International Journal of Approximate Reasoning 187 (2025) 109547

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

GTransformer: Multi-view functional granulation and self-attention for tabular data modeling

Liang Liao [a], Yumin Chen [a,*], Yingyue Chen [b], Yiting Lin [a]

[a] School of Computer and Information Engineering, Xiamen University of Technology, Xiamen, Fujian 361024, China
[b] School of Economics and Management, Xiamen University of Technology, Xiamen, Fujian 361024, China

ARTICLE INFO

Keywords:
Granular computing
Self-attention
Multi-view representation
Tabular data
Deep learning

ABSTRACT

To bridge the performance gap between deep learning models and tree ensemble methods in tabular data tasks, we propose GTransformer, a novel deep architecture that innovatively integrates granular computing and self-attention mechanisms. Our approach introduces a scalable granulation function set, from which diverse functions are randomly sampled to construct multi-view feature granules. These granules are aggregated into granule vectors, forming a multi-view functional granulation layer that provides comprehensive representations of tabular features from multiple perspectives. Subsequently, a Transformer encoder driven by granule sequences is employed to model deep interactions among features, with predictions generated via a hierarchical multilayer perceptron (MLP) classification head. Experiments on 12 datasets show that GTransformer achieves an average AUC of 92.9%, which is comparable to the 92.3% performance of LightGBM. Compared with the current mainstream deep model TabNet, the average AUC gain is 2.74%, with a 14.5% improvement on the Sonar dataset. GTransformer demonstrates strong robustness in scenarios with noise and missing data, especially on the Credit and HTRU2 datasets, where the accuracy decline is 24.73% and 17.03% less than that of MLP-Head respectively, further verifying its applicability in complex real-world application scenarios.

1. Introduction

Deep learning has achieved significant breakthroughs in various fields such as image recognition, speech processing, and natural language processing (NLP) [1]. In 2012, Krizhevsky et al. proposed AlexNet [2], which achieved remarkable success in the ImageNet image recognition competition, greatly promoting the application of deep learning in computer vision. Subsequently, Mikolov et al. introduced Word2Vec [3] in 2013, which includes the Continuous Bag-of-Words (CBOW) and Skip-Gram models, effectively mapping words into low-dimensional vector spaces to better capture semantic relationships, thus providing valuable tools for NLP applications. In 2015, Mozilla Research proposed Deep Speech2 [4], achieving high accuracy in recognizing both English and Mandarin. Further advancement came in 2017 with Transformer, introduced by Vaswani et al. [5], which leveraged self-attention mechanisms to effectively capture long-range dependencies in sequential data, revolutionizing NLP. Following this, BERT (Bidirectional Encoder Representations from Transformers) was proposed by Devlin et al. in 2018 [6], which introduced a bidirectional Transformer encoder pre-trained on large-scale text data, significantly enhancing NLP performance.

* Corresponding author.
E-mail address: cym0620@163.com (Y. Chen).

https://doi.org/10.1016/j.ijar.2025.109547
Received 13 May 2025; Received in revised form 15 July 2025; Accepted 5 August 2025

Available online 8 August 2025

Despite these achievements, the application of deep learning to tabular data has been relatively underexplored and slower in progress [7]. To bridge this gap, researchers have been actively exploring deep learning techniques for structured data. In 2020, Google's research team proposed TabNet [8], a high-performance and interpretable framework specifically designed for tabular data. By employing sequential attention mechanisms, TabNet achieves effective feature selection at each decision step. In 2021, Gorishniy et al. introduced FT-Transform [9], an improved Transformer-based model for tabular data, achieving state-of-the-art performance across various datasets.

Tabular data, characterized by rows and columns where each column represents a feature and each row represents a sample, is widely used across domains such as recommendation systems, online advertising, finance, and healthcare [10]. Numerous machine learning competitions, including Kaggle [11], KDD-CUP, and Tianchi, are also centered around tabular data tasks. Traditional tree-based ensemble methods, such as XGBoost [12], LightGBM [13], and CatBoost [14], dominate these tasks due to their fast training speed, interpretability, and strong predictive accuracy [15]. However, tree-based models face several limitations: (1) They are unsuitable for continuous training scenarios involving streaming data [16]; (2) They struggle to perform effective end-to-end learning in multimodal contexts, particularly when integrating tabular data with image or text encoders [17]; (3) They provide limited robustness and capability in handling noisy or missing features, with relatively sparse research addressing these issues [18].

The concept of granular computing was introduced by Zadeh through fuzzy set theory to simulate human reasoning and manage uncertainty. In 1979, Zadeh further extended this concept by introducing information granularity [19]. Around the same time, Pawlak, the founder of rough set theory, also developed the notion of granularity in information processing [20]. Subsequently, granular computing became a popular research area, with various theories and models proposed to enhance data representation and processing [21–24].

Modern applications of granular computing emphasize its role in improving machine learning models. For example, Pedrycz et al. [25] emphasized the importance of optimal granularity for reliable learning results, while Li et al. [26] proposed a double-quantitative feature selection approach for multi-granularity ordered decision systems. Liang et al. [27] investigated information granules and entropy theory within information systems, characterizing and representing granules under various binary relations and providing an axiomatic definition. Shi et al. [28] applied a support vector machine-based information granulation approach to the intelligent prediction of surrounding rock deformation in shallow-buried highway tunnels. Jiang et al. [29] introduced a novel stochastic sigmoid function-based granulation method, integrating it with the extreme learning machine algorithm for weather prediction. Fu et al. [30] proposed a granular random forest algorithm based on width granulation to address the issue that random forests have poor classification performance in handling uncertain data. Furthermore, Deng et al. [31] developed an incremental fuzzy concept-cognitive learning model, enhancing learning performance by utilizing fuzzy attribute granulation.

The success of machine learning models largely depends on effective data representation [32]. Inspired by the attention mechanism in the Transformer encoder [33], which effectively highlights essential features in structured data, we integrate this mechanism into our model. Additionally, leveraging the principles of granular computing theory [34], we propose a novel approach for extracting robust and comprehensive feature representations [35]. By defining multi-view feature granules where each granulation function represents a distinct perspective, our approach captures diverse feature representations, which are then aggregated to form a holistic view [36]. This process is particularly effective in handling noisy or incomplete data [37,38].

In this paper, we propose a novel deep learning architecture called GTransformer, which integrates granular computing and self-attention mechanisms to bridge the performance gap between deep learning models and tree-based ensemble methods for tabular data modeling. The primary contributions of this paper are summarized as follows:

- We introduce a novel architecture, GTransformer, which applies multi-view functional granulation to individual features, generating feature granules that are aggregated into granule vectors and subsequently segmented into granule sequences for Transformer encoding. This approach enhances representational power while effectively mitigating computational complexity in high-dimensional data scenarios.
- Through extensive experiments on ten publicly available datasets, we demonstrate that GTransformer outperforms the state-of-the-art deep learning model TabNet [8] by an average AUC margin of 1.91%, with a remarkable improvement of 10.4% on the Sonar dataset. Moreover, GTransformer achieves performance comparable to or better than tree-based ensemble models like LightGBM.
- GTransformer demonstrates strong robustness, especially on the Credit dataset, where its average classification accuracy drop under the two scenarios of noise and missing data is reduced by 16.365% less compared to MLP Head, demonstrating its applicability to real-world scenarios with imperfect data.

2. Multi-view functional granulation

In classical set theory, elements are considered unique and unordered, which limits the flexibility and applicability of set-related operators in complex learning scenarios. To overcome these limitations, we introduce the concept of a granule, defined as an ordered finite set that permits repetitions. Furthermore, we extend this concept by defining a granule vector, which is an ordered sequence composed of granules.

To enable the effective utilization of these structures in machine learning tasks, we propose novel granule and granule vector operators [30], which enhance the expressiveness and robustness of granular representations. This framework provides a powerful foundation for constructing granular classifiers, facilitating improved performance and interpretability in various learning scenarios.

2.1. Granules and granule vectors

Definition 1. Suppose we have a set $g = \{r_1, r_2, \dots, r_n\}$, where each $r_j \in R$. If the elements in this set are ordered and repetitions are allowed, then the set is defined as a granule [39] denoted as:

$$g = \{r_j\}_{j=1}^{n}. \tag{1}$$

Definition 2. Given granules $g_i = \{r_j\}_{j=1}^{n}$, an m-dimensional granule vector G [40] is defined as:

$$G = (g_1, g_2, \dots, g_m) = (g_i)_{i=1}^{m}. \tag{2}$$

A granule vector is a finite ordered sequence of granules, where each element of the granule vector is itself a granule. Since granules are sets, the elements of a granule vector are sets rather than scalar real numbers as in traditional vectors.

2.2. Multi-view functional granulation and multi-view granules

Given an information space $IS = (U, F)$, where $U = \{x_1, x_2, \dots, x_n\}$ represents the sample set, and $F = \{f_1, f_2, \dots, f_m\}$ denotes the feature set. For any feature $f \in F$, let $n(x, f)$ represent the standardized value (mean of 0 and variance of 1) of sample x on feature f.

Definition 3. Let $\varphi(n(x, f))$ be a granulation function. A granulation function set consisting of $\theta \geq 1$ granulation functions is defined as:

$$\begin{aligned} Funcset =& \{\varphi_1(n(x, f)), \varphi_2(n(x, f)), \dots, \varphi_\theta(n(x, f))\} \\ =& \{\varphi_i(n(x, f))\}_{i=1}^{\theta} \end{aligned} \tag{3}$$

where the granulation functions in this set are ordered and non-repetitive.

Definition 4. Given an information space $IS = (U, F)$, for any sample $x \in U$ and any feature $f \in F$, let $\beta = \{\beta_1(n(x, f)), \beta_2(n(x, f)), \dots, \beta_k(n(x, f))\} \subseteq Funcset$ be a randomly sampled subset of granulation functions. Then, the multi-view granule formed by applying multi-view functional granulation on feature f is defined as:

$$g_f(x) = \{r_j\}_{j=1}^{k} = \{r_1, r_2, \dots, r_k\}, \tag{4}$$

where $r_j = \beta_j(n(x, f))$ denotes the functional granulation of sample x on a specific feature.

Definition 5. Given an information space $IS = (U, F)$, for any sample $x \in U$ and a feature subset $A \subseteq F$, where $A = \{a_1, a_2, \dots, a_m\}$, the granule vector for sample x on the feature subset A is defined as:

$$G = G_A(x) = \left(g_{a_1}(x),\, g_{a_2}(x),\, \dots,\, g_{a_m}(x)\right)^T, \tag{5}$$

where $g_{a_m}(x)$ denotes the granule formed by sample x on feature a_m. Granule vectors consist of granules, and each granule is an ordered set generated by functional granulation. Thus, unlike traditional vectors whose elements are real numbers, the elements of granule vectors are ordered sets. For convenience, the feature set $A = \{a_1, a_2, \dots, a_m\}$ can be indexed using integers. Thus, a granule vector can also be expressed as:

$$\begin{aligned} & G(x) = (g_1(x), g_2(x), \dots, g_m(x))^T \\ \text{or} \quad & G = (g_1, g_2, \dots, g_m)^T. \end{aligned} \tag{6}$$

Definition 6. Given an information space $IS = (U, F)$, for any sample $x \in U$ and a single feature $f \in F$, the sigmoid granulation function for feature f is defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-n(x, f)}}. \tag{7}$$

Definition 7. Given an information space $IS = (U, F)$, for any sample $x \in U$ and a single feature $f \in F$, the cosine granulation function for feature f is defined as:

$$cos(x) = cos(n(x, f)). \tag{8}$$

Definition 8. Given an information space $IS = (U, F)$, for any sample $x \in U$ and a single feature $f \in F$, the tanh granulation function for feature f is defined as:

Table 1
Information Space.

U	a	b	c
X1	0.1	0.5	0.3
X2	0.2	0.3	0.7
X3	0.7	0.6	0.8
X4	0.9	0.4	0.1

$$tanh(x) = \frac{e^{n(x,f)} - e^{-n(x,f)}}{e^{n(x,f)} + e^{-n(x,f)}}. \tag{9}$$

Example 1. Given an information space $IS = (U, F)$ as shown in Table 1, where the sample set is $U = \{x_1, x_2, x_3, x_4\}$ and the feature set is $F = \{a, b, c\}$. Suppose a randomly sampled subset of the granulation function set $Funcset$ is defined as:

$$\beta = \{sigmoid(x),\ cos(x),\ tanh(x)\},$$

as specified in Definitions 6–8.

Given the sample set $U = \{x_1, x_2, x_3, x_4\}$, the multi-view granules constructed according to feature a are as follows:

$$g_1 = g_a(x_1) = \{sigmoid(0.1), cos(0.1), tanh(0.1)\},$$
$$g_2 = g_a(x_2) = \{sigmoid(0.2), cos(0.2), tanh(0.2)\},$$
$$g_3 = g_a(x_3) = \{sigmoid(0.7), cos(0.7), tanh(0.7)\},$$
$$g_4 = g_a(x_4) = \{sigmoid(0.9), cos(0.9), tanh(0.9)\}.$$

The multi-view granules constructed according to feature b are:

$$g_5 = g_b(x_1) = \{sigmoid(0.5), cos(0.5), tanh(0.5)\},$$
$$g_6 = g_b(x_2) = \{sigmoid(0.3), cos(0.3), tanh(0.3)\},$$
$$g_7 = g_b(x_3) = \{sigmoid(0.6), cos(0.6), tanh(0.6)\},$$
$$g_8 = g_b(x_4) = \{sigmoid(0.4), cos(0.4), tanh(0.4)\}.$$

The multi-view granules constructed according to feature c are:

$$g_9 = g_c(x_1) = \{sigmoid(0.3), cos(0.3), tanh(0.3)\},$$
$$g_{10} = g_c(x_2) = \{sigmoid(0.7), cos(0.7), tanh(0.7)\},$$
$$g_{11} = g_c(x_3) = \{sigmoid(0.8), cos(0.8), tanh(0.8)\},$$
$$g_{12} = g_c(x_4) = \{sigmoid(0.1), cos(0.1), tanh(0.1)\}.$$

If $A = \{a, b, c\}$, then the granule vector of sample x_1 on A is:

$$\begin{aligned} G_A(x_1) &= (g_a(x_1), g_b(x_1), g_c(x_1))^T \\ &= (\{sigmoid(0.1), cos(0.1), tanh(0.1)\}, \\ &\quad \{sigmoid(0.5), cos(0.5), tanh(0.5)\}, \\ &\quad \{sigmoid(0.3), cos(0.3), tanh(0.3)\}). \end{aligned}$$

If $A = \{a, b, c\}$, then the granule vector of sample x_2 on A is:

$$\begin{aligned} G_A(x_2) &= (g_a(x_2), g_b(x_2), g_c(x_2))^T \\ &= (\{sigmoid(0.2), cos(0.2), tanh(0.2)\}, \\ &\quad \{sigmoid(0.3), cos(0.3), tanh(0.3)\}, \\ &\quad \{sigmoid(0.7), cos(0.7), tanh(0.7)\}). \end{aligned}$$

If $A = \{a, b, c\}$, then the granule vector of sample x_3 on A is:

$$\begin{aligned} G_A(x_3) &= (g_a(x_3), g_b(x_3), g_c(x_3))^T \\ &= (\{sigmoid(0.7), cos(0.7), tanh(0.7)\}, \end{aligned}$$

$$\{sigmoid(0.6), cos(0.6), tanh(0.6)\},$$
$$\{sigmoid(0.8), cos(0.8), tanh(0.8)\}).$$

If $A = \{a, b, c\}$, then the granule vector of sample x_4 on A is:

$$\begin{aligned} G_A(x_4) &= (g_a(x_4), g_b(x_4), g_c(x_4))^T \\ &= (\{sigmoid(0.9), cos(0.9), tanh(0.9)\}, \\ &\quad \{sigmoid(0.4), cos(0.4), tanh(0.4)\}, \\ &\quad \{sigmoid(0.1), cos(0.1), tanh(0.1)\}). \end{aligned}$$

2.3. Granular operations

Since arithmetic operations such as addition, subtraction, multiplication, and division of real numbers are closed under the set of real numbers, the proposed granular operations should similarly exhibit closure within the set of granules. Furthermore, the defined granule vector operations are also expected to maintain this property, ensuring consistency and robustness in granular computations.

Definition 9. Given two granules $s = \{s_j\}_{j=1}^n$ and $t = \{t_j\}_{j=1}^n$, granular addition, subtraction, multiplication, and division are defined as follows:

$$s + t = \{s_j + t_j\}_{j=1}^n, \tag{10}$$
$$s - t = \{s_j - t_j\}_{j=1}^n, \tag{11}$$
$$s \times t = \{s_j \times t_j\}_{j=1}^n, \tag{12}$$
$$s/t = \{s_j/t_j\}_{j=1}^n. \tag{13}$$

Definition 10. Given two m-dimensional granule vectors $G = (g_1, g_2, \dots, g_m) = (g_i)_{i=1}^m$ and $Q = (q_1, q_2, \dots, q_m) = (q_i)_{i=1}^m$, the granule vector operations of addition, subtraction, multiplication, and division are defined as follows:

$$G + Q = (g_1 + q_1,\ g_2 + q_2, \dots,\ g_m + q_m), \tag{14}$$
$$G - Q = (g_1 - q_1,\ g_2 - q_2, \dots,\ g_m - q_m), \tag{15}$$
$$G \times Q = (g_1 \times q_1,\ g_2 \times q_2, \dots,\ g_m \times q_m), \tag{16}$$
$$G/Q = (g_1/q_1,\ g_2/q_2, \dots,\ g_m/q_m). \tag{17}$$

3. GTransformer architecture

GTransformer comprises three core components: the Multi-view Functional Granulation Layer, the Granule Sequence-driven Transformer Encoder, and the Hierarchical Multilayer Perceptron (MLP) Classification Head. Through comprehensive interaction modeling of multi-view granule sequences, GTransformer significantly improves the performance of tabular data classification tasks by capturing complex feature relationships from multiple perspectives.

The design of the Transformer Encoder is inspired by the original architecture proposed by Vaswani et al. [5], while the overall framework structure draws inspiration from the Vision Transformer (ViT) [41]. The detailed architecture of GTransformer is depicted in Fig. 1.

3.1. Multi-view functional granulation layer

In this paper, we propose a novel granulation method termed Multi-view Functional Granulation, designed to enhance the representational capacity of tabular data by generating diverse multi-view feature representations. Each granulation function acts as an independent observer, producing features from a distinct perspective. By aggregating these multiple viewpoints, the multi-view functional granulation process constructs a diversified feature subspace, effectively modeling observer differences and mitigating biases introduced by single-view representations.

Given an input sample $X = [x_1, x_2, \dots, x_d] \in \mathbb{R}^d$, where d denotes the feature dimensionality, each feature is first standardized through a normalization layer as follows:

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \quad i = 1, 2, \dots, d, \tag{18}$$

where μ_i and σ_i represent the mean and standard deviation of the i^{th} feature, respectively. The resulting normalized feature vector is $\hat{X} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_d]$, which serves as the input to the Multi-view Functional Granulation Layer.

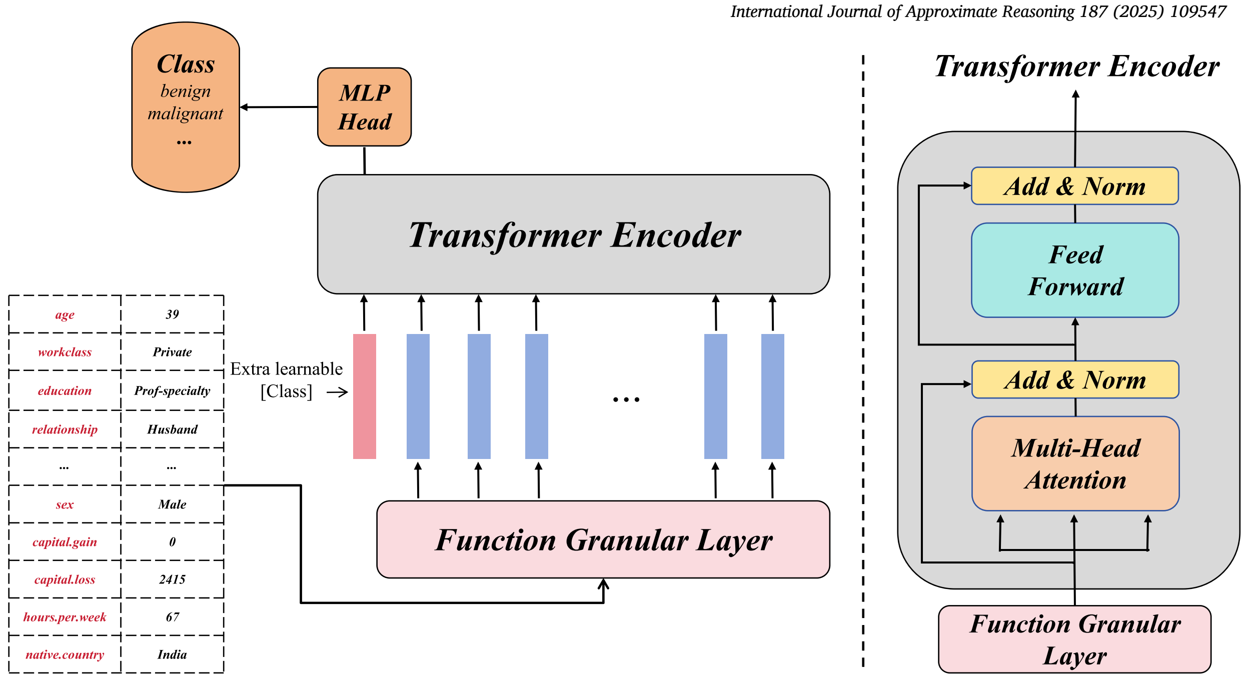

Fig. 1. Framework of GTransformer.

The core idea of this layer is to randomly sample n granulation functions from a predefined Granulation Function Set $F = \{f_1, f_2, \dots, f_m\}$. Each granulation function f_i is treated as an independent observer, generating distinct multi-view granules $g \in \mathbb{R}^n$ and corresponding granule vectors $G \in \mathbb{R}^{d\times n}$. To obtain granule sequences, the granule vector G is segmented according to observers, resulting in the Granule Sequence Matrix:

$$\hat{G} = [f_1(\hat{X}), f_2(\hat{X}), \dots, f_n(\hat{X})], \tag{19}$$

where each generated granule sequence $p_j = f_j(\hat{X})$, and $p_j \in \mathbb{R}^d$ represents the j^{th} granule sequence's d-dimensional representation. The overall process is illustrated in Fig. 2.

3.2. Granule sequence-driven transformer encoder

The Transformer encoder consists of a multi-head self-attention layer and a position-wise feed-forward network. In the self-attention mechanism, the input granule sequence matrix $\hat{G} \in R^{m\times d}$ is linearly mapped to Key, Query, and Value matrices through the learned weight matrices $\mathbf{W}_K$, $\mathbf{W}_Q$, and $\mathbf{W}_V$, respectively:

$$\mathbf{K} = \hat{G}\mathbf{W}_K, \quad \mathbf{Q} = \hat{G}\mathbf{W}_Q, \quad \mathbf{V} = \hat{G}\mathbf{W}_V, \tag{20}$$

where $\mathbf{K}, \mathbf{Q} \in R^{m\times k}$, $\mathbf{V} \in R^{m\times v}$, and k, v are dimensions of the key and value vectors respectively. The attention weight matrix $\mathbf{A} \in R^{m\times m}$ is computed using scaled dot-product:

$$\mathbf{A} = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{k}}\right), \tag{21}$$

where the scaling factor $\sqrt{k}$ prevents overly large dot-products leading to gradient vanishing. The attention output is then computed by multiplying this weight matrix with the Value matrix:

$$Attention(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = \mathbf{A}\mathbf{V}. \tag{22}$$

In the multi-head self-attention mechanism, outputs from multiple attention heads are concatenated and linearly transformed to the target dimension:

$$\begin{aligned} MultiHead(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = {} & Concat(head_1, head_2, \dots, head_h) \\ & \times \mathbf{W}_O, \end{aligned} \tag{23}$$

where

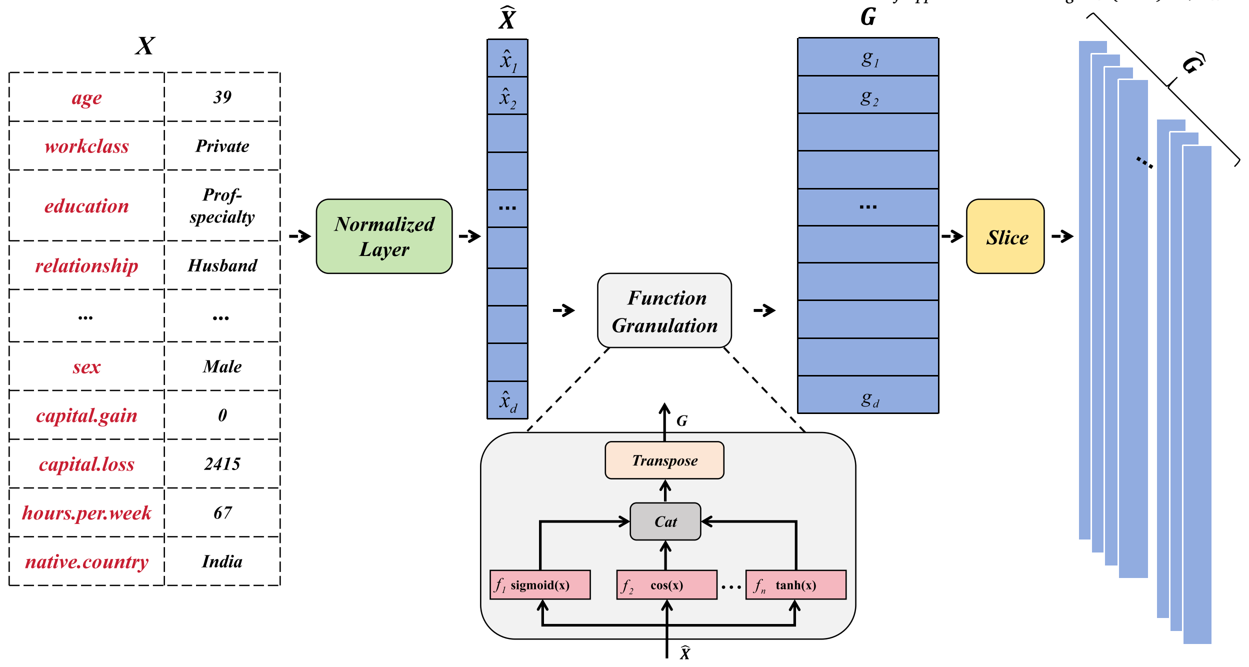

Fig. 2. Multi-view Functional Granulation.

$$head_i = Attention(\mathbf{K}_i, \mathbf{Q}_i, \mathbf{V}_i), \tag{24}$$

and h denotes the number of attention heads, and $\mathbf{W}_O \in R^{hv\times d}$ is a learnable weight matrix. Following the attention layer is a position-wise feed-forward network (FFN), which applies nonlinear transformations to each granule sequence's hidden representation:

$$FFN(\mathbf{z}) = ReLU(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{25}$$

where $\mathbf{z} \in R^d$ denotes the hidden state of each granule sequence, $\mathbf{W}_1 \in R^{d\times d_f}$, $\mathbf{W}_2 \in R^{d_f\times d}$ are learnable weight matrices, $\mathbf{b}_1 \in R^{d_f}$, $\mathbf{b}_2 \in R^d$ are biases, d_f is the hidden dimension of the FFN, and $ReLU(\cdot)$ denotes the rectified linear unit activation function.

The output of the Transformer encoder is stabilized using residual connections and layer normalization, expressed as:

$$H_{out} = LayerNorm\left(H_{in} + SubLayer(H_{in})\right), \tag{26}$$

where H_{in} and H_{out} represent the input and output features, respectively, and $SubLayer(\cdot)$ denotes the combined multi-head attention layer and the position-wise feed-forward network. Through the Transformer encoding layer, the model gradually learns high-order interactions and global dependencies among granule sequences. Inspired by BERT, a learned classification token $p_{[CLS]} \in R^d$ is prepended to the granule sequence. This token aggregates global contextual information through self-attention and ultimately serves as the input feature to the classifier.

3.3. Hierarchical multilayer perceptron classification head

To map the high-order features output by the Transformer encoder to the target classification space, we propose a hierarchical multilayer perceptron (MLP) head. The hierarchical structure comprises two layers: a hidden expansion layer and an output layer. Firstly, a feature alignment layer maps the classification token $p_{[CLS]}$ into a suitable dimensional space, obtaining the hidden representation:

$$h_1 = ReLU(W_1 p_{[CLS]} + b_1), \quad W_1 \in R^{d\times d}, \quad b_1 \in R^d. \tag{27}$$

Next, the feature expansion layer enhances the representational capability through dimensional expansion:

$$h_2 = ReLU(W_2 h_1 + b_2), \quad W_2 \in R^{d\times 2d}, \quad b_2 \in R^{2d}. \tag{28}$$

Finally, the classification output layer maps expanded features to the classification space based on the task requirement (e.g., C-class classification):

$$O = W_3 h_2 + b_3, \quad W_3 \in R^{2d\times C}, \quad b_3 \in R^C. \tag{29}$$

Class probabilities are computed via the softmax function:

Table 2
Description of UCI Datasets.

Datasets	Sample	Feature	Category
WDBC	569	30	2
Blood	748	4	2
Raisin	900	7	2
Ionosphere	351	34	2
Spambase	4601	57	2
Sonar	208	60	2
Online	12330	17	2
HTRU2	17898	8	2
QSAR_Biodegradation(QSAR)	1055	41	2
Dry_Bean(Bean)	13611	16	7
Iranian_Churn(Iranian)	3150	13	2
Credit_Card_Clients(Credit)	30000	23	2

$$P(y=c|X)=\frac{exp(O_c)}{\sum_{i=1}^{C} exp(O_i)}, \quad c=1,2,\dots,C. \tag{30}$$

4. Experimental analysis

In the experiments, we utilize 12 tabular datasets from various domains obtained from the UCI Machine Learning Repository [42]. Detailed descriptions of these datasets are provided in Table 2. Each dataset is randomly split into training and validation sets with a ratio of 70% to 30%, respectively.

The experimental setup is structured as follows: In Section 4.1, we investigate the impact of the number of granulation functions on model performance and compare GTransformer with the deep tabular model TabNet, demonstrating the effectiveness of our proposed granulation method. In Section 4.2, we evaluate the robustness of GTransformer under conditions of noisy and incomplete data, assessing its resilience and stability. In Section 4.3, demonstrates the effectiveness of the GTransformer model through comparisons with existing models.

4.1. Impact of granulation function modeling

Data normalization is applied as a preprocessing step to standardize input features before granulation. Each sample is processed through granulation by randomly sampling granulation functions from a predefined function set, thereby generating granule sequences. In this experiment, both the number of encoding layers and attention heads in GTransformer were fixed at 1, while the sequence dimension was defined as the number of randomly sampled granulation functions. The final results represent the average of five-fold cross-validation performed on each dataset. The number of granulation functions ranged from 1 to 10. Experimental results across different datasets are shown in Fig. 3, where the x-axis denotes parameter k (number of granulation functions) and the y-axis indicates classification accuracy.

In the WDBC dataset, when k is 10, the classification accuracy of GTransformer reaches its peak at 0.9885. As k varies, the classification accuracy of GTransformer shows significant fluctuations, indicating that GTransformer is sensitive to the number of granulation functions. Different numbers of granulation functions may lead to different classification performances of the model. When k is in the range of 1 to 10, the classification accuracy of GTransformer is significantly higher than that of CatBoost.

In the Blood dataset, when k is 3, the classification accuracy of GTransformer reaches its peak at 0.7988. As k varies, the classification accuracy of GTransformer shows considerable fluctuations, indicating that GTransformer is sensitive to the number of granulation functions. When k is in the range of 1 to 3, the classification accuracy of GTransformer increases, and it shows a downward trend when k is greater than 3, but the classification accuracy is still higher than that of the three comparison models.

In the Raisin dataset, when k is 2, the classification accuracy of GTransformer reaches its peak at 0.8946, which is 2.33% higher than that of TabNet. As k varies, the classification accuracy of GTransformer shows obvious fluctuations, indicating that GTransformer is sensitive to the number of granulation functions. However, the classification accuracy is still higher than that of the three comparison models.

In the Ionosphere dataset, when k is 6, the classification accuracy of GTransformer reaches its maximum at 0.9317, which is 2.3% higher than that of TabNet. As k varies, the classification accuracy of GTransformer shows considerable fluctuations, indicating that GTransformer is sensitive to the number of granulation functions. When k is in the range of 1 to 6, the classification accuracy of GTransformer increases with slight fluctuations, and it shows a downward trend when k is greater than 6.

In the Spambase dataset, when the k parameter is 8, the classification accuracy of GTransformer reaches its peak at 0.9378, which is 1.9% higher than that of TabNet. As k increases, the classification accuracy of GTransformer fluctuates slightly, indicating that GTransformer is not sensitive to the number of granulation functions. Moreover, the classification accuracy is always higher than that of TabNet.

In the Sonar dataset, when the k parameter is 8, the classification accuracy of GTransformer reaches its maximum at 0.8918, which is 17.15% higher than that of TabNet. As k increases, the classification accuracy of GTransformer fluctuates significantly, indicating

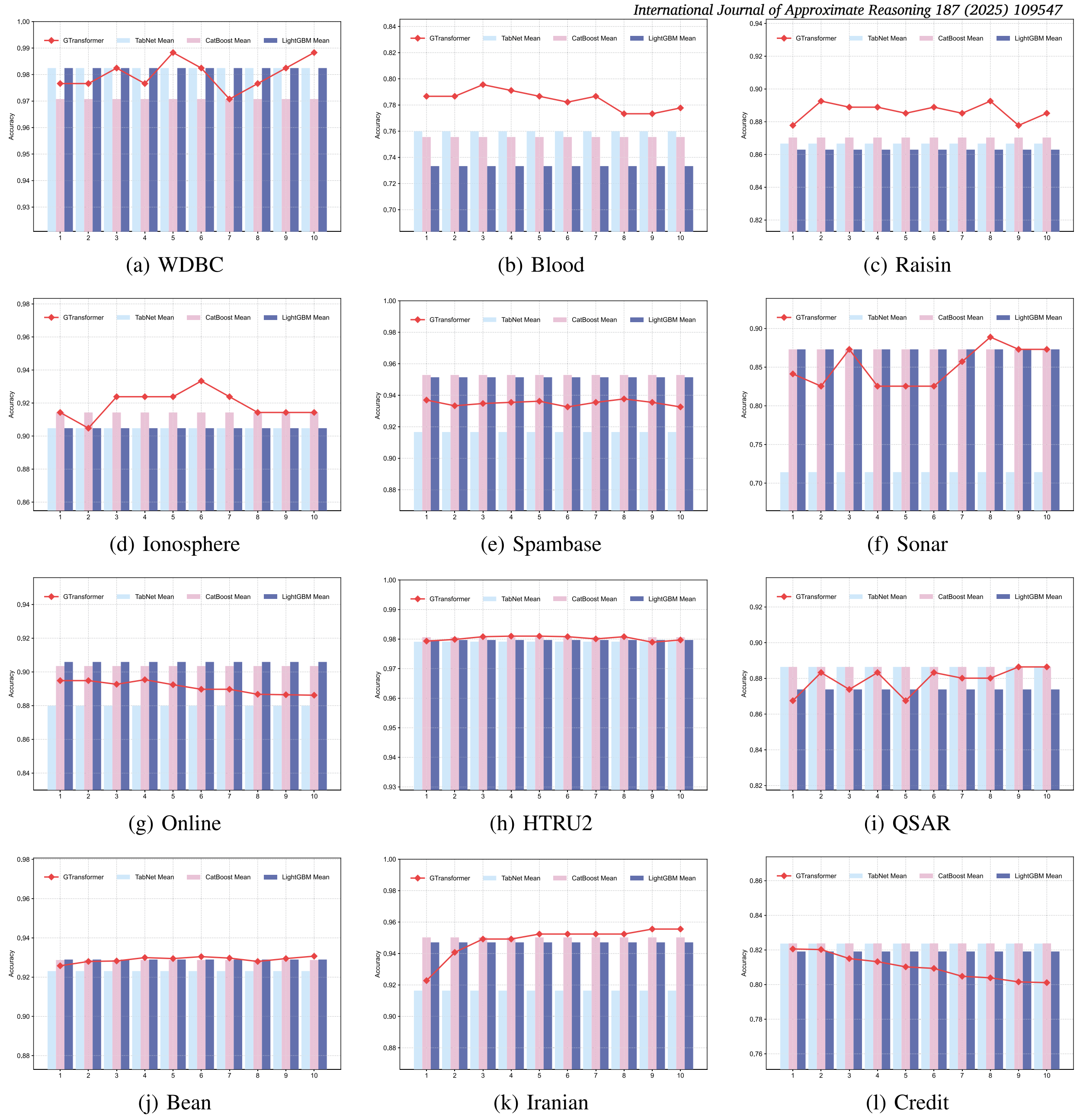

Fig. 3. Comparison of accuracy variations across different k values on 12 datasets. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

that GTransformer is sensitive to the number of granulation functions. However, the classification accuracy is always higher than that of TabNet.

In the Online dataset, when the k parameter is 4, the classification accuracy of GTransformer reaches its maximum at 0.8983, which is 1.8% higher than that of TabNet. As k increases, the classification accuracy of GTransformer fluctuates slightly, indicating that GTransformer is not sensitive to the number of granulation functions. Moreover, the classification accuracy is always higher than that of TabNet.

In the HTRU2 dataset, when the k parameter is 5, the classification accuracy of GTransformer reaches its maximum at 0.9815. As k increases, the classification accuracy of GTransformer remains relatively stable, maintaining at around 0.98, indicating that GTransformer is not sensitive to the number of granulation functions.

In the QSAR dataset, when the k parameter is 9 and 10, the classification accuracy of GTransformer reaches the maximum value of 0.8865. As the k value increases, the classification accuracy of GTransformer shows an upward trend with small fluctuations.

Table 3
Dataset Characteristics and Optimal k Values.

Dataset	Feature Dimension	Sample Size	Optimal k Value
WDBC	30	569	10
Blood	4	748	3
Raisin	7	900	2
Ionosphere	34	351	6
Spambase	57	4601	8
Sonar	60	208	8
Online	17	12330	4
HTRU2	8	17898	5
QSAR	41	1055	9 and 10
Bean	16	13611	10
Iranian	13	3150	10
Credit	23	30000	1

In the Bean dataset, when the k parameter is 10, the classification accuracy of GTransformer reaches the maximum value of 0.9305. As the k value increases, the classification accuracy of GTransformer shows a gradually increasing trend, and the classification accuracy is higher than that of TabNet.

In the Iranian dataset, when the k parameter value is 10, the classification accuracy of the GTransformer model reaches the peak value of 0.9583, which is 4% higher than that of TabNet. The increase is significant when the k value is in the range of 1 to 5, but it tends to saturate and only increases by 0.3% when the k value is greater than 5. This indicates that in the Iranian dataset, appropriately increasing the number of granulation functions can enhance the model's ability to represent features, but further increasing the k value has limited contribution to improving the model performance.

In the Credit dataset, when the k parameter is 1, the classification accuracy of GTransformer reaches the maximum value of 0.8205. As the k value increases, the classification accuracy of GTransformer shows a downward trend, suggesting that increasing the number of granulation functions may lead to overfitting and weaken the classification performance.

Although different datasets and parameter k values can affect the classification accuracy, GTransformer can consistently outperform TabNet by adjusting the k value. As shown in Table 3, for datasets with complex nonlinearities, k values between 5 and 10 can be prioritized, while for low-dimensional and linear datasets with simple decision boundaries, k values between 1 and 5 can be prioritized. As the k value continues to increase, the classification accuracy of GTransformer shows a stable and upward trend on most datasets, verifying the universality and advantages of the multi-visual function granulation strategy.

4.2. Robustness experiments

To further demonstrate that the introduction of multi-perspective function granulation and self-attention in GTransformer can enhance its robustness against noisy and missing data, a comparison was made with a GTransformer that had the multi-perspective function granulation layer and self-attention layer removed. The GTransformer without these layers was equivalent to an MLP-Head. In this experiment, the number of granulation functions randomly selected by GTransformer was fixed at 8, the number of encoding layers was fixed at 1, and the number of attention heads was fixed at 4.

(1) Noisy Data

On the test examples, the data was contaminated by replacing a certain number of values with randomly generated values from the corresponding columns. The noisy data was then passed to the trained GTransformer to calculate the accuracy. Fig. 4 shows the comparison of experimental results on five different datasets. On the HTRU2, Online, and Spambase datasets, GTransformer performed better in terms of prediction accuracy as the noise increased, demonstrating greater robustness than MLP-Head.

Notably, on the Iranian dataset, MLP-Head had a better prediction accuracy than GTransformer when the noise was between 0 and 15%, but GTransformer's prediction accuracy surpassed MLP-Head as the noise increased. On the Credit dataset, MLP-Head's prediction accuracy dropped from 0.7889 to 0.5186, a decrease of 27.03%, while GTransformer's prediction accuracy decreased from 0.8156 to 0.7926, a decrease of 2.3%, which was 24.73% less than MLP-Head.

(2) Missing Data

Similarly, in the test data, some missing values were artificially selected, and the data with missing values were sent to the trained GTransformer to calculate the prediction score. For the handling of missing values, the mode was used for imputation across all classes in the corresponding column [43][44]. The results of five data sets are shown in Fig. 5. It can be seen that on the Credit and Online datasets, GTransformer demonstrated better stability in handling missing values as the missing rate increased compared to MLP-Head.

In particular, on the Iranian and WDBC datasets, MLP-Head had a better prediction accuracy than GTransformer when the missing rate was between 0 and 15%, but GTransformer's prediction accuracy surpassed MLP-Head as the missing rate increased. On the HTRU2 dataset, both models had the same accuracy when the missing rate was 0. As the missing rate increased, GTransformer's prediction accuracy decreased from 0.9876 to 0.9126, a decrease of 7.5%, while MLP-Head's prediction accuracy dropped from 0.9876 to 0.7423, a decrease of 24.53%, which was 17.03% less than MLP-Head.

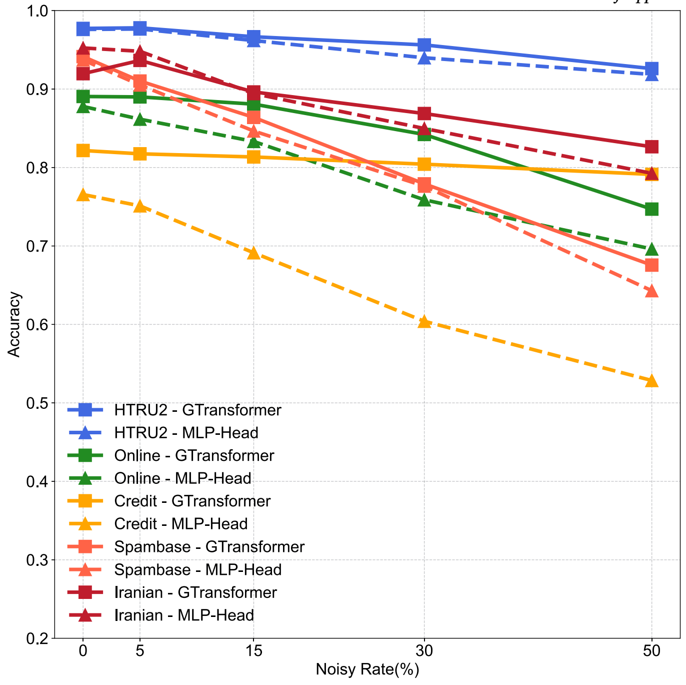

Fig. 4. Performance Comparison between GTransformer and MLP Head under Noisy Data.

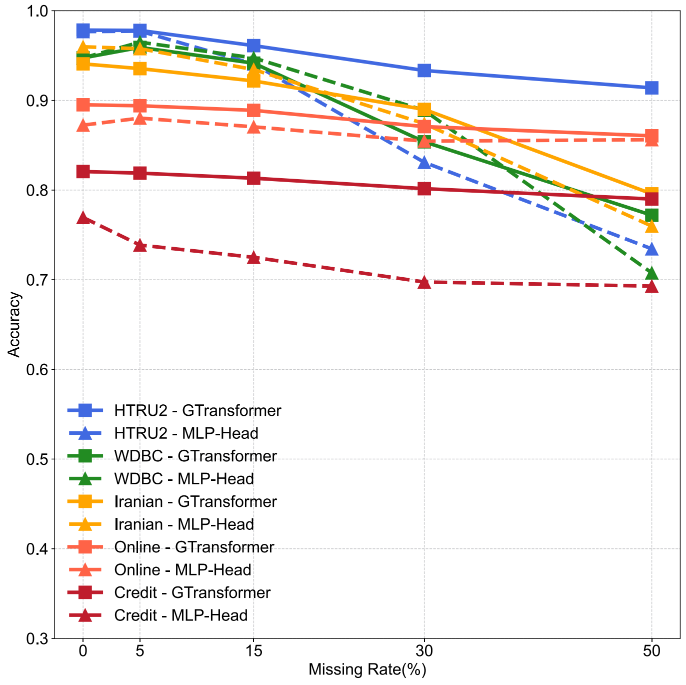

Fig. 5. Performance Comparison between GTransformer and MLP Head under Missing Data.

The experiments show that the multi-perspective function granulation layer enhances the representation ability of features, and the robustness comes from the fusion of multi-perspective features. Even if a feature is noisy, it can extract information from the correct features, allowing for a certain degree of correction.

Table 4
Accuracy Comparison of Different Models on 12 Datasets.

Datasets	GTransformer	TabNet	MLP-Head	LightGBM	CatBoost	SVM	GBDT	LR	KNN
WDBC	**0.9836**	0.9691	0.9415	0.9632	0.9508	0.9356	0.9567	0.9473	0.9239
Blood	**0.8033**	0.7671	0.7637	0.7423	0.7812	0.7210	0.7288	0.7348	0.7128
Raisin	**0.8905**	0.8623	0.8535	0.8636	0.8678	0.8481	0.8556	0.8519	0.8296
Ionosphere	**0.9248**	0.8985	0.9023	0.9151	0.9234	0.9028	0.9004	0.9022	0.8585
Spambase	0.9450	0.9261	0.9341	0.9522	**0.9602**	0.9239	0.9399	0.9203	0.8088
Sonar	**0.8571**	0.8254	0.8413	0.8475	0.8495	0.7619	0.8095	0.8243	0.7826
Online	0.8961	0.8648	0.8621	0.8965	**0.9121**	0.8673	0.8789	0.8640	0.8629
HTRU2	**0.9804**	0.9783	0.9793	0.9801	0.9782	0.9763	0.9773	0.9787	0.9716
QSAR	**0.8707**	0.8644	0.8517	0.8675	0.8612	0.8668	0.8549	0.8611	0.8075
Bean	**0.9308**	0.9212	0.8645	0.9292	0.9305	0.9142	0.8994	0.9155	0.7215
Iranian	**0.9612**	0.9511	0.9016	0.9600	0.9524	0.9089	0.9270	0.9015	0.8645
Credit	0.8196	**0.8226**	0.7818	0.8203	0.8173	0.7722	0.8200	0.7802	0.7556

Table 5
Precision Comparison of Different Models on 12 Datasets.

Datasets	GTransformer	TabNet	MLP-Head	LightGBM	CatBoost	SVM	GBDT	LR	KNN
WDBC	**0.9785**	0.9686	0.9485	0.9610	0.9610	0.9657	0.9582	0.9671	0.9472
Blood	**0.7288**	0.7041	0.7097	0.6600	0.7041	0.7072	0.6350	0.7100	0.6442
Raisin	**0.8694**	0.8629	0.8574	0.8332	0.8592	0.8593	0.8402	0.8515	0.8627
Ionosphere	**0.9302**	0.8712	0.9037	0.9091	0.9287	0.9104	0.8749	0.9183	0.8999
Spambase	0.9258	0.9281	0.9245	0.9449	**0.9501**	0.9284	0.9388	0.9163	0.8932
Sonar	0.8530	0.8257	0.7993	**0.8546**	0.8079	0.7416	0.8442	0.7273	0.7500
Online	0.7848	0.7896	0.7913	0.8118	**0.8239**	0.8158	0.8060	0.8227	0.7370
HTRU2	**0.9776**	0.9669	0.9347	0.9474	0.9522	0.9540	0.9398	0.9480	0.9481
QSAR	**0.8556**	0.7886	0.8057	0.8275	0.8309	0.8235	0.8240	0.8245	0.8175
Bean	**0.9493**	0.9457	0.9411	0.9418	0.9444	0.9413	0.9405	0.9356	0.9345
Iranian	**0.9414**	0.8527	0.8740	0.8787	0.8989	0.9355	0.8780	0.8437	0.9006
Credit	0.7396	**0.7684**	0.7041	0.7578	0.7584	0.7544	0.7551	0.7511	0.6840

Table 6
Recall Comparison of Different Models on 12 Datasets.

Datasets	GTransformer	TabNet	MLP-Head	LightGBM	CatBoost	SVM	GBDT	LR	KNN
WDBC	**0.9608**	0.9486	0.9408	0.9531	0.9531	0.9605	0.9557	0.9527	0.9307
Blood	**0.6849**	0.5825	0.6054	0.6073	0.6335	0.5142	0.5927	0.5285	0.6013
Raisin	0.8522	0.8547	0.8440	0.8307	**0.8571**	0.8427	0.8388	0.8474	0.8446
Ionosphere	**0.9428**	0.8802	0.9312	0.9212	0.9375	0.9195	0.9081	0.8460	0.8019
Spambase	0.9268	0.9271	0.9292	0.9419	**0.9506**	0.9235	0.9344	0.9040	0.8874
Sonar	0.8463	0.8000	0.7778	**0.8565**	0.8009	0.7361	0.8096	0.7315	0.7269
Online	0.7720	0.7424	0.7900	0.7895	**0.8052**	0.6640	0.7818	0.6576	0.6183
HTRU2	**0.9353**	0.8900	0.9259	0.9143	0.9127	0.8999	0.9105	0.8853	0.9073
QSAR	0.8532	0.8446	0.8437	0.8505	**0.8557**	0.8435	0.8452	0.8392	0.8497
Bean	0.9363	**0.9426**	0.9378	0.9364	0.9398	0.9337	0.9348	0.9274	0.9304
Iranian	0.9154	**0.9262**	0.8987	0.8693	0.9110	0.6501	0.8522	0.6684	0.9080
Credit	**0.6634**	0.6613	0.6090	0.6572	0.6595	0.6362	0.6573	0.6080	0.6174

4.3. Comparison with existing models

In order to comprehensively evaluate the performance of GTransformer, extensive horizontal comparison experiments are carried out on 12 datasets. The AUC, accuracy, precision, recall, and F1 score are used as evaluation indicators, and the average of five-fold cross-validation of each group of data is used as the final result. In this experiment, the number of randomly sampled granulation functions in GTransformer uses the best parameters in 4.1, the number of encoding layers is fixed to 1, and the number of attention heads is fixed to 1. In experiments, GTransformer is compared with a variety of existing models, including TabNet, MLP-Head, LightGBM, CatBoost, Support Vector Machine (SVM), Gradient Boosting Decision Tree (GBDT), Logistic regression (LR) and k-Nearest Neighbor (KNN). The specific parameters are set as follows: Comparison of the evaluation results of different models on 12 datasets shows that GTransformer has the best comprehensive performance, followed by CatBoost and LightGBM, and the other models do not achieve competitive results. The best results are marked in bold and shown in Table 4 to 7.

In the AUC comparison on 12 datasets, GTransformer is better than TabNet in most scenarios, as shown in Table 8. The Sonar dataset performs particularly well with an AUC gain of 14.5%. Although TabNet slightly outperforms HTRU2 and Bean datasets, GTransformer performs better on 12 datasets with an average AUC gain of 2.74%.

Table 7
F1 score Comparison of Different Models on 12 Datasets.

Datasets	GTransformer	TabNet	MLP-Head	LightGBM	CatBoost	SVM	GBDT	LR	KNN
WDBC	**0.9696**	0.9585	0.9446	0.9570	0.9570	0.9631	0.9569	0.9598	0.9389
Blood	**0.7062**	0.6376	0.6534	0.6326	0.6669	0.5955	0.6131	0.6060	0.6220
Raisin	**0.8607**	0.8588	0.8506	0.8319	0.8581	0.8509	0.8395	0.8494	0.8536
Ionosphere	**0.9365**	0.8757	0.9172	0.9151	0.9331	0.9149	0.8912	0.8807	0.8481
Spambase	0.9263	0.9276	0.9268	0.9434	**0.9503**	0.9259	0.9366	0.9101	0.8903
Sonar	0.8496	0.8126	0.7884	**0.8555**	0.8044	0.7388	0.8265	0.7294	0.7383
Online	0.7783	0.7653	0.7906	0.8005	**0.8144**	0.7321	0.7937	0.7309	0.6725
HTRU2	**0.9560**	0.9269	0.9303	0.9306	0.9320	0.9262	0.9249	0.9156	0.9273
QSAR	**0.8544**	0.8156	0.8243	0.8388	0.8431	0.8334	0.8345	0.8318	0.8333
Bean	**0.9428**	0.9441	0.9394	0.9391	0.9421	0.9375	0.9376	0.9315	0.9324
Iranian	**0.9282**	0.8879	0.8862	0.8740	0.9049	0.7671	0.8649	0.7459	0.9043
Credit	0.6994	**0.7108**	0.6531	0.7039	0.7055	0.6903	0.7028	0.6720	0.6490

Table 8
AUC Comparison Between GTransformer and TabNet on 12 Datasets.

Datasets	GTransformer	TabNet	Gain (%)
WDBC	99.3	98.6	**0.7**
Blood	79.9	76.4	**3.5**
Raisin	94.0	92.8	**1.2**
Ionosphere	97.6	91.4	**6.2**
Spambase	98.1	97.0	**1.1**
Sonar	89.7	75.2	**14.5**
Online	90.2	88.7	**1.5**
HTRU2	97.6	98.0	-0.4
QSAR	93.0	91.6	**1.4**
Bean	99.4	99.5	-0.1
Iranian	97.3	95.4	**1.9**
Credit	78.2	76.8	**1.4**

Table 9
Mean AUC Comparison Across 12 Datasets.

Model Name	Mean AUC (%)
GTransformer	**92.9**
TabNet	90.1
MLP-Head	91.5
LightGBM	**93.0**
CatBoost	**93.2**
SVM	88.7
GBDT	92.1
Logistic Regression	89.2
KNN	85.8

In the comparison results of the average AUC value of 12 datasets, CatBoost has the best performance with an average AUC of 93.2%, followed by LightGBM, and GTransformer is almost equal to LightGBM. The first three significantly outperform the other models, and the Top3 results are shown in Table 9 marked in bold. MLP-Head performs slightly worse than GTransformer, indicating that the combination of granular computing and attention mechanism has more advantages in complex feature extraction.

5. Conclusion

This paper presents GTransformer, a novel deep tabular data modeling framework that integrates granular computing theory with self-attention mechanisms to overcome the limitations of traditional deep learning models in tabular data tasks. By incorporating the multi-view functional granulation technique, GTransformer effectively captures complex feature interactions through diversified feature representations, thereby enhancing its generalization capability. Extensive experimental evaluations demonstrate that GTransformer consistently outperforms existing deep learning models such as TabNet across multiple datasets, while achieving comparable performance to tree-based ensemble models like LightGBM. Moreover, GTransformer exhibits remarkable robustness when handling noisy and incomplete data, highlighting its practical applicability in real-world scenarios. Future work will focus on optimizing the selection of granulation functions to minimize their impact on model performance. Additionally, we will explore the adaptability of GTransformer to unsupervised learning tasks, further expanding its applicability to diverse learning paradigms.

CRediT authorship contribution statement

Liang Liao: Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization. **Yumin Chen:** Writing – review & editing, Project administration, Methodology. **Yingyue Chen:** Writing – review & editing, Validation, Investigation. **Yiting Lin:** Writing – review & editing, Writing – original draft, Resources.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yumin Chen reports financial support was provided by Fujian Provincial Natural Science Foundation of China. Yumin Chen reports financial support was provided by Natural Science Foundation of Xiamen Municipality, China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Fujian Provincial Natural Science Foundation of China (2024J011192) and the Natural Science Foundation of Xiamen Municipality, China (3502Z202473069).

Data availability

Data will be made available on request.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[2] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2012) 84–90.
[3] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space. [Online]. Available: https://arxiv.org/abs/1301.3781, 2013.
[4] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, et al., Deep speech 2: end-to-end speech recognition in English and Mandarin, in: Proceedings of the International Conference on Machine Learning (ICML), PMLR, 2016, pp. 173–182.
[5] A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), 2017, pp. 6000–6010.
[6] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), vol. 1, 2019, pp. 4171–4186.
[7] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: a survey, IEEE Trans. Neural Netw. Learn. Syst. 35 (6) (2022) 7499–7519.
[8] S.O. Arik, T. Pfister, Tabnet: attentive interpretable tabular learning, Proc. AAAI Conf. Artif. Intell. 35 (8) (2021) 6679–6687.
[9] Y. Gorishniy, I. Rubachev, A. Babenko, On embeddings for numerical features in tabular deep learning, in: Advances in Neural Information Processing Systems (NeurIPS), 2022, pp. 24 991–25 004.
[10] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference on Machine Learning (ICML), 2006, pp. 161–168.
[11] K. Competitions, Kaggle: your machine learning and data science community, https://www.kaggle.com/, 2023.
[12] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 785–794.
[13] G. Ke, Q. Meng, T. Finley, et al., Lightgbm: a highly efficient gradient boosting decision tree, in: Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), 2017, pp. 3149–3157.
[14] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, Adv. Neural Inf. Process. Syst. 31 (2018).
[15] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, in: Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS), vol. 37, 2022, pp. 507–520.
[16] Z. Chen, M. Ma, T. Li, H. Wang, C. Li, Long sequence time-series forecasting with deep learning: a survey, Inf. Fusion 97 (2023) 101819.
[17] D. Ramachandram, G.W. Taylor, Deep multimodal learning: a survey on recent advances and trends, IEEE Signal Process. Mag. 34 (6) (2017) 96–108.
[18] D.P. Kingma, D.J. Rezende, S. Mohamed, et al., Semi-supervised learning with deep generative models, Adv. Neural Inf. Process. Syst. 27 (2014) 3581–3589.
[19] L.A. Zadeh, Fuzzy sets and information granularity, in: Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems, 1996, pp. 433–448.
[20] Z. Pawlak, Rough sets, Int. J. Comput. Inf. Sci. 11 (1982) 341–356.
[21] J.R. Hobbs, Granularity, in: Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI), 1985, pp. 432–435.
[22] T.Y. Lin, Granular computing: from rough sets and neighborhood systems to information granulation and computing with words, in: Proceedings of the European Congress on Intelligent Techniques & Soft Computing (EUFIT), 1996, pp. 1602–1606.
[23] D. Miao, Y. Zhao, Y. Yao, et al., Relative reducts in consistent and inconsistent decision tables of the Pawlak rough set model, Inf. Sci. 179 (24) (2009) 4140–4150.
[24] M. Song, W. Pedrycz, Granular neural networks: concepts and development schemes, IEEE Trans. Neural Netw. Learn. Syst. 24 (4) (2013) 542–553.
[25] W. Pedrycz, X. Wang, Optimal granularity of machine learning models: a perspective of granular computing, IEEE Trans. Fuzzy Syst. 32 (4) (2024) 2176–2186.
[26] W. Li, C. Deng, W. Pedrycz, O. Castillo, C. Zhang, T. Zhan, Double-quantitative feature selection approach for multigranularity ordered decision systems, IEEE Trans. Artif. Intell. 5 (5) (2024) 2385–2396.
[27] J. Liang, Y. Qian, Information granules and entropy theory in information systems, Sci. China, Ser. F, Inf. Sci. 51 (10) (2008) 1427–1444.
[28] S. Shi, R. Zhao, S. Li, X. Xie, L. Li, Z. Zhou, H. Liu, Intelligent prediction of surrounding rock deformation of shallow buried highway tunnel and its engineering application, Tunn. Undergr. Space Technol. 90 (2019) 1–11.
[29] H. Jiang, Y. Chen, H. Jiang, Y. Ni, H. Su, A granular sigmoid extreme learning machine and its application in a weather forecast, Appl. Soft Comput. 147 (2023) 110799.

[30] X. Fu, Y. Chen, J. Yan, Y. Chen, F. Xu, Bgrf: a broad granular random forest algorithm, J. Intell. Fuzzy Syst. 44 (5) (2023) 8103–8117.

[31] X. Deng, J. Li, Y. Qian, J. Liu, An emerging incremental fuzzy concept-cognitive learning model based on granular computing and conceptual knowledge clustering, IEEE Trans. Emerg. Top. Comput. Intell. 8 (3) (2024) 2417–2432.

[32] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.

[33] J. Chen, Z. Wang, Z. Lin, Rethinking attention mechanism for spatio-temporal modeling: a decoupling perspective in traffic flow prediction, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM), 2024, pp. 3032–3041.

[34] W. Pedrycz, Granular computing: an introduction, in: Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol. 3, 2001, pp. 309–328.

[35] J. Yao, A triarchic theory of granular computing, Granul. Comput. 1 (2016) 145–157.

[36] Y. Li, M. Yang, Z. Zhang, A survey of multi-view representation learning, IEEE Trans. Knowl. Data Eng. 31 (10) (2019) 1863–1883.

[37] J. Yoon, J. Jordon, M. van der Schaar, Gain: missing data imputation using generative adversarial nets, in: Proceedings of the 35th International Conference on Machine Learning (ICML), vol. 80, 2018, pp. 5689–5698.

[38] S. Patnaik, H. Changwal, M. Aggarwal, et al., Cabinet: content relevance based noise reduction for table question answering, in: The 12th International Conference on Learning Representations (ICLR), 2024.

[39] L. He, Y. Chen, C. Zhong, K. Wu, Granular elastic network regression with stochastic gradient descent, Mathematics 10 (15) (2022) 1–15.

[40] Y. Chen, X. Zhang, Y. Zhuang, B. Yao, B. Lin, Granular neural networks with a reference frame, Knowl.-Based Syst. 260 (2023) 1–14.

[41] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al., An image is worth 16x16 words: transformers for image recognition at scale, in: International Conference on Learning Representations (ICLR), 2020.

[42] D. Dua, C. Graff, Uci machine learning repository, http://archive.ics.uci.edu/ml, 2017.

[43] A.R.T. Donders, G.J.M.G. van der Heijden, T. Stijnen, et al., A gentle introduction to imputation of missing values, J. Clin. Epidemiol. 59 (10) (2006) 1087–1091.

[44] N. Ipsen, P.A. Mattei, J. Frellsen, How to deal with missing data in supervised deep learning?, in: The 10th International Conference on Learning Representations (ICLR), 2022.