

Article

An Inversion Method Based on Prior Knowledge for Deep Cascading Pipeline Defects

Pengchao Chen, Wenbo Xuan *, Rui Li , Fuxiang Wang and Kuan Fu

PipeChina Institute of Science and Technology, Langfang 065000, China; chenpc@pipechina.com.cn (P.C.); lirui@pipechina.com.cn (R.L.); wangfx@pipechina.com.cn (F.W.); fukuan@pipechina.com.cn (K.F.)

* Correspondence: xuanwb@pipechina.com.cn; Tel.: +86-15932648972

Abstract: With the robust growth of the national economy, the demand for oil and natural gas continues to rise, heightening the significance of pipeline transportation in the energy sector. However, long-term pipeline operations are often subjected to factors such as corrosion, aging, and damage, which can result in leaks and safety incidents, posing significant threats to life, property, and environmental integrity. Consequently, timely and precise detection of pipeline defects and estimation of their sizes hold paramount practical importance. This research endeavors to employ advanced information technology and artificial intelligence to explore methods for pipeline defect detection and size estimation grounded in prior knowledge. The aim is to enhance the accuracy and reliability of pipeline defect diagnosis and ensure the safe operation of pipelines. The primary innovative work includes the development of a preprocessing method based on prior knowledge, the design of an adaptive algorithm for estimation of defect size, and the creation of an algorithm for estimation of deep cascade pipeline defect size. These methods effectively combine traditional mechanisms and data-driven approaches, leveraging the strengths of both to improve performance, accuracy, and robustness. The proposed methodology demonstrates superior accuracy and stability in defect inversion, providing strong technical support for the quantitative assessment of pipeline defects, which is significant for fault diagnosis and the precise maintenance of long-distance pipelines.

Keywords: defect inversion; magnetic flux leakage detection; prior knowledge; deep learning



Citation: Chen, P.; Xuan, W.; Li, R.; Wang, F.; Fu, K. An Inversion Method Based on Prior Knowledge for Deep Cascading Pipeline Defects. *Electronics* **2024**, *13*, 4781. <https://doi.org/10.3390/electronics13234781>

Academic Editors: Fabio Mendonca, Morgado Dias and Sheikh Shanawaz Mostafa

Received: 29 October 2024
Revised: 22 November 2024
Accepted: 24 November 2024
Published: 3 December 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In this era of rapid economic development, the demand for oil and natural gas continues to rise, underscoring the critical importance of pipeline transportation in the energy sector. However, pipelines operating over extended periods often suffer from corrosion, aging, and damage, leading to leaks and safety incidents that pose significant threats to life, property, and the environment. Timely and accurate defect detection and size estimation have become paramount for ensuring pipeline safety and operational reliability [1–4].

Existing defect inversion methods for pipelines face significant challenges. Traditional approaches struggle with complex defect characteristics, noisy data, and limited generalizability to diverse operating conditions. These methods often rely heavily on expert knowledge and manual interventions, which limit their scalability and robustness. While advancements in data-driven methodologies such as deep learning have shown promise, there remains a critical gap in effectively integrating domain-specific prior knowledge with advanced computational techniques to achieve higher accuracy and robustness in defect detection and size estimation [5–7].

This research aims to bridge this gap by developing a novel defect inversion method that combines the strengths of traditional domain knowledge and modern deep learning techniques. By leveraging prior knowledge, the proposed method addresses the limitations of purely data-driven approaches, such as sensitivity to noise and lack of interpretability, while maintaining the flexibility and precision offered by deep learning [8–10].

The primary objective of this study was to design a deep cascade pipeline defect inversion algorithm guided by prior knowledge. The key innovations include the following:

1. Developing a preprocessing method based on prior knowledge to address data anomalies and normalize defect sizes [11–13];
2. Constructing a residual network model with a multi-scale hybrid attention mechanism to enhance feature extraction and fusion [14–16];
3. Introducing a cascade structure that integrates domain knowledge with deep learning for adaptive estimation of defect size [17–20].

These contributions not only improve defect detection accuracy but also enhance robustness against noise and other operational uncertainties, providing strong technical support for fault diagnosis and precise maintenance of long-distance pipelines. This research has significant implications for advancing pipeline safety monitoring and operational efficiency in the energy sector.

2. Methodology

2.1. Algorithmic Framework

In this chapter, a deep cascade pipeline defect inversion algorithm guided by prior knowledge is proposed; the overall algorithmic framework is shown in Figure 1. Combining the prior-knowledge-based self-coding network proposed in Chapter 3 with the residual network based on the multi-scale attention mechanism proposed in this chapter, the risk resistance of a single network can be improved and robustness increased through a multi-network parallel approach. At the same time, a large model migration strategy is incorporated to improve the operational speed of the whole network.

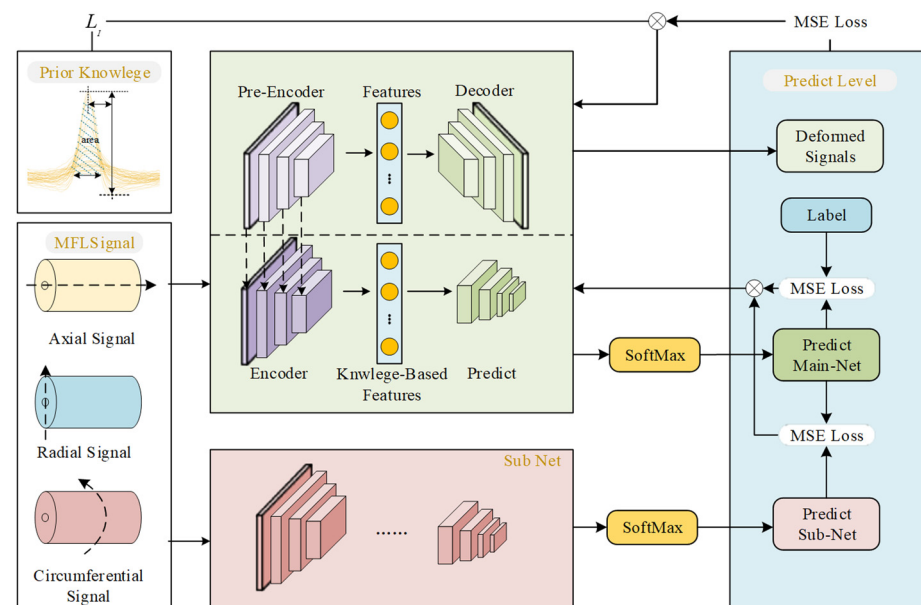


Figure 1. Framework diagram of defect size estimation algorithm for deep cascaded pipeline.

2.2. Deep Residual Network Construction

Figure 2 illustrates the fundamental structure of the residual block used in the ResNet architecture, a core component of the proposed methodology. The residual block has been designed to address the issues of vanishing gradients and degradation in deep neural networks by enabling the network to learn residual mappings instead of direct mappings [21,22]. In this framework, the input feature map undergoes a series of convolutional operations followed by batch normalization and activation functions (e.g., *ReLU*). A skip connection allows the original input to bypass these operations and be added directly to the output of the residual block. This addition operation ensures that the gradient can

flow through the network more effectively during backpropagation, thus alleviating the vanishing gradient problem [23].

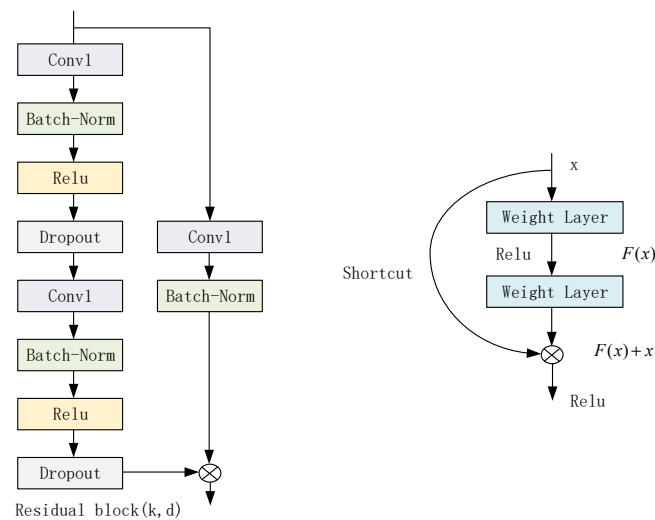


Figure 2. Residual module structure.

By stacking multiple residual blocks, the ResNet architecture can construct deeper networks capable of learning complex feature representations while maintaining stability during training. The residual structure also allows the network to focus on learning the incremental difference (residuals) between the input and the desired output, which simplifies the optimization process. In the proposed methodology, the residual block forms the foundation for capturing multi-scale features and enhancing the accuracy of defect size estimation in pipeline defect detection [24].

During the training process, the residual network can build very deep networks by stacking multiple residual blocks, and it does not suffer from the degradation problem encountered in traditional deep networks. ResNet architecture has become standard for many computer vision and deep learning tasks, such as image classification, object detection, and semantic segmentation.

In a traditional deep neural network, $H(x)$ is assumed to be the mapping function to be learned, where x is the input and $y = H(x)$ denotes the network output. A traditional network would learn the mapping $H(x)$ directly, whereas the ResNet learns the residual mapping $F(x) = H(x) - x$. Thus, the output of the ResNet can be expressed as:

$$y = F(x) + x \quad (1)$$

where y is the output of the residual block, x is the input, and $F(x)$ is the residual mapping, which represents the residual part that the network needs to learn. The ResNet is generated by stacking multiple residual blocks, and when the stacking layer is deepened, each residual block's input at this time turns into the output of the previous residual block; at this time, the residual block can be expressed as follows:

$$output = ReLU(conv(ReLU(conv(input)))) + input \quad (2)$$

$$y_{i+1} = F(y_i, Conv_i) + y_i \quad (3)$$

The basic residual block structure of the ResNet is as follows:

$$output = ReLU(conv(ReLU(conv(input)))) + input \quad (4)$$

where *conv* denotes the convolution operation and *ReLU* denotes the modified linear unit activation function. In each residual block, the two convolutional layers and the *ReLU*

activation function are applied first and then, the output is added to the input and the *ReLU* activation function is applied again.

Residual networks effectively utilize the feature map information at each level by stacking residual blocks, preventing the loss of information and thus maintaining good gradient propagation in deep networks. However, as the depth of the network increases, some redundant information or unnecessary feature responses may be introduced, affecting the learning efficiency and generalization ability of the network.

In the following, an attention mechanism is introduced to further optimize the feature learning process. The attention mechanism allows the network to dynamically adjust its focus in order to concentrate on the extraction and utilization of key information. By introducing the attention mechanism, the network can adaptively assign the weights of different features to increase the focus on important features while suppressing the responses to redundant or irrelevant features.

2.3. Residual Network Modeling Based on Multiscale Hybrid Attention Mechanisms

Attention mechanism (AM) is a technique that mimics the mechanism of human attention and is used in deep learning, especially when dealing with sequential or multi-channel data, to improve the performance and representational capabilities of the model. The mechanism enables the model to dynamically focus its attention on important parts of the input data to learn and utilize the information more efficiently.

The basic idea of the attention mechanism is to mimic the process of allocating human attention when processing information. In deep learning, the attention mechanism allows the model to selectively focus on the most relevant or important parts by assigning different weights to different parts of the input data. This improves the efficiency and accuracy of the model's processing of the input data.

Common attention mechanisms include self-attention, channel attention, spatial attention, and so on.

2.3.1. Self-Attention Mechanism

This type of attention mechanism for sequence modeling, initially applied to the transformer model, is designed to learn the dependencies and importance between different positions in the input sequence [25]. Figure 3 illustrates the basic framework of a self-attention mechanism.

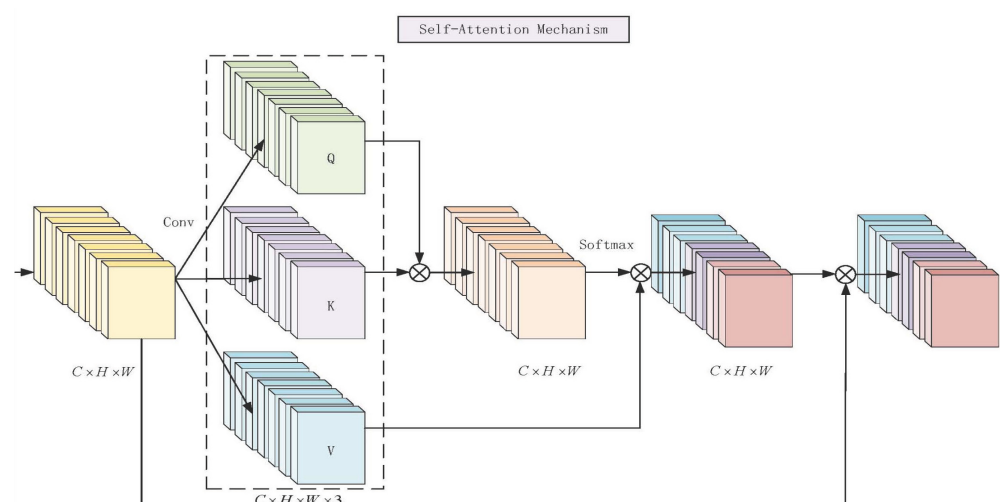


Figure 3. Structure of ECA mechanism.

A self-attention mechanism enables the model to dynamically compute attention weights for each position in the input sequence in relation to all other positions [26]. This mechanism is particularly effective for capturing long-range dependencies, as it allows the

model to focus on the most relevant parts of the sequence for a given position. Specifically, the process involves the following steps:

- (1) Each input element in the sequence is mapped to three distinct vectors—Query, Key, and Value—via learned linear transformations. These vectors represent the core components of the attention mechanism [27];
- (2) For each position, the attention weight is computed by taking the dot product between the Query vector of the current position and the Key vectors of all positions, followed by a softmax operation to ensure normalization. This step determines the relative importance of other positions in the sequence compared with the current position;
- (3) The final representation of each position is obtained by performing a weighted summation of the Value vectors, where the weights are the attention scores computed in the previous step.

By enabling dynamic adjustments based on sequence context, the self-attention mechanism enhances the model's ability to represent complex relationships within sequential data. This capability is vital for tasks requiring the integration of information over extended dependencies, such as pipeline defect detection with spatially distributed features.

(1) Basic Principles

The core of the self-attention mechanism is to obtain a representation of the sequence by computing the attentional weights between each pair of positions in the sequence. Suppose an input sequence $X = [x_1, x_2, \dots, x_i]$ is given, where x_i denotes the feature vector of the i th position in the sequence. The self-attention mechanism maps the feature representation of each position in the sequence to three different spaces by linear transformation: query space (Query), key space (Key), and value space (Value).

(2) Computation of Query, Key, and Value

For each position, calculate the corresponding query vector (Query), key vector (Key) and value vector (Value). The formula for calculating the query vector is as follows:

$$q_i = W_q x_i \quad (5)$$

The formula for the key vector is as follows:

$$k_i = W_k x_i \quad (6)$$

The formula for the value vector is as follows:

$$v_i = W_v x_i \quad (7)$$

where W_q , W_k , and W_v are learnable weight matrices for mapping input features to query space, key space, and value space.

(3) Calculation of attention weights

Calculate the attentional weight a_{ij} between position i and position j :

$$a_{ij} = \text{softmax} \left(\frac{q_i^\top k_j}{\sqrt{d_k}} \right) \quad (8)$$

where d_k is the dimension of the key vector and the softmax function is used to normalize the attentional weights to ensure that the weights sum to 1.

(4) Self-attention representation

The output representation of the self-attention mechanism is obtained by weighted summation of the value vectors using the attention weights:

$$\text{SelfAttention}(X) = \sum_{i=1}^M \left(\sum_{j=1}^N (a_{ij} v_{ij}) \right) \quad (9)$$

This can be simplified to:

$$\text{SelfAttention}(X) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (10)$$

where Q , K , and V are query, key, and value matrices obtained by linear transformation.

2.3.2. Channel Attention Mechanism

Figure 4 illustrates the efficient channel attention (ECA) mechanism, a critical component used to enhance the performance of neural networks, especially in tasks involving multi-channel data such as image processing or speech recognition [28]. This mechanism dynamically adjusts the importance of each channel in the feature map to emphasize the most informative channels and suppress less relevant ones [29].

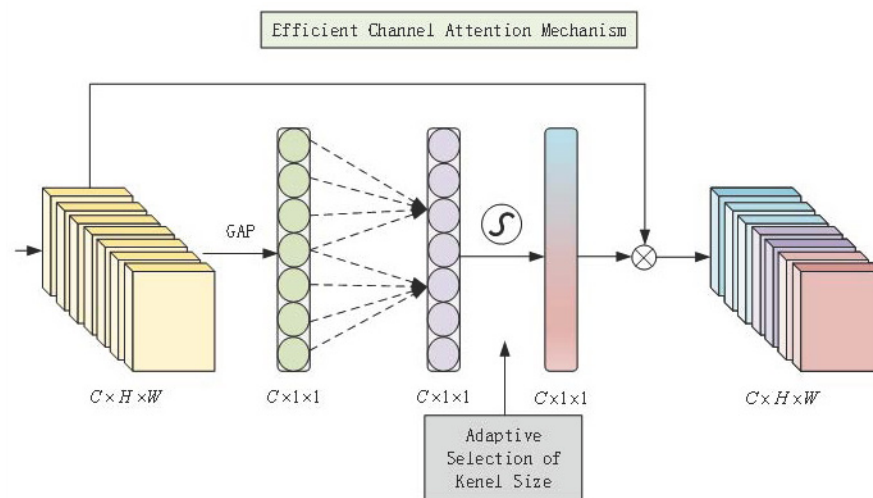


Figure 4. Structure of ECA mechanism.

The ECA mechanism begins by applying global average pooling to the input feature map, which aggregates global contextual information across spatial dimensions while preserving channel-specific information. This step results in a compact global representation for each channel. Instead of relying on fully connected layers to compute channel weights, ECA employs a lightweight, adaptive, one-dimensional convolution with a kernel size determined by the number of channels. This design ensures that channel interactions are efficiently captured without introducing additional computational overhead. The computed weights are then applied to the original feature map through element-wise multiplication, allowing the network to focus on key channels that contribute most significantly to the task [30].

By avoiding the complexity of fully connected layers, the ECA mechanism offers a streamlined and effective solution for channel attention. In the proposed methodology, it enhances the network's ability to extract and utilize critical features, contributing to improved performance in defect detection and size estimation tasks.

(1) Basic Principles

Consider a multi-channel feature $X \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels and H and W are the height and width of the feature graph, respectively. The channel attention mechanism weights the features of each channel by introducing attention weights and computes the weighted feature map to improve the network performance. The ECA mechanism is introduced under the convolution operation in the network. The structure diagram is shown in Figure 4.

The attention module of the channel attention mechanism is involved in two key steps: computing the channel weights and applying the weights to the feature map.

(2) Calculate the channel weights

First, the features of each channel are pooled using the global average pooling operation to obtain the global feature representation of each channel. Suppose the output after the pooling operation is $Z \in \mathbb{R}^C$, where Z_c is the global feature representation of the i th channel. The adaptive one-dimensional convolution kernel k is obtained using Z_c computation:

$$k = \phi(c) = \left\lfloor \frac{\log_2(c)}{\gamma} + \frac{b}{\gamma} \right\rfloor \quad (11)$$

The unnormalized channel weights of size $C \times 1 \times 1$ are obtained by mapping Z itself using the adaptive convolution of $\text{Kernel_size} = k$ with the following formula:

$$Z' = \text{Conv1d}(k, Z) \quad (12)$$

Next, the attention weight $A \in \mathbb{R}^C$ is computed for each channel by introducing an activation function:

$$A = \sigma(W \cdot f_{\text{ReLU}}(Z')) \quad (13)$$

where W is the parameter matrix for learning, f_{ReLU} is the *ReLU* activation function, and σ is the softmax function to ensure that the weights are normalized and interpretable.

(3) Apply weights to the feature map

The computed attention weight A is applied to the original feature map X . The weighted feature map Y is obtained by element multiplication operation:

$$Y_c = A_c \cdot X_c \quad (14)$$

where Y_c is the weighted feature map of the c th channel.

Subsequently, the weighted feature maps Y are used as inputs to the subsequent network layers, thus achieving the purpose of dynamically adjusting the importance of each channel. The channel attention mechanism can effectively improve the performance of neural networks in various tasks, especially when dealing with multi-channel data; it can better capture relationships and importance between features and improve the model's representation ability and generalization performance.

2.3.3. Spatial Attention Mechanism

Figure 5 depicts the spatial attention mechanism, which is widely used in computer vision tasks to enhance model performance and generalization. The primary goal of this mechanism is to efficiently identify and focus on important regions within an image by learning the dependencies between different spatial locations. By dynamically adjusting the feature responses at various spatial positions, the mechanism helps the network concentrate on critical areas while ignoring less relevant information.

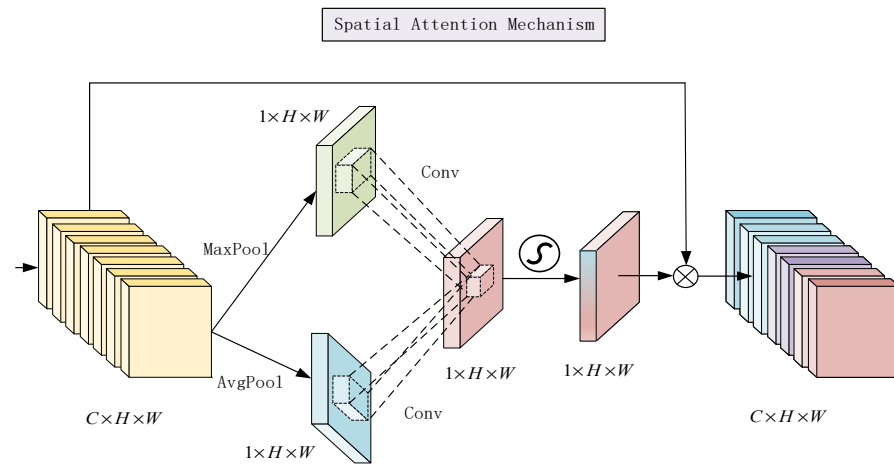


Figure 5. Structure of SAM mechanism.

The spatial attention mechanism typically operates by first applying global average pooling and global max pooling to the input feature map along the channel dimension. This results in two spatial feature maps that encode complementary information about the input data. These maps are then concatenated and passed through a convolutional layer to produce a spatial attention map, which highlights the most important regions in the image. The attention map is subsequently applied to the original input feature map via element-wise multiplication, enhancing the network's focus on key spatial regions.

In the proposed methodology, the spatial attention mechanism plays a crucial role in optimizing feature extraction, particularly in tasks involving complex spatial patterns, such as defect detection. By enabling the model to adaptively emphasize relevant spatial features, this mechanism significantly contributes to the accuracy and robustness of the pipeline defect size estimation process.

(1) Basic Principles

The basic principle of the spatial attention mechanism is to dynamically adjust the feature representation of each location by calculating the attention weights of different locations in the image. In CNNs, the spatial attention mechanism is usually combined with the channel attention mechanism to enhance the model.

(2) Calculation of positional attention weights

Given an input feature map $X \in \mathbb{R}^{H \times W \times C}$, where H and W denote the height and width of the feature map, respectively, and C denotes the number of channels (number of features), for each location (h, w) according to the spatial attention mechanism, the formula for calculating the attention weight a_{hw} is as follows:

$$a_{hw} = \text{softmax}(f(x_{hw})) \quad (15)$$

where f is a mapping function, usually an operation containing convolution and nonlinear activation functions, used to extract the feature representation x_{hw} at position (h, w) . Here, the global maximum pooling operation and the global average pooling operation are used, and the two feature maps obtained are spliced with the following formula:

$$f(x_{hw}) = \text{AvgPool}(x_{hw}) + \text{MaxPool}(x_{hw}) \quad (16)$$

The feature responses are normalized by a softmax function to ensure that the attention weights sum to 1.

(3) Spatial attention representation

The formula for characterizing the output of the spatial attention mechanism by weighted summation of the input feature maps using the attention weights is shown below:

$$\text{SpatialAttention}(X) = \sum_{h=1}^H \sum_{w=1}^W a_{hw} x_{hw} \quad (17)$$

The spatial attention mechanism focuses on important regions and can automatically learn the location of important regions in the image and dynamically adjust the feature representation, which helps to improve the model's focus on key regions. The spatial attention mechanism can also enhance the model's ability to represent the input image and adapt to features of different scales and locations.

The whole multi-scale hybrid attention network model is shown in Figure 6. The residual network consists of two parts, the main branch of forward propagation and the sub-branch of jump propagation. The sub-branch transfers the feature map information in layer i to layer $i + 2$ and fuses the feature information of the network in layer $i + 1$. In the whole network, sub-branch feature maps exist at different scales from the feature maps of the main branch, and the dependencies in the feature maps are strengthened by using the self-attention mechanism for both the sub-branches and the main branch of the network, respectively, to handle feature maps of different scales. In the face of multi-scale feature maps, the larger-scale feature maps are downsampled to reduce the redundancy of the whole network and unify the feature scales.

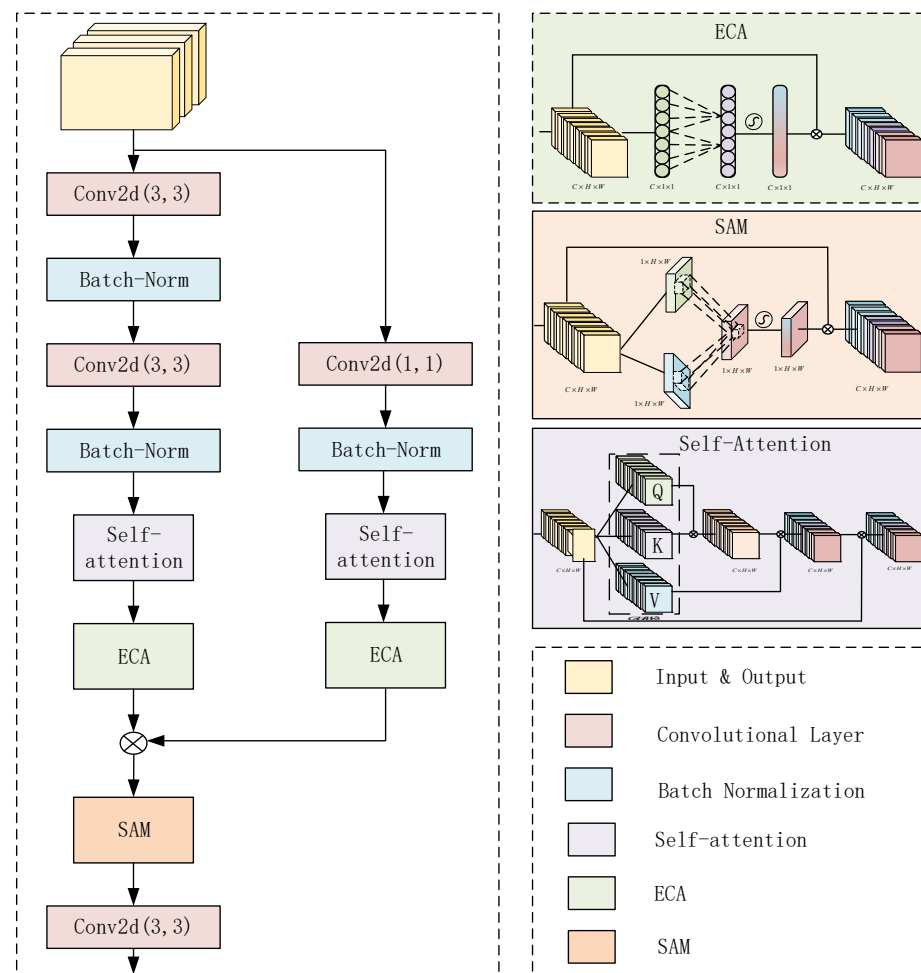


Figure 6. Multi-scale hybrid attention network model.

Considering the structure of the residual network, while the addition of sub-branches introduces certain redundant information or unnecessary feature responses, in order to deal with such redundant information and ensure that the network focuses on important features, a channel attention mechanism is introduced in the feature information fusion part. The feature map information of layer i and the feature information of layer $i + 1$ are regarded as the feature information of different channels, respectively, and the multiple weights of the channel features are dynamically adjusted to facilitate the combination of the two parts.

After obtaining the fused residual features, the spatial attention mechanism is used to strengthen the focus on the key feature information; the spatial attention mechanism is also used to adjust the whole network's residual fusion features, which makes better use of the key feature information and further optimizes the feature extraction process of the network.

2.4. Research on Network Fusion Algorithms

In order to improve the performance of the whole network, a cascade learning approach has been designed to combine the two networks by using a sparse stacked self-coder network based on prior knowledge as the main network and a residual network model based on a multiscale hybrid attention mechanism as the sub-network, via a process of network fusion.

The more common network fusion methods include feature-level fusion, which is realized by simple feature concatenation, weighted summation, splicing, or other complex feature fusion strategies; model-level fusion, which is realized by fusing or combining the weights of two models, or by connecting two models' layers; voting fusion, which involves votes on the predictions of the two networks and selection of the category with the highest number of votes as the final prediction; and stacked fusion, which stacks the outputs of the two networks and feeds these stacked outputs as new inputs into a separate layer or network.

Considering that feature-level fusion and model-level fusion undoubtedly bring new burdens to the network and increase the difficulty of network training, the following discussion compares two other types of network fusion algorithm (voting fusion and cascade fusion).

2.4.1. Voting Fusion Methods

Voting fusion algorithms are a common integrated learning method used to combine the predictions of multiple basic classifiers or models to improve the accuracy and stability of the overall prediction. Common voting fusion algorithms include hard voting, which uses multiple classifiers, where each basic classifier makes a categorical prediction for an input sample and the final prediction is determined by simple majority voting, i.e., the category that receives the most votes is selected as the final output. Soft voting is similar to hard voting [31], with the difference that it considers the probability value of each classifier's prediction, where each basic classifier provides a prediction probability for the input sample of each category, and the final prediction is based on a weighted average or vote of the predicted probabilities of each classifier; this method is usually used for integration of probabilistic predictions. In weighted voting, the voting weight of each basic classifier can be assigned based on its performance or confidence; more accurate classifiers can be assigned higher weights to increase their impact in integration. Adaptive voting algorithms dynamically adjust the weights or voting strategies of basic classifiers based on their performance on validation datasets [32], a method that can be automatically adjusted based on the performance of the classifiers to improve the performance and stability of the integrated model.

These voting fusion algorithms can be selected and adapted according to task requirements and data characteristics for building more robust and robustly integrated models.

Outcome majority voting is similar to real-life multi-person voting, but it is necessary to ensure that the number of fused models is odd in order to ensure that a clear voting result can be produced, which may lead to undetermined results if the number of models is even.

Direct averaging of results aims to average the predictions of individual models to improve the predictive power of the overall model; unlike majority voting of results, direct averaging of results does not emphasize the strengths of individual models but can compensate for the weaknesses between models. For example, if a model is overfitting and another model is underfitting, averaging can combine the two and effectively prevent overfitting and underfitting, but the direct averaging method finds it difficult to reflect the advantages of one of the models for individual data, which leads to limited improvement in accuracy.

Taking the adaptive voting method as an example, the steps and formulas are shown below:

(1) Model training and evaluation:

Train several different neural network models, e.g., model $M_1, M_2, \dots, M_k$, where k is the number of models. In this paper, $k = 2$. The performance of each model is evaluated on a validation set or a cross-validation set, using accuracy Acc_i or other evaluation metrics such as error Err_i .

(2) Calculate the model weights:

Based on the performance evaluation of the models, the weights of each model w_i are calculated. A common approach is to use the inverse of the performance metrics as the weights of the models, i.e.,:

$$w_i = \frac{1}{Acc_i} \quad (18)$$

Doing so allows better performing models to receive higher weights.

(3) Weighted voting:

For a given test sample x , each model M_i provides one prediction:

$$y_i = M_i(x) \quad (19)$$

The predictions of each model are weighted using the model weights w_i to obtain the weighted prediction $\hat{y}(x)$:

$$\hat{y}(x) = \sum_{i=1}^k w_i y_i = \sum_{i=1}^k \frac{y_i}{Acc_i} \quad (20)$$

where y_i is the prediction of the model $M_i(x)$ and w_i is the weight of the model M .

Through this process of weighted voting, the prediction results of the model are influenced by the performance of multiple models, and the model with better performance has more influence in the voting, thus making the overall model prediction results more accurate and stable. With this method, due to the large difference in the inversion effect of each base network on different datasets, if weighted voting is used, it is easy to make the overall network biased in the direction of poorer accuracy.

2.4.2. Cascade Fusion Methodology

The network cascade approach is a strategy commonly used in deep learning to form a cascade structure by connecting multiple models or sub-models in a sequential manner to solve complex tasks or improve model performance. This approach is commonly used to process complex input data or tasks to improve model performance and generalization by processing the data in stages or extracting features step by step. A multi-stage processing approach can generally be used, where the whole task is decomposed into multiple stages or sub-tasks and different models are designed to handle each stage. The output of each stage is used as the input for the model in the next stage, forming a staged processing flow. In a feature pyramid approach [33], different levels of features are extracted at different

network stages to form a feature pyramid. Features extracted at lower stages may be more localized and detailed, while features extracted at higher stages are more abstract and global, such that the feature pyramid helps to better represent complex data patterns. Information flow transfer methods allow information to flow between different stages, and the output of each stage can be used as input in the next stage to progressively extract higher level features or solve more complex subtasks. This information flow transfer helps the model to learn complex tasks better. A model integration approach can improve overall performance by integrating multiple models. Different model structures or parameter settings can be used at each stage and are eventually integrated into an overall model to achieve better prediction or classification results. The task decomposition approach, where a complex task is decomposed into multiple simple tasks, each of which is processed by a separate model, is a processing strategy that helps to improve the interpretability and generalization of the model. End-to-end training models can be trained in an end-to-end manner, meaning that the entire network can be trained by joint optimization without the need to train each stage of the model separately.

In the procedure presented in this paper, in order to simplify the model so that a single base network has the feature extraction capability of the whole cascade network, inspired by the large model migration approach, the knowledge learned from a pre-trained large neural network model in one task or domain is migrated to another related task or domain, and the choice has been made to use the method of knowledge distillation to combine the two networks. The whole framework of knowledge distillation is shown in Figure 7.

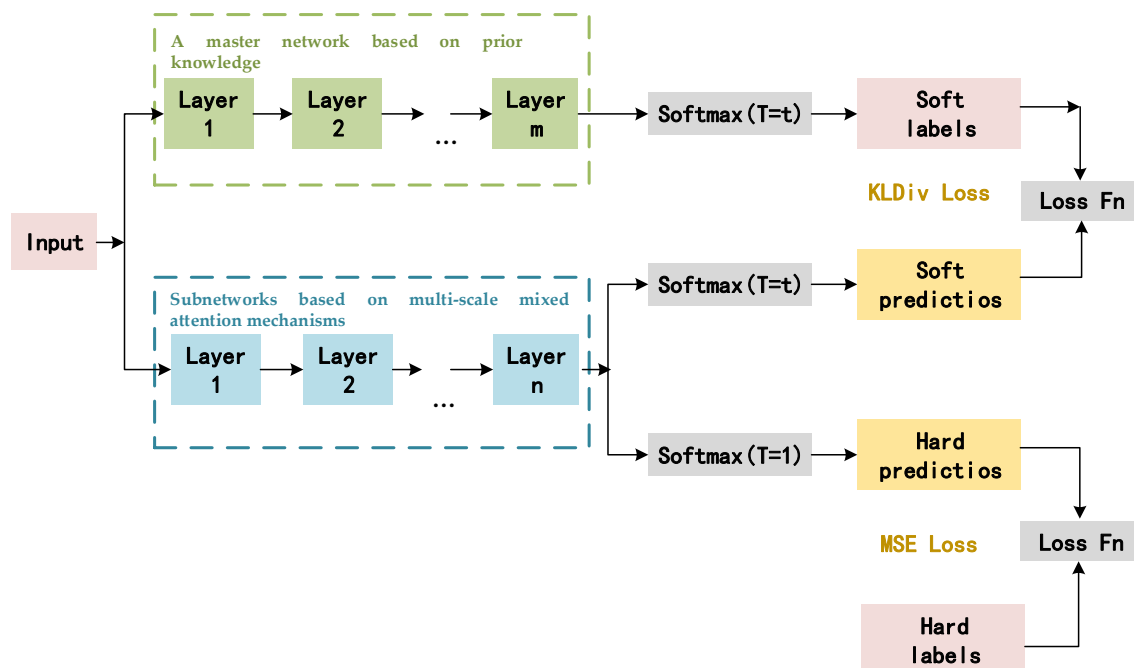


Figure 7. Knowledge distillation.

In knowledge distillation, it is often necessary to set up an instructor network and a student network, where the instructor network has richer knowledge and high performance while the student network aims to learn from the instructor network and keep running relatively fast. In this study, we set the main network based on prior knowledge as the teacher network and the residual sub-network based on a multi-scale hybrid attention mechanism as the student network.

A teacher network based on prior knowledge was set as f_{Tech} and a residual student network based on a multi-scale hybrid attention mechanism f_{Stu} . They aimed to learn the same task, and the student network aimed to learn from the teacher network through a distillation process based on the teacher network's feature representation ability.

The core idea of knowledge distillation is to train student networks using soft targets of the teacher network, which can be the output probability distribution of the teacher network or other representations that can convey more information and knowledge. The training goal of the student network is to minimize the distance between its predictions and the output of the teacher's network, and the difference in predictions is usually measured using cross-entropy or other distance metrics. In this paper, we use the KL (Kullback–Leibler) scatter to compute the distributional error between the predictions of the student network and the teacher network.

KL scatter, also known as relative entropy, is used to measure the difference or distance between two probability distributions. The formula for calculating KL scatter for discrete random variables is shown below.

Suppose there are two discrete probability distributions P and Q with probability mass functions $p(x)$ and $q(x)$, respectively; the KL dispersion is given by:

$$D_{KL}(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \quad (21)$$

where $D_{KL}(P||Q)$ denotes the KL dispersion from probability distribution Q to probability distribution P .

Given the input sample x , the output of the teacher network f_{Tech} is $z_{Tech} = f_{Tech}(x)$, and the output of the student network f_{Stu} is $z_{Stu} = f_{Stu}(x)$. Soft label $\tilde{z}_{Tech}$ is used to represent the output of the teacher network, and soft label $\tilde{z}_{Stu}$ is used to represent the output of the student network. The soft labels were obtained by calculating the probability distribution of the network label using softmax at temperature $T = t$, that is, “softening” the network label. At the same time, softmax was used to “harden” the student network label at temperature $T = 1$. The student network was trained by minimizing the following loss function:

$$\begin{aligned} Loss &= KLDiv(\tilde{z}_{Tech}, \tilde{z}_{Stu}) + MSE(z_{Stu}, z_{real}) \\ &= D_{KL}(\tilde{z}_{Tech}||\tilde{z}_{Stu}) + Dis(z_{Stu}, z_{real}) \end{aligned} \quad (22)$$

where $KLDiv$ represents the loss function for calculating KL divergence, calculating the KL divergence between the student network soft label $\tilde{z}_{Stu}$ and the teacher network soft label $\tilde{z}_{Tech}$, and MSE represents the mse loss between the predicted label and the real label, that is, the distance between the two labels. This training process allows the student network to learn effectively from the knowledge of the teacher network, improving the model's generalization ability and performance.

The process of knowledge distillation can add additional regularization terms or adjust temperature parameters during training to control the distribution of soft targets in order to further optimize the learning effect of the student network.

In cascading learning mode, sub-networks can enhance the performance and stability of the main network. By constructing networks of different depths and using them in parallel, the overall network can learn multi-scale features. This strategy fully utilizes the advantages of multiple base networks, enabling the network to integrate the empirical feature extraction capabilities of the manually designed main network, while also fully tapping into the feature extraction potential of deep networks. Due to the potential problem of insufficient information when a single architecture base network completes prediction tasks, preliminary experiments have shown that deep sub-networks based on channel attention mechanisms with different architectures from the main network perform better in certain situations. In addition, due to its concentrated focus, a single network may perform poorly in dealing with various changes and uncertainties. By combining multiple base networks, the overall robustness of the network can be significantly improved, thereby better responding to various interferences. Finally, the strategy of integrating two networks not only helps to reduce network complexity while achieving the same functionality, but also makes the network more lightweight.

In summary, by constructing parallel sub-networks to enhance the performance and stability of the main network, this multi-scale feature learning strategy combines the advantages of prior knowledge and deep learning. By utilizing feature representations at different levels, the network's performance in complex tasks is improved. At the same time, the introduction of deep sub-networks based on the channel attention mechanism and the combination of multiple networks provide comprehensive improvements in network performance, robustness, and lightweight design.

3. Experimental Validation

3.1. Description of Experimental Data

The experimental data consisted of 479 samples, each with a size of 88×50 , derived from real pipeline data collected in northern China. In this part of the study, the advanced network based on prior knowledge was combined with the network based on multi-scale attention mechanism proposed in this paper by using cascade expressions to improve the accuracy and robustness of a single network, and the network performance was verified by comparative experiments. The software experimental environment used in this paper was based on the TensorFlow Keras 2.10.0 framework, while the hardware experimental environment consisted of an NVIDIA GeForce RTX 3070 GPU (8 GB) manufactured by NVIDIA Corporation in Santa Clara, CA, USA, and a 12th Gen Intel(R) Core(TM) i7-12700KF CPU manufactured by Intel Corporation in Santa Clara, CA, USA.

3.2. Comparative Experiment and Performance Analysis

In order to evaluate the effectiveness of the prior knowledge-based deep cascade pipeline defect inversion method, referred to as Model 3, the proposed model was compared with several advanced models. These included the adaptive defect inversion method based on prior knowledge (referred to as Model 1), which utilizes domain-specific knowledge to adapt to the unique characteristics of defects in the dataset, providing a baseline for comparison in terms of both accuracy and interpretability. Additionally, the residual network with a multi-scale mixed attention mechanism proposed in this chapter (referred to as Model 2) was included in the comparison. This model focuses on capturing complex patterns in the data by employing a residual network architecture, combined with attention mechanisms that dynamically emphasize important features across multiple scales, thereby enhancing the model's ability to focus on relevant defect patterns. Furthermore, stacking random forest (Stacking RF), an ensemble learning method, was used for a comparison. Stacking RF aggregates the predictions of several base learners, such as decision trees, in a stacking framework to improve overall prediction accuracy, offering a traditional baseline to evaluate against the deep learning-based methods. The experimental results are presented based on two additional approaches: the dominant feature inversion algorithm (Dominant), which focused on identifying the most important features contributing to defect detection, and an algorithm that merged Model 1 and Model 2 using a voting method (referred to as Voting), providing a comparison of performance with that of combined models.

3.2.1. Precision Comparison Experiment

The results of precision experiments are very important for evaluating the performance of the proposed algorithm. Through precision experiments, we can gain insight into the accuracy and reliability of the algorithm in pipeline defect detection and quantitative analysis tasks.

The proposed algorithm was compared with other comparison algorithms across the following five indexes: the mean error of defect inversion (MAE), standard deviation of the defect inversion error (Std), the defect inversion error with 80% confidence interval (Pof0.8), the defect inversion error with 90% confidence interval (Pof0.9), and the root mean square error of the defect inversion (RMSE). The specific experimental results are shown in the radar map in Figure 8.

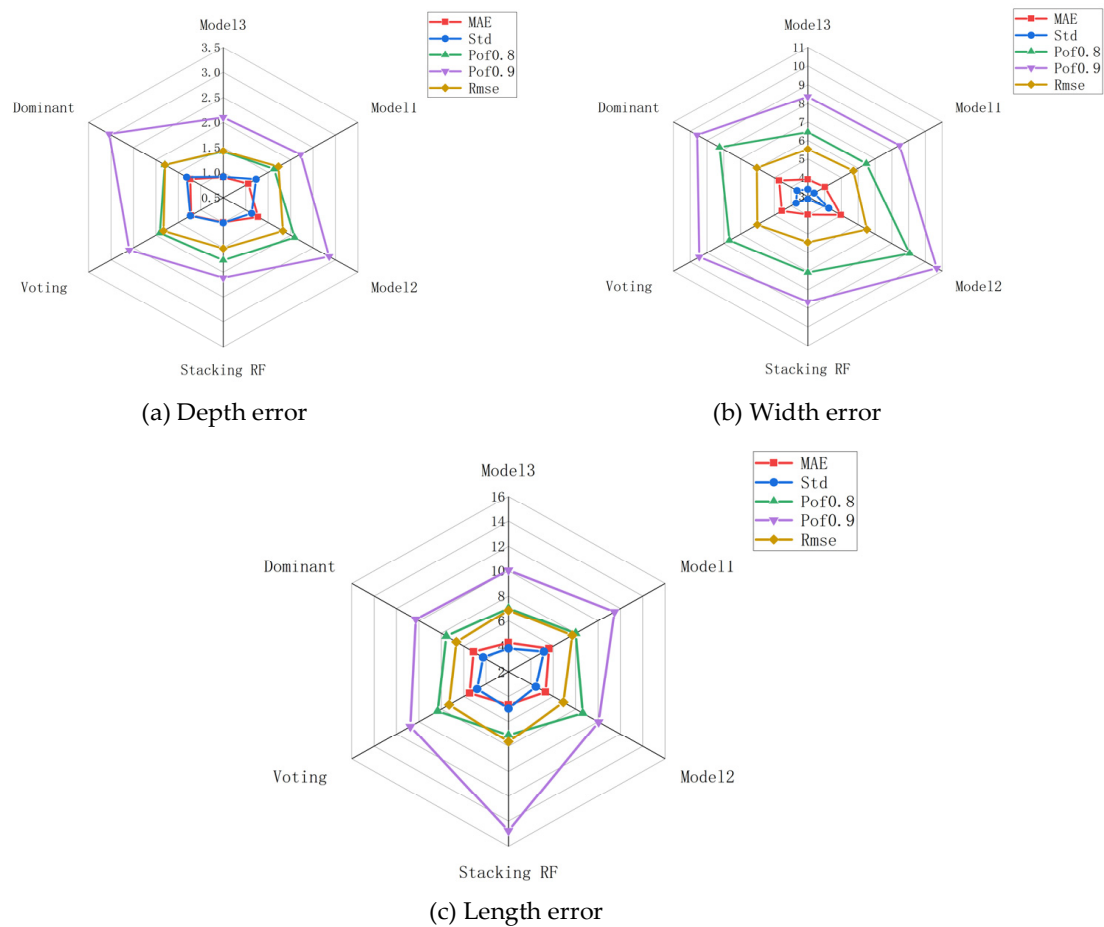


Figure 8. Accuracy comparison.

It was observed that the proposed algorithm was superior to the other compared algorithms in different indexes of length, width, and depth labels. In the depth label, the average error of defect inversion of the proposed algorithm was 0.9070 mm, while the average error of the Dominant algorithm was 1.2334 mm. The performance of the proposed algorithm improved by 26.46%, and the average error of stacked random forest was 0.9961 mm, an improvement of 8.95%.

Compared with Model 1 and Model 2, the average error of these two algorithms was 1.0571 mm and 1.2777 mm, respectively, and the proposed algorithms were both improved, by 14.2% and 29.02%, indicating that the proposed algorithm combined the advantages of the two algorithms and successfully improved the accuracy of defect inversion. Compared with the average error of the voting method, at 1.2174 mm, the proposed method was improved by 25.50%, showing the performance of the cascade expression involved. It effectively combined the advantages of the two networks, combining the prior feature-based knowledge of the prior knowledge network with the deeper feature knowledge of the deep residual network, and further improved the performance of the algorithm.

The specific experiments are shown in Table 1, and the best performance is highlighted in bold in the table.

Compared with the other methods, the proposed method also performed well under different confidence intervals. At an 80% confidence interval, the inversion error of the proposed method was 1.4189 mm, while the error values of Model 1, Model 2, stacked random forest, Voting, and Dominant were 1.6329 mm, 2.0886 mm, 1.7555 mm, 1.9108 mm, and 1.8016 mm, respectively. At a 90% confidence interval, the performance of the proposed algorithm was also superior to the other comparison algorithms, effectively verifying that the designed algorithm successfully combined the two algorithms by using cascade

expressions to obtain features with stronger representation ability, and also demonstrating the necessity of guiding the network with prior knowledge for deeper mining.

Table 1. The results of the accuracy experiment.

Target	Algorithm	MAE (mm)	Std (mm)	Pof0.8 (mm)	Pof0.9 (mm)	RMSE (mm)
Depth	Model 3	0.9070	0.9150	1.4189	2.0988	1.4286
	Model 1	1.0571	1.2315	1.6329	2.2253	1.7312
	Model 2	1.2777	1.1323	2.0886	2.8601	1.8300
	Stacking RF	0.9961	1.0047	1.7555	2.1115	1.5137
	Voting	1.2174	1.2319	1.9108	2.5927	1.8306
	Dominant	1.2334	1.3135	1.8016	3.0411	1.7968
Width	Model 3	3.9104	3.3918	6.4499	8.3688	5.5203
	Model 1	4.0137	3.3755	6.4835	8.4615	5.7192
	Model 2	4.9642	4.2453	9.0646	10.6794	6.5175
	Stacking RF	3.9657	3.1402	7.0677	8.6550	5.4360
	Voting	4.5409	3.6983	7.6620	9.4584	6.0062
	Dominant	4.7110	3.6406	8.2523	9.5863	6.0341
Length	Model 3	4.2721	3.8386	7.0124	10.0435	6.8556
	Model 1	5.6233	5.1577	8.0102	11.4795	7.7025
	Model 2	5.3068	4.4455	8.6370	10.0607	6.9079
	Stacking RF	4.6642	4.9473	7.0806	14.7612	7.6246
	Voting	5.4701	4.8066	8.3286	10.7751	7.3102
	Dominant	5.1243	4.2705	7.5631	10.2847	6.6563

The standard deviation (Std) metric is often used to measure the dispersion of the values in a dataset, and it provides information about the distribution of the data, thus helping to evaluate the stability and consistency of the model. The root mean square error (RMSE) index is a common index to measure the difference between the predicted value and the actual value. It was observed that the algorithm proposed in this paper did not reach the optimal level under the Std and RMSE indexes, but its performance was still satisfactory in relative terms, which further proves that the algorithm has good stability. The effective combination of the two methods makes the algorithm more applicable in industrial scenarios.

3.2.2. Robustness Comparison Experiment

In a pipeline, factors such as media flow, vibration, collision, and disturbance in the pipeline can affect the magnetic leakage (MFL) signal collected by the in-pipeline detector (PIG) running in the pipeline, resulting in strong background noise in the signal. In order to evaluate the anti-noise capability of the system, a series of comparative experiments were carried out. In order to compare the sensitivity of different inversion methods to noise, three-order white Gaussian noise was introduced into the test samples, and the SNR was set to 20, 10, and 5 dB, respectively. The error distribution of noiseless defect inversion is presented in the form of histogram, and the results obtained under different noises are displayed in the form of error distribution, marked with different colors.

It can be seen from Figure 9 that the proposed inversion algorithm performed best when there is no noise. In theory, with the decrease in SNR, the features of the MFL signal become more difficult to extract, which leads to degradation of the performance of the inversion algorithm. It can be seen from the figure that with the addition of noise, the error of the proposed algorithm changed little. Compared with other comparison algorithms, the error of the proposed algorithm on the three labels of length, width, and depth was smaller than that of the others, which means that the proposed algorithm has good robustness.

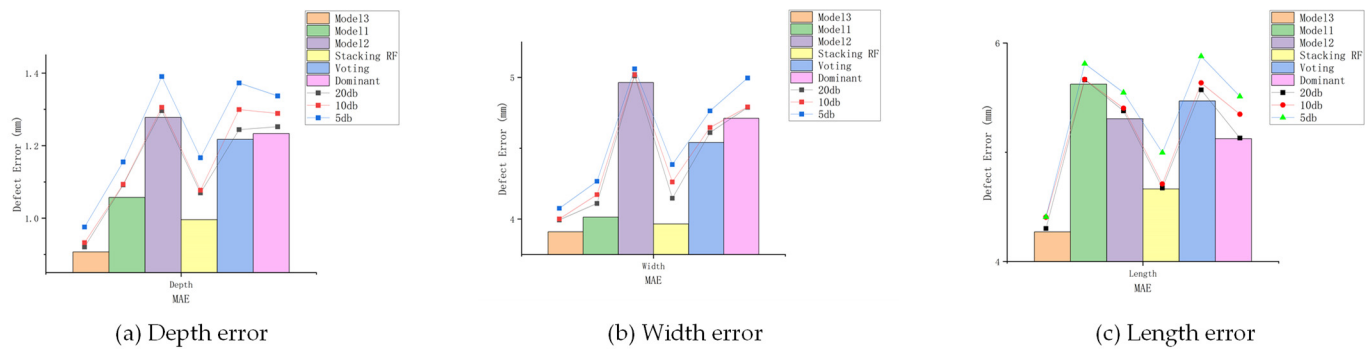


Figure 9. MAE comparison of inversion errors under different noise levels.

Taking depth labeling as an example, the average error of the proposed method under 20 db noise was 0.9208 mm, while the average error of stacked random forest inversion in ensemble learning was 1.0918 mm, representing a 15.66% improvement in the performance of the proposed algorithm. The voting method had an average error of 1.2442 mm, while the dominant feature inversion method had an error of 1.2522 mm. It was observed that the proposed algorithm effectively improved the anti-interference ability of the network by mining the deep features and using the parallel method of multiple base networks.

Meanwhile, as shown in Figures 10 and 11, comparing Pof0.8 and Pof0.9 indexes, we found that the proposed method showed small errors under different confidence intervals after the addition of noise, indicating a robust network under strong interference, and its performance was far better than the other methods. This method achieved better performance by combining prior knowledge and abstract features. At the same time, under the input of damaged signal (similar to noise), the method was able to reconstruct the original signal well, and learn strong characteristics from it, with stronger adaptability to non-stationary signals. This feature enhances the recognition and anti-interference ability of MFL signal under noisy working conditions, which is of great significance to improving the accuracy and reliability of pipeline health monitoring and can provide strong support for engineering practice and scientific research in related fields.

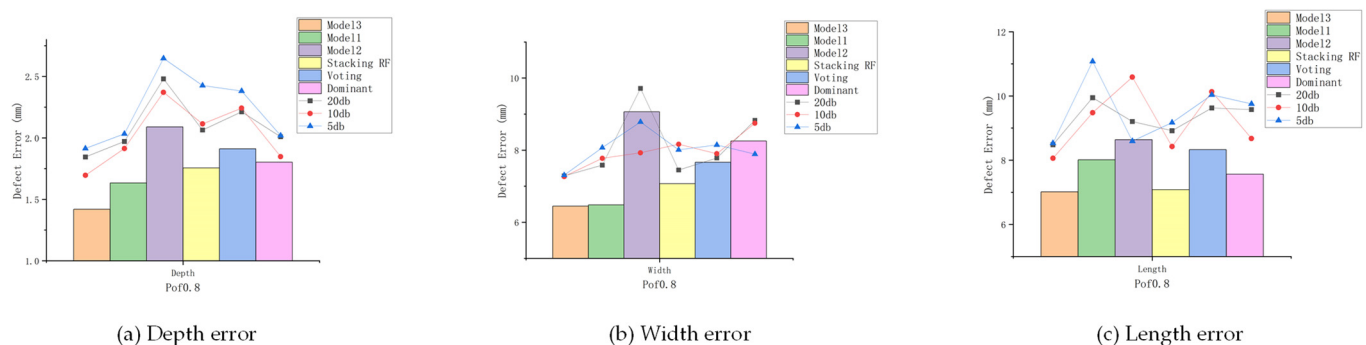


Figure 10. Pof0.8 comparison of inversion errors under different noise levels.

The global robustness inversion results are listed in the Table 2 below. With the introduction of noise, the inversion accuracy of defect data in all models decreased. However, it is difficult to summarize the variation rules of the other defect indexes. Under different noises, the average error, 80% confidence error, and 90% confidence error of the proposed algorithm were lower than the other algorithms, which indicates the algorithm's better performance and applicability than other algorithms. However, through the RMSE index, it was observed that the algorithm still had the problem of uneven error distribution. Even if the inversion effect of some defective data is good, there are still situations where the inversion error of specific data is large; so, the algorithm still needs to be further improved. The main reasons for this analysis include (1) the limitations of industrial pipeline data,

leading the algorithm to focus on typical data and ignore small-sample data, and (2) lack of optimization and adjustment of algorithms in a specific noise environment.

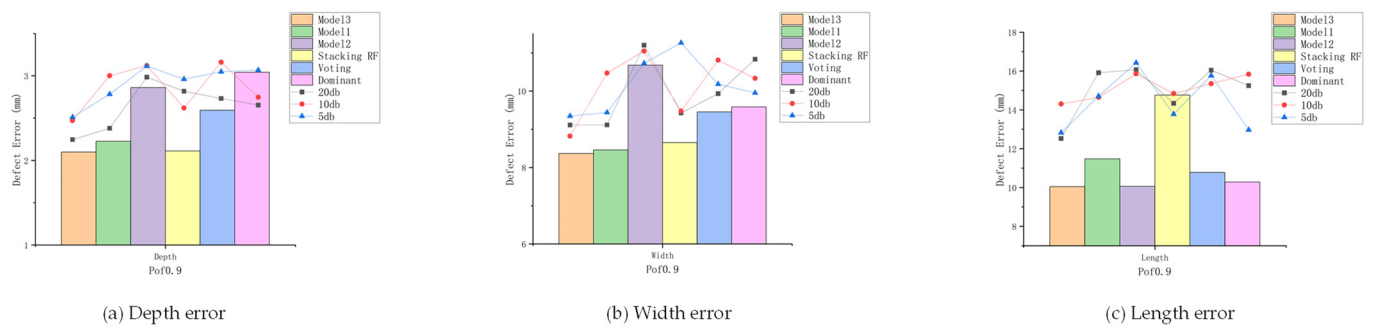


Figure 11. Pof0.9 comparison of inversion errors under different noise levels.

Table 2. The results of inverting different indicators under different noise levels.

Noise	Target	Algorithm	MAE (mm)	Std (mm)	Pof0.8 (mm)	Pof0.9 (mm)	RMSE (mm)
20 db	Depth	Model 3	0.9208	0.8910	1.8446	2.2454	1.4553
		Model 1	1.0918	0.8069	1.9710	2.3803	1.4546
		Model 2	1.2965	1.3062	2.4799	2.9843	1.9177
		Stacking RF	1.0698	1.0584	2.0640	2.818	1.6242
		Voting	1.2442	1.1065	2.2123	2.7323	1.7362
		Dominant	1.2522	1.1330	2.0090	2.6539	1.6847
	Width	Model 3	3.9940	2.9159	7.2851	9.1142	5.7537
		Model 1	4.1100	2.8873	7.5851	9.1162	5.4407
		Model 2	5.0111	4.1752	9.7122	11.2009	6.6503
		Stacking RF	4.1471	3.0654	7.4507	9.4279	5.5058
		Voting	4.6105	3.3082	7.7823	9.9355	5.8225
		Dominant	4.7883	3.5833	8.8275	10.8362	6.2882
	Length	Model 3	4.3027	4.7623	8.4819	12.5266	6.4503
		Model 1	5.6640	5.6895	9.9463	15.9137	8.5408
		Model 2	5.3796	5.3776	9.2048	16.0727	8.2434
		Stacking RF	4.6742	3.4222	8.9186	14.3370	7.9483
		Voting	5.5718	5.5836	9.6255	16.0432	8.4421
		Dominant	5.1308	4.5571	9.5773	15.2480	7.6621
10 db	Depth	Model 3	0.9327	0.9705	1.6954	2.4703	1.4167
		Model 1	1.0935	0.9655	1.9137	3.0010	1.5697
		Model 2	1.3055	1.4466	2.3709	3.1206	2.0207
		Stacking RF	1.0771	1.0268	2.1139	2.6196	1.6608
		Voting	1.2995	1.3060	2.2423	3.1608	1.8952
		Dominant	1.2888	1.1080	1.8476	2.7473	1.6958
	Width	Model 3	4.0007	2.9182	7.2690	8.8244	5.2448
		Model 1	4.1723	4.1355	7.7747	10.4724	6.2289
		Model 2	5.0205	4.3530	7.9276	11.0491	6.6300
		Stacking RF	4.2620	2.8794	8.1651	9.4785	5.5286
		Voting	4.6464	4.2942	7.9012	10.8107	6.4794
		Dominant	4.7917	3.3774	8.7484	10.3351	6.0483
	Length	Model 3	4.4058	4.5423	8.0600	14.3044	7.6244
		Model 1	5.6674	5.5190	9.4794	14.6466	8.7115
		Model 2	5.4032	6.0442	10.5881	15.8582	8.8197
		Stacking RF	4.7096	4.9377	8.4254	14.8403	7.8306
		Voting	5.6353	5.8816	10.1338	15.3524	8.8656
		Dominant	5.3484	5.9987	8.6753	15.8386	8.8311

Table 2. Cont.

Noise	Target	Algorithm	MAE (mm)	Std (mm)	Pof0.8 (mm)	Pof0.9 (mm)	RMSE (mm)
5 db	Depth	Model 3	0.9757	0.8059	1.9142	2.5110	1.7238
		Model 1	1.1550	1.2977	2.0336	2.7823	1.7935
		Model 2	1.3905	1.4471	2.6474	3.1151	2.0014
		Stacking RF	1.1667	1.1669	2.4264	2.9613	1.3846
		Voting	1.3728	1.4724	2.3808	3.0487	1.9975
		Dominant	1.3370	1.2413	2.0189	3.0678	1.8200
	Width	Model 3	4.0759	3.5751	7.3064	9.3469	5.5555
		Model 1	4.2668	3.1702	8.0695	9.4419	6.0015
		Model 2	5.0608	4.3093	8.7817	10.7307	6.7095
		Stacking RF	4.3849	4.0916	8.0102	11.2654	6.3517
		Voting	4.7638	4.0422	8.1441	10.1863	6.4555
		Dominant	4.9958	3.9584	7.8896	9.9534	6.4816
	Length	Model 3	4.4091	4.6427	8.5932	12.8268	7.2056
		Model 1	5.8113	4.5808	11.0762	14.7053	7.7065
		Model 2	5.5463	6.0021	8.8291	16.4260	8.6409
		Stacking RF	4.9968	3.6908	9.1702	13.7775	7.0001
		Voting	5.8788	5.4914	10.0347	15.7656	8.3737
		Dominant	5.5112	4.6363	9.7542	12.9718	7.7147

4. Conclusions

This study addresses critical gaps in pipeline defect detection, particularly the limitations of traditional inversion methods in handling complex defect characteristics, noise, and reliance on manual expertise. Motivated by these challenges, we developed a prior-knowledge-guided deep cascade pipeline defect inversion method. The proposed approach integrates domain-specific knowledge with advanced deep learning techniques, leveraging a residual network with a multi-scale hybrid attention mechanism and a cascade learning framework to enhance the accuracy and robustness of defect size estimation.

The main findings of this research are summarized as follows. Firstly, the proposed method demonstrated superior performance in experimental evaluations, achieving higher accuracy in defect size estimation compared with traditional methods and baseline models. Secondly, robustness tests under various noise conditions confirmed the method's ability to maintain high accuracy and reliability, outperforming competing approaches in both mean error and confidence interval metrics. Thirdly, the cascade learning framework effectively combined domain-specific prior knowledge with deep learning, enabling efficient defect inversion with minimal dependence on manual intervention. Finally, the integration of multi-scale hybrid attention mechanisms significantly improved the network's feature extraction and fusion capabilities, resulting in more precise defect characterization.

Despite these promising results, this study has certain limitations. The method's performance is influenced by the quality and diversity of the training data, and further optimization is required for specific noise environments and rare defect scenarios. Additionally, the computational complexity of the proposed model warrants future work to enhance its efficiency for real-time applications. Nonetheless, the findings of this research have significant implications for engineering and industrial applications. By combining deep learning and domain knowledge, the proposed approach provides a practical and scalable solution for pipeline safety monitoring and defect assessment. This method offers strong technical support for improving the safety and operational efficiency of long-distance pipelines in the energy sector.

Author Contributions: Conceptualization, R.L.; Methodology, F.W.; Validation, K.F.; Writing—original draft, P.C.; Writing—review & editing, W.X.; Project administration, R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: For reasons of privacy and confidentiality, we regret that we cannot publish the data used in this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Chen, P.; Feng, W.; Yan, B. Construction of a comprehensive integrity management system for oil and gas pipelines. *Oil Gas Storage Transp.* **2020**, *39*, 40–47.
- Li, X.; Chen, R.; Liu, L.; Zhou, C.; Bate, B. A non-Darcy flow CFD–DEM method for simulating ground collapse induced by leakage through underground pipeline defect. *Comput. Geotech.* **2023**, *162*, 105695. [\[CrossRef\]](#)
- Sun, H.; Peng, L.; Huang, S.; Li, S.; Long, Y.; Wang, S.; Zhao, W. Development of a physics-informed doubly fed cross-residual deep neural network for high-precision magnetic flux leakage defect size estimation. *IEEE Trans. Ind. Inform.* **2021**, *18*, 1629–1640. [\[CrossRef\]](#)
- Wang, T.; Wang, X.; Li, Z.; Zhang, Y.; Liu, J.; Zhao, L. Comparative analysis of long-distance oil and gas pipeline failures at home and abroad. *Oil Gas Storage Transp.* **2017**, *36*, 1258–1264.
- Jiang, L.; Zhang, H.; Liu, J.; Xiao, Q.; Shen, X.; Xu, H. A Physics-Guided MFL Deformed Defect Recovery Method. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 2113–2124. [\[CrossRef\]](#)
- Liu, B.; Ge, Q.; Wu, Z.; Lian, Z.; Yang, L.; Geng, H. Signal characteristics of oil and gas pipeline leakage detection based on magneto-mechanical effects. *Sensors* **2021**, *23*, 1857. [\[CrossRef\]](#) [\[PubMed\]](#)
- Gao, B.; Zhao, H.; Miao, X. A novel multi-model cascade framework for pipeline defects detection based on machine vision. *Measurement* **2023**, *220*, 113374. [\[CrossRef\]](#)
- Ren, Y.; Liu, J.; Zhang, H.; Wang, J. Task-based bias domain adaptation network under industrial small samples. *IEEE Trans. Ind. Inform.* **2022**, *18*, 6109–6119. [\[CrossRef\]](#)
- Shi, Y.; Zhang, C.; Li, R.; Cai, M.; Jia, G. Theory and application of magnetic flux leakage pipeline detection. *Sensors* **2015**, *15*, 31036–31055. [\[CrossRef\]](#)
- Zhang, S.; Liu, J.; Zhang, X. Adaptive compressive sensing: An optimization method for pipeline magnetic flux leakage detection. *Sustainability* **2023**, *15*, 14591. [\[CrossRef\]](#)
- Mao, B.; Lu, Y.; Wu, P.; Mao, B.; Li, P. Signal processing and defect analysis of pipeline inspection applying magnetic flux leakage methods. *Intell. Serv. Robot.* **2014**, *7*, 203–209. [\[CrossRef\]](#)
- Rempel, R.G. Anomaly detection using magnetic flux leakage technology. In Proceedings of the 2005 Conference, Rio de Janeiro, Brazil, 1 July 2005.
- Usarek, Z.; Warnke, K. Inspection of gas pipelines using magnetic flux leakage technology. *Adv. Mater. Sci.* **2017**, *17*, 37–45. [\[CrossRef\]](#)
- Gloria, N.B.S.; Areiza, M.C.L.; Miranda, I.V.J.; Rebello, J.M.A. Development of a magnetic sensor for detection and sizing of internal pipeline corrosion defects. *NDT E Int.* **2009**, *42*, 669–677. [\[CrossRef\]](#)
- Jing, L.; Zhao, M.; Li, P.; Xu, X. A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement* **2017**, *111*, 1–10. [\[CrossRef\]](#)
- Peng, X.; Anyaoha, U.; Liu, Z.; Tsukada, K. Analysis of magnetic-flux leakage (MFL) data for pipeline corrosion assessment. *IEEE Trans. Magn.* **2020**, *56*, 1–15. [\[CrossRef\]](#)
- Wu, D.; Liu, Z.; Wang, X.; Su, L. Composite magnetic flux leakage detection method for pipelines using alternating magnetic field excitation. *NDT E Int.* **2017**, *91*, 148–155. [\[CrossRef\]](#)
- Khodayari-Rostamabad, A.; Reilly, J.P.; Nikolova, N.K.; Hare, J.R.; Pasha, S. Machine learning techniques for the analysis of magnetic flux leakage images in pipeline inspection. *IEEE Trans. Magn.* **2009**, *45*, 3073–3084. [\[CrossRef\]](#)
- Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Chen, Y.; Zeng, W.; Yu, P. Generalizing to unseen domains: A survey on domain generalization. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 8052–8072. [\[CrossRef\]](#)
- Sun, Z.; Wang, P.; Vuran, M.C.; Al-Rodhaan, M.A.; Al-Dhelaan, A.M.; Akyildiz, I.F. MISE-PIPE: Magnetic induction-based wireless sensor networks for underground pipeline monitoring. *Ad Hoc Netw.* **2011**, *9*, 218–227. [\[CrossRef\]](#)
- Mangai, U.; Samanta, S.; Das, S.; Chowdhury, P.; Ug, M. A survey of decision fusion and feature fusion strategies for pattern classification. *IETE Tech. Rev.* **2010**, *27*, 293–304. [\[CrossRef\]](#)
- Fauvel, M.; Chanussot, J.; Benediktsson, J.A. Decision fusion for the classification of urban remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2006**, *44*, 2828–2838. [\[CrossRef\]](#)
- Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
- He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
- Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31, pp. 1–8.

26. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
27. Zhang, Z.; Cui, P.; Zhu, W. Deep Learning on Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* **2019**, *34*, 249–270. [[CrossRef](#)]
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.A.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. In Proceedings of the Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
29. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
30. Luong, M.T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal, 1–4 September 2015; pp. 1412–1421.
31. Wang, Q.; Dong, X.; Fu, C.; Zhang, L.; Xu, M.; Yang, J. Efficient Channel Attention for Visual Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 11534–11542.
32. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
33. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.