Journal of Data Science 23 (1), 109–129
February 2025
DOI: 10.6339/24-JDS1122
Data Science in Action

BIE: Binary Image Encoding for the Classification of Tabular Data

James Halladay[1,*], Drake Cullen[1], Nathan Briner[1], Darrin Miller[1], Riley Primeau[1], Abraham Avila[1], Warin Watson[1], Ram Basnet[1], and Tenzin Doleck[2]

[1]*Department of Computer Science and Engineering, Colorado Mesa University (CMU), Grand Junction, CO 81501, USA*
[2]*Faculty of Education, Simon Fraser University, Burnaby, BC V5A 1S6, Canada*

Abstract

There has been remarkable progress in the field of deep learning, particularly in areas such as image classification, object detection, speech recognition, and natural language processing. Convolutional Neural Networks (CNNs) have emerged as a dominant model of computation in this domain, delivering exceptional accuracy in image recognition tasks. Inspired by their success, researchers have explored the application of CNNs to tabular data. However, CNNs trained on structured tabular data often yield subpar results. Hence, there has been a demonstrated gap between the performance of deep learning models and shallow models on tabular data. To that end, Tabular-to-Image (T2I) algorithms have been introduced to convert tabular data into an unstructured image format. T2I algorithms enable the encoding of spatial information into the image, which CNN models can effectively utilize for classification. In this work, we propose two novel T2I algorithms, Binary Image Encoding (BIE) and correlated Binary Image Encoding (cBIE), which preserve complex relationships in the generated image by leveraging the native binary representation of the data. Additionally, cBIE captures more spatial information by reordering columns based on their correlation to a feature. To evaluate the performance of our algorithms, we conducted experiments using four benchmark datasets, employing ResNet-50 as the deep learning model. Our results show that the ResNet-50 models trained with images generated using BIE and cBIE consistently outperformed or matched models trained on images created using the previous State of the Art method, Image Generator for Tabular Data (IGTD).

Keywords *computer vision; DeepInsight; IGTD; native representation; ResNet-50; tabular-to-image*

1 Introduction

Since AlexNet achieved state-of-the-art accuracy during the Large Scale Visual Recognition Challenge in 2012, deep learning techniques have rapidly advanced in the domain of image recognition (Alom et al., 2018; Krizhevsky et al., 2012). However, applying techniques like these to structured data was much less successful due to the lack of properties that encode information about the data such as a temporal or spatial ordering (Sun et al., 2019). Tabular data is a common form of structured data arranged into rows and columns of numerical or categorical values. In contrast, images are unstructured data typically represented as a collection of pixels

*Corresponding author. Email: jehalladay112@gmail.com.

Received October 13, 2023; Accepted February 13, 2024

with varying intensities and colors. Many methods have been proposed for deep learning on structured data, but they often rely on hand-engineered features or complex neural architectures that may be computationally expensive and difficult to interpret. These approaches do not typically outperform traditional machine learning methods like Random Forest or XGBoost (Sun et al., 2019).

Converting tabular data into images, however, has recently gained attention as a potential approach for enhancing the efficacy of deep learning models when dealing with classification tasks involving tabular data, such as distinguishing between malignant and benign data, categorical/species classification, rankings, etc (Sun et al., 2019; Sharma et al., 2019; Zhu et al., 2021). By transforming tabular data into an image format, we can leverage the powerful feature extraction capabilities of CNNs (Noroozi and Favaro, 2016). This provides the opportunity to encode spatial information into the image, which the model can use for classification. CNNs can draw inferences from several features of an image, such as local correlation between pixels or global structures that emerge from groups of correlated pixels. They are also robust to noise and transformation invariant due to the pooling operations between convolutional layers, such as max or average pooling (Alom et al., 2018). This motivates research to create tabular-to-image (T2I) transformations that take advantage of these features.

The core idea behind T2I transformations is to encode a 1D feature vector into a 2D format. One promising T2I transformation algorithm, IGTD (Zhu et al., 2021), achieves this by assigning each feature to a single pixel and arranging them in a 2D space based on their correlation (Zhu et al., 2021). By doing so, IGTD takes advantage of the powerful feature extraction capabilities of CNNs in computer vision tasks. The pixel arrangements based on correlations also add spatial information to the image, further enhancing classification performance (Zhu et al., 2021). This imposes an ordering upon the data into the image that imitates how pixel values in a typical image are highly correlated with their surrounding neighbors.

Simpler T2I approaches such as flow-wrapping or one-hot encoding also exist (Krupski et al., 2021). Flow-wrapping reshapes a feature vector of length N into a 2D vector of shape M-rows by K-columns, where M x K = N the length of the feature vector (Krupski et al., 2021). Similarly, one-hot encoding assigns each feature m buckets and represents categorical and continuous features alike by filling in a single bucket with a value of 1 while leaving the other m empty (Krupski et al., 2021). This transforms each feature vector into a vector of shape n by m, where the majority of the values are empty. These methods all attempt to represent a number as a feature vector to create a 2D vector, but overlook a simple way to represent each feature as a vector: the native binary representation of their data format.

In this paper, we introduce two novel T2I techniques, Binary Image Encodings (BIE) and Correlated Binary Image Encoding (cBIE), for transforming structured data into a corresponding image format. BIE converts a 1D feature vector of length n into a 2D vector representing a grayscale image. Given a binary representation with precision p, for each feature, we convert it into a vector of length p where each element corresponds to a binary digit in the feature's binary representation. These vectors are then concatenated to create an n by p image.

Since each element of this vector corresponds to a digit in the binary representation, they are all correlated with each other. This is similar to IGTD, which seeks correlation between all pixels in 2D space; however, our method only achieves this in one dimension. To fully achieve correlation between pixels in both dimensions, we can follow the same path as IGTD and impose an ordering based on the correlation between the features. The vectors for each feature are then concatenated with each other based on this ordering. Since every pixel is correlated to the pixels generated from the same feature, as well as the feature vector concatenated above and below its

position, this algorithm imitates how pixels in an image are correlated with their surrounding neighbors, hence the name cBIE.

There are several advantages to using binary encodings for tabular classification tasks. By converting tabular data into images, we can leverage powerful pre-trained networks such as ResNet-50, which have been trained on large-scale image datasets and have learned to extract meaningful features from them. These features can then be used for tabular data classification, potentially improving the performance of the models. BIE and cBIE capture complex relationships between variables in tabular data that may not be easily captured by traditional machine learning methods. These techniques also allow CNNs to take advantage of the structure of the data since structured data types can be broken into separate parts. For example, floating point values have their structure broken down into a sign bit, an exponent, and a mantissa. Then, since these discrete parts will be stacked on top of each other in the final image, CNN filters can draw inference based on how parts of the structure differ. In addition, cBIE captures correlations between the features, encoding them in a way that preserves the correlation structure between them. This can be particularly useful in domains such as finance, where variables are often highly correlated (Albanese et al., 2013).

Furthermore, BIE and cBIE are simple and computationally efficient. They represent the data in a highly compact form and do not require complex hand-engineering. The features do not need normalization or other transformation techniques to take advantage of this algorithm since binary floating-point numbers and integers can represent an incredibly large space of possible numbers. This also allows us to take advantage of the highly optimized bitwise operations available in modern processors, making the transformation process computationally efficient.

Our paper introduces two novel methods, BIE and cBIE, for classification tasks that address several limitations of current deep learning approaches for structured data. Unlike existing methods that require data normalization and may lose information (Das and Spanos, 2022), our methods utilize a native representation of data that retains all the original information. Furthermore, our approach does not require domain-specific knowledge, making it accessible to a wider range of users. In previous methods, T2I algorithms such as Deep Insight perform averages over each feature value to calculate the pixel values, which creates a lossy representation. Our compact representation, which can be applied to any structured data type with fixed length, takes advantage of the architecture of a CNN to create a feature neighborhood structure. Our experiments demonstrate that our methods perform as well as, if not better than, current state-of-the-art algorithms.

The rest of the paper is organized as follows. In Section 2, we review related works on deep learning for structured data, with a focus on methods that use image-based representations. Section 3 provides a detailed description of our proposed BIE and cBIE methods. In Section 4, we describe the architecture used in our experiments. Sections 5, and 6 cover our experiments and results. We discuss limitations of our experiments and suggest future work in Section 7. Finally, Section 8 presents our conclusion.

2 Related Works

In this section, we discuss related works and techniques used to develop computer vision algorithms and their application to tabular data. In Section 2.1, we highlight the significant advancements in computer vision achieved through Deep Learning (DL) methods, including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks

(RNN), and Long Short-Term Memory (LSTM). We then provide an overview of some of the most prominent DL models in computer vision, including AlexNet, VGGnet, and ResNet, which have significantly contributed to the field's progress. The subsequent subsections delve into the techniques of converting tabular data into image form. We provide insights into the research and development of these techniques and their potential for application in computer vision.

2.1 Computer Vision

DL has revolutionized the field of Computer Vision with powerful brain-inspired learning methods (Alom et al., 2018). DL models such as DNNs, CNNs, RNNs, and LSTMs have been widely used for photo classification, face recognition, and other computer vision tasks.

In 2012, Alex Krizhevsky's AlexNet, a deeper and wider CNN model, won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), marking a groundbreaking moment in the history of DL (Alom et al., 2018; Krizhevsky et al., 2012). AlexNet's architecture featured convolution, max pooling, and Local Response Normalization (LRN) in the first two layers and Fully Connected (FC) layers with dropout in the last two layers (Krizhevsky et al., 2012). LRN and dropout were key innovations that distinguished AlexNet from other models at the time until the introduction of Batch normalization (Alom et al., 2018; Simon et al., 2016).

In 2014, the Visual Geometry Group (VGGnet) also participated in ILSVRC and established itself as a prominent model due to the better classification accuracy achieved by deeper networks (Alom et al., 2018). VGGnet's architecture consisted of two convolutional layers with ReLU activation functions, followed by a max pooling layer (Simonyan and Zisserman, 2014). This was then repeated with additional convolutional and ReLU layers, and another max pooling layer at the end, followed by three FCs. VGGnet's versions were numbered according to the number of layers, for example VGG-16 has 13 convolutional layers, with 3 FC layers at the end of the architecture (Alom et al., 2018; Simonyan and Zisserman, 2014). However, the computational expense of the larger VGGnets, like VGG-19, which has 138M weights and 15.5M MACs, was a significant drawback (Alom et al., 2018).

In 2015, He et al. (2016a) introduced Residual Network (ResNet), which won the ILSVRC challenge with its innovative architecture (Alom et al., 2018; He et al., 2016a,b). ResNet used many layers, ranging from 34 to 1202. The most popular version, ResNet-50, contained 49 convolutional layers and 1 FC output layer (Alom et al., 2018; He et al., 2016a). ResNet was able to mitigate or prevent the issue of vanishing gradient, which was a major problem with previous DL models (Alom et al., 2018). It achieved this by introducing residual connections that enable learning of the residual functions with respect to the layer inputs (Alom et al., 2018; He et al., 2016a). Moreover, ResNet used Batch Normalization (instead of AlexNet's LRN) before being put into an activation function (e.g., ReLU) (He et al., 2016a). The entire ResNet-50 model has 25.5M weights and 3.9M MACs (He et al., 2016a). Improvements to ResNet include the aggregated residual transform, proposed in 2016 by Xie et al. (2017).

2.2 Tabular-to-Image Algorithms

The following subsections discuss some common methods for converting tabular data into image format for training DL models.

2.2.1 One-Hot Encoding

One-Hot Encoding is a widely employed technique utilized to convert categorical data into a numerical representation better suited for training models. This technique is particularly well-suited for categorical data with fixed, preferably limited, finite categories as it transforms the original information into a set of binary features (Krupski et al., 2021). A new feature represents each possible category in the original data; for each sample, only the feature corresponding to the sample's category is assigned a value of one, while all other possible categories are assigned a value of zero. This binary representation can easily be transformed into an image where ones are represented as a white pixel and zeros as black pixels. It is worth noting, however, that one-hot encoding also has certain limitations. When the cardinality of categorical features is high, it can significantly strain a computer's memory resources. Furthermore, in domains with sparse and high-dimensional features, storing parameter vectors for one-hot encoded data can become challenging, even for simple models (Seger, 2018).

2.2.2 Flow-Wrapping

Flow wrapping has been commonly used in the field of network traffic classification (Krupski et al., 2021; Moskalenko et al., 2020) and was first used by Wang et al. (2017). This method was practiced in many accredited papers which saw great use in DL models. The major drawback to this method of tabular-to-image is the necessity of normalizing the data.

The paper by Moskalenko et al. (2020) use network flow (tuples of packet data) to convert their datasets into images via flow wrapping. In flow wrapping, data samples must be normalized into an interval, thus the process of converting the sample into images is not robust to outliers and adds significant bias in production based on the training distribution. The original values undergo min-max normalization such that all values are in the rage [0, 1] which are then multiplied by 255 to bring them into a greyscale pixel space. The sample is transformed from a 1xN vector where N is the number of features into a MxM matrix where $M = ceil(\sqrt{N})$ by simply padding the feature vector with $M^2 - N$ empty values and reshaping it, shown in Figure 1. This gives us a square greyscale image which can be consumed by DL models to perform classification or

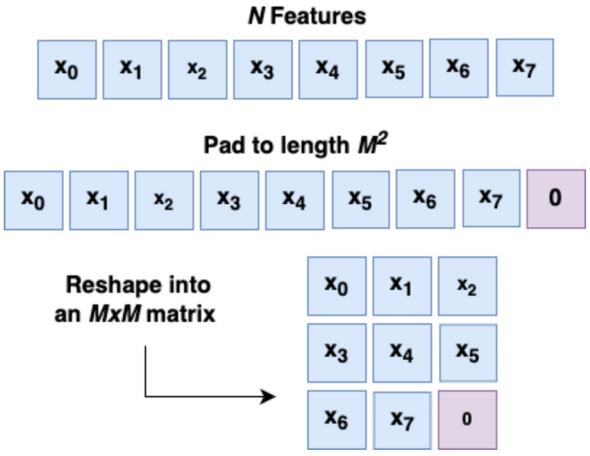

Figure 1: After the features are brought into pixel space, they are padded and reshaped into a square matrix.

regression tasks.

Moskalenko et al. (2020) used each network flow as packets. They limited the length of each flow to 784 bytes so the data can be transformed into 28 by 28 pixel images. This was then put through a LeNet-5 feature extractor to calculate pixel activation of each channel from OMP or LARS. After training models for classification, they found the accuracy to be 96.1% on decision rules of malware traffic detection.

2.2.3 DeepInsight

Sharma et al. (2019) introduced the DeepInsight architecture for transforming non-image data into images that can be used as inputs for CNNs. This architecture clusters the dataset's most similar features and spreads out any dissimilar features. The features clustered in a 2D plane can be converted into an M by N image and processed into a CNN for classification into the image's corresponding category. This method of data transformation has proven useful in domains such as genetics (gene-expression), vowels, texts, and other artificial data.

Rabbah et al. (2022) utilized DeepInsight to convert data into images and input them into a deep neural network for classifying potential churners for service providers. Gokhale et al. (2023) introduced Improved DeepInsight to convert gene-expression data into images for a Vision Transformer (ViT) to classify. As a result, their proposed method outperformed nine state-of-the-art methods for classification in the domain of gene-expression.

2.2.4 TAC: Tabular Convolution

Buturović and Miljković (2020) introduced TAC, a novel CNN method for tabular data classification. TAC transforms input vectors into kernels and performs convolutions on a fixed base image, saving the resulting images. TAC outperformed shallow learning methods on gene expression data, and they claim that the additional computation time is inconsequential with contemporary computer architecture.

2.2.5 SuperTML and DWTM

Sun et al. (2019) propose a new method called SuperTML for classification on tabular data. The method involves projecting the features of tabular data into two-dimensional embeddings like an image and then feeding this image into fine-tuned two-dimensional CNN models for classification. The experimental results have shown that the proposed SuperTML method had achieved state-of-the-art results on both large and small datasets.

Iqbal et al. (2022) introduce the Dynamic Weighted Tabular Method (DWTM), a novel feature embedding technique for applying CNNs to tabular datasets. DWTM dynamically assigns feature weights based on their correlation to class labels, and converts data points into images for input into a CNN architecture. The paper demonstrates the effectiveness of DWTM by applying it to six benchmark tabular datasets using three different CNN architectures and achieving an average accuracy of 98

2.2.6 IGTD

Zhu et al. (2021) proposed an innovative technique called Image Generator for Tabular Data (IGTD) to enhance existing image generation methods. The optimization algorithm transforms tabular data into images by assigning each feature to a pixel based on the ranking of pairwise

distances between features and the assigned pixels. The algorithm aims to minimize the difference between these two measurements by calculating pairwise distances through a distance measure such as Euclidean distance or Pearson Correlation Coefficient. This process assigns similar features to adjacent pixels and dissimilar features to distant pixels.

The effectiveness of this method is attributed to a greedy iterative process that swaps pixel assignments of features to optimally reduce the distance between them. In contrast to DeepInsight, IGTD generates dense image representations where each pixel represents a distinct feature, resulting in smaller images that require less time for training CNNs. IGTD does not necessitate domain knowledge and exhibits exceptional feature preservation as closer features are more alike. Furthermore, the size and shape of the generated image can be adjusted, making it applicable to a wide range of domains.

Zhu et al. (2021) evaluated the performance of IGTD against CNNs trained with DeepInsight and REFINED (REpresentation of Features as Images with NEighborhood Dependencies) images on datasets comprising gene expression profiles of cancer cell lines and molecular descriptors of drugs. CNNs trained on IGTD demonstrated comparable or superior prediction performance relative to other T2I methods. Choi et al. (2022) utilized IGTD to create images for predicting optimal charge level in electric vehicles. EfficientNet trained with IGTD generated images outperformed a RNN and basic CNN as well as traditional shallow models such as k-nearest neighbors. Similarly, Taheri et al. (2022) used IGTD to predict critical temperature in superconductors, outperforming the XGBoost method in the R2 criterion. They also found that IGTD outperformed DeepInsight when training their CNN with generated images.

3 Binary Encoding

The subsequent sections provide an in-depth exploration of the inner mechanics of the proposed BIE and cBIE T2I techniques. These techniques address the limitations of prior work by preserving all data information without normalization and requiring no domain-specific knowledge. They apply to all structured data types with fixed length and have the capacity to outperform current state-of-the-art algorithms. Section 3.1 presents a comprehensive overview of BIE, followed by a detailed explanation of its algorithm in Section 3.2. Subsequently, the cBIE technique is introduced and juxtaposed against the conventional BIE methodology. Lastly, a discussion section is included for further insights and analysis.

3.1 Binary Image Encoding Overview

We introduce BIE, a novel methodology for converting binary images into a format that is optimally compatible with CNNs. Our approach draws inspiration from the one-hot encoding technique mentioned in Krupski et al. (2021). One-hot encoding involves the transformation of data into a two-dimensional image through the application of one-hot encoding to each feature of the sample. In contrast to traditional one-hot encoding, BIE leverages the inherent power of the binary representation for data types such as floating-point, integer, and boolean values. As shown in Figure 2, a double-precision binary number consists of a sign bit, an exponent determining the magnitude, and a mantissa representing the significant digits of the value. Figure 2 illustrates representations of integer and boolean values, where all fields are either '1' or '0' for booleans. Notably, the binary value is propagated throughout the entire row, influencing the convolutional filter as it traverses the image. By utilizing the unique structure of the data type, BIE effectively

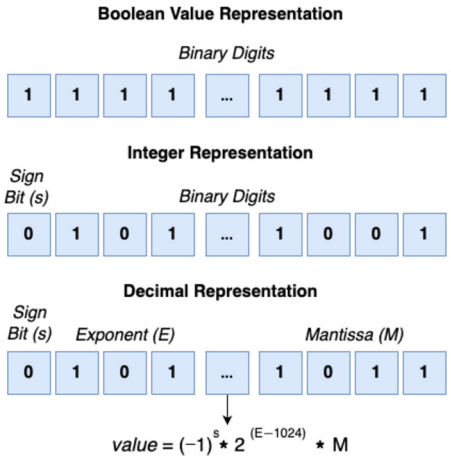

Figure 2: BIE scheme for encoding boolean, integer, and floating point values.

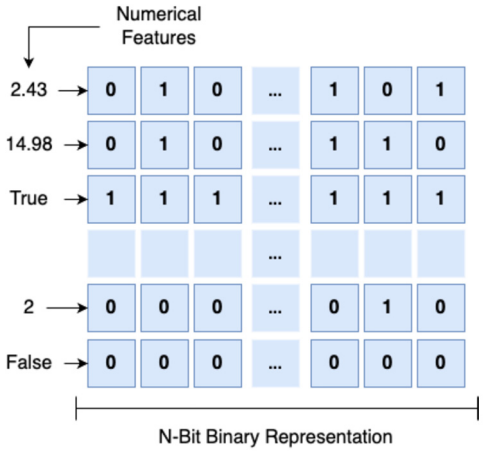

Figure 3: BIE applied to a single sample.

encodes binary images for utilization in CNNs, which makes it a highly effective technique for enhancing CNN-based image processing tasks.

The technique employed by BIE involves the transformation of numerical sample values into their corresponding binary strings (floating-point, integer, or binary). These binary strings are then combined vertically to form a two-dimensional matrix, which can be interpreted as an image representation where dark-colored pixels denote zero (0) values, and white-colored pixels denote one (1) values by default (though the function can be parameterized to represent zero and one values as any pixel value). The process of converting numerical feature values into n-bit binary strings, where n represents the precision used, and stacking them on top of each other to form the complete BIE image is illustrated in Figure 3. It is important to note that the figure depicts a single sample from a dataset, serving as a representative example of the BIE image creation process.

Algorithm 1 Binary Image Encoding (BIE).

```
function BIE(sample, precision)
    sample_out = [ ]
    for f ∈ sample do
        if f is floating point then
            br = f.binary("floating_point", precision)
        else if f is integer then
            br = f.binary("int", precision)
        else if f is bool then
            if f is True then
                br = [1 for _ in range(precision)]
            else
                br = [0 for _ in range(precision)]
            end if
        end if
        sample_out.append(br)
    end for
    Return sample_out
end function
```

Inputs:

- *sample*: An array or list of data elements. Each element in the array can be of type floating point, integer, or boolean.
- *precision*: An integer value that determines the number of bits used to represent each element in binary form.

Outputs: An image *sample_out* representing the data sample after BIE has been applied.

3.2 Algorithm Details

The Binary Image Encoding (BIE) algorithm is given below as Algorithm 1. BIE accepts a dataset, sample-by-sample, and first inspects their data type before converting the data into the appropriate binary representation. We handle the canonical data types, integers, floating points, and booleans in the algorithm. For most data types, the data is simply converted into its binary representation with the given precision. For data types like boolean values that have a static size, we broadcast its representation into a vector of size precision. This is done for every column in the sample, and then the columns are concatenated together. Ultimately, an image is generated utilizing the values present in sample_out and preserved in the specified directory with the designated label.

3.3 Correlated Binary Image Encoding

Prior to converting each sample to its binary representation, the cBIE algorithm sorts the columns by their correlation to a feature. This is achieved by using the order_columns_by_correlation algorithm given as Algorithm 2.

Algorithm 2 Order Columns by Correlation.

function ORDER_COLUMNS_BY_CORRELATION(df, $label$)
    $new\_column\_order = []$
    $columns = df.columns$
    $current = label$
    $corr = df$.correlations() // Create a correlation table
    while length($corr$) < 1 **do**
        c_idx = argmax($corr[current]$)
        $corr$.drop([$current$])
        $current = columns[c\_idx]$
        new_column_order.append($current$)
    end while
    Return new_column_order
end function

Input: A Pandas dataframe df and a target label $label$.
- df: A Pandas dataframe containing the target dataset.
- $label$: The target label of the dataset.

Output: A list new_column_order representing the order of columns sorted by their correlation with the target label.

This algorithm simply creates a new ordering for the columns based on their Pearson correlations. Column c_0 is the column most correlated with the label column. If the dataset has no label, a representative column must be chosen first. Then every c_i is assigned the column most correlated with column c_{i-1}. Each column may only be used once.

Once the algorithm computes the new column order, the rest of the process is the same as in the BIE methodology. Our algorithm is given using pseudocode that assumes familiarity with the Pandas library in Python.

3.4 Binary Encoding Discussion

We propose that utilizing binary-encoded vectors to represent feature values can provide several advantages for Tabular classification and regression tasks. First, this approach does not rely on normalization. This is beneficial because normalization can diminish the range of potential feature values and can be heavily influenced by outliers, resulting in information loss. Second, a binary decimal string separates the magnitude of a value (exponent) from the precise value of each digit (mantissa) in its floating-point representation. Thus, a CNN can better ascertain the difference between quantities of different magnitudes. Furthermore, boolean values are dispersed throughout the row, and integer values can accommodate large numbers without the need for normalization. Last, this technique enhances the information content of each value by dividing it into meaningful components. Unlike IGTD, where each pixel corresponds to a given feature, our method does not limit the amount of information that the image can convey based on the number of available features.

4 Architecture

4.1 Residual Networks

As research on DNNs has progressed, it has generally been observed that increasing the depth of these networks leads to improved classification accuracy. These deeper networks generally use combinations of non-linear activation functions to enable the model to represent complex non-linear behaviors. However, deeper neural networks are more challenging to train due to the problem of vanishing or exploding gradients which arise from the non-linear nature of the activation function (He et al., 2016a). ResNet is a residual network that utilizes rectified linear unit (ReLU) as its non-linear activation function and mitigates the vanishing gradient problem with residual connections.

The activation function ReLU produces a gradient of zero when the input to a neuron is non-positive. Backpropagation, which is used to compute the gradient with respect to the loss function, relies on the chain rule and follows a specific order of operations to optimize efficiency (Sharma et al., 2019). If a neuron has a non-positive weight, the gradient is reduced to zero due to the nature of the chain rule. If this issue occurs among several neurons, the gradients can become too small to be used effectively in backpropagation. A gradient of zero does not provide a direction for the gradient descent algorithm, causing the model to lose its ability to learn.

ResNet is a convolutional neural network designed to mitigate the drawback of vanishing/exploding gradients through residual connections. Residual connections allow the gradients of the network to bypass one or more layers (He et al., 2016a,b; Xie et al., 2017). These residual layers receive input from both the previous weight layer and the residual layer. When the residual layer is activated, it takes the output of the weight layer to which it is directly connected and sums it with the output of the last residual layer, accessible through the skip connection (He et al., 2016a). The sum of these two outputs is then passed through an activation function to produce the output of the residual layer. This prevents vanishing gradients as the information added from previous layers ensures that the gradient does not become too small. As a result, ResNet models can maintain greater depth than several other neural networks and demonstrate significantly higher accuracy than previous non-residual models (He et al., 2016a).

5 Experiments

In the following subsection, we briefly describe the model architecture and benchmark datasets employed in our experiments. The selection of data sets was based on their wide popularity and frequent use in various research works, thus providing a robust benchmark to evaluate our methodology against. Additionally, the datasets were carefully chosen for their diversity and ability to present distinct challenges, thereby enhancing the validity and generalizability of our findings. For each dataset, we present a sample image exemplifying the three T2I methods (two novel methods and a current state of the art technique). It is noteworthy that, for the BIE and cBIE methods, additional rows of zeros may be appended to images, as needed, in order to maintain consistency in the number of rows and columns across all samples. Subsequently, we will explore experimental design and the performance metrics utilized to assess our models. Lastly, we will investigate the training procedures for our models and the methods used to optimize their performance. We make the datasets used and code publicly available, see the supplementary material section.

5.1 Model Architecture

We utilize a version of ResNet in our experiments primarily to take advantage of the efficiency and accuracy that stem from residual networks. ResNet-50 is the 50-layer version of ResNet that we use throughout the experiments. It is composed of 49 convolutional layers and one fully connected layer (He et al., 2016a). Although ResNet-50 is not the most effective version of ResNet, its fewer layers provide shorter training times while still maintaining low error (Koonce, 2021). One key feature of ResNet-50 that helps reduce computational expense is Bottleneck Blocks. The shallower versions of ResNet use two convolutional layers with 3x3 filters. ResNet-50 improves on this by introducing bottleneck blocks that apply three convolution layers in a stack like so: 1x1, 3x3, 1x1 (He et al., 2016a; Koonce, 2021). From a purely mathematical perspective, this approach is less powerful than using 3x3 filters. However, 1x1 layers are much easier to implement and allow for a significant increase in efficiency. The 1x1 layers are responsible for reducing and restoring dimensions, which leaves the 3x3 layer a bottleneck with smaller input and output dimensions (He et al., 2016a). As a result, ResNet-50 is able to run four times as many filters using these bottleneck layers which ultimately makes the model more powerful (Koonce, 2021). Due to this feature and its middling number of layers, we chose ResNet-50 for our experiments since it is computationally well-suited for our architecture and maintains accuracy.

5.2 Iris

The Iris dataset (Fisher, 1988), presented in R.A. Fisher's seminal 1936 paper, is a simple dataset that has become a popular model dataset for testing different ML methods (Fisher, 1936). The dataset consists of 150 samples with 4 features, encompassing the length and width of both the petal and sepal, as well as the label for three species of Iris. The dataset is balanced, containing 50 instances of each of the 3 classes. Notably, one class is linearly separable from the other two; however, the latter two are not linearly separable from each other (Fisher, 1988). Figure 4 illustrates the first Versicolor sample converted to images using BIE, cBIE, and IGTD. The BIE image consists of four rows that correspond to each feature present in the dataset with additional padding added to create a square image. On the other hand, cBIE is a modified version of the BIE image where rows have been rearranged based on feature correlation. Additionally, the IGTD format comprises four squares, each depicting an individual image. Further details on the BIE and cBIE formats can be found in previous sections.

Figure 4: An Iris Versicolor sample generated using BIE, cBIE, and IGTD.

Figure 5: A Cultivar 1 sample generated using BIE, cBIE, and IGTD.

Figure 6: A malignant sample generated using BIE, cBIE, and IGTD.

5.3 Wine Cultivar

The Wine Cultivar dataset consists of 178 samples from 3 unbalanced classes. This dataset has become another common baseline for testing new classification techniques and is available from the UCI machine learning repository (Aeberhard and Forina, 1991). The data consists of the numerical results of a chemical analysis of wines produced in the same region of Italy but derived from three different cultivars.

The analysis determined the quantities of 13 constituents found in each of the three types of wines. Figure 5 denotes a type 1 sample from the wine cultivar dataset that has been transformed using BIE, cBIE, and IGTD. Similar to the depiction in Figure 4, the BIE and cBIE images represent each feature in the dataset as a separate row, while in the IGTD format, every feature is represented as a shaded square.

5.4 Breast Cancer

The Breast Cancer Wisconsin dataset (Wolberg et al., 1995) encompasses measurements and attributes of fine needle aspirates of breast masses derived from digitized images. For data cleaning, we omitted the case id feature as it does not influence the manner in which a breast mass is measured or characterized. All remaining features (totaling 31) were converted into images and trained by the model. Figure 6 displays the same malignant sample as represented by the three T2I methodologies. As observed in the malignant sample, it is evident that as the number of features increases, the corresponding information gain in the BIE and cBIE formats also increases proportionately.

Figure 7: An NTP DDoS sample generated using BIE, cBIE, and IGTD.

5.5 DDoS

The CIC-DDoS2019 dataset encompasses benign and DDoS attack flows from 13 distinct contemporary DDoS attack types (Sharafaldin et al., 2019). We excluded the WebDDoS attack type from our analysis due to its significantly smaller sample size relative to the other attack types. The CIC-DDoS2019 dataset was selected for our research owing to its substantial number of attack types and samples, rendering it well-suited for multiclass classification. Additionally, the dataset was developed by a renowned institution and has been utilized in numerous related works, facilitating improved comparability of our findings. Prior to training the classifiers, the data underwent cleaning, pruning, and downsampling procedures as discussed in the work by Halladay et al. Halladay et al. (2022). The data that remained had all of the samples with infinite, null, or NaN (Not a Number) values removed, columns with data such as IP address and port number were removed to prevent overfitting, and columns that consisted of a single unique value such as 0 were removed.

The same NTP DDoS sample can be visualized when converted using the T2I algorithms in Figure 7. In this sample, we can see that a significant proportion of the features contain zero values. Notably, both BIE and cBIE formats display the same features but in a different order, while the IGTD format provides relatively less information.

5.6 Performance Metrics

We employ a set of essential metrics that serve as indicators of our models' performance and efficiency. These metrics include accuracy, F1-score, AUC, Loss, and T2I encoding time. The approach for evaluating these metrics involves computing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) as prescribed by Hossin and Sulaiman (2015). F1-score, which represents the harmonic mean of precision and recall, is a vital metric that provides insight into how well our model performs by correctly classifying positive classifications and accurately predicting true positives (Hand et al., 2021).

The use of ROC curves enables us to visually depict the TP rate in relation to the FP rate, with AUC calculated as the total area under a ROC curve. Both F1-score and AUC are robust metrics that help mitigate the impact of unbalanced data and assist in identifying potential overfitting to training data (Ling et al., 2003; He and Garcia, 2009; Szegedy et al., 2016). Our preferred metric for comparing model performance is F1-score, given its reliability and widespread use in the research community. Loss is a widely-used metric to assess and enhance trained models. It can be computed using various methods, such as mean squared error and binary cross-entropy. The ResNet-50 model employs categorical cross-entropy loss. The

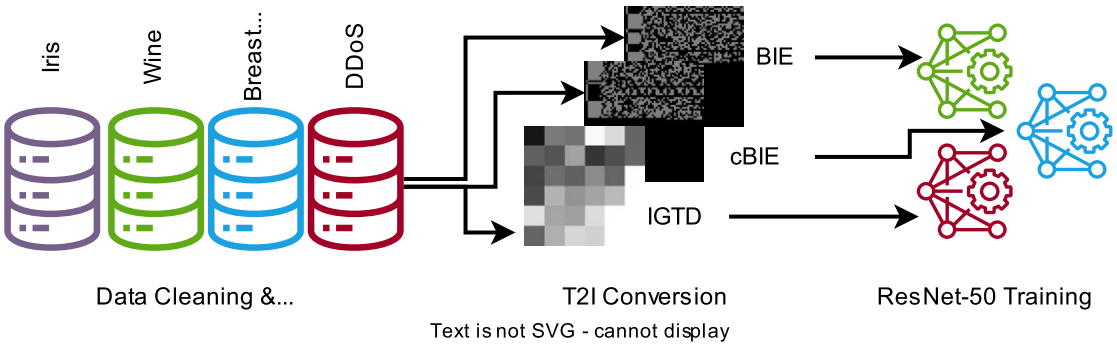

Figure 8: Our Research Methodology. We clean the datasets, transform the samples into images using a T2I technique, and fine-tune ResNet-50 to perform classification. Min-Max Normalization is applied to only the IGTD images.

loss metric indicates the average degree of error in the model's predictions, with a lower score indicating better performance.

Furthermore, we also report the T2I encoding time (in seconds) as an important metric to consider for real-time anonymous traffic detection and continuous model learning. This metric reflects the efficiency of our method and its ability to handle large amounts of data in a timely manner, thus serving as a key performance indicator.

5.7 Experiment Design

In our experimental methodology, as visualized in Figure 8, we adhered to the data cleaning procedures outlined in the preceding sections to ensure data quality. To investigate the efficacy of our methods, we generated three distinct collections of images for each dataset utilizing BIE, cBIE, and IGTD, respectively. Each of these image collections was trained independently on a ResNet-50 model. Our experiments were conducted on an Ubuntu 20.04.3 LTS machine equipped with an Intel i7-7700k CPU, GTX 1080 GPU, and 16 Gigabytes of RAM.

To optimize performance for our specific task, we designed a custom fully connected network to process the output of ResNet-50. This network consisted of two dense layers with respective output sizes of 512 and n units, where n represents the number of classes in the tabular data. The first dense layer incorporated a ReLU activation function to introduce non-linearity (He et al., 2016a), and the second dense layer utilized the softmax activation function for multi-class classification (Szegedy et al., 2016).

We conducted experiments to evaluate the efficacy of various optimizers, including adaptive moment estimation, adaptive gradient, and stochastic gradient descent (SGD). Our experiments found SGD to be the most effective optimizer among the tested methods. To further improve our model's performance, we employed the ReduceLROnPlateau callback to monitor the learning rate and decrease the learning rate if the validation loss hasn't decreased for 5 epochs. This callback automatically reduces the learning rate if the loss fails to improve. Our model was set to train for a maximum of 100 epochs with an early stopping callback to terminate training if the loss did not improve in five subsequent epochs. We used a learning rate of 0.1 for all experiments, as we did not find learning rate to significantly affect results. We use a batch size of 64 since some of our datasets like Iris had only a couple hundred values. We saved the model weights after each epoch and used Tensorboard to generate real-time graphs that allowed us to

Table 1: Performance Metrics for Three T2I Methodologies on our 4 Datasets.

Data	T2I Methods	Accuracy	F1	AUC	Loss	T2I Time
Iris	BIE	1.0	1.0	1.0	0.0004	0.0531
	cBIE	1.0	1.0	1.0	0.0065	0.071
	IGTD	.9667	0.9505	0.9967	0.1058	0.21
Wine Cultivar	BIE	1.0	1.0	1.0	0.0004	0.0531
	cBIE	1.0	1.0	0.999	0.0068	0.0576
	IGTD	0.9931	0.9948	0.9998	0.0231	0.3720
Breast Cancer	BIE	1.0	1.0	1.0	0.0003	0.4519
	cBIE	1.0	1.0	1.0	0.0012	0.508
	IGTD	1.0	1.0	1.0	0.0022	1.34
DDoS	BIE	0.7704	0.7650	0.9876	0.4701	55.36
	cBIE	0.7686	0.7646	0.9872	0.4776	57.42
	IGTD	0.6783	0.6480	0.9774	0.6333	24.8

monitor the model's progress. To ensure the generalizability of our model, we utilized a 5-fold cross-validation technique during training (Cawley and Talbot, 2010). This approach involved dividing the data into five subsets, with each subset used for testing once while the others were used for training.

6 Results

In this section, we present our findings and offer an in-depth discussion of the implications of our results.

6.1 Model Performance

In Table 1, we provide a comprehensive overview of the overall performance of the three T2I algorithms across the four datasets. The Iris dataset yielded impressive results, with both the BIE and cBIE methods achieving a perfect score of 100% in terms of accuracy, F1-score, and AUC, using both BIE and cBIE. The IGTD variation had a slightly lower accuracy rate of 96.67% but achieved a notable F1-score of 95.05%. In terms of loss and T2I time, the standard BIE variation performed the best, with the lowest values of 0.0060 and 0.063 seconds, respectively. As discussed earlier, the high performance of the BIE and cBIE methods can be attributed to the linear separability among two of the three classes in this dataset, while IGTD performed relatively worse due to the limited information it could extract from the 9 squares in the image.

We observed a similar trend in Wine Cultivar dataset where both the BIE and cBIE methods achieve a perfect score of 100% for both accuracy and F1-score, and IGTD exhibiting a slightly lower accuracy rate of 99.31%, but still managing to achieve an impressive F1-score of 99.48% and an AUC of 99.98%. Once again, BIE had the lowest loss and T2I time. The classifiers performed well on this dataset, likely due to the well-behaved class structures and the relatively simple task of distinguishing between only three cultivars.

For the Breast Cancer dataset, all methods achieved a perfect score of 100% for accuracy, F1-score, and AUC. BIE performed the best, with the lowest loss of 0.0003 and the lowest T2I time of 0.4519 seconds. Once again, all T2I techniques were highly effective, most likely due to the well-defined features and the binary classification problem at hand.

The DDoS dataset posed a unique challenge due to its large number of classes (13) and diverse data. Across all variations, the IGTD algorithm achieved a relatively lower accuracy, F1-score, and AUC with the highest values achieved by the BIE at 77.04%, 76.60%, and 98.76% respectively. It is noteworthy that IGTD had the lowest T2I time at 24.8 seconds. As the dataset expands, the overhead for T2I time through the correlation algorithm in IGTD diminishes, resulting in faster execution when compared to BIE and cBIE.

In summary, the BIE and cBIE methods consistently outperformed IGTD in terms of accuracy, F1-score, and AUC across most datasets. However, for certain datasets, such as the Wine Cultivar and Breast Cancer datasets, the IGTD method performed comparably well. Our findings highlight the importance of selecting the appropriate T2I algorithm for a given dataset, as well as the significance of carefully evaluating algorithm performance across a range of metrics.

6.2 Discussion

Our results demonstrate the effectiveness of the proposed T2I conversion method in improving classification accuracy and reducing loss for tabular data. Our approach outperforms the state-of-the-art shallow learning classifier, XGBoost, on the DDoS dataset with the same features, achieving an accuracy of 77.04% compared to XGBoost's 74.08% (Wolberg et al., 1995). Moreover, BIE and cBIE performed better than IGTD in terms of T2I conversion time for small datasets, while IGTD had superior training times on larger datasets due to the overhead involved in finding feature correlation. With further optimizations, BIE and cBIE could be much faster and more efficient. T2I conversion time is a crucial factor to consider, particularly for real-time applications where tabular data needs to be converted before feeding it to the classifier. Lower T2I times are beneficial for organizations with limited computational resources or those dealing with large amounts of data.

As a drawback, the images produced by BIE and cBIE can vary in size depending on the number of features and the precision used. IGTD usually performs best on datasets with a large number of features, owing to a 1-to-1 feature-pixel mapping. BIE, on the other hand, has been demonstrated to be effective with as few as four features. Datasets with numerous features and high precision values could take longer to generate images and train models than other methods. Nonetheless, IGTD also produces varying image sizes depending on the dataset. For example, the training times for DDoS were $3,892$, $3,895$, and $4,848$ seconds for BIE, cBIE, and IGTD, respectively. In contrast, for the small Iris dataset, the training times were 151, 154, and 94 seconds, respectively. The impact of training time should be especially considered in batch learning.

BIE and cBIE have demonstrated superior performance over the established IGTD method in our experiments. Furthermore, BIE is versatile as it can be applied to any classification problems similar to IGTD. Empirical results suggest that the proposed BIE and cBIE are viable and effective new techniques for T2I conversion with superior performance.

7 Limitations and Future Work

Due to computational constraints, we performed limited hyper-parameter tuning for ResNet-50 models and experimented with few different image sizes. Future work could involve conducting more extensive hyper-parameter tuning, exploring a wider range of image sizes, and optimizing the technique's T2I performance metric.

Another avenue for future research is to test the proposed technique on additional datasets. Currently, our evaluation was conducted on only four benchmark datasets. As a future work, testing on diverse datasets with varying data types, structures, and characteristics could provide a more comprehensive assessment of the technique's generalizability and applicability across different domains. One potential issue with using the BIE method on datasets with a high number of features is that each feature is stacked on top of one another. As a result, BIE may generate images with significantly disproportionate dimensions, where one dimension is much larger than the other. To address this challenge, it may be beneficial to stack samples horizontally as needed, to create more square-like images. It is worth noting that simply increasing the precision arbitrarily to achieve more square images may not be an effective solution, as this approach may encounter diminishing returns.

One way to enhance the BIE and cBIE algorithms could be to assign grayscale values to the pixels rather than the default values of 0 or 1. This could involve mapping the values to the statistical distribution of the dataset after normalization. Additionally, multiple channels could be assigned to the image, with each channel utilizing a different approach to assigning values between 0 and 1.

By considering the common structures among binary representations of features with the same data type, we could potentially enhance the feature extraction process and improve the accuracy of the resulting BIE images. Exploring different methods for aggregating features, such as aggregating all double precision values together, followed by booleans and integers, or other forms of reordering, could help determine the optimal feature ordering for generating BIE images and potentially lead to better performance.

Evaluating the proposed T2I techniques on other deep learning architectures–pre-trained and non-pretrained networks–could provide more insight on the robustness of the proposed techniques. Additionally, conducting a comparative analysis with other existing techniques in the field, as listed in the related works, could highlight the advantages and limitations of the proposed techniques, providing a comprehensive understanding of their efficacy and competitiveness in the current machine-learning research landscape.

A potential area of future research involves utilizing feature extraction techniques to convert an image into a tabular dataset. Subsequently, BIE could be employed to generate an encoded image from the tabular dataset. Finally, the encoded image could be concatenated with the original image, allowing BIE to enhance CV tasks. This technique may be particularly useful in scenarios where the original image contains complex or high-dimensional features that are difficult to analyze directly.

Furthermore, exploring the use of natural language data to create tabular datasets and applying the proposed technique to generate images could open up new possibilities for applications beyond traditional tabular data, such as natural language processing tasks, and further extend the applicability of the approach to diverse data types and structures.

Addressing these limitations and exploring the suggested future work directions, would further contribute to the advancement and refinement of the proposed technique and its potential applications in various domains.

8 Conclusion

We introduced two novel T2I conversion algorithms, BIE and cBIE, and comprehensively evaluated their performance against the state-of-the-art IGTD technique. The empirical results demonstrate that BIE and cBIE outperforms IGTD in four benchmark datasets showcasing their efficacy as effective T2I conversion methods. Notably, BIE and cBIE provide several inherent advantages, including minimal data loss due to the absence of data normalization, simple and straightforward algorithms, utilization of a native data representation, and a broad applicability to structured and tabular datasets. Our findings show that BIE is remarkably effective and robust even on a small dataset (Iris) with just 150 samples and 4 features. These findings significantly contribute and push the state-of-the-art in the field of T2I conversion and deep learning applications.

Moreover, the implementations of the algorithms used in this study have been made publicly available for reproducibility and further scrutiny in a variety of contexts and applications (Halladay et al., 2023). The T2I conversion methods were trained using ResNet-50, a widely used deep neural network architecture, which indicates the potential applicability of the proposed algorithms in practical real-world scenarios.

Supplementary Material

We provide the code and datasets separately in the supplementary material. Included in the code is also all of the figures included in the paper in svg, pdf, and png format. The code reflects the contents of the github repository used for these experiments at the time of publication (Halladay et al., 2023).

Funding

This work was supported by the State of Colorado through funds appropriated for cybersecurity by a piece of legislation dubbed "Cyber Coding Cryptology for State Records."

References

Aeberhard S, Forina M (1991). Wine. UCI Machine Learning Repository. https://doi.org/10.24432/C5PC7J

Albanese C, Li D, Lobachevskiy E, Meissner G (2013). A comparative analysis of correlation approaches in finance. *The Journal of Derivatives*, 21(2): 42–66. https://doi.org/10.3905/jod.2013.21.2.042

Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, et al. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches.

Buturović L, Miljković D (2020). A novel method for classification of tabular data using convolutional neural networks. *BioRxiv*, 2020–05. https://doi.org/10.1101/2020.05.02.074203

Cawley GC, Talbot NL (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11: 2079–2107.

Choi S, Fang C, Haddad D, Kim M (2022). Predictive modeling of charge levels for battery electric vehicles using cnn efficientnet and igtd algorithm. arXiv preprint: https://arxiv.org/abs/2206.03612

Das HP, Spanos CJ (2022). Improved dequantization and normalization methods for tabular data pre-processing in smart buildings. In: *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 168–177.

Fisher RA (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2): 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Fisher RA (1988). Iris. UCI Machine Learning Repository. https://doi.org/10.24432/C56C76

Gokhale M, Mohanty SK, Ojha A (2023). Genevit: Gene vision transformer with improved deepinsight for cancer classification. *Computers in Biology and Medicine*, 155: 106643. https://doi.org/10.1016/j.compbiomed.2023.106643

Halladay J, Cullen D, Briner N, Warren J, Fye K, Basnet R, et al. (2022). Detection and characterization of ddos attacks using time-based features. *IEEE Access*, 10: 49794–49807. https://doi.org/10.1109/ACCESS.2022.3173319

Halladay J, Cullen D, Briner N, Watson W, Miller D, Primeau R (2023). Binary image transformation: Github repository.

Hand DJ, Christen P, Kirielle N (2021). F*: An interpretable transformation of the f-measure. *Machine Learning*, 110(3): 451–456. https://doi.org/10.1007/s10994-021-05964-1

He H, Garcia EA (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9): 1263–1284. https://doi.org/10.1109/TKDE.2008.239

He K, Zhang X, Ren S, Sun J (2016a). Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

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

Hossin M, Sulaiman MN (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2): 1. https://doi.org/10.5121/ijdkp.2015.5201

Iqbal MI, Mukta MSH, Hasan AR, Islam S (2022). A dynamic weighted tabular method for convolutional neural networks. *IEEE Access*, 10: 134183–134198. https://doi.org/10.1109/ACCESS.2022.3231102

Koonce B (2021). *ResNet 50*, 63–72. Apress, Berkeley, CA

Krizhevsky A, Sutskever I, Hinton GE (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, (F Pereira, CJ Burges, L Bottou, KQ Weinberger, eds.) 25.

Krupski J, Graniszewski W, Iwanowski M (2021). Data transformation schemes for cnn-based network traffic analysis: A survey. *Electronics*, 10(16): 2042. https://doi.org/10.3390/electronics10162042

Ling CX, Huang J, Zhang H (2003). Auc: A better measure than accuracy in comparing learning algorithms. In: *Advances in Artificial Intelligence: 16th Conference of the Canadian Society for Computational Studies of Intelligence* (Y Xiang, (B Chaib-Draa, eds.), volume 16 of *Proceedings*, AI 2003, Halifax, Canada, June 11–13, 2003, 329–341. Springer.

Moskalenko A, Moskalenko V, Shaiekhov A, Zaretskyi M (2020). Multi-layer model and training method for information-extreme malware traffic detector. In: *CMIS*, 288–299.

Noroozi M, Favaro P (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In: *European Conference on Computer Vision*, 69–84. Springer.

Rabbah J, Ridouani M, Hassouni L (2022). A new churn prediction model based on deep insight features transformation for convolution neural network architecture and stacknet. *Interna-*

tional Journal of Web-Based Learning and Teaching Technologies (IJWLTT), 17(1): 1–18. https://doi.org/10.4018/ijwltt.300342

Seger C (2018). An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing.

Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*, 1–8. IEEE.

Sharma A, Vans E, Shigemizu D, Boroevich KA, Tsunoda T (2019). Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, 9(1): 11399. https://doi.org/10.1038/s41598-019-47765-6

Simon M, Rodner E, Denzler J (2016). Imagenet pre-trained models with batch normalization. arXiv preprint: https://arxiv.org/abs/1612.01452

Simonyan K, Zisserman A (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint: https://arxiv.org/abs/1409.1556

Sun B, Yang L, Zhang W, Lin M, Dong P, Young C, et al. (2019). 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*. 2973–2981.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016). Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Taheri A, Ebrahimnezhad H, Sedaaghi MH (2022). Prediction of the critical temperature of superconducting materials using image regression and ensemble deep learning. *Materials Today Communications*, 33: 104743. https://doi.org/10.1016/j.mtcomm.2022.104743

Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017). Malware traffic classification using convolutional neural network for representation learning. In: *2017 International Conference on Information Networking (ICOIN)*, 712–717. IEEE.

Wolberg W, Mangasarian O, Street N, Street W (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B

Xie S, Girshick R, Dollár P, Tu Z, He K (2017). Aggregated residual transformations for deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1492–1500.

Zhu Y, Brettin T, Xia F, Partin A, Shukla M, Yoo H, et al. (2021). Converting tabular data into images for deep learning with convolutional neural networks. *Scientific Reports*, 11(1): 11325. https://doi.org/10.1038/s41598-021-90923-y