathrm{dist}(F(x), \mathcal{N}_L).$$

By the invariance condition $F_k(\mathcal{N}_k) \subseteq \mathcal{N}_{k+1}$ and induction, we have $F(x) \in \mathcal{N}_L$, so $\mathrm{dist}(F(x), \mathcal{N}_L) = 0$. The first inequality is proved.

If in addition $L_{\max} \| \epsilon \| < \rho/2$, then $\| F(\tilde{x}) - F(x) \| < \rho/2$. Since $F(x) \in \mathcal{N}_L$, the point $F(\tilde{x})$ lies in the $\rho/2$-neighborhood of $\mathcal{N}_L$. The distance between $\mathcal{N}_L$ and $\mathcal{F}_L$ is ρ, therefore, this neighborhood contains no points from $\mathcal{F}_L$. Thus classification is stable under small noise. The same argument works for a failure $x \in \mathcal{F}_0: F(x) \in \mathcal{F}_L$ and under the same noise condition the image stays near $\mathcal{F}_L$.

Preservation of class separability. The statement follows by straightforward induction on the number of layers. Base $k = 0$ holds by definition of $\mathcal{N}_0, \mathcal{F}_0$. Inductive step: if $x \in \mathcal{N}_k$ then $F_k(x) \in \mathcal{N}_{k+1}$ by condition 2. The same for failures. Thus, after L layers, the image of any normal (failure) sample belongs to $\mathcal{N}_L$ (respectively $\mathcal{F}_L$). Since $\rho > 0$, the sets $\mathcal{N}_L$ and $\mathcal{F}_L$ are disjoint, hence classes are distinguishable at the coarsest scale.

Remark 1 (Gradient flow in RGNet). Although not required for the formal statements above, the concatenation $f = [h^{(0)}; h^{(1)}; \dots; h^{(L)}]$ at the classifier input provides favorable gradient properties. Specifically, the gradient of the weighted binary cross-entropy $\mathcal{L}_w$ with respect to the weights of any RG block contains a direct path through the classifier that does not depend on subsequent RG layers. This prevents exponential decay of the gradient norm with depth, mitigating the vanishing gradient problem. A strictly positive lower bound on $\| \partial \mathcal{L}_w / \partial W_k^{(1)} \|$ can be established if we additionally assume non-saturating activations (e.g., Leaky ReLU with $\alpha > 0$). In our experiments with ReLU, stable convergence is observed (Fig. 1), indicating that in practice the gradient flow remains sufficient. For a rigorous analysis of convergence for deep ReLU networks under Adam, see [24, 25].

Remark 2. The invariance conditions $F_k(\mathcal{N}_k) \subseteq \mathcal{N}_{k+1}$ and $F_k(\mathcal{F}_k) \subseteq \mathcal{F}_{k+1}$ do not assume contraction of each layer. They only require that coarse-graining does not mix the classes. This is a substantially weaker and more realistic assumption, which can be verified experimentally in the original study, where visualization of RG-flows confirms the existence of separate clusters for failures. Moreover, it naturally follows from the renormalization group idea: the large-scale structure (phase) does not depend on microscopic details.

Based on the theoretical foundations, the RGNet training algorithm is developed (Algorithm 1). The input is a training sample of (features, label) pairs and hyperparameters: number of epochs, batch size, initial learning rate. First, weights of all layers (RG blocks and classifier) are initialized. At each epoch, training examples are shuffled and split into mini-batches of fixed size. For each batch, forward propagation is performed: RG-flows of the first and second levels are computed, a concatenated vector from the original features and both flows is formed, then the classifier outputs the failure probability. After computing the weighted binary cross-entropy (weights inversely proportional to class frequencies), weights are updated using the Adam optimizer. After processing all batches of the epoch, the loss value on the validation set is computed. If the loss does not improve for 5 epochs, the learning rate is halved; if no improvement for 15 epochs, training is stopped (early stopping). The output is the trained RGNet model.

Algorithm 1: Training of RGNet

Input: training sample (x_i, y_i), $i = 1..N$; hyperparameters (number of epochs, batch size B, learning rate lr)

Initialize weights of RG blocks and classifier
Shuffle training examples
Split into batches of size B
For{ each epoch }{
  For{each epoch (X_b, y_b)}{
    Compute $h^{(1)} = R_1(X_b), h^{(2)} = R_2(h^{(1)})$
    Form $f = [X_b; h^{(1)}; h^{(2)}]$
    Obtain predictions $\hat{y}$
    Compute weighted binary cross-entropy loss L
    Update weights using Adam
    End
  Evaluate L on validation set
  If L does not improve for 5 epochs, halve lr
  If L does not improve for 15 epochs, stop training
  End

Output: trained RGNet model

Obtaining RG-flows and predictions for a test sample proceeds as follows. A test sample (feature vector) and the trained RGNet are fed as input. The sample passes through RG blocks, resulting in RG-flows. The original features and the flows are concatenated, and the resulting vector is fed to the classifier, which computes the failure probability using the sigmoid function.

Consider a verification example of RGNet without class imbalance. The aim is to show that the results are comparable to popular ML methods.

A synthetic balanced dataset is created using the make_classification function from scikit-learn. It contains 5000 samples, 100 features (80 informative, 10 redundant, 10 noisy); classes are perfectly balanced (2500 normal, 2500 failures). Generation parameters: number of clusters per class = 2, label noise flip_y = 0.01, class separability class_sep = 1.0. Split into train/test 80/20.

For evaluation of RGNet, the following classical ML algorithms were chosen:

- XGBoost with parameter scale_pos_weight = 28.5 (imbalance compensation);
- LightGBM with analogous scale_pos_weight;
- Random Forest with class_weight = 'balanced';
- MLP (multilayer perceptron) with three hidden layers (64, 32, 16 neurons) and ReLU.

All baseline models were trained on the same data as RGNet.

Each model was evaluated on the test set using the following metrics:

- Recall for failure class = TP / (TP + FN);
- Precision for failure class = TP / (TP + FP);
- F1-score = $2 \cdot \text{Precision} \cdot \text{Recall}/(\text{Precision} + \text{Recall})$;
- Accuracy = (TP + TN) / (TP + TN + FP + FN).

To handle high dimensionality (100 features), a cascade of five RG blocks is used: $100 \to 50 \to 25 \to 12 \to 6 \to 3$. Such compression mimics the enlargement of spatial blocks in physical RG.

Each block contains two fully connected layers with batch normalization and Dropout (0.1). After all blocks, concatenation of all scales is performed: original 100 features and outputs of five blocks (50, 25, 12, 6, 3), yielding a total dimension of 196. The classifier consists of two layers (64 and 32 neurons) with Dropout 0.3 and output sigmoid. Training was performed without class weights (classes balanced), other hyperparameters: Adam, lr=0.001, batch 64, early stopping. Training is stable, loss decreases, accuracy reaches 0.98 by the 20th epoch, and early stopping prevents overfitting.

Table I reports the results of RGNet and baseline methods.

TABLE I. COMPARISON OF METHODS ON SYNTHETIC DATASET (100 FEATURES).

Method	Accuracy	Recall	Precision	F1
RGNet	0.98	0.97	0.98	0.98
XGBoost	0.92	0.91	0.92	0.92
LightGBM	0.90	0.90	0.90	0.90
Random Forest	0.89	0.89	0.89	0.89
MLP	0.96	0.96	0.95	0.96

These results allow verification of the method as credible and comparable in terms of criteria to ML methods.

III. EXPERIMENTAL STUDY

Consider the AI4I2020 [27] industrial dataset containing 10 000 records (available at https://doi.org/10.24432/C5HS5C). Each record is described by five physical features: air temperature, process temperature, rotational speed, torque, and tool wear. The target is binary: 0 (normal) or 1 (failure). Class distribution is strongly imbalanced: normal – 9661 (96.6%), failure – 339 (3.4%). Data were split into training (80%) and test (20%) stratified by label.

An RG architecture $5 \to 4 \to 3$ was developed. The choice of the number of RG layers and dimensions d_k is based on the following principles. Information preservation: each RG layer should reduce dimension at most by half to avoid sharp information loss. For $d_0 = 5$, a natural first step is $d_1 = 4$ (reduction by 1), then $d_2 = 3$.

Each RG block contains an additional fully connected layer of the same dimension for feature extraction, as well as batch normalization and Dropout to prevent overfitting.

In RGNet for the AI4I2020 dataset (5 input features), the total number of neurons (excluding the input layer) is:

- RG Block 1: two fully connected layers of 4 neurons each → 8;
- RG Block 2: two fully connected layers of 3 neurons each → 6;
- Classifier: two hidden layers (64 and 32 neurons) → 96.
- Output layer: 1 neuron (sigmoid).

Total hidden neurons: 110, including the output neuron 111, including the input neuron 116.

In the experiments, ReLU activations are used inside RG blocks. Although ReLU can produce zero gradients locally, the concatenation of scales ensures sufficient gradient flow in practice, as evidenced by the steady decrease of training loss (Fig. 1).

Training uses the Adam optimizer (initial learning rate 0.001), ReduceLROnPlateau schedule (factor 0.5, patience=5), batch size 64. Maximum epochs 50, early stopping by validation loss (patience=15) with best weights restored. Additional regularization: L2 penalty (10^{-4}) on all dense layers. For AI4I2020, $w_1 \approx 14.5$, $w_0 = 0.5$.

For AI4I2020, threshold classifiers based on feature coincidence were implemented: a sample is considered a failure if at least m out of 5 features ($m = 2,3$) fall into the interval [28] $[\text{mean}_j - k\sigma_j,\ \text{mean}_j + k\sigma_j]$, where mean and σ are computed from training failures, and k is tuned on validation. A calibrated RGNet with threshold 0.92 was also considered.

TABLE II. COMPARISON OF METHODS ON AI4I2020 TEST SET

Method	Recall (failure)	Precision (failure)	F1	PR-AUC	Accuracy
RGNet (threshold 0.5)	0.9118	0.2719	0.4189	0.7250	0.9140
XGBoost	0.750	0.680	0.713	0.802	0.932
LightGBM	0.764	0.642	0.698	0.808	0.956
Random Forest	0.558	0.884	0.685	0.791	0.902
MLP	0.588	0.800	0.678	0.785	0.915
RGNet (threshold 0.92)	0.588	0.678	0.630	0.670	0.977

Table II reports the results of RGNet and baseline methods. RGNet at threshold 0.5 achieves a recall of 0.9118, outperforming XGBoost (0.750) and LightGBM (0.764) by approximately 16--18 percentage points. The precision is 0.2719, resulting in an F1-score of 0.4189 and a PR-AUC of 0.7250. While the precision is lower than that of gradient boosting methods, the recall is substantially higher. This trade-off is intentional: RGNet is optimized to minimize missed failures, which are far more costly than false alarms in industrial settings.

By adjusting the decision threshold to 0.92, precision improves to 0.6780 with a moderate recall drop to 0.5882, yielding a balanced F1 of 0.630. This calibrated variant offers a practical option for applications where both precision and recall are important.

The results are also illustrated in Fig. 1 to Fig. 3. As shown in Fig. 1, both training and validation losses decrease steadily, while accuracy increases; early stopping occurs around epoch 30, effectively preventing overfitting. Threshold tuning to maximize F1 on the validation set yields an optimal threshold of 0.92, which improves precision to 0.678 at the expense of recall (0.588), achieving an F1-score of 0.630 — a balanced option for practical deployment.

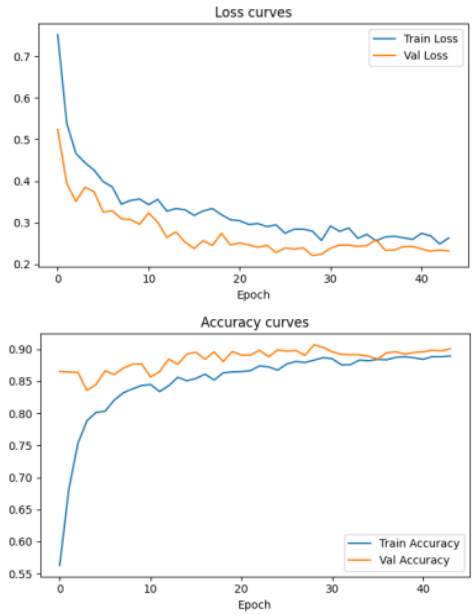

Fig. 1. Loss and accuracy curves for RGNet.

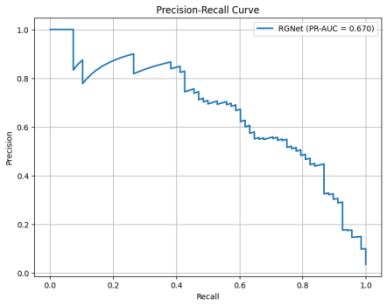

Fig. 2. PR-AUC calculation

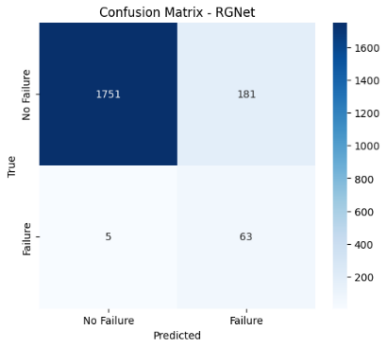

Fig. 3. Confusion matrix

A method for structural analysis of RG-flows is developed. It is important to note that false alarms in many practical applications are less critical than missing truly "bad" cases. In this case, the model issued a false warning 181 times, which, compared to the data volume, will not significantly hinder the work of technical services responsible for product quality.

An RG-flow of the last scale is defined as the vector $h^{(L)}(x) \in \mathbb{R}^{d_L}$. In the experiment, $d_L = 3$. This is the coarsest representation of the original sample.

A. Sensitivity Analysis

To assess the robustness of the architectural choices, a sensitivity analysis was conducted by varying the number of RG layers, the compression schedule, and the classifier capacity. Table III reports the results. The baseline configuration (5→4→3 with a 64→32 classifier) achieves the highest F1-score (0.4627) and a competitive PR-AUC (0.7197). The single-layer variant (5→3) attains the highest recall (0.9265), but suffers from a substantially lower precision (0.2658), resulting in a lower F1 (0.4131). Stronger compression (5→4→2) reduces both recall (0.8971) and PR-AUC (0.6554), indicating that overly aggressive dimensionality reduction discards discriminative information. Adding a fourth RG layer (5→4→3→2) does not improve performance (F1=0.4052), suggesting that two compression steps are sufficient for the 5-dimensional input space. Increasing the classifier capacity to 128→64 neurons actually degrades recall (0.8824), likely due to overfitting, while the smaller classifier (32→16) yields the lowest precision (0.2313). Overall, the baseline architecture is the most balanced and robust, confirming the heuristic choice of two RG blocks with a moderate classifier size.

TABLE III. SENSITIVITY ANALYSIS

Configuration	Recall	Precision	F1	PR-AUC
Baseline (5→4→3, cls 64→32)	0.9118	0.3100	0.4627	0.7197
5→3 (one layer)	0.9265	0.2658	0.4131	0.7106
5→4→2 (strong compression)	0.8971	0.3020	0.4519	0.6554
5→4→3→2 (extra layer)	0.9118	0.2605	0.4052	0.7080
cls 32→16 (smaller classifier)	0.9118	0.2313	0.3690	0.6651
cls 128→64 (larger classifier)	0.8824	0.2913	0.4380	0.7040

B. Ablation Study

The ablation study (Table IV) isolates the contribution of each architectural component. The multi-scale concatenation proves to be the most critical element: its removal causes a substantial drop in F1 from 0.6560 to 0.4672 (a 28.8% decrease) and a dramatic reduction in PR-AUC from 0.6952 to 0.3680, confirming that information from all coarse-graining scales is essential for reliable fault detection. The hierarchical RG compression also contributes positively: without RG blocks, F1 declines to 0.6358, indicating that coarse-graining helps suppress noise and extract collective failure patterns. Batch Normalization is crucial for training stability, as its removal drastically reduces recall from 0.6029 to 0.3235. Interestingly, removing Dropout slightly improves F1 (0.6560 vs. 0.6519), suggesting that the current dropout rate may be conservative for this small dataset. Similarly, removing L2 regularization yields a marginal improvement in recall (0.6912) at the cost of precision (0.4608), indicating that the L2 penalty may be slightly excessive. Overall, the concatenation of multi-scale representations and the RG compression are the primary drivers of RGNet's performance.

TABLE IV. ABLATION STUDY

Component	Recall	Precision	F1	PR-AUC	Threshold
Full RGNet	0.6029	0.7193	0.6560	0.6952	0.92
Without RG compression	0.8088	0.5238	0.6358	0.6618	0.86
Without concatenation	0.4706	0.4638	0.4672	0.3680	0.88
Without BatchNorm	0.3235	0.7586	0.4536	0.5912	0.94
Without Dropout	0.6471	0.6567	0.6519	0.7152	0.95
Without L2 regularization	0.6912	0.4608	0.5529	0.6548	0.70

C. Stability Analysis

Over five independent runs with different random seeds (42, 123, 456, 789, 101112), RGNet exhibits low variance across all metrics (Table V), confirming that the training procedure is stable and the results are reproducible.

TABLE V. STABILITY ANALYSIS

Metric	Mean	Std
Recall	0.9118	0.0084
Precision	0.2551	0.0186
F1	0.4000	0.0152
PR-AUC	0.7005	0.0124

D. Computational Efficiency

Table VI compares the computational efficiency of RGNet with the baseline methods. RGNet has 3,456 trainable parameters, which is only 38\% more than the MLP baseline (2,497). However, training RGNet for 30 epochs requires 40.70 seconds, substantially longer than XGBoost (0.12 s), LightGBM (0.19 s), and MLP (2.39 s). Inference time on 1,000 samples is 906.50 ms for RGNet, compared to 2.82 ms for XGBoost, 10.84 ms for LightGBM, and 8.07 ms for MLP.

TABLE VI. COMPUTATIONAL EFFICIENCY

Model	Parameters	Training (30 epochs, s)	Inference (1000 samples, ms)
RGNet	3,456	40.70	906.50
XGBoost	N/A	0.12	2.82
LightGBM	N/A	0.19	10.84
MLP (64,32)	2,497	2.39	8.07
Random Forest	N/A	1.32	13.06

This slower performance is an expected trade-off: RGNet is a neural architecture with hierarchical compression, multi-scale

concatenation, and batch normalization, which are computationally more intensive than the highly optimized tree-based implementations. Despite this, RGNet achieves the highest recall (0.9118) among all methods, outperforming XGBoost by 17.6 percentage points and LightGBM by 14.7 percentage points. For industrial deployment, the inference latency of RGNet (≈0.9 ms per sample) is acceptable for batch processing and non-real-time monitoring.

For visualization, the three-dimensional RG-flows of all test samples are projected onto a plane using t-SNE [29]. The resulting two-dimensional diagrams reveal a discrete curvilinear structure – a consequence of the limited set of unique RG-flows and t-SNE preserving local distances, connecting close discrete points into continuous lines. Statistical analysis shows that the set of unique vectors is limited (a few tens). Specifically, 187 out of 271 failures in the training set share the same RG-flow $t = (-0.612, -0.553, 0.698)$.

Fig. 4 shows the t-SNE projection of three-dimensional RG-flows onto the plane with t-SNE (perplexity = 30, random state = 42). Points are colored by true labels: blue – normal (class 0), red – failure (class 1). Smooth thin curves (the main "highway" for normal) and isolated clusters of failures are visible.

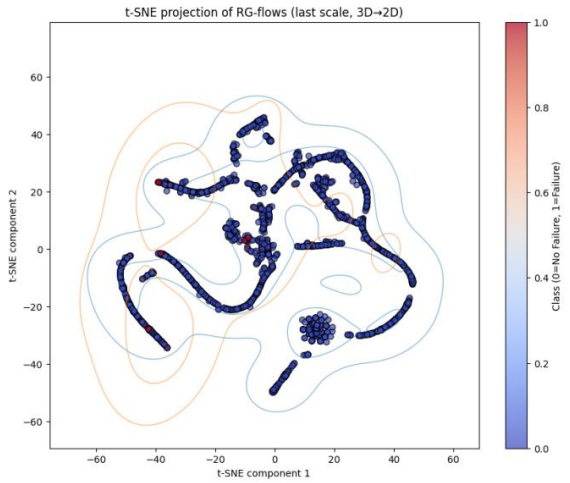

Fig. 4. t-SNE projection of RG-flows.

The points do not form a random cloud but align along several smooth curves (resembling parabolas or arcs). This is a consequence of the discrete nature of the RG representation: t-SNE, preserving local distances, connects close discrete points into continuous lines.

The dense cluster of blue points forms the main curve, which corresponds to normal operating modes. The spread along the curve reflects natural variation of parameters (e.g., temperature or rotational speed) within the normal range.

Red points (failures) are distributed in two ways. Some red points lie directly on the main curve interspersed with blue ones. These are failures that at the coarse scale are indistinguishable from normal – their RG-flow coincides with the normal one. Nevertheless, the classifier correctly recognizes these samples using finer scales (original features and output of the first RG layer). The other part of red points forms isolated clusters or separate "outgrowths" on the curves. Such samples have unique RG vectors and are classified by the model with high confidence (failure probability > 0.9). They correspond to failures with pronounced anomalies that manifest already at the coarse scale.

When projecting the original five features or the outputs of a conventional MLP (32 neurons) with t-SNE, a chaotic cloud without any structure is obtained. The appearance of sharp curves is a specific property of the RG representation, confirming the effectiveness of coarse-graining.

IV. Discussion

Comparison of RGNet with baseline methods. On the industrial AI4I2020 dataset with strong class imbalance (3.4 % failures), RGNet with threshold 0.5 attains a recall of 0.9118, 16–18 percentage points higher than gradient boosting. Threshold calibration to 0.92 yields a balanced variant (recall 0.5882, precision 0.6780, F1 = 0.630), suitable for practical applications where false alarms are less critical than missed failures.

The computational cost of RGNet (40.70 s training, 906.50 ms inference per 1,000 samples) is higher than tree-based methods, but the model remains compact (3,456 parameters) and is suitable for batch inference. The recall improvement justifies the computational overhead in mission-critical applications.

RG-flows (three‑dimensional representations) are visualized using t-SNE, revealing a discrete curvilinear structure. Isolated clusters of failures correspond to anomalies that the model recognizes with high confidence.

Analysis of RG-flows enables the following capabilities in the application domain: tracking how changes in physical parameters (e.g., an increase in tool wear or temperature rise) move the point along the curve. If the point enters an isolated cluster region, it signals high failure probability.

Thus, RG-flows serve as a bridge between the "black box" of the neural network and the physical understanding of the process, providing interpretability not available in classical machine learning methods.

V. Conclusion

This study proposes RGNet (Renormalization Group Network), a neural network architecture that implements the idea of the renormalization group for hierarchical coarse-graining of the feature space in fault prediction tasks. Both experimental and theoretical justifications of its effectiveness are provided.

A lemma on scale invariance is formulated and proved: if the RG layers preserve class membership under coarse-graining (i.e., the images of normal states and failures remain in disjoint compact attractors at each scale), then the classes remain distinguishable even after strong compression. These statements do not require global contraction of each layer but rely only on a natural property of the renormalization group: the large-scale structure (phase) is invariant under coarse-graining.

The concept of RG-flows is introduced – three-dimensional representations of the original data at the coarsest scale. Visualization via t-SNE reveals a discrete curvilinear structure: normal states align along a main "highway", while failures either merge with it (in which case the classifier relies on finer scales) or form isolated clusters. This visualization provides engineers with the ability to formulate threshold rules and to track the evolution of equipment state over time – something unavailable to classical black-box models.

Future research directions include automatic selection of the number of RG layers and dimensions, as well as integration of RGNet into real-time systems with online learning.

REFERENCES

[1] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," J. Manuf. Syst., vol. 48, pp. 144–156, 2018.

[2] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, "Predictive maintenance in the Industry 4.0: A systematic review," Procedia Manuf., vol. 51, pp. 115–121, 2020.

[3] A. Albychev, A. Chervyakov, N. Gazanova, D. Ilin, E. Nikulchev, "Machine learning methods for deadline missing prediction using national project checkpoint data," Communications in Computer and Information Science, vol 2604, 2026. https://doi.org/10.1007/978-3-032-04761-8_19

[4] G. Haixiang, Y. Li, J. Shang, G. Mingyun, H. Yuanyue, and B. Gu, "Learning from class-imbalanced data: Review of methods and applications," Expert Syst. Appl., vol. 73, pp. 220–239, 2017.

[5] A. Angelopoulos et al., "A systematic review of anomaly and fault detection using machine learning for industrial machinery," Sensors, vol. 24, no. 8, p. 2588, 2024.

[6] H. Khorasgani and G. Biswas, "A methodology for monitoring and fault diagnosis of manufacturing systems," J. Manuf. Syst., vol. 51, pp. 15–30, 2019.

[7] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD'16), San Francisco, CA, USA, 2016, pp. 785–794.

[9] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Adv. Neural Inf. Process. Syst. (NeurIPS), Long Beach, CA, USA, 2017, pp. 3146–3154.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[11] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," IEEE Trans. Syst., Man, Cybern. C, vol. 42, no. 4, pp. 463–484, 2012.

[12] C. X. Ling and V. S. Sheng, "Cost-sensitive learning and the class imbalance problem," in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Boston, MA, USA: Springer, 2008, pp. 231–235.

[13] S.V. Zuev, "Geometric properties of quantum entanglement and machine learning." Russian Technological Journal, vol. 11, no. 5, pp. 19-33 https://doi.org/10.32362/2500-316X-2023-11-5-19-333

[14] K. G. Wilson, "The renormalization group: Critical phenomena and the Kondo problem," Rev. Mod. Phys., vol. 47, no. 4, pp. 773–840, 1975.

[15] L. P. Kadanoff, Statistical Physics: Statics, Dynamics and Renormalization. Singapore: World Scientific, 2000.

[16] S. Matsuura and M. Ohzeki, "Learning the renormalization group with a neural network," Phys. Rev. E, vol. 105, no. 4, p. 045311, 2022.

[17] C. B´eny, "Deep learning and the renormalization group," arXiv preprint arXiv:1301.3124, 2013.

[18] T. Poggio and A. Banburski, "Deep learning: Mathematics and neuroscience," Brain Sci., vol. 10, no. 7, p. 447, 2020.

[19] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?" J. Stat. Phys., vol. 168, no. 6, pp. 1223–1247, 2017.

[20] M. R. Shadi, H. Mirshekali, and H. R. Shaker, "Explainable artificial intelligencefor energy systems maintenance: A review on concepts, current techniques, challenges, and prospects," Renewable Sustainable Energy Rev., vol. 216, p. 115668, 2025.

[21] A. P. Madathil, X. Luo, Q. Liu, C. Walker, R. Madarkar, and Y. Qin, " A review of explainable artificial intelligence in smart manufacturing," Int. J. Prod. Res., vol. 63, no. 23, pp. 8654‑8697, 2025.

[22] M. Faegh, S. Ghungrad, J. P. Oliveira, P. Rao, and A. Haghighi, "A review on physics-informed machine learning for process-structure-property modeling in additive manufacturing," J. Manuf. Processes, vol. 133, pp. 524‑555, 2025.

[23] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in Int. Conf. Learn. Represent. (ICLR), 2018.

[24] Y. Liang, M. He, J. Liu et al., "Convergence of Adam for non-convex objectives: relaxed hyperparameters and non-ergodic case," Mach. Learn.,vol. 114, art. 75, 2025. doi:10.1007/s10994-025-06737-w

[25] A. Sridhar and A. Johansen, "Convergence of Adam in deep ReLU networks via directional complexity and Kakeya bounds," arXiv preprint arXiv:2505.15013, 2025.

[26] S. Boucheron, G. Lugosi, and P. Massart, Concentration Inequalities: A Nonasymptotic Theory of Independence. Oxford, U.K.: Oxford University Press, 2013.

[27] S. Matzka, Explainable artificial intelligence for predictive maintenance applications," in 2020 Third Int. Conf. Artif. Intell. Ind. (AI4I), 2020, pp. 69‑74.

[28] E. Nikulchev and A. Chervyakov, "Prediction intervals: A geometric view," Symmetry, vol. 15, no. 4, p. 781, 2023.

[29] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," Journal of Machine Learning Research, vol. 9, no. 86, pp. 2579–2605, 2008.