Vortex Feature Positioning: Bridging Tabular IIoT Data and Image-Based Deep Learning

Vortex Feature Positioning: Bridging Tabular IIoT Data and Image-Based Deep Learning

Jong-Ik Park[a], Sihoon Seong[b], JunKyu Lee[c], Cheol-Ho Hong[b,*]

[a]*Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, 15213, PA, United States*
[b]*Department of Intelligent Semiconductor Engineering, Chung-Ang University, 84 Heukseok-ro, Dongjak District, Seoul, 06974, Korea*
[c]*Institute for Analytics and Data Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom*

Abstract

Tabular data from IIoT devices are typically analyzed using decision tree-based machine learning techniques, which struggle with high-dimensional and numeric data. To overcome these limitations, techniques converting tabular data into images have been developed, leveraging the strengths of image-based deep learning approaches such as Convolutional Neural Networks. These methods cluster similar features into distinct image areas with fixed sizes, regardless of the number of features, resembling actual photographs. However, this increases the possibility of overfitting, as similar features, when selected carefully in a tabular format, are often discarded to prevent this issue. Additionally, fixed image sizes can lead to wasted pixels with fewer features, resulting in computational inefficiency. We introduce Vortex Feature Positioning (VFP) to address these issues. VFP arranges features based on their correlation, spacing similar ones in a vortex pattern from the image center, with the image size determined by the attribute count. VFP outperforms traditional machine learning methods and existing conversion techniques in tests across seven datasets with varying real-valued attributes.

Keywords: IIoT tabular data, data augmentation, convolutional neural networks.

*Corresponding author

1. Introduction

The industrial Internet of Things (IIoT) collects vast amounts of sensor data commonly presented in a tabular form [1, 2, 3]. This industry data is often analyzed using traditional machine learning (ML) techniques based on decision tree algorithms [4], supported by voting algorithms [5], and enhanced by ensemble techniques such as Gradient Boosting [6], XGBoost [7], LightGBM [8], and CatBoost [9]. These techniques identify defective goods and detect anomalies in IIoT systems. However, the increasing complexity and breadth of attributes in IIoT data, along with the need for high-resolution data, often exceed the capabilities of traditional ML methods. For example, in predictive maintenance, sensors may record high-frequency vibration data that is highly granular, posing a challenge for traditional ML methods to process [10]. Similarly, in optimizing wafer fabrication processes, precise control of numerous parameters based on real-time sensor data is crucial for yield improvement, requiring more sophisticated solutions [11].

When the input data is numeric and highly complex, a deep learning approach becomes more suitable for analyzing tabular data [12]. Many researchers have explored using Convolutional Neural Networks (CNNs) [10] by converting tabular data into images. CNNs are a deep learning technique with numerous parameters, excelling particularly in handling image data that exhibit distinct patterns within pixels [13]. However, applying CNNs to tabular data is challenging because convolution kernels are designed to detect specific spatial patterns in multidimensional data. Typically, tabular data is represented as a vector, and even when reshaped into a 2-D matrix using conventional methods, the resulting 'image' often lacks meaningful spatial patterns. Consequently, for CNNs to perform effectively, the attributes of the tabular data must be strategically arranged in the reshaped matrix to create recognizable spatial patterns [10, 14].

Previous studies, such as DeepInsight [15], REFINED [16], IGTD [17], and SuperTML [18], have proposed methods to convert 1-D tabular data into 2-D image data by considering the positions of attributes. Specifically, DeepInsight, REFINED, and IGTD focus on grouping similar attributes with high correlations in specific locations within a 2-D matrix. This enables a CNN model to learn patterns within these similar attributes. However, this approach can lead to overfitting if similar attributes are intensively grouped in one area, making it difficult for the model to train on universal patterns between dissimilar attributes [19]. Additionally, these methods con-

vert tabular data into fixed-size images, which can result in wasted pixels for smaller datasets or insufficient space to represent all attribute values for larger datasets. Wasted pixels lead to computational inefficiency, as the CNN processes empty or irrelevant areas of the image, consuming resources without contributing to learning valuable patterns. Conversely, SuperTML carves tabular features (or attributes) of a sample (or row) onto an empty black image by assigning different font sizes to features based on their importance. While SuperTML has shown improved performance on some datasets, it has limitations: it may not be applicable when there are too many attributes to fit on a reasonably sized image, and its performance can vary depending on the font type.

This paper presents a novel approach, Vortex Feature Positioning (VFP), which converts tabular data into images tailored for CNNs while accounting for attribute correlations. With the increasing number of sensors in IIoT, there is a corresponding rise in attributes with real value sensor data [20]. To accommodate such IIoT applications, VFP arranges the attributes of the tabular data in a vortex shape, rotating from the center based on their Pearson correlation coefficient (PCC). This arrangement facilitates convolution operations, allowing for the optimal extraction of essential patterns. By positioning low-correlated features near the center of the converted image, VFP creates a convex-like loss landscape. A convex-like loss landscape helps prevent overfitting by ensuring that the optimization process is smoother and less likely to get stuck in local minima. It encourages the model to find broader, more generalizable solutions rather than narrowly focusing on specific patterns that may not apply to unseen data. Additionally, this vortex shaping allows for the formation of images with flexible sizes, ensuring that the spatial representation is optimally adapted to the number of attributes. This flexibility further contributes to the efficient training of the CNN model.

We evaluated VFP on datasets related to typical industrial environments and general datasets to ensure its applicability across a broad range of IIoT

Table 1: Overview of datasets in this study, detailing the number of attributes, sample count, and labels, and noting any missing data.

Name	Iris	Wine	Dry Bean	HELOC	HIGGS	Epileptic Seizure	SECOM	DARWIN	Eating
# of attributes	4	12	16	23	28	178	591	450	6,373
# of samples	150	6,497	13,611	11,459	11,000,000	11,500	1,567	174	945
# of labels	3	2	7	2	2	5	2	2	7
Missing data	No	Yes	No	No	No	No	Yes	No	No

environments. These datasets were sourced from the UCI Machine Learning Repository [21], Kaggle [22], and OpenML [23]. The industrial datasets include Wine, Dry Bean, and SECOM (semiconductor manufacturing data). For general IIoT applications, we included Epileptic Seizure, Eating, DARWIN, Iris, HELOC, and HIGGS. Detailed descriptions of each dataset are provided in Table 1. As shown in the evaluation section, VFP outperforms traditional ML techniques such as XGBoost and CatBoost, and it generally excels across most datasets compared to techniques utilizing CNNs, such as DeepInsight, REFINED, and IGTD.

Machine and deep learning approaches aim to effectively discover complex but generalized patterns in training data. In this context, VFP makes three significant contributions:

- VFP outperforms traditional ML techniques for tabular data and previous methods for converting tabular data into images for CNNs.
- VFP can convert tabular data with varying numbers of attributes into images with an optimized number of pixels.
- Since VFP is a data format transformation technique, it can be utilized with any state-of-the-art CNNs and training methods.

With these contributions, VFP presents a valuable and versatile tool for transforming tabular data into images suitable for use with CNNs.

The rest of the paper is structured as follows: In Section 2, we discuss related works. We present our proposed VFP method in Section 3. Section 4 shows the strategic positioning of correlated features and convergence analysis of VFP, Section 5 describes the experimental evaluation, and in Section 6, we conclude the paper and discuss future work.

2. Related Work

2.1. Machine Learning Techniques for Tabular Data

Traditional ML methods for analyzing tabular data include Gradient Boosting [6], XGBoost [7], LightGBM [8], and CatBoost [9], all based on decision tree algorithms [4]. These techniques are widely used and remain dominant over CNNs when handling tabular data [24]. They mitigate overfitting in regression or classification tasks by employing ensemble methods such as boosting and bagging [6].

Despite their strengths, these models face significant challenges when handling high-dimensional data with numerous real-valued attributes. The primary mechanism in these tree-based models involves splitting the data at various thresholds to maximize criteria like information gain or Gini impurity. However, many real-valued attributes exponentially increase the number of potential split points. This elevates the computational burden and increases the risk of overfitting, as the models may select splits that fit the training data too closely, failing to generalize to unseen data [6, 7, 9]. Also, the curse of dimensionality becomes particularly problematic in high-dimensional spaces. As the number of attributes grows, the data points become increasingly sparse, making it difficult for the models to find statistically significant and generalizable splits. This sparsity reduces the models' ability to capture meaningful patterns, leading to degraded performance on test data [1, 25]. Moreover, these models are inherently sensitive to feature correlation, a common characteristic of real-valued attributes in industrial sensor data. Highly correlated features can lead to redundant splits, where the model repeatedly selects similar features, adding complexity without contributing to the model's predictive power. This redundancy not only increases computational costs but also hampers the interpretability and efficiency of the model [8].

LightGBM, with its leaf-wise growth algorithm, was designed to improve training speed by selecting optimal leaves to split rather than following a level-wise approach. However, this method often struggles to capture the detailed interactions between features crucial in high-dimensional spaces, resulting in performance that does not consistently surpass that of XGBoost and CatBoost [9]. As a result, there is a pressing need for new methods to handle the challenges of tabular data with numerous real-valued attributes more effectively.

2.2. Converting Tabular Data into Images for CNNs

Industries such as semiconductor manufacturing generate extensive tabular data characterized by real-number attributes, which differ from integer-based categorical attributes. These data sets, often collected from many sensors at high frequencies, present significant challenges due to their scale and complexity. Traditional machine learning techniques struggle with such data, particularly as the dimensionality increases, leading to inefficiency in capturing meaningful patterns [1, 10, 25].

CNNs are specifically designed to handle complex patterns in 2-D spatial data, such as images [13], but their effectiveness is limited when applied

to 1-D tabular data [10, 24]. This limitation arises because CNNs excel at detecting spatial hierarchies and local patterns within grid-like structures where the proximity of data points is meaningful. In contrast, traditional machine learning models often need help with high-dimensional tabular data, where interactions between features can be complex and non-linear, making it difficult for these models to generalize effectively.

To leverage the strengths of CNNs in capturing intricate feature interactions, researchers have proposed various methods to convert tabular data into images where spatial relationships can be encoded. Techniques such as DeepInsight [15], REFINED [16], IGTD [17], and SuperTML [18] have shown superior performance compared to traditional machine learning approaches, particularly in scenarios with numerous attributes. Transforming tabular data into 2-D representations allows CNNs to exploit their inherent ability to model local dependencies and complex patterns, which are challenging for conventional models to capture in high-dimensional spaces. This approach enhances predictive accuracy and enables more effective feature learning and representation in complex datasets. ‘

However, these methods can lead to overfitting or the oversight of global patterns [10]. A common approach among these methods is assigning each attribute a fixed position within a 2D (or 3D) matrix by grouping similar attributes. While this may seem logical, it can hinder learning complex patterns across different attributes, as convolutional operations might overly focus on patterns among adjacent, similar features [10, 26]. This strategy contrasts with traditional machine learning practices, which often recommend dropping highly correlated attributes to prevent overfitting [10, 27].

SuperTML [18] engraves features of tabular data onto an empty black image (i.e., a zero 2-D matrix), with each feature engraved in varying sizes based on its importance. Although SuperTML performed better than traditional ML techniques in some datasets, it faces challenges in determining the extent to increase image size with a growing number of attributes. It also requires prioritizing attributes due to considerations of font size when the choice of font type influences engraving features and its performance. These constraints highlight the necessity for a more generalized method.

Despite the enhanced performance achieved by DeepInsight, REFINED, IGTD, and SuperTML, it remains to be seen whether the improvement stems from CNNs' ability to detect complex patterns through 2-D convolution operations or primarily from their feature positioning methods. To address this uncertainty, we introduce 12 different feature positioning scenarios in CNNs

in Section 3. These scenarios aim to demonstrate that the effectiveness of CNNs in detecting complex patterns is not solely responsible for improved test performance; how features are positioned also plays a critical role. Based on these insights, we introduce a new method for positioning features called Vortex Feature Positioning (VFP). VFP directly transforms all attributes of tabular data into images based on their correlations, reducing the risk of overfitting by considering the varying degrees of correlations. Additionally, VFP adjusts the image size based on the number of attributes, directly converting all tabular data attributes into images based on their correlations and accounting for their correlation degrees, unlike prior methods with fixed image sizes.

3. Vortex Feature Positioning

As the number of real value attributes in tabular data increases, traditional ML techniques exhibit lower performance and slower training speeds [10]. CNNs can overcome these limitations by converting tabular data into images and exploiting the benefits of 2-D convolution operations to capture complex patterns in tabular data [15, 17].

Previous methods of converting tabular data into images aggregate similar features in specific locations. However, because tabular data is inherently heterogeneous, its features do not correspond to pixels in a literal image, where similar pixels are naturally grouped together to form coherent patterns. Aggregating similar features of tabular data in this context may lead to overfitting, especially when dealing with highly correlated features [10, 27]. Our new method, Vortex Feature Positioning (VFP), addresses these issues by taking into account two critical factors:

- Tabular data is heterogeneous because it comes from distinct sensors, and its features are not like the pixels in an actual image. Therefore, we interpret the functionality of 2-D convolutions as forming appropriate patterns based on tabular features such as numerical values, categories, or textual information rather than merely extracting patterns like an actual image.

- Highly correlated features of tabular data should be positioned far away from each other.

First, we explain how to embed features into a 2-D matrix while considering convolution operations in Section 3.1. Then, we describe how to arrange features based on the correlation of attributes in an image in Section 3.2.

3.1. Embedding Features Considering Convolution Operations

CNNs, such as ResNet [28] and DenseNet [29], typically employ 3×3 kernels in the convolution layer. The results of convolution operations with 3×3 kernels and feature maps in the first layer impact the kernels of every subsequent layer and, consequently, influence the final inferences [13]. Therefore, in the first layer, we need to consider the number of features in an image per convolution operation to determine the complexity of patterns. To simplify the explanation, assume there are $m \times n$ attributes in tabular data, where m and n represent the numbers of rows and columns in the feature matrices, respectively. We examine three methods for embedding features into a 2-D matrix: zero-padding with sizes of 1 and 2, as well as distancing, as illustrated in Fig. 1.

Zero Padding of Size 1 (ZPOS1)

$$\begin{aligned} N_4^{Pad1} &= 4 \\ N_6^{Pad1} &= 2 \times (m-2) + 2 \times (n-2) = 2m + 2n - 8 \\ N_9^{Pad1} &= (m-2) \times (n-2) = mn - 2m - 2n + 4 \end{aligned} \tag{1}$$

Zero Padding of Size 2 (ZPOS2)

$$\begin{aligned} N_1^{Pad2} &= 4 \qquad N_2^{Pad2} = 8 \qquad N_4^{Pad2} = 4 \\ N_3^{Pad2} &= 2 \times (m-2) + 2 \times (n-2) = 2m + 2n - 8 \\ N_6^{Pad2} &= 2 \times (m-2) + 2 \times (n-2) = 2m + 2n - 8 \\ N_9^{Pad2} &= (m-2) \times (n-2) = mn - 2m - 2n + 4 \end{aligned} \tag{2}$$

Distancing

$$\begin{aligned} N_1^{Dist} &= m \times n = mn \\ N_2^{Dist} &= n \times (m-1) + m \times (n-1) = 2mn - m - n \\ N_4^{Dist} &= (m-1) \times (n-1) = mn - m - n + 1, \end{aligned} \tag{3}$$

Here, N_i^{case} represents each case's convolution operations involving i features. The 3×3 kernels of convolution layers can handle up to nine features.

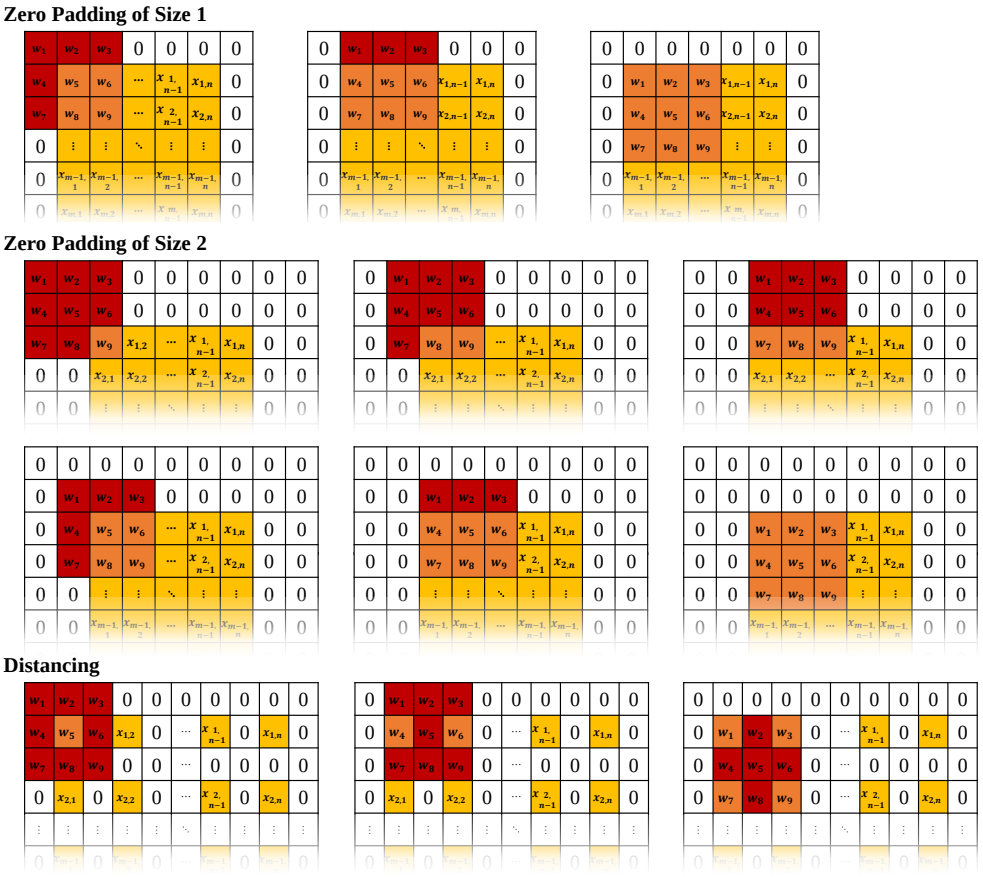

Figure 1: Three cases of feature positioning considering the number of features per convolution operation. From the top: zero padding of sizes 1-2 and distancing.

The numbers of convolution operations per the number of features dealt with at once are calculated by Equations 1, 2, and 3, and the results are shown in Table 2.

ZPOS1 uses three cases of the number of features for convolution operations totaling mn. ZPOS2 uses six cases of features for convolution operations totaling $mn + 2m + 2n + 4$. Distancing uses three cases of the number of features for convolution operations totaling $4mn - 2m - 2n + 1$. The number of total convolution operations denotes how many patterns CNNs have and the number of features used at once, which are related to the complexity of patterns. For example, if nine features ($m = 3$ and $n = 3$) exist, ZPOS2 and distancing perform twenty-five convolution operations identically. How-

Table 2: Number of convolution operations per the number of features in cases with zero padding of sizes 1 and 2, and distancing.

Cases	Zero Padding of Size 1	Zero Padding of Size 2	Distancing
1 feature	N/A	4	mn
2 features	N/A	8	$2mn - m - n$
3 features	N/A	$2m + 2n - 8$	N/A
4 features	4	4	$mn - m - n + 1$
6 features	$2m + 2n - 8$	$2m + 2n - 8$	N/A
9 features	$mn - 2m - 2n + 4$	$mn - 2m - 2n + 4$	N/A
All cases	mn	$mn + 2m + 2n + 4$	$4mn - 2m - 2n + 1$

ever, if there are more than nine features, distancing performs many more convolution operations and uses fewer features than ZPOS2. Therefore, distancing considers many rough patterns between features, while ZPOS2 considers complex patterns as combinations of various features. Since ZPOS1 uses only four, six, and nine features and performs mn convolution operations, it is beneficial for avoiding overfitting compared to ZPOS2, which finds excessively detailed patterns. We also embed features in an image with no padding or distancing. However, convolution operations only use nine features at once, and the number of convolution operations is $(m-2)(n-2)$, the smallest among the methods. Overall, the number of features used in a single convolution operation and the complexity of patterns should be considered when embedding features into a 2-D matrix.

3.2. Arranging Features Considering Correlations of Attributes

The traditional ML techniques may exhibit overfitting when heavily relying on highly correlated attributes without dropping them during training [27, 30]. Previous methods of converting tabular data into images have mainly focused on placing similar features together, resulting in better performance [10, 15]. The high ability of CNNs to capture critical patterns contributes to increased performance. However, we cannot conclude that gathering similar features increases performance because each attribute in tabular data, which is heterogeneous, differs from pixels in an actual image, which represents homogeneous data [10].

VFP performs convolution operations on low-correlated attributes by placing features with low correlation at a blank image's center (in ascend-

Algorithm 1: Vortex Feature Positioning: This algorithm arranges attributes based on the sum of Pearson correlation coefficients (PCCs). It takes a set of attributes $\mathbb{X}$ and outputs the attributes arranged in a vortex pattern $\mathbb{X}_{ordered}$. The vector $\mathbf{c}$ contains the sum of absolute PCCs for each feature c_i, which determines the order.

1: **Input:** $\mathbb{X} = \{\mathbf{x}_i | i = 1, 2..., k\}$
2: **Output:** $\mathbb{X}_{ordered}$
3: $\mathbf{c} = \{c_i | i = 1, 2, ..., k\}$
4: **for** $i \in 1, 2, ..., k$ **do**
5: $\quad c_i \leftarrow 0$
6: $\quad$ **for** $j \in 1, 2, ..., k$ **do**
7: $\quad\quad c_i \leftarrow c_i + |r(\mathbf{x}_i, \mathbf{x}_j)|$
8: $\quad$ **end for**
9: **end for**
10: $\mathbb{X}_{ordered} \leftarrow \mathbb{X}[rank_{ascending}(\mathbf{c})]$ or $\mathbb{X}[rank_{descending}(\mathbf{c})]$

ing order). This is because 2-D convolution operations use center-located features more than edge-located features.

In the context of investigating attribute relationships, the Pearson Correlation Coefficient (PCC), denoted as $r(\mathbf{a}, \mathbf{b})$ in Equation 4, is employed. This coefficient measures the linear correlation between two attributes, $\mathbf{a}$ and $\mathbf{b}$, which are sets of data points in a dataset. The PCC is calculated by dividing the sum of the products of the deviations of each data point from their respective means ($\mathbf{a}_i - \overline{\mathbf{a}}$ and $\mathbf{b}_i - \overline{\mathbf{b}}$) by the square root of the product of the sums of the squared deviations.

$$r(\mathbf{a}, \mathbf{b}) = \frac{\sum^{n}(\mathbf{a}_i - \overline{\mathbf{a}})(\mathbf{b}_i - \overline{\mathbf{b}})}{\sqrt{(\sum^{n}(\mathbf{a}_i - \overline{\mathbf{a}})^2)(\sum^{n}(\mathbf{b}_i - \overline{\mathbf{b}})^2)}}, \tag{4}$$

Algorithm 1 outlines the arranging process, which sorts attributes in ascending or descending order based on ***the sum of absolute values of PCCs*** aiming to investigate whether gathering similar values improves performance more than spreading them out.

$\mathbb{X}$ is a tabular dataset composed of attributes $\mathbf{x}_i$ (*line 1* in Algorithm 1). $\mathbf{c}$ is a vector composed of c_i, which is the sum of PCCs of $\mathbf{x}_i$ and other attributes (*line 3*). $rank_{ascending}(\mathbf{c})$ and $rank_{descending}(\mathbf{c})$ are functions that

arrange $\mathbf{c}$ according to c_i size in ascending or descending order and provide ordered indices (*lines 4-9*). We arrange $\mathbb{X}$ based on the order stored in $\mathbf{c}$ (*line 10*).

After arranging attributes based on the PCC, features of attributes are embedded in an image. CNNs perform more convolution operations on features at the center of the image than on exterior features. Therefore, placing features in a vortex shape from the center of the image exploits desired features for many convolution operations, while undesired features are the opposite. For example, arranging features in ascending order according to the sum of PCCs performs many convolution operations with low-correlated features such as x_1^c, x_2^c, x_3^c, and x_4^c, which are similar to feature selection in ML techniques for tabular data [30]. In contrast, features x_n^c and x_{n-1}^c, which have high PCCs and are located near the edge, are used for relatively fewer convolution operations.

We use a single-channel image (i.e., a grayscale image) in VFP. However, state-of-the-art CNNs (i.e., ResNet and DenseNet) require 3-channel images. This paper employs Pre-activated ResNet-18 [28]. Therefore (R), green (G), and blue (B) channels all have the same feature map, as shown in Fig. 2.

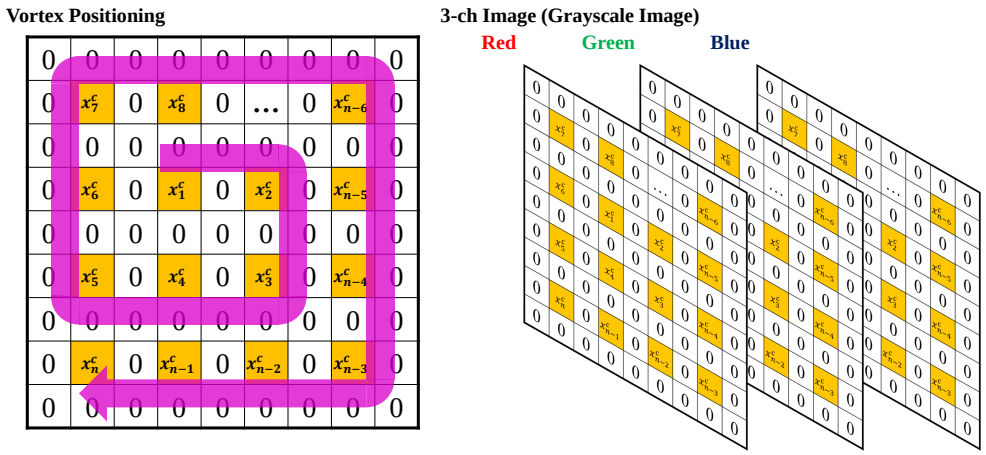

Figure 2: Vortex Feature Positioning and forming a 3-channel image by copying a 2-D matrix.

4. Analysis of Vortex Feature Positioning

The VFP method optimizes the arrangement of features within an image representation of tabular data to fully leverage CNNs' strengths. This section provides a detailed analysis of the key design choices in VFP, specifically the strategic positioning of correlated features and the utilization of CNNs' inherent central bias. Using these design choices, we analyze the potential for improved convergence properties in CNNs, particularly when using stochastic optimization techniques like SGD.

4.1. Strategic Positioning of Correlated Features

4.1.1. Mathematical Justification

A critical aspect of VFP is the deliberate spatial separation of correlated features within the image. This strategy aims to reduce redundancy in the information processed by CNNs, thereby minimizing the risk of overfitting and enhancing the model's generalization capability. In CNNs, the output of a convolutional operation for a given pixel $y(i, j)$ in the feature map is given by:

$$y(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} x(i+m-1, j+n-1) \cdot w(m,n) + b,$$

where $y(i, j)$ is the output feature at position (i, j), $x(i + m - 1, j + n - 1)$ represents the input values within the receptive field, $w(m, n)$ are the kernel weights, $M \times N$ is the kernel size, b is the bias term.

When correlated features are adjacent, the convolution operation tends to reinforce these correlations, which can cause the model to focus excessively on localized patterns, leading to overfitting. By placing correlated features far apart, VFP ensures that CNNs process these features in different parts of the image, forcing the network to learn more diverse, global patterns. This decorrelation can be expressed as minimizing the covariance $\text{Cov}(f_i, f_j)$ between feature activations f_i and f_j across the network:

$$\text{Cov}(f_i, f_j) = \frac{1}{N} \sum_{n=1}^{N} (f_i^n - \mu_{f_i})(f_j^n - \mu_{f_j}),$$

where N is the number of samples, μ_{f_i} and μ_{f_j} are the mean activations of features f_i and f_j, respectively.

By reducing the covariance between features, VFP promotes a more generalized learning process, enabling CNNs to capture complex interactions across the entire feature space.

4.1.2. Leveraging CNNs' Central Bias

CNNs naturally emphasize features located near the center of an image due to the expansion of the receptive field as layers deepen. This characteristic is strategically utilized in VFP by positioning critical features near the center of the image, where they are more likely to influence the network's output.

4.1.3. Receptive Field Expansion

The receptive field R_l at any given layer l can be calculated recursively as:

$$R_l = R_{l-1} + (k_l - 1) \prod_{i=1}^{l-1} s_i,$$

where k_l is the kernel size at layer l, s_l is the stride at layer l, and R_{l-1} is the receptive field of the previous layer.

For a convolutional layer with a stride of 1 ($s_l = 1$), this simplifies to:

$$R_l = R_{l-1} + (k_l - 1)$$

As the network depth increases, the receptive field expands, particularly overlapping more in the image's central region. This overlap results in a bias towards the central features, which influence more neurons in deeper layers of the network.

4.1.4. Impact of Global Average Pooling

In many CNNs, global average pooling (GAP) is used to reduce the spatial dimensions of the feature maps before the fully connected layers. The output of a GAP layer for a channel c is calculated as:

$$\text{GAP}(c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} y_c(i, j),$$

where $H \times W$ are the height and width of the feature map, $y_c(i, j)$ is the feature map value at position (i, j) for channel c.

Due to the denser overlap of receptive fields at the center, central features disproportionately contribute to the final output after GAP, reinforcing the central bias.

4.1.5. Summary

VFP exploits this central bias by positioning essential features near the center of the image, ensuring they receive more attention during the convolution process. This approach maximizes CNNs' ability to capture critical patterns and relationships within the data, leading to improved model performance. By combining decorrelating feature positions and leveraging the central bias, VFP enhances CNNs' effectiveness in handling complex, high-dimensional tabular data. This dual approach is key to the improved performance observed in our experiments, as discussed in Section 5.

4.2. Convergence Analysis

4.2.1. Preliminaries: Vortex Feature Positioning

Vortex Feature Positioning (VFP) arranges tabular data into an image format by positioning attributes outward from the center, with features closer to the center being less correlated. This spatial layout facilitates the learning of feature relationships through convolution operations.

4.2.2. Assumptions

The convergence analysis of Stochastic Gradient Descent (SGD) in the context of VFP relies on the following assumptions:

- **Locally Convex Regions**: While the entire loss surface of a CNN trained on VFP images might be non-convex, regions close to the center (where less correlated attributes reside) are approximated as locally convex.

- **Bounded Gradients**: The gradient magnitude does not grow indefinitely. This is a reasonable assumption given the structured nature of VFP images and the bounded activations in CNNs.

- **Lipschitz Continuity of Gradients**: Despite the spatial rearrangements due to VFP, we assume that small changes in the input lead to proportionally small changes in the output. This is represented by a Lipschitz constant L such that:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

- **Diagonal Dominance in Hessian for Central Features**: For features positioned centrally in VFP images (and thus less correlated),

their Hessian matrix is approximately diagonal, indicating limited interactions with other features. Consider two features, x_t^k and x_t^l, with their effect on the loss f represented as a second-order term in the Hessian matrix H:

$$H_{ij} = \frac{\partial^2 f}{\partial x_t^k \partial x_t^l}$$

For uncorrelated features x_t^k and x_t^l, H_{kl} is close to zero. A function is convex if its Hessian is positive semi-definite everywhere. With VFP, central features (less correlated) have a Hessian structure that's approximately diagonal:

$$\mathbf{H} \approx \text{diag}(\lambda_1, \lambda_2, ..., \lambda_m), \quad \lambda_n \geq 0$$

where λ_n are eigenvalues. Local convexity around these features is suggested if these eigenvalues are non-negative.

4.2.3. Convolution's Role in Convergence

In CNNs, convolution operations, especially in the initial layers, focus on the local regions of the input image. Given the VFP structure, these convolutions will predominantly operate on the less correlated central features in their early stages. This behavior ensures:

- **Unique Feature Capture**: Convolutions can effectively capture the unique characteristics of these less correlated features, providing more informative gradient signals during backpropagation.
- **Gradient Quality**: Gradients derived from less correlated features are likely more distinct, reducing the chances of vanishing or exploding gradients, especially in the early layers.

4.2.4. Correlation-Induced Convexity

The combined influence of less correlated features x_t^k and x_t^l on the loss f can be depicted by the Hessian entry H_{ij}, which is expected to be close to zero. The diagonal-like Hessian for central features in VFP images suggests potential local convex regions.

4.2.5. SGD Convergence Analysis with VFP Images

In the context of VFP, we analyze the convergence of Stochastic Gradient Descent (SGD), which updates the model using randomly sampled mini-batches, incorporating the spatial structure of VFP images into the gradient computation.

Consider the general SGD update rule:

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(x_t, \theta_t),$$

where θ_t are the model parameters, and x_t is a mini-batch of the feature vectors sampled at iteration t, η_t is the learning rate, and $\nabla f(x_t, \theta_t)$ is the stochastic gradient of the loss function with respect to both x and θ at iteration t.

In the case of VFP, the gradient incorporates the interaction between features:

$$\nabla f(x_t, \theta_t) = \frac{\partial f}{\partial x_t}(\theta_t) + \sum_{k \neq l} C_{kl} \frac{\partial f}{\partial x_t}(\theta_t),$$

where C_{kl} represents the interaction coefficient between features x_t^k and x_t^l based on their spatial positioning in the VFP image. The interaction term C_{kl} captures the correlations between features. When features x_t^k and x_t^l are weakly correlated, the interaction term C_{kl} becomes small.

In mini-batch SGD, these interaction terms tend to cancel out in expectation due to random sampling:

$$\mathbb{E}\left[\sum_{k \neq l} C_{kl} \frac{\partial f(x_t)}{\partial \theta_t}\right] = 0.$$

Thus, the gradient estimate $\nabla f(x_t; \theta_t)$ is a noisy approximation of the true gradient of the full loss function $F(\theta_t)$, but remains unbiased:

$$\mathbb{E}[\nabla f(x_t; \theta_t)] = \nabla F(\theta_t),$$

where $F(\theta_t)$ is the full objective function, and the expectation $\mathbb{E}[\cdot]$ is taken over the randomness in the selection of the mini-batch x_t.

The SGD update rule, considering the VFP structure, becomes:

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{\partial f}{\partial x_t}(\theta_t) + \sum_{k \neq l} C_{kl} \frac{\partial f}{\partial x_t}(\theta_t) \right).$$

We analyze the squared norm of the parameter difference:

$$\|\theta_{t+1} - \theta^*\|^2 = \|\theta_t - \eta_t \nabla f(x_t; \theta_t) - \theta^*\|^2.$$

Expanding this:

$$\|\theta_{t+1} - \theta^*\|^2 = \|\theta_t - \theta^*\|^2 - 2\eta_t \nabla f(x_t; \theta_t)^\top (\theta_t - \theta^*) + \eta_t^2 \|\nabla f(x_t; \theta_t)\|^2.$$

Using the convexity assumption:

$$f(x^*) \geq f(x_t) + \nabla f(x_t; \theta_t)^\top (\theta^* - \theta_t),$$

we obtain:

$$\nabla f(x_t; \theta_t)^\top (\theta_t - \theta^*) \geq f(x_t) - f(x^*).$$

Substituting this inequality into the squared norm expression gives:

$$\mathbb{E}[\|\theta_{t+1} - \theta^*\|^2] \leq \|\theta_t - \theta^*\|^2 - 2\eta_t \mathbb{E}[f(x_t) - f(x^*)] + \eta_t^2 \mathbb{E}[\|\nabla f(x_t; \theta_t)\|^2].$$

We assume that the gradient norm is bounded:

$$\mathbb{E}[\|\nabla f(x_t; \theta_t)\|^2] \leq \sigma^2,$$

leading to the simplified inequality:

$$\mathbb{E}[f(x_t) - f(x^*)] \leq \frac{1}{2\eta_t} \left(\|\theta_t - \theta^*\|^2 - \mathbb{E}[\|\theta_{t+1} - \theta^*\|^2] \right) + \frac{\eta_t}{2} \sigma^2.$$

With the assumption that expectations are linear and distributed over sums, summing the inequality over t from 1 to T yields:

$$\sum_{t=1}^{T} \mathbb{E}[f(x_t) - f(x^*)] \leq \frac{\|\theta_1 - \theta^*\|^2}{2\eta_t} + \frac{\eta_t \sigma^2 T}{2}.$$

With a diminishing learning rate $\eta_t = \frac{\eta}{\sqrt{t}}$, the convergence rate becomes:

$$\mathbb{E}[f(x_t) - f(x^*)] = O\left(\frac{1}{\sqrt{t}} \right).$$

4.2.6. Summary

The interaction terms in the gradient estimate cancel out due to randomness in mini-batch selection, ensuring that the gradient remains unbiased. Therefore, despite spatial correlations between features in VFP, the stochastic gradient descent maintains the typical $O(1/\sqrt{t})$ convergence rate.

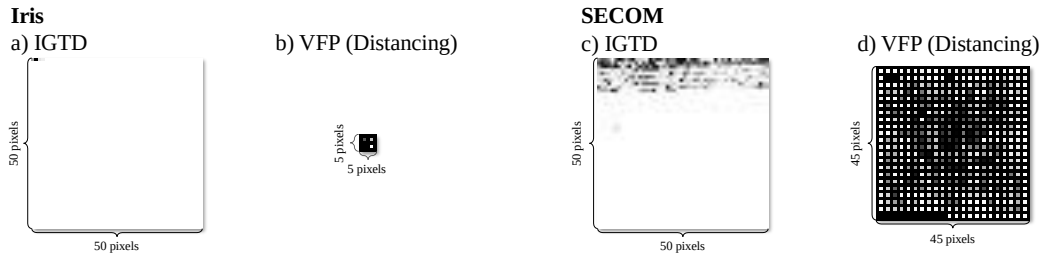

Figure 3: Converted images of IGTD and VFP (with distancing) using Iris and SECOM datasets.

5. Experimental Evaluation

In this evaluation, we analyze nine benchmark tabular datasets, categorized by the number of attributes and samples. These datasets are sourced from the UCI Machine Learning Repository [21], OpenML [23], Kaggle [22], and a peer-reviewed paper [31] (Penglung). The datasets include Iris, Wine, Dry Bean, and SECOM, widely used in industrial applications. Additionally, we evaluate the Epileptic Seizure, DARWIN, and Eating datasets, known for their challenges in medical data training, the HELOC dataset, which highlights the disparity between current market value and purchase price in the housing sector, and the HIGGS dataset, which is used to distinguish between processes that produce Higgs bosons and those that do not.

Each dataset's details are summarized in Table 1. Iris, Wine, Dry Bean, HELOC, and HIGGS have relatively small numbers of attributes, while Epileptic Seizure and SECOM have 178 and 591 attributes, respectively. We also consider cases where the number of attributes far exceeds the number of instances, such as DARWIN and Eating. This scenario is particularly important because in cases with more attributes than instances, deterministic methods such as constructing the Hessian matrix are not feasible, leading to the necessity of using machine learning or deep learning approaches [32, 31]. Therefore, by comparing the performance across different methods, we can empirically determine which approaches are more effective in handling high-dimensional data with limited instances. All datasets contain real-valued attributes, which are challenging to train using ML techniques for tabular data. Each dataset is randomly split into training and testing sets in a 0.8:0.2 ratio, and a Min-Max scaler is applied to normalize the data.

Fig. 3 shows converted images for the Iris and SECOM datasets using IGTD and VFP with distancing. IGTD [17] is a recent method for converting

Table 3: Averaged testing accuracies (with standard deviations) using VFP. Feature arrangement methods include ascending (less correlated features near the center), descending (more correlated features near the center), and random(where features were randomly positioned). Each case was tested with eight trials, and the cases achieving the top two accuracies for each dataset are bolded.

Optimizer		SGD							AdamW	
Data Name		Iris	Wine	Dry Bean	HELOC	HIGGS	Epileptic Seizure	SECOM	DARWIN	Eating
Distancing	Ascending	**96.7 (4.08)**	**99.7 (0.22)**	**93.5 (0.33)**	72.6 (0.71)	**78.3 (0.07)**	**76.1 (0.50)**	93.0 (0.78)	**91.1 (2.83)**	**63.0 (3.00)**
	Descending	**95.8 (3.23)**	**99.7 (0.21)**	**93.5 (0.40)**	**72.7 (0.88)**	77.8 (0.05)	**76.1 (0.90)**	**93.1 (0.83)**	**90.4 (4.02)**	**61.5 (3.37)**
	Random	**97.1 (2.14)**	**99.8 (0.19)**	**93.5 (0.43)**	**72.7 (0.81)**	77.7 (0.10)	76.0 (0.85)	**93.1 (0.84)**	**90.4 (2.13)**	**60.6 (4.13)**
Zero Padding of Size 1	Ascending	**94.6 (4.70)**	**99.7 (0.22)**	93.4 (0.34)	72.2 (0.81)	**77.7 (0.06)**	**74.9 (0.99)**	**93.0 (0.84)**	**92.1 (3.97)**	**62.1 (2.80)**
	Descending	94.6 (3.70)	99.6 (0.21)	**93.5 (0.37)**	**72.3 (0.82)**	77.6 (0.07)	**75.0 (0.84)**	**93.1 (0.73)**	**93.2 (3.03)**	61.3 (2.97)
	Random	**96.3 (4.15)**	**99.7 (0.23)**	**93.5 (0.31)**	72.2 (0.89)	**77.8 (0.14)**	75.5 (0.99)	**93.1 (0.83)**	92.1 (2.96)	59.3 (3.00)
Zero Padding of Size 2	Ascending	**97.1 (2.60)**	**99.7 (0.22)**	**93.5 (0.34)**	**72.5 (0.81)**	**77.8 (0.10)**	**75.3 (0.47)**	**93.1 (0.73)**	**90.4 (2.13)**	61.9 (3.94)
	Descending	**97.5 (2.20)**	**99.8 (0.19)**	93.5 (0.30)	**72.4 (1.00)**	77.6 (0.04)	75.0 (0.43)	**93.1 (0.83)**	93.2 (2.13)	60.8 (3.05)
	Random	**95.4 (3.96)**	99.7 (0.18)	**93.5 (0.45)**	72.5 (0.90)	77.8 (0.10)	75.1 (0.42)	**93.1 (0.95)**	91.1 (4.69)	60.8 (1.99)
No Padding & Distancing	Ascending	**96.3 (2.00)**	**99.7 (0.26)**	**93.5 (0.41)**	72.6 (0.80)	**78.1 (0.06)**	**75.4 (0.73)**	**93.0 (0.78)**	**89.3 (3.97)**	**61.8 (3.07)**
	Descending	95.0 (5.53)	99.7 (0.18)	93.5 (0.29)	**72.7 (0.87)**	77.9 (0.06)	**75.1 (0.32)**	93.0 (0.74)	**92.1 (3.33)**	59.8 (2.33)
	Random	94.6 (5.02)	**99.7 (0.27)**	93.4 (0.47)	72.5 (0.89)	77.8 (0.03)	75.3 (0.94)	**93.1 (0.82)**	**90.4 (3.03)**	60.0 (0.85)

tabular data into images that has shown better performance and smaller image sizes than other methods such as DeepInsight [15] and REFINED [16]. The image size generated by IGTD is fixed at 50×50 pixels, regardless of the number of attributes in the dataset. Although the authors [17] mention that the image size can be changed according to the number of attributes, there is no proposed generalized rule for image sizes across different datasets. In contrast, VFP, with distancing, adjusts the image size based on the number of attributes, resulting in image sizes of 5×5 and 45×45 pixels for the Iris and SECOM datasets, respectively.

5.1. Models

We evaluate performance using Pre-activated ResNet-18 (for Iris, Wine, Dry Bean, HELOC, HIGGS, and Epileptic Seizure) and ResNet-10 (for DARWIN and Eating) architectures, both constructed by stacking residual blocks. He et al. [28] used four types of residual blocks to build ResNets. In our setup, ResNet-18 consists of 2 blocks for each type, while ResNet-10 consists of 1 block per type. We use these simpler ResNet variants to demonstrate the effectiveness of CNNs compared to traditional ML techniques for tabular data.

Table 4: Averaged testing accuracies (with standard deviations) of VFP, traditional machine learning techniques, and previous deep learning-based image conversion techniques. The cases achieving the top two accuracies for each dataset are bolded.

Data Name		Iris	Wine	Dry Bean	HELOC	HIGGS	Epileptic Seizure	SECOM	DARWIN	Eating
VFP (Ascending)	Distancing	**96.7 (4.08)**	**99.7 (0.22)**	**93.5 (0.33)**	72.6 (0.71)	**78.3 (0.07)**	**76.1 (0.50)**	93.0 (0.78)	**91.1 (2.83)**	**63.0 (3.00)**
	Zero Padding of Size 1	**94.6 (4.70)**	**99.7 (0.22)**	93.4 (0.34)	72.2 (0.81)	**77.7 (0.06)**	**74.9 (0.99)**	**93.0 (0.84)**	**92.1 (3.97)**	**62.1 (2.80)**
	Zero Padding of Size 2	**97.1 (2.60)**	**99.7 (0.22)**	**93.5 (0.34)**	**72.5 (0.81)**	**77.8 (0.10)**	**75.3 (0.47)**	**93.1 (0.73)**	**90.4 (2.13)**	61.9 (3.94)
	No Padding & Distancing	**96.3 (2.00)**	**99.7 (0.26)**	**93.5 (0.41)**	72.6 (0.80)	**78.1 (0.06)**	**75.4 (0.73)**	**93.0 (0.78)**	**89.3 (3.97)**	**61.8 (3.07)**
ML Techniques	XGBoost	90.8 (6.18)	99.5 (0.21)	93.0 (0.21)	72.2 (0.86)	76.2 (0.02)	70.8 (0.44)	**93.0 (0.89)**	87.9 (3.43)	60.8 (4.06)
	CatBoost	92.9 (3.89)	99.5 (0.14)	93.0 (0.38)	72.6 (0.74)	73.9 (0.02)	71.6 (0.53)	**93.0 (0.78)**	88.2 (2.65)	**62.7 (4.9)**
DL Image Converting Techniques	DeepInsight	96.2 (0.03)	99.3 (0.00)	91.6 (0.00)	**73.0 (0.01)**	71.8 (0.04)	67.2 (1.59)	92.8 (0.01)	90.0 (0.04)	14.8 (0.00)
	REFINED	**97.1 (3.51)**	99.3 (0.24)	91.1 (0.79)	71.7 (0.93)	**78.2 (0.06)**	70.7 (1.05)	**93.1 (0.86)**	**90.0 (4.29)**	46.6 (2.07)
	IGTD	95.8 (5.46)	99.7 (0.17)	93.0 (0.42)	72.0 (0.65)	78.0 (0.07)	73.9 (1.07)	**93.1 (0.97)**	86.8 (2.81)	51.0 (2.94)

One modification we made is changing the kernel size in the first layer to 3×3 with a stride of 1 to be consistent with VFP's convolution operations.

5.2. Experimental Setups

Our experiments were conducted on an NVIDIA A100 40GB GPU. We applied a cosine annealing warm-up with restarts for optimization, using the SGD optimizer with a learning rate varying between 0.01 and 0.001 for the Iris, Wine, Dry Bean, HELOC, HIGGS, Epileptic Seizure, and SECOM datasets. These learning rates were reset every five epochs. For the DARWIN and Eating datasets, we used the AdamW optimizer, with fixed learning rates of 0.005. The experiments were run with a mini-batch size of 128. We trained each dataset for 20 epochs (HIGGS), and 200 epochs for the others. To ensure the reliability of the results, each experiment was repeated eight times using different random seeds ranging from 1000 to 8000, incremented by 1000.

5.3. Converting Tabular Data into Images

To convert tabular data into images, we first embed one sample of each tabular dataset into a 2-D matrix with different types of zero padding and distancing: zero padding of size 1 (ZPOS1), zero padding of size 2 (ZPOS2), and distancing. We then convert the resulting 2-D matrix into a 3-channel image by copying itself.

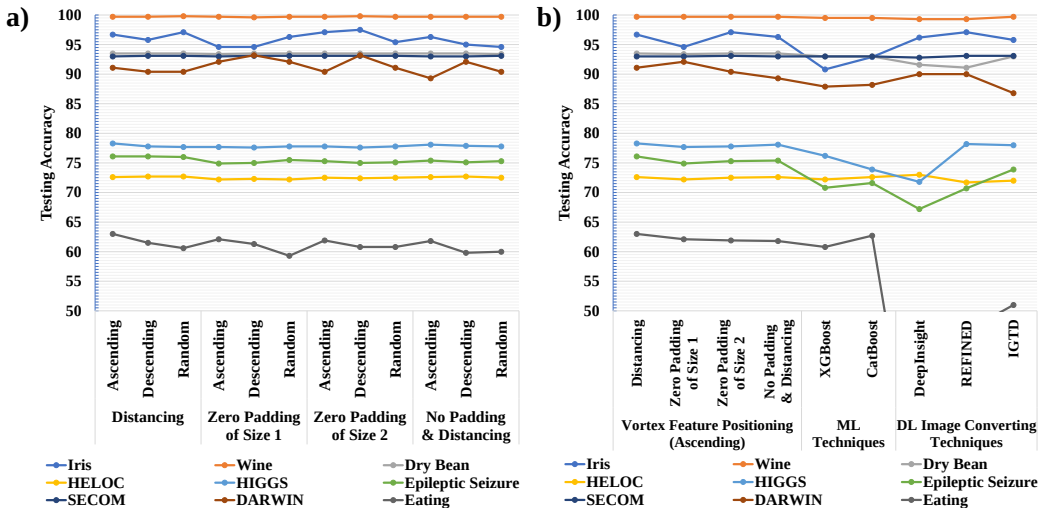

Figure 4: Part **a)** lines correspond to the test accuracies presented in Table 3, while part **b)** lines relate to the test accuracies in Table 4. VFP consistently shows better testing performance than other methods.

5.4. Employing Vortex Feature Positioning

VFP improves performance by conducting convolution operations on selected features. The test results, presented in Table 3 (which highlights the top two testing accuracies for each scenario, categorized by dataset and feature positioning method) and in Fig. 4 part **a)**, demonstrate that arranging features in ascending order based on their correlation from the center leads to better performance.

Among 8 trials, arranging features in ascending order achieved a top-two accuracy in 30 cases, compared to 20 for descending order and 18 for random feature arrangement out of 36 testing scenarios. In these scenarios, descending order did not show a significant difference from random feature arrangement, whereas ascending order consistently outperformed. The choice of strategy—whether distancing, zero-padding (sizes 1 or 2), or no padding—varied depending on the dataset's complexity. As shown in Table 3, distancing achieved top performance in 22 cases, zero-padding size 1 in 17 cases, zero-padding size 2 in 15 cases, and no padding (or distancing) in 14 cases. This suggests that distancing consistently delivers moderately strong results, even when using only 1, 2, or 4 features. This simplicity helps avoid overfitting and enhances overall performance. In conclusion, a tailored embedded space that leverages specific feature relationships leads to more

effective patterns and improved performance.

5.5. Comparison Performance with Machine Learning Techniques for Tabular Data

We evaluated the effectiveness of VFP against traditional machine learning techniques (XGBoost and CatBoost) as well as previously established deep learning conversion methods (DeepInsight, REFINED, and IGTD). For traditional machine learning techniques, we modified the maximum number of leaves in CatBoost from 16 to 64 and adjusted the minimum child weight in XGBoost from 1 to 9. Additionally, we varied the maximum tree depth from 4 to 16 for both CatBoost and XGBoost. For established deep learning conversion methods, we applied the same training configuration as that used in VFP cases.

These comparisons are detailed in Table 3 and illustrated in part **b)** of Fig. 4. For the machine learning techniques, we conducted eight repeated experiments at various tree depths, ranging from one to sixteen. Similarly, we repeated the tests eight times for the deep learning approaches, maintaining the same training conditions used for VFP.

The results indicate that VFP surpasses other methods across most of the datasets tested. Particularly, compared to traditional machine learning techniques, VFP achieved test accuracy improvements in the following ranges: Iris (4.5–6.9%), Wine (0.2–0.2%), Dry Bean (0.5–0.5%), HELOC (0.0–0.6%), HIGGS (2.8–6.0%), Epileptic Seizure (6.3–7.5%), SECOM (0.1–0.1%), DARWIN (4.4–4.8%), and Eating (0.5–3.6%). Compared to deep learning approaches, VFP also showed accuracy improvements of Iris (0.0–1.4%), Wine (0.0–0.4%), Dry Bean (0.5–2.6%), HELOC (-0.5–1.3%), HIGGS (0.1–9.1%), Epileptic Seizure (3.0–13.2%), SECOM (0.0–0.3%), DARWIN (2.3–6.1%), and Eating (23.5–325.7%).

Overall, our evaluation demonstrates the robustness and superiority of VFP in handling diverse datasets, offering a significant improvement over traditional machine learning techniques and existing deep learning-based image conversion methods. The flexibility in adjusting image sizes based on the number of attributes and the strategic arrangement of features contributes to VFP's enhanced performance.

By capitalizing on the strengths of both CNNs and VFP's unique feature arrangement strategy, our method provides a reliable and efficient approach to transforming tabular data into a format that maximizes the potential of deep learning models. This leads to better generalization, improved accuracy,

and a reduction in overfitting, making VFP a valuable tool for a wide range of applications in industrial and other domains.

6. Conclusions

This paper introduces Vortex Feature Positioning (VFP), a novel method for converting tabular data into images based on attribute correlations. VFP strategically positions attributes to fully exploit the strengths of convolutional neural networks (CNNs), leading to performance improvements across various datasets. Specifically, VFP achieves performance increases ranging from 0.0% to 7.5% compared to traditional machine learning techniques.

When compared to existing methods for converting tabular data into images, VFP demonstrates performance improvements ranging from -0.5% to 325.7%. These results suggest that VFP effectively mitigates overfitting by considering the correlations among attributes. Additionally, VFP can dynamically adjust the size of the converted images to accommodate different numbers of attributes, making it a flexible and adaptable solution for complex industrial tabular data, including that generated by Industrial Internet of Things systems.

VFP's capability to transform tabular data into a format optimized for CNNs enhances model performance. Our experiments show that VFP consistently outperforms traditional machine learning methods and other deep learning-based image conversion techniques. This makes VFP a valuable tool for a wide range of applications in industrial and other domains where handling complex, high-dimensional tabular data is crucial. Therefore, future work will focus on optimizing VFP and exploring its applications to broaden its utility and impact in various data-driven industries.

References

[1] J. Wang, W. Zhang, Y. Shi, S. Duan, J. Liu, Industrial big data analytics: challenges, methodologies, and applications, arXiv preprint arXiv:1807.01016 (2018).

[2] L. V. H. Vardhan, S. Kok, Generating privacy-preserving synthetic tabular data using oblivious variational autoencoders, Proceedings of the Workshop on Economics of Privacy and Data Labor at the 37 th International Conference on Machine Learning (2020).

[3] S. Mahanthappa, B. Chandavarkar, Data formats and its research challenges in iot: A survey, Evolutionary Computing and Mobile Sustainable Networks: Proceedings of ICECMSN 2020 (2021) 503–515.

[4] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1986) 81–106.

[5] T. G. Dietterich, Ensemble methods in machine learning, in: International workshop on multiple classifier systems, Springer, 2000, pp. 1–15.

[6] J. H. Friedman, Stochastic gradient boosting, Computational statistics & data analysis 38 (2002) 367–378.

[7] 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, 2016, pp. 785–794.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems 30 (2017).

[9] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, Advances in neural information processing systems 31 (2018).

[10] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).

[11] K. Kitchat, C.-Y. Lin, M.-T. Sun, Defective wafer detection using sensed numerical features, in: 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), IEEE, 2021, pp. 1–6.

[12] D. Babaev, M. Savchenko, A. Tuzhilin, D. Umerenkov, E.t.-rnn: Applying deep learning to credit loan applications, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2183–2190. URL: https://doi.org/10.1145/3292500.3330693. doi:10.1145/3292500.3330693.

[13] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.

[14] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, Advances in neural information processing systems 35 (2022) 507–520.

[15] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, T. Tsunoda, Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture, Scientific reports 9 (2019) 1–7.

[16] O. Bazgir, R. Zhang, S. R. Dhruba, R. Rahman, S. Ghosh, R. Pal, Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks, Nature communications 11 (2020) 1–13.

[17] Y. Zhu, T. Brettin, F. Xia, A. Partin, M. Shukla, H. Yoo, Y. A. Evrard, J. H. Doroshow, R. L. Stevens, Converting tabular data into images for deep learning with convolutional neural networks, Scientific reports 11 (2021) 1–11.

[18] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, J. Dong, Supertml: Two-dimensional word embedding for the precognition on structured tabular data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[19] S.-H. Moon, Y.-H. Kim, An improved forecast of precipitation type using correlation-based feature selection and multinomial logistic regression, Atmospheric Research 240 (2020) 104928.

[20] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, B. Qureshi, An overview of iot sensor data processing, fusion, and analysis techniques, Sensors 20 (2020) 6076.

[21] University of California, Irvine, Uci machine learning repository, `https://archive.ics.uci.edu/ml/index.php`, 2023. Accessed: May 2023.

[22] Kaggle, Kaggle datasets, https://www.kaggle.com/datasets, 2023. Accessed: May 2023.

[23] OpenML, Openml datasets, https://openml.org/search?type=data, 2024. Accessed: August 2024.

[24] R. Shwartz-Ziv, A. Armon, Tabular data: Deep learning is not all you need, Information Fusion 81 (2022) 84–90.

[25] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference, and prediction, 2017.

[26] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, arXiv preprint arXiv:1603.07285 (2016).

[27] L. Yu, H. Liu, Feature selection for high-dimensional data: A fast correlation-based filter solution, in: Proceedings of the 20th international conference on machine learning (ICML-03), 2003, pp. 856–863.

[28] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, Springer, 2016, pp. 630–645.

[29] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[30] K. Grabczewski, N. Jankowski, Feature selection with decision tree criterion, in: Fifth International Conference on Hybrid Intelligent Systems (HIS'05), IEEE, 2005, pp. 6–pp.

[31] H. M. Zawbaa, E. Emary, C. Grosan, V. Snasel, Large-dimensionality small-instance set feature selection: A hybrid bio-inspired heuristic approach, Swarm and Evolutionary Computation 42 (2018) 29–42.

[32] P. Bühlmann, S. Van De Geer, Statistics for high-dimensional data: methods, theory and applications, Springer Science & Business Media, 2011.