IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 November 2022, accepted 15 December 2022, date of publication 20 December 2022,
date of current version 29 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3231102

RESEARCH ARTICLE

A Dynamic Weighted Tabular Method for Convolutional Neural Networks

MD. IFRAHAM IQBAL[1], MD. SADDAM HOSSAIN MUKTA[2], AHMED RAFI HASAN[2], AND SALEKUL ISLAM[2], (Senior Member, IEEE)
[1]Department of Data Science, Friedrich Alexander University of Erlangen, 91054 Erlangen, Germany
[2]Department of CSE, United International University (UIU), Dhaka 1212, Bangladesh

Corresponding author: Salekul Islam (salekul@cse.uiu.ac.bd)

This research is funded by the Institute for Advanced Research Publication Grant of United International University, Bangladesh, Ref. No.: IAR-2022-Pub-042.

ABSTRACT Traditional Machine Learning (ML) models are generally preferred for classification tasks on tabular datasets, which often produce unsatisfactory results in complex tabular datasets. Recent works, using Convolutional Neural Networks (CNN) with embedding techniques, outperform the traditional classifiers on tabular dataset. However, these embedding techniques fail to use an automated approach after analyzing the importance of the features in the dataset accurately. This study introduces a novel feature embedding technique named Dynamic Weighted Tabular Method (DWTM), which dynamically uses feature weights based on their strength of the correlations to the class labels during applying any CNN architectures on the tabular datasets. DWTM converts each data point into images and then feeds to a CNN architecture. It dynamically embeds the features of the tabular dataset based on their strength and assigns pixel positions to the appropriate features in the image canvas space instead of using any static configuration. In this paper, DWTM embedding method is applied over six benchmark tabular datasets independently by using three different CNN architectures (i.e., ResNet-18, DenseNet and InceptionV1) and an outstanding performance (an average accuracy of 98%) has obtained, which outperforms any traditional and CNN based classifiers as well.

INDEX TERMS Tabular convolution, feature associativity, convolutional neural networks, tabular data to image, image embedding, image classification.

I. INTRODUCTION

In the era of big data, data analysis has become one of the fundamental tools for extracting information over the last few decades. Due to the abundance of data availability, data analysis and machine learning have become critical components for data-driven decision-making [1]. Data is stored as both structured (i.e., excel sheets, databases) and unstructured (i.e., text, email, social media, images, videos) data. Structured data is stored as tabular data, with each column containing a distinct feature and each row containing a distinct instance. Usually, traditional classifiers such as Support Vector Machine (SVM) [2], Logistic Regression (LogReg) [3] and tree-based algorithms [4] are preferred for tabular data analysis. These models provide satisfactory results with limited amounts of data and they generally outperform deep learning (DL) models on tabular datasets [5].

However, the traditional classifiers tend to give a poor performance on large datasets [6], while, DL models [7] comparatively perform better on large datasets due to their ability to learn complex patterns amongst the data [7]. In most cases, Convolutional Neural Networks (CNN) [8] can show outstanding performance in classifying images [9]. CNNs have become highly effective in analyzing unstructured data [10]. CNNs stack multiple convolutional and pooling layers on top of each other. The use of convolutional kernels and the stacking of multiple layers lead to the hierarchical feature abstraction of the input. Initially, the layers find low-level features like edges and corners within an image input. This information is then passed to the deeper layers, which are then

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

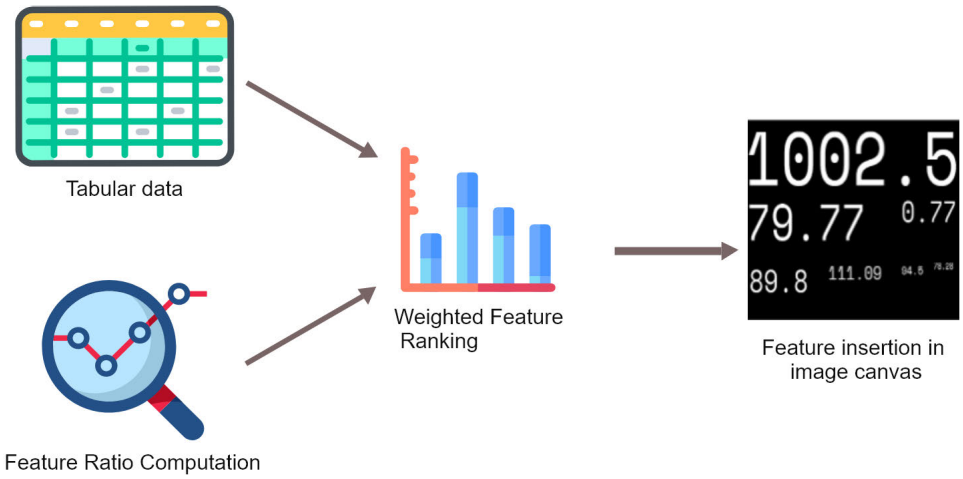


FIGURE 1. High-level overview of the DWTM methodology.

able to detect high-level features by combining the low-level features. All these attributes make CNN highly effective in analyzing data that have spatial relationships amongst their features. However, in its initial form, tabular data do not contain spatial relationships amongst their features, making CNN unsuitable for tabular data. Due to this, the full strength of CNNs' learning capacities could not be utilized yet on tabular data. Furthermore, CNNs perform much better with big data than traditional classifiers. The rise in the availability of data further emphasises the use of CNNs on large tabular datasets.

Tabular data is the most popular form of available data [11]. Unfortunately, CNNs could not be directly applied on tabular data for the reasons mentioned above. This motivated researchers to convert the tabular data into images using embedding techniques. While converting data into images, each feature is assigned a pixel position within the image and then an image dataset is created from the tabular dataset. This enables CNNs to use the relationships within the features and learn accordingly. To the best of our knowledge, three recent studies have introduced embedding techniques that make CNN suitable for tabular data analysis [11], [12], [13]. In [12] the authors introduce DeepInsight, a technique where non-image datasets are converted into image datasets and are forwarded to DL models. A similar approach is introduced in the Image Generator for Tabular Data (IGTD) [13]. Features are assigned to pixel positions and then, each instance is converted into an image. Similar features are placed next to each other in the image. Values of different features are assigned to pixel intensities. However, IGTD is specifically designed for gene expression prediction and not as a generic solution for tabular datasets. This issue is tackled in [11] where the authors introduce two techniques (i.e., Equal Font-SuperTML (EFTML) and Variable Font-SuperTML (VFTML)). VFTML gives greater image space to the more relevant features, while EFTML provides equal space for each feature. Although, in theory, VFTML is likely to produce better results than EFTML, the method cannot perform better in practice. Further analysis of their method shows that the TML possesses a few shortcomings. The method is not the most space-efficient when converting data points to images. Additionally, no parameters are used to assign the canvas space for each feature in VFTML. These shortcomings are also applicable for the IGTD and DeepInsight techniques.

Feature analysis is critical for tabular data analysis. None of the studies [11], [12], [13] use statistical tools for feature analysis in their experiments. IGTD groups the features adjacently based on similarity. However, they do not consider the associativity of the features with the class. SuperTML applies variable font-size to the features but does not consider the statistical relations amongst them. Furthermore, none of the previous studies focus on the applications of the exploratory data analysis and their impact on the performance of CNNs. The methods mentioned above also require manual assigning of pixel positions for each feature, which is a tedious, time-consuming and error-prone process. However, the results from [11] indicate that CNNs are much more effective than traditional classifiers for tabular tasks. CNNs can largely produce results which are better result than the traditional classifiers. The manual dataset creation and the feature associativity error are the current issues with SuperTML. The details of these shortcomings are explained in section II. This study aims to develop a usable but robust method for tabular dataset creation.

In this paper, the Dynamic Weighted Tabular Method (DWTM) is proposed for applying Convolutional Neural Networks (CNN) on tabular datasets. A high-level, pictorial description of the DWTM methodology is shown in Figure 1. DWTM is the first of its kind that uses feature weights to create images for applying CNNs. After computation of feature weights each instance in the dataset is converted into an image with spaces assigned to each feature based on the weights. The features are inserted into the image canvas

accordingly. The primary emphasis of this study is to create a tabular dataset using an automated procedure that enables using CNN models for tabular data analysis while prioritizing the essential features. Additionally, the designed system should be robust to deal with datasets of all types and sizes. The proposed method uses statistical techniques (i.e., Pearson Correlation and Chi-Square) to compute the weights of each feature. The features are then arranged in descending order based on the calculated weights. Each feature is assigned a portion of space in the image canvas based on the ratio of their corresponding weights. The features are inserted accordingly based on their weights. The algorithm that DWTM has developed is based on the best-fit approach to ensure the maximum utilization of the image canvas space. All data points are converted into images and are fed into CNNs (i.e., Resnet-18 [14], DenseNet [15] and Inception [16]) for analysis. To the best of our knowledge, no previous studies have proposed this approach. In this study, DWTM is applied on six benchmark datasets and results have been compared with both previous studies and traditional classifiers. DWTM demonstrates better results for all datasets. In short, the study has the following contributions:

- We build a novel automated embedding technique, DTWM, for applying CNN models on tabular datasets.
- We develop a tool that classifies tabular dataset by using CNN models based on dynamically selected feature importance for both categorical and continuous feature set.
- We apply DWTM over six benchmark tabular classification datasets which demonstrates significant improvement over popular methods (i.e., SuperTML, IGTD) and the traditional classifiers.

Rest of the paper is organized as follows: section 2 briefly describes the previous works on similar topics, section 3 displays the materials and methodology used for this technique, section 4 shows the experiments and results obtained using this method, section 5 contains the discussion section, and finally, the study is concluded in section 6.

II. LITERATURE REVIEW

In recent times, DL has become the fundamental tool for machine learning applications [17], [18]. DL is applied in a wide domain such as computer vision (CV) [19], natural languages processing (NLP) [20], [21] and speech recognition (SR) [22]. In this section, the main ideas from previous studies related to Convolutional Neural Networks (CNN) methods for tabular data tasks are discussed. CNNs are popular due to their unparalleled success with classifying images. Architectures like the AlexNet [23], VGG [24] and deep Residual Networks [14] achieve state-of-the-art performance on the ImageNet[1] dataset. Additionally, due to CNNs success in extracting features from given vectors [25], CNNs are now the ideal choice for Natural Language Processing (NLP) and Image classification tasks.

[1]http://www.image-net.org/

Despite the success of CNN's in other fields, much previous research attempts to use CNN for tabular data analysis. However, in most cases, traditional ML models work far better than CNN models [5]. Hence, CNNs for tabular data remained unexplored for an extended period. However, significant development is being made in this sector in recent times. Xu et al. [26] introduce the novel Conditional GAN (CTGAN) that uses mode-specific normalization and a conditional generator. Mode-specific normalization deals with multimodal and Non-Gaussian distributions, while conditional generator deals with imbalanced columns. The authors find that their model can learn better distributions than the Bayesian network-based models in their results. Butorovic et al. [27] propose a novel method then for using CNN on tabular data analysis known as Tabular Convolution (TAC). Feature vectors are transformed into kernels using the Kernel method and convolved using the base image. The authors use Resnet with TAC for classifying gene expression and found that the results using TAC are similar to the best results that ML classifiers produce.

In another study, the DeepInsight method is proposed [12]. The method converts non-image data to images and uses them as input for CNNs. In this method, CNNs could simultaneously work on different types of data, including tabular data. However, this method does not work well if the dataset is small, as it will create a limited number of images for input. In [13] the authors proposed the image generator for tabular data (IGTD). The method uses an embedding technique to convert tabular data into images by assigning feature positions to pixels based on the similarity of the features. CNNs were applied to the converted image dataset and they outperformed the DeepInsight [12] and traditional classifiers for predicting cancer cell lines and molecular descriptors of drugs.

Sun et al. [28] proposed the Super Characters Method which is used for sentiment classification, whereby texts are converted into images using two-dimensional embeddings. This idea removes the issue of adding another separate step for word embedding as the images are used as input to CNNs. Further investigation shows that the results obtained using this method consistently outperform the other methods for sentiment analysis. As an update based on this work, the authors introduce the SuperTML method [11]. The same idea of two-dimensional embedding is used. However, this time the SuperTML method is applicable for tabular data. For each instance, a separate image is created. A different textbox is allocated in the image for each feature, where the values in each row are inserted. In this paper, the authors propose two variations of the SuperTML method, the EF and VF variations. The features are given equal importance in the EF variation, while the most important feature has the largest feature size in the VF variation. Results show that SuperTML produces state-of-the-art performance on benchmark tabular datasets.

Despite the recent success of DL methods in tabular data, none emphasizes feature importance correctly, which is a

critical element in tabular data analysis. Previous studies predominantly use static methods, which may not work well for all tabular datasets. The method proposed in this study dynamically allocates image canvas space to the features based on their strength of association with the class label. We present a comparison result in Table 1 which shows different tabular based CNN models for classification and regression problems and their corresponding performance.

To the best of our knowledge DeepInsight [12] was the first method to apply CNNs on tabular data by converting them to images. The DeepInsight method successfully classified RNA sequences with a 99% accuracy. However, the robustness of this method was not tested in the study. The method was successful on a large dataset but it has not been applied on smaller datasets to date. Furthermore, the success of DeepInsight is only seen on one type of tabular data. Another limitation of the DeepInsight method is that it produces large images; hence, the computation cost and learning time for the CNN architectures are quite high [13]. IGTD [13] tackles the issue of large images and creates much more compact images from the same tabular data [13]. The method is much more flexible and uses the added benefit of clustering similar features together to boost the learning capabilities of CNNs. A similar problem to DeepInsight remains as the IGTD is yet to be tested on multiple types of tabular datasets. Furthermore, the achieved performance is not the best.

[11] proves to be much more effective compared to the IGTD and DeepInsight methods. Firstly, the method is applied to four different types of tabular datasets. Large, small and multi-class datasets were tested. The SuperTML provided over 90% accuracy on the smaller datasets (i.e., Iris and Wine) but failed to match this performance on the larger dataset (Adult). The VF variation uses the largest feature size for the most important feature and reduces the font size for the lesser important features. The EF variation does not take feature size into consideration. The VF variation should work better in this regard but in the paper, it was found that the EF variation was performing much better. Further analysis of the method showed that the VF-SuperTML has some technical flaws with its feature importance method leading to poorer performances. Figure 2 is a VF-SuperTML generated image of the Adult dataset. The reduced feature sizes mean that some of the features are disturbed chaotically within the image. They also may be too small when compared to their importance or relevance scores. The issue arises from the fact that the VF-TML does not use any mathematical computation of the feature importance to determine their text size accordingly. Instead, variable font sizes are used based on the priority list of the features. Furthermore, it also affects the image in a negative manner as more blank spaces are left within the image which the CNN has no use for learning from the dataset. The EF-SuperTML may work better in this case as when using equal fonts the reduced font sizes of the less priority features will be increased and hence it leads to much better use of the image canvas space. In contrast, a sample image of the DWTM method of this paper is shown in Figure 4 to show how the method uses the image canvas space more efficiently to create much better images for tabular datasets. This directly affects the results achieved from the CNNs as shown in Table 8. The DWTM outperforms the SuperTML on all three similar datasets.

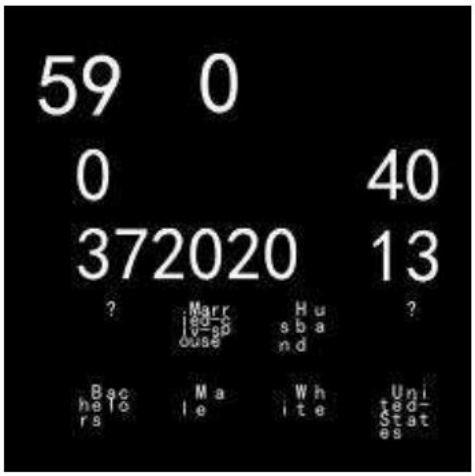


FIGURE 2. VFTML Image example on the adult dataset.

III. METHODOLOGY

This section gives the methodology for the Dynamic Weighted Tabular Method (DWTM) in detail. Furthermore, details of the datasets used to evaluate DWTM are also provided. Tabular datasets contain multiple features; some are more associated with the class than others. In the experiments for this study, statistical techniques (i.e., *pearson correlation, chi-square test*) are used to compute the weights of each feature. The features are then arranged in descending order based on the calculated weights. Each feature is assigned space in the image canvas based on its corresponding weights. The method requires 4 inputs: the *length* and *height* of the image, the *r-score* and the *maximum number of characters* required for each feature. In our embedding technique, we embed the features into the canvas based on their strength (r_{score}). Many of our features have a long floating-point sequence because the value of r_{score} ranges from 0-1. For the sake of space, we keep a portion of the sequence after the floating-point as characters which are embedded into the canvas and the rest is trimmed. The maximum number of characters refers to the maximum number of such character sequence that is required for each feature in the tabular dataset. It is an important parameter to consider when converting to images as the number of characters determines the space allocation of the features by updating their length and height accordingly. DWTM calculates the length, height and area required for each feature by using the ratio of the weights of each feature to the sum of the total weights of all features and distributes the image canvas space accordingly.

The overview of the methodology is shown earlier in Figure 1. Algorithm 1 represents the algorithm used for

TABLE 1. Comparison of related works.

Method Name	Classification	Robustness	Result (Accuracy/R^2)
DeepInsight	Yes	Only gene expression data	99% Accuracy
IGTD	No	Only vitro drug screening data	0.798 R^2 (Average)
SuperTML	Yes	Applicable on most tabular datasets	92.75% Accucary (Average)

Algorithm 1 The DWTM Method

Input:
  l as length of the Image
  h as height of the Image
  X_t as $\sum$ of $pearson$,$r-score$ of all features.
  F is a vector <*contains Feature*> containing *r-score*, max no of char in F, area, length and height of each feature
Output:
  s contains Starting Point for each feature
  f contains Font Size for each feature

Initialize:
  Feature Attributes:
  F_R as $r-score$ for corresponding feature
  $F.X_r$ as the ratio of $r-Score$ to X_t for corresponding feature
  $F.X_A$ as percentage of area corresponding feature requires
  $F.X_L$, $F.X_H$ as measurement of the length and height of the area feature takes
  $F.C$ as max number of character required for each feature
  Box Attributes:
  B as Feature Box size for each feature containing the length and height of each feature
  $B.X_L$ as length of corresponding feature box
  $B.X_H$ as height of corresponding feature box

Procedure:
for $f \leftarrow F$ **do** Feature-Box-Computation ▷ Using Algorithm 2
end for
SB = sortLargest(B) ▷ Feature Box sorted in descending order

while (SB != $NULL$)

  Feature-Insertion-in-Image-Canvas ▷ Use Algorithm 3
end while

Algorithm 2 Feature Box Computation

Input:
  l as length of the Image
  h as height of the Image
  X_t as $\sum$ of $pearson$,$r-score$ of all features.
  F is a vector <*contains Feature*>
  F as Features containing $r-score$, max no of char in F, area, length and height of each feature

Output:
  SB is a vector <*contains Feature Boxes sorted by their associativity to the class*>
  SB as Feature Box size for each feature containing the length, height, area and position of each feature

Procedure:
$X_t = \sum$ of $F.r-score$ ▷ Add Summation Range
for $x \leftarrow F$ **do**
  $F.X_r = X.RS/X_t$ ▷ Ratio of feature's r-score to $\sum$ of r-score
  $F.X_A = X_r * m * n$ ▷ Computing Feature Box Size in Image
  $F.X_H = \sqrt{(F*A)/N}$ ▷ Computing Height of Feature Box
  $F.X_L = N * F.X_H$ ▷ Computing Length of Feature Box
end for
Insert $F.X_L$, $F.X_H$, $F.X_A$ in B
$F.X_A = [\{X_L\} * \{X_H\}]$
SB = sortLargest(B) ▷ Feature Box sorted in descending order

this method. The repository provided with this paper is autonomous and robust and requires only a single input of the tabular dataset. When a new dataset is loaded, the method identifies the present feature set and computes the feature importance from the latest dataset for generating the image dataset.

A. STRUCTURED DATA TO IMAGE ALGORITHM

B. WEIGHT COMPUTATION OF THE FEATURES

The Pearson Correlation coefficient is used to calculate the weights of each feature by determining the associativity between each feature and the class. The Pearson Correlation technique is widely used in research for finding the associativity between attributes or variables [29]. The Pearson *r-score* ranges from -1 to $+1$ and corresponds to the strength of associativity of the features with the class label. Negative value refers to negative associativity. The *r-scores* of the selected features are calculated. Afterward, the weights for each feature are computed using their calculated *r-scores*. However, Pearson Correlation is not applicable for associativity calculation on categorical data [30]. Hence, the Chi-Square test is used when there are categorical features in

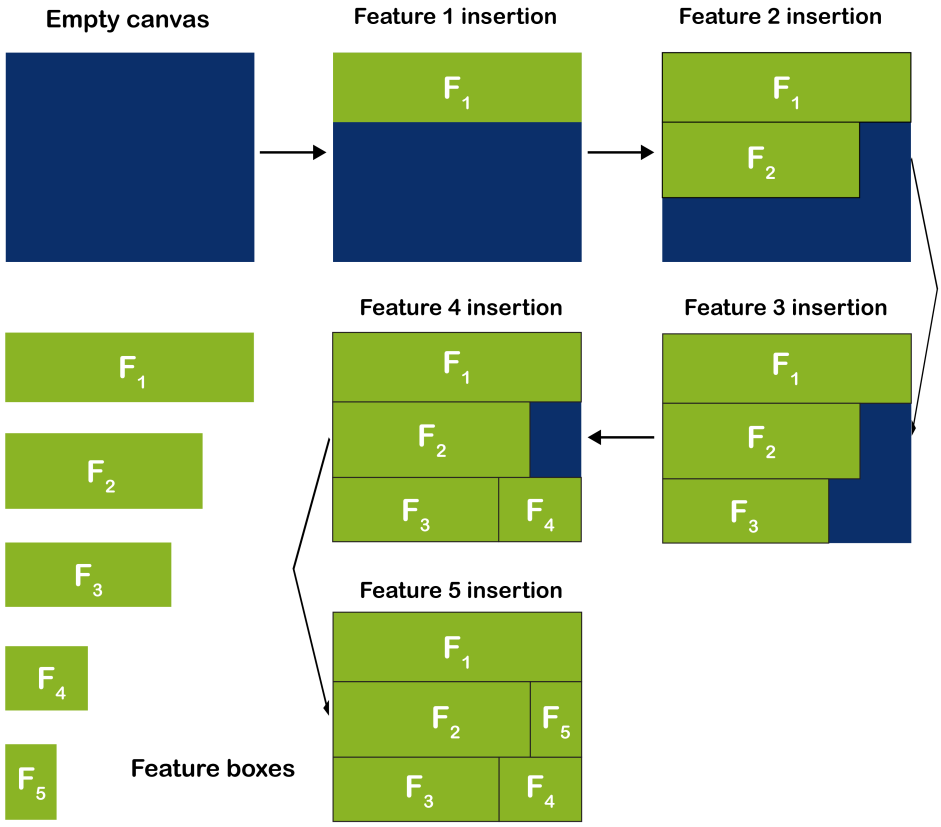


FIGURE 3. A sample simulation of feature insertion into canvas space using DWTM.

the dataset. Afterward, Cramer's V is calculated to find the strength of associativity between the two variables [31].

$$F_{ratio} = \frac{r_i}{\sum r_{1-n}} \tag{1}$$

Equation 1 is used to determine the weights for the features. Here, F_{ratio} represents the weight, r_i represents the r_{score} of the *i-th* feature and the denominator is the summation of all the r_{score}. In the case of categorical variables, the *r-score* is replaced by Cramer's V.

C. CANVAS SIZE ALLOCATION

In this subsection, the area required for each feature in the image canvas is calculated using the F_{ratio}. The formula for calculating the total area of each feature required in the image canvas is calculated using Equation 2. In this study, it is assumed that the length and height of the total image are l and h, respectively. Afterward, the length and height required for each feature in the image are quantified. The height for each feature is the square root of the ratio of the total area and characters required for the corresponding feature and is calculated by using Equation 4. Next, the length of each feature is calculated using the product of the height and character number for that feature. For this study, the monospace [32] font type is used as this font consists of the same length and height for each character. Algorithm 2 is used for this subsection. For example, if there exist three features, with weights of 0.5, 0.3 and 0.2, and characters required 2, 3 and 4, respectively. Assuming the image is of size 128 by 128, the area of feature-1 is calculated by using Equation 2. The calculated area for feature 1 is 8192 units. Then the height using Equation 4 (64 units) and then the length using Equation 5 (128 units) are calculated. The same process is repeated for features 2 and 3. After the process is complete, all the dimensions (i.e., length, height, area) for each feature are acquired.

$$F_{Area} = r_i * l * h \tag{2}$$

$$F_{Area} = F_{Height} * F_{Length} \tag{3}$$

$$F_{Height} = \sqrt{\frac{F_{Area}}{F_{char}}} \tag{4}$$

$$F_{Length} = F_{Height} F_{char} \tag{5}$$

D. OPTIMIZED CANVAS AREA DIVISION

After computing all the dimensions for each feature, it is now possible to insert the feature into each image. However, the points at which these features are to be inserted into the image canvas for the optimal solution are still unknown. Hence, the best fit solution to the problem is designed. Each feature is assumed to be a box of length F_l and height F_h. The image is considered as an empty canvas of size m and n. At the start of the procedure, the features are arranged in descending order based on their area. Then the algorithm 3 is used to insert the

Algorithm 3 Feature Insertion in Image Canvas

Input:
l as length of the Image
h as height of the Image
SB is a vector <*contains Sorted Feature Boxes*>
B as Feature Box size for each feature containing the no. of characters, length, height, area and position of each feature

Output:
s contains Starting Point of each feature
f contains Font Size of each feature

Procedure:

I = [m]*[n] ▷ Image Space
for *k* ← *SB* **do** ▷ Check Canvas availability for SB[k]
  if *flag* == 1 **then**

    Canvas not available
    Continue

  end if

  if *flag* == 0 **then**
    Canvas is available
    Insert feature into image canvas
    pop SB[k] from SB ▷ Feature Inserted
  end if

  if SB != empty **then**
      Feature-Trim ▷ Using Algorithm 4

  end if
end for

features into the available space in the image canvas. In this procedure, each pixel (assuming each pixel as a value in an array) in the image is iterated until the maximum possible space for each feature is found. The feature is inserted into that space. For this algorithm, the flag is initially 0 for all positions in the image and they are only converted to an integer value based on the number of feature(s) being inserted there.

A sample simulation for this procedure is shown in Figure 3. Features are inserted one by one into the image using a descending order of their feature weights. When the required space is found, it is checked if the corresponding rows and columns equalling the length and height of the feature box are empty or not. If there is not enough space the features are systematically trimmed until they can be inserted. If the image space is empty, the feature is added to the canvas space and the space is marked as filled for the corresponding feature. The starting point is also stored. The process is repeated until all the features are given space in the image canvas. In the end, the dimensions and the starting points for all the features are

Algorithm 4 Feature Trim

Input:
UB is a vector <*contains uninserted Feature Boxes*>
UB as Feature Box size for each feature containing the no. of characters (n), length (l), height (h), and area (a) of each feature

Output:
TB is a vector <*contains Trimmed Feature Boxes*>
TB as Feature Box size for each feature containing the no. of characters (n), length (l), height (h), and area (a) of each feature

Procedure:

for *b* ← *UB* **do**
      b.h = b.h - 1
  b.l = bl. - n
  b.a = b.l * b.h
end for

acquired. The height of each feature is equivalent to the font size in the image.

In some cases, some of the features fail to be inserted by the algorithm. This situation occurs due to the unavailability of abundant space in one location to insert the feature box. Hence, the feature box cannot be inserted despite enough overall blank space availability in the image. The features which face this difficulty fall in the lower half of the priority list. To overcome this issue, a trim feature is used. In the trim feature font size of each uninserted feature is decreased by one until all the features are inserted into the image. To do this, algorithm 4 is used which reduces the font size of the remaining features by 1. Again the Weighted Feature Insertion procedure is called. This process continues until each feature is inserted or has a font size of 0. The result is that the least important features heuristically have lesser space or are removed altogether.

E. IMAGE CREATION

The algorithm returns each feature's starting point, area, length, and height. Then using OpenCV [33], each feature is inserted into the image one after another into the image canvas using the information determined earlier. An image is created by using all the features of each data point. These values are rounded to the maximum character sizes allowed for those features and then inserted using a *monospace* font, which is available in *OpenCV*. Figure 4 presents a sample image produced by this method on a classified dataset.

F. THE CONVOLUTIONAL NEURAL NETWORKS USED IN THE EXPERIMENTS

The image dataset is created based on the above methodology. The dataset is then split into training and testing sets and forwarded to the DL models for training. In this case the popular Residual Network-18 [14], Densely Connected

FIGURE 4. Sample image of a datapoint using DWTM.

Convolution Networks [15] and Inception [16] model are used. Residual Network (ResNet) is a model which uses Residual Learning to address the degradation problem in deep neural networks. ResNet introduces shortcut connections that enable the training of deeper networks. This idea helps the DL models perform better on classification tasks due to adding more layers. The study in [14] introduces networks with up to 1000 layers in depth. The ResNet-18 model is preferred for this study.

The study in [15] builds on the idea of the shortcut connections between layers and proposes a Densely Connected Convolutional Network (DenseNet). All the layers with matching feature maps are connected in this network, thus significantly boosting the information flow between the layers. Results from the study show that DenseNet performs better than other state-of-the-art architectures in terms of computational efficiency. In [16] the authors proposed the GoogLeNet, also known as Inception Network. This network applies multiple kernels on the same level, thus enabling the network to tackle overfitting and reducing Deep Neural Networks' computational expense. As a result, it is feasible to use Inception in big data tasks [34]. An updated version of the network, Inception-v3 [35], provides state-of-the-art results for computer vision tasks while being much cheaper computationally. In this study, ReLU activation is used. ReLU has a linear scale for positive values and a value of 0 for negative instances. LR values between 0.0001 and 0.00005 and 30 epochs are used to train the models. Additionally, the Stochastic Gradient Descent (SGD) with exponential learning rate decay, Adam and Adamax are used for optimizing the CNN architectures.

G. BENCHMARK DATASET SELECTION

Several standard image datasets (MNIST, CIFAR and ImageNet) are considered for evaluating state-of-the-art CNN networks on image data. However, no such specific dataset for evaluating tabular methods exist [36]. Hence, to evaluate DWTM, a standard set of datasets need to be considered for tabular data analysis. DWTM is applied over the selected benchmark datasets. Finally, the results of DWTM are compared with the results of the traditional classifiers and other DL-based techniques.

The number of features, number of instances, number of classes and the type of input need to be considered when selecting benchmark tabular datasets. Previous studies demonstrated that traditional classifiers are not well suited to dealing with categorical data [36]. Furthermore, multiclass problems tend to raise the difficulties of classification, especially when the data is limited. All these issues are considered when selecting the datasets in this study to evaluate DWTM. Future researchers are encouraged to use these datasets for evaluating classification methods on tabular data, thus making these datasets the benchmark for evaluation. The details of the selected datasets are summarized in Table 2.

The Iris dataset is the most commonly used dataset in ML literature and hence it has been used for evaluation.[2] The dataset consists of data from 150 iris plants. 4 features are recorded for each instance. Three classes of iris plants, each having 50 instances, are available in this dataset. The Wine dataset[3] requires the method to effectively classify wine samples using the available chemical data of wine grown in a region in Italy. Both the Iris and Wine datasets test a method's effectiveness in dealing with multiclass problems while using a limited amount of data. The Iris dataset also tests the method's effectiveness in classification using only four features. In the Iris dataset, the class label is a categorical variable.

Medical data analysis is essential using Machine Learning at present and so the emphasis is given to using medical datasets. It is critical in disease diagnosis for models not to be biassed towards one class [37]. As a result, the scopes of a proposed method for disease diagnosis are also evaluated. The medical datasets selected for this study are the Cleveland dataset,[4] the Early Stage Diabetes Risk Prediction [38] and the Breast Cancer Wisconsin [39] Dataset. The Cleveland dataset consists of data from 303 patients correlated to the patient's risk of heart disease. In this study, the 14 features recommended on the UCI website by previous studies are used for the experiments. The class column, *condition* contains values from zero to five, which indicates the patient's risk of heart disease. The values are converted from 1-4 to one to identify the presence of heart disease in a patient. The Early Stage Diabetes Risk Prediction [38] dataset was conducted in the Sylhet Diabetes Hospital in Bangladesh. To create the dataset, doctors used a questionnaire on 520 patients. Eventually, 17 features are recorded using the tests and questionnaires. Afterward, patients are tested to check for diabetes, and the results of this test represent the class for each patient. The Breast Cancer Wisconsin [39] Dataset contains

[2]https://archive.ics.uci.edu/ml/datasets/iris
[3]https://archive.ics.uci.edu/ml/datasets/wine
[4]https://archive.ics.uci.edu/ml/datasets/heart+disease

TABLE 2. Details of the selected datasets.

Dataset	# of Inst.	# of features	# of Classes	% of train. inst.
Cleveland	303	14	2	85%
Diabetes	520	17	2	80%
Iris	150	4	3	85%
Wine	178	13	3	85%
Breast Can.	699	10	2	80%
Adult	48842	14	2	67%

data of 699 patients. The data is collected using a Fine Needle Aspirate of a breast mass. There are ten features in the dataset. Class values two and four indicate that breast cancer is malignant or benign. The Breast Cancer dataset contains all numerical data. The Diabetes and Cleveland datasets contain both numerical and categorical variables thus testing the durability of the method on all types of medical datasets.

To test the robustness of the proposed method, datasets that contain less than 1000 instances are selected alongside large datasets. The Adult dataset[5] is one of the most popular dataset used in a DL survey for tabular data [36]. The datasets contain information of 48842 individuals and the class represents their income. The class column is binary, with zero representing individuals who earn less than $50K per year. The dataset contains numerical and categorical variables. The Adult dataset is the largest dataset used for evaluation in this study.

IV. RESULT AND ANALYSIS

Numerous ablation studies are conducted to fine-tune the DWTM methodology. The Cleveland dataset is used to study for ablation studies as it is an imbalanced dataset and is quite difficult to classify. Afterward, the models with the best parameters are selected and tested on the benchmark datasets to evaluate the performance of DWTM.

A. ABLATION STUDY

To determine the effect of the parameters on the CNN models, we fine tune several several hyperparameters. We change several hyperparameters like the optimizers, training type, font type, font size, image size, and type of network to select the best available configuration.

1) SELECTED CNN ARCHITECTURES

We apply three different CNN models for image classification by using the DWTM algorithm (i.e., ResNet-18, Inception, DenseNet). The results from the analysis of the CNN models are shown in Table 3. The ResNet-18 produces the best results initially. Later on, when testing the datasets with more random seeds, we find that the DenseNet shows more robustness. ResNet-18 still produces the best results initially but through each training phase, the decrease in loss value is much lower than the ones achieved by the DenseNet model. On the other hand, the DenseNet model produces poor performances initially but the performance improves substantially after each epoch. The DenseNet presents a consistent result regardless of the dataset. The Inception model shows a moderate performance which ranges between the ResNet-18 and DenseNet models. The network also produces high acccuracy regardless of the dataset being used. However, we observe that the Inception network tends to suffer from bias issues when random weights are used. In contrast, the ResNet-18 and DenseNet models may require longer to be trained with random weights but they always manage to produce significant scores. In the light of above performance, we recommend to use the DenseNet for consistency and train it for up to 30 epochs.

[5]https://archive.ics.uci.edu/ml/datasets/adult

2) OPTIMIZERS AND LEARNING RATE

We test several optimizers to minimize the loss functions of the CNN models. We find that the Stochastic Gradient Descent (SGD) with exponential learning rate decay (ELRD) is the best performing optimizer [40] and Adam optimizers performed most consistently in previous studies [41]. The results from our experiments are shown in Table 4. For each dataset, a separate set of parameters are required. Hence, it is a time-consuming and tedious process. In some scenarios, when varying random seed values, the SGD without ELRD failed to improve performance after running for a certain number of epochs. We also use ELRD was and the SGD achieved outstanding performance for some random seeds. However, the Adam and Adamax optimizers perform well regardless of the dataset and random seed used with the default parameters mentioned in [42]. The AdaGrad and RMSProp also perform well in many circumstances. However, Adam combines the advantages of AdaGrad and RMSProp while also using bias correction to ensure divergence does not occur. On the Cleveland dataset, the Adam optimizer trains the network much faster than the Adamax variant. Thus, the Adam optimizer is preferred for this study. Learning Rates of 0.001 and 0.0005 for the SGD and Adam were used respectively to find the optimal results on the benchmark datasets.

3) IMAGE SIZE

DWTM creates image datasets automatically, unlike the methods proposed in [11], [12], and [13]. We create image datasets in multiple sizes. Increasing the image size means

TABLE 3. Ablation study for selecting CNN architectures.

Model	Loss in each Epoch					Best Validation Accuracy
	1	2	3	4	5	
ResNet-18	0.4419	0.0174	0.0006	0.0005	0.0005	1.000
squeezenet1_0	0.8589	0.8609	0.7136	0.7041	0.6986	0.5333
mobilenet_v2	4.4156	2.4203	2.0301	2.0998	1.4554	0.5333
AlexNet	0.8071	0.7037	0.6907	0.6903	0.6907	0.5333
convnext_base	0.7937	0.7113	0.6513	0.6303	0.6275	0.8333
densenet121	2.4693	0.2690	0.0276	0.0288	0.0285	1.000
efficientnet_b0	6.4369	3.8097	1.6584	1.2948	0.8840	0.5667
Inception V1	1.3432	0.6834	0.0006	0.0003	0.0002	1.000
mnasnet1_3	6.7030	6.6173	6.5577	6.5005	6.4436	0.5333
VGG16_BN	1.7225	0.7750	0.7002	0.7164	0.7061	0.5333

TABLE 4. Ablation study for selecting Optimizers.

Optimizer	Loss in each Epoch					Best Validation Accuracy
	1	2	3	4	5	
Sgd with ELRD	0.6668	0.5895	0.5340	0.5246	0.5213	0.8666
Sgd with Momentum	0.6412	0.5280	0.4657	0.4609	0.4609	0.8500
Sgd with Momentum+ELRD	0.6717	0.5303	0.4550	0.4435	0.4412	0.8500
AdaGrad	0.3654	0.0199	0.0048	0.0037	0.0030	1.000
Adadelta	0.5165	0.1199	0.0333	0.0258	0.0202	0.7333
RMSProp	0.7167	0.5117	0.0221	0.0046	0.0025	1.000
Adam	0.3223	0.0002	0.0001	0.0001	0.0001	1.000
Adamax	0.3536	0.1020	0.0020	0.0011	0.0011	1.000

that the pixel positions can be provided more accurately and errors due to rounding are minimized. Furthermore, CNNs can learn from the data more efficiently if the images are larger. The *caveat* is that large image sizes lead to increased training time for the CNNs. The DWTM method also takes much longer to assign pixel positions to each feature as larger images require the method to iterate through a larger area. Hence, small image size is considered at the start and is gradually increased until the loss function for the CNN models is minimized. The results of our experiments are shown in Table 5 which indicates that image sizes of 128 by 128 are the best for tabular datasets. Larger images were also used, but we find that the loss per epoch values for image sizes of 128 and 256 are quite similar. The DWTM method, however, required almost four times as much time to assign pixel positions and create images. When using images of dimensions 64 and 96, we observe that the CNNs took much longer to minimize the loss values. As a result, the overall training time of the method increases.

4) FONT TYPE AND FONT SCALE

Four font types (i.e., plain, simplex, complex, and duplex) are considered for this study. Table 6 presents the results from analyzing these fonts. The complex and duplex font types achieve better loss and validation accuracy values than the plain and simplex font types. The plain and simplex performs similarly, so only one result was shown in the table. The complex and duplex font types provided identical results in majority of the cases of our experiments. Hence, both are viable and effective options for applying DWTM. Image sizes of 128 by 128 are used to study the effect of font types. Furthermore, the font scale is also varied. The font scale increases the thickness of the font. It is observed that with increasing font scale, the loss value improves. The increased thickness of the fonts representing the feature value means that the CNNs have more pixels to capture the feature information.

5) INDIVIDUAL FEATURE FONTS

The effect of font size for individual features is also considered when comparing the performance of DWTM to SuperTML. SuperTML also uses the ResNet-18 architecture. SuperTML was applied on the Iris and Adult datasets. It is observed that the ResNet-18 achieves equivalent performance for both techniques on the Iris dataset. However, there is a huge boost in performance on the Adult dataset and DWTM-ResNet-18 produces 100% accuracy scores compared to the 87.60% accuracy of SuperTML. This shows the remarkable effect of DWTM on the performance of CNN models on tabular datasets. The weighted feature embedding based on relevance provides the CNNs with more pixels to learn the important features.

6) PRETRAINED VS RANDOM WEIGHTS

The three CNN networks used for this study were trained in two ways. Initially, the networks were trained from scratch with random weights. Afterward, a new network is tested with the pre-trained weights on the Image Net dataset. Previous studies have shown the effectiveness of transfer learn-

TABLE 5. Ablation study for selecting image size.

Validation Loss						
Image Size	64*64		96*96		128*128	
Epoch	Resnet18	Densenet121	Resnet18	Densenet121	Resnet18	Densenet121
1	0.5307	1.0390	0.5354	1.2365	0.4397	1.1116
2	0.7022	0.0228	0.2081	0.0124	0.0001	0.0105
3	0.2147	0.0032	0.0003	0.0033	0.0001	0.0035
4	0.1042	0.0037	0.0003	0.0035	0.0000	0.0049
5	0.0439	0.0038	0.0003	0.0035	0.0000	0.0033
Adam	0.3223	0.0002	0.0001	0.0001	0.0001	1.000
Adamax	0.3536	0.1020	0.0020	0.0011	0.0011	1.000

TABLE 6. Ablation study for font type.

	FONT_HERSHEY_PLAIN		FONT_HERSHEY_DUPLEX		FONT_HERSHEY_COMPLEX	
Epoch	Val Accuracy		Val Accuracy		Val Accuracy	
	Resnet18	Densenet121	Resnet18	Densenet121	Resnet18	Densenet121
1	0.8333	0.5333	0.9000	0.5333	0.9333	0.5667
2	0.8167	0.9333	1.0000	1.0000	1.0000	1.0000
3	0.9667	1.0000	1.0000	1.0000	1.0000	1.0000
4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.0439	0.0038	0.0003	0.0035	0.0000	0.0033
Adam	0.3223	0.0002	0.0001	0.0001	0.0001	1.000
Adamax	0.3536	0.1020	0.0020	0.0011	0.0011	1.000

TABLE 7. Recommended parameters for DWTM.

Parameters	Values
Optimizer	Adam
Learning Rate	0.001
Image Size	128 × 128
Weights	Pre-trained
Font Type	Complex
Font Scale	4
CNN Model (Consistency)	DenseNet
CNN Model (Rapid Classification)	ResNet-18
CNN Model (Alternative)	Inception

ing for CNN models [43]. A similar observation was made during the conducted experiments. All three models can minimize the weights much more quickly when using the pre-trained weights of the Image Net dataset. The ResNet-18, in particular, shows a significant improvement in the loss per epoch and converges to its best minimum loss value within 30 epochs when using pre-trained weights. Hence, the pre-trained weights for the CNNs are preferred for tabular data analysis.

7) BEST PARAMETERS

A summary of the findings from the conducted ablation studies and the final parameter values used in the experiments on the benchmark datasets is shown in Table 7.

B. RESULTS

The selected parameters as mentioned in subsection IV-A are used to test the performance of DWTM on the selected benchmark tabular datasets in the following subsection IV-B1. Afterward, DWTM is also applied on Kaggle competitions to see how it matches up against the state-of-the-art methods used in tabular data. The results on these competitions are discussed later in subsection IV-B2.

1) RESULTS ON THE BENCHMARK DATASETS

The results from the experiments are displayed in Table 8. DWTM+R, DWTM+D, and DWTM+I refer to the use of the ResNet-18, DenseNet and InceptionV1 models, respectively. Results from the TML [11] and IGTD [13] methods are also included for the Iris, Wine and Adult datasets. We also apply three traditional classifiers (i.e., Logistic Regression (LogReg), Random Forests (RF) and Support Vector Machine (SVM)) on these datasets and the results are presented here for comparison.

From Table 8, it can be observed that DWTM provides significantly better results on the Cleveland dataset than the traditional classifiers. The DenseNet, Inception and ResNet-18 produce 100% accuracy scores. The results show that DWTM is a viable option for disease diagnosis due to its ability to be unbiased. The Cleveland dataset has only 303 instances. DWTM assists the CNN models in providing better results than the traditional classifiers on the Cleveland dataset. The method again proves successful in disease diagnosis on the Diabetes dataset. The CNN models provide balanced results, which the traditional classifiers fail to deliver. Results on the Diabetes dataset provide further evidence of the model's ability to provide state-of-the-art results on a medical dataset. DWTM shows further robustness on the

TABLE 8. Result comparison of the CNN models and traditional classifiers.

Dataset	Model	Acc.	Sns.	Spc.	Prec.	F1.
Cleveland	DWTM+R	100	100	100	100	100
	DWTM+D	100	100	100	100	100
	DWTM+I	100	100	100	100	100
	LogReg	83.33	78.57	87.50	83.48	83.16
	RF	83.33	71.42	93.75	84.93	82.86
	SVM	66.66	42.86	87.50	69.31	64.11
Diabetes	DWTM+R	100	100	100	100	100
	DWTM+D	100	100	100	100	100
	DWTM+I	100	100	100	100	100
	LogReg	82.69	82.76	82.61	82.44	82.53
	RF	95.19	98.48	89.47	95.67	94.72
	SVM	94.23	96.55	91.30	94.39	94.12
Breast Can.	DWTM+R	98.58	98.91	97.96	98.91	98.91
	DWTM+D	100	100	100	100	100
	DWTM+I	100	100	100	100	100
	LogReg	97.14	95.83	97.82	96.83	96.83
	RF	94.29	87.50	97.82	94.60	93.52
	SVM	95.71	91.67	97.83	95.69	94.7
Iris	DWTM+R	93.33	-	-	93.33	93.33
	DWTM+D	100	-	-	100	100
	DWTM+I	100	-	-	100	100
	IGTD [13]	93.33	-	-	-	-
	TML [11]	93.33	-	-	-	-
	LogReg	100	-	-	100	100
	RF	100	-	-	100	100
	SVM	100	-	-	100	100
Wine	DWTM+R	100	-	-	100	100
	DWTM+D	100	-	-	100	100
	DWTM+I	100	-	-	100	100
	IGTD [13]	96.30	-	-	-	-
	TML [11]	97.30	-	-	-	-
	LogReg	96.30	-	-	97.44	95.64
	RF	94.44	-	-	93.71	94.46
	SVM	96.30	-	-	95.38	94.44
Adult	DWTM+R	100	100	100	100	100
	DWTM+D	100	100	100	100	100
	DWTM+I	100	100	100	100	100
	TML [11]	87.60	-	-	-	-
	LogReg	80.55	26.59	97.14	77.63	61.87
	RF	84.85	61.70	91.97	79.45	77.99
	SVM	85.30	57.74	93.77	80.93	75.76

Breast Cancer dataset as each CNN model produces almost the maximum possible results using the proposed method.

The Iris dataset is the most popular dataset from UCI. However, it is also the smallest dataset containing only 150 instances and four features. Furthermore, the Iris dataset has three classes of Iris plants. Despite the minimal information available, DWTM achieves maximum performance on the dataset, thus matching the performance of traditional classifiers. The Wine Dataset contains only 178 instances and three classes similar to the Iris dataset. Table 8 shows our experiments' results produced on the Wine dataset. It is observed that the CNNs consistently produce results that are usually better or equivalent to the results from the traditional classifiers. The Wine and Iris dataset results also show DWTM's ability to deal with multiclass tabular datasets. The results of the experiments on the Adult dataset are shown in Table 8. The Adult dataset contains over 48000 instances, the largest dataset used for evaluation in this study. The performance of DWTM models is far better than those of the traditional classifiers. The performance on this dataset shows that DWTM can be applied to datasets of any dimension. A summary of the loss

per epoch of the CNNs on the test datasets are shown in Figure 5.

In our experiment, in the majority of the cases, we get high test accuracy, i.e., greater than 98%, which seems to be overfitting. There might be a few reasons for such overfitting. If the test and train data sets have common instances to be tested. Another reason might be that if the dataset is imbalanced and the test data is largely from the majority class, then there is a high chance of being overfitted. Since we consider popular datasets and they are well balanced. We also applied the cross validation approach accordingly during testing. However, these well known datasets have features which can be separated distinctly from one another. Therefore, we were able to achieve outstanding results for four of the datasets while two datasets show less than 100% accuracy. For more acceptability of our process, we considered multiple random seeds and the model achieves such high accuracies in the four of the cases on the testing and validation sets as mentioned in Section IV-A.

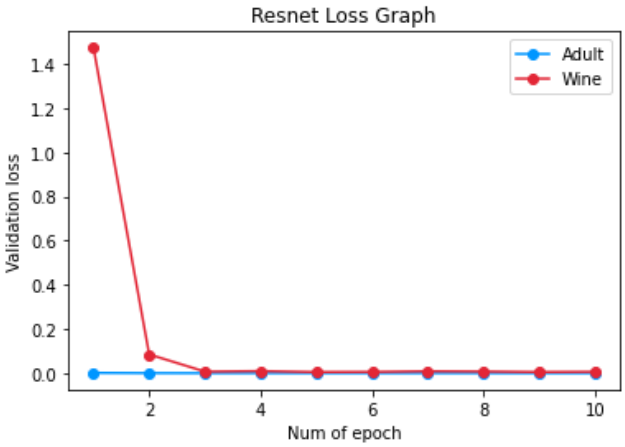
2) RESULTS ON KAGGLE COMPETITIONS

DWTM is applied on live competitions in Kaggle to observe how they generalize on tabular datasets. For this study, the ongoing competition Spaceship Titanic is selected. To this date (20-07-2022) there are 19282 entries on 2322 teams. DWTM is applied on the training set to receive the feature weights. The same weights are used on the test dataset. Afterward, the dataset is converted to images. ResNet-18, DenseNet and Inception methods are applied on the dataset and it is observed that ResNet-18 and DenseNet methods achieve 100% accuracy on the validation sets. The highest accuracy on the on leaderboards has a score of 86.509. This shows the immense potential of DWTM to achieve high performance levels on any type of tabular data.

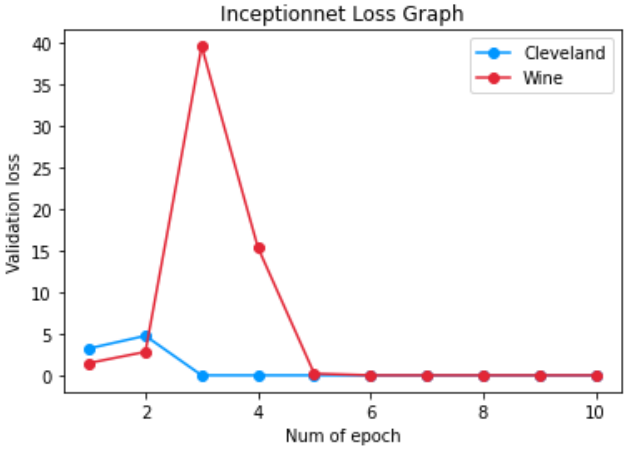
V. DISCUSSION

In the past, CNN models were not very effective with tabular datasets. Our study shows that CNN models consistently outperform the traditional classifiers on tabular datasets. Furthermore, in the past, it was assumed that DL only performed well with large amounts of data [44]. The results contradict this statement and provide further evidence of the usability of CNN models for small datasets.

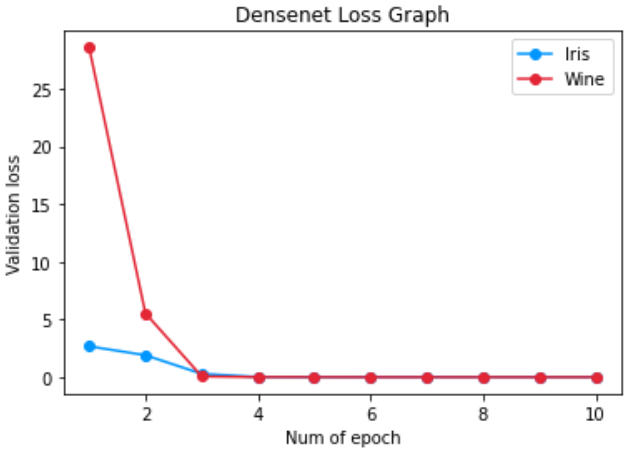
Deep learning and traditional machine learning models have the major difference in the feature selection and classification steps. Deep learning methods, i.e., CNN, has an automated feature extraction process during the convolution steps [45]. The convolution step applies several filters to extract prominent features. The convolution layer finds features easily if the objects are distinguishable in the image canvas without much effort. Alternatively, images from different classes represent different distributions of the objects. In this connection, our DWTM embedding approach successfully performs the task during feature positioning on the canvas in an easily recognizable manner by the filters in the convolution layers. Thus, our DWTM helps to classify the instances easily



(a) ResNet Loss per Epoch on Adult and Wine Datasets



(b) Inception Loss per Epoch on Cleveland and Wine Datasets



(c) DenseNet Loss per Epoch on Iris and Wine Datasets

FIGURE 5. Loss curves of the CNNs on the test datasets.

in the later stages of CNN such as flatten layer and during applying the softmax activation function.

The loss for each epoch on the Benchmark datasets is shown in Figure 5. All three CNN models demonstrate similar performance on the Wine dataset eventually. However, the ResNet and DenseNet converges much faster than Inception. Furthermore, in smaller datasets the CNNs take longer to

reach minimal loss values. The Adult dataset which is quite large in comparison is much more suited to the CNNs as they reach optimum loss values rapidly. This suggests that the DWTM is much more suited to larger tabular datasets.

DWTM is also a viable option for classification tasks on medical datasets. The method effectively deals with bias much better than the traditional classifiers. The model outperforms the traditional classifiers in the Cleveland, Diabetes, and Breast Cancer datasets. Notably, in the Cleveland dataset, the traditional classifiers all produced Sensitivity scores below 0.80. On the other hand, the CNN models produce high sensitivity and specificity scores. DWTM also shows it can perform multiclass tasks with ease. The method provides outstanding results in both the Iris and Wine datasets. The results produced are better (or similar) than those provided by SuperTML and traditional classifiers. The Adult dataset is the only dataset used with a vast number of instances. DWTM surpasses the performance of the traditional classifiers and the performance of SuperTML by a considerable margin. All these results show the robustness of the model as it works effectively in small, large and multiclass datasets. Furthermore, the results also indicate that DWTM makes the CNN models much more beneficial than the traditional classifiers for tabular data tasks.

From these experiments, it can also be noted that the Adamax and Adam optimizers produce better results, followed by the Stochastic Gradient Descent (SGD) with Exponential Learning Rate Decay. Our recommendation based on the experiments is to use the Adam optimizer with default parameters initially. If Adam fails to provide decent results, the learning rate and bias values vary. However, theoretically, the SGD with momentum achieves minima most effectively if the proper parameter values are assigned. If the Adam optimizer fails to provide good results, experiments should be conducted with the SGD to find its best parameters. Even then, if good results are not achieved, we can consider recent variants of Adam like NAdam [46], ND-Adam [47] and combining Adam with SGD [48]. A learning rate of 0.001 alongside the Adam optimizer tends to reach the desired outcomes more often. Learning rate values closer to 0.0005 works the best with SGD.

ResNet-18 is highly efficient with tabular data as per the experiments. However, if the models are trained up to 30 epochs, DenseNet proves to be the best option. It is recommended to use DenseNet on tabular datasets with DWTM. ResNet-18 is the best option when rapid or real-time classification solutions are required. Furthermore, large datasets require a significant amount of time to be trained. ResNet-18 is a more realistic solution for big data and larger datasets. ResNet-18 also has more room for improvement due to the availability of deeper ResNet-18 networks [14]. The Inception network is the third best network for tabular data and it is recommended to use this network if DenseNet fails to perform.

Although previous studies of SuperTML [11], DeepInsight [12] and IGTD [13] use the idea of converting non-image datasets into image datasets for deep learning applications, none of the studies utilize feature weights for classification. IGTD is an updated version of DeepInsight for tabular data. However, IGTD was specifically designed for gene expressions. The study [13] has limited applications on tabular datasets and so IGTD is not a proper solution for all tabular datasets. On the other hand, SuperTML was applied to numerous tabular datasets. Furthermore, it achieved high accuracy levels when compared to the traditional classifiers. As a result, SuperTML is considered the benchmark for applying CNNs on tabular data. On top of this, the study in [36] states that typically it is challenging to deal with categorical data using DL models. DWTM embedding technique allows CNNs to take categorical data as inputs and deal with them efficiently. To the best of our knowledge, this makes DWTM the only technique which can handle categorical data with CNNs on structured datasets.

The IGTD and SuperTML provided different ideas for assigning the feature values into the images. SuperTML used texts and inserted them with OpenCV. IGTD used pixel intensities to represent the values within the features. The method provided by SuperTML is used initially for DWTM due to the reasons mentioned above of SuperTML being the benchmark technique. Later on, the pixel intensities method of IGTD is also considered. However, the technique has not dealt with categorical data and at present, there is no way of representing categorical data with pixel intensities during embedding without creating numerical relationships between them. Hence, the pixel intensities technique is not used.

The results show that DWTM provides similar or better outcomes when compared to SuperTML. Comparing the performance of the methods on the Adult dataset shows that the use of weighted font sizes has a significant impact on the performance of the CNNs. Without feature weights, the CNN provides an accuracy of 87.60% as shown by the results of SuperTML on the tabular dataset. On the other hand, DWTM uses weighted feature importance based on their correlation significance to the class label. The font size is determined mathematically from these feature weights and each CNN model produces 100% accuracy. DWTM is also more dynamic and robust compared to SuperTML. It uses a novel approach by using the feature weights to create images. Compared to the DeepInsight technique, it also performs remarkably on small datasets. DWTM provides the most significant space to the most important features based on the assigned weights. As a result, the CNN models are likely to learn the more complex patterns from the essential features. To the best of our knowledge, no previous study used feature weights for CNN models. On top of this, DWTM uses an entirely automated procedure, unlike the previous techniques.

This study provides the foundation of feature analysis for DL models. In this study, Pearson Score Coefficient is used for assigning weights. Future studies can use other statistical techniques and evaluate which options are the best for calculating feature weights. The Pearson Correlation and Cramer's V are preferred for these experiments as they can find the

strength of associativity of a feature to the class. Alternatively, future researchers can use Fishers Correlation [49] which uses a transformation technique to deal with highly correlated features. This method works much better than Pearson Correlation with highly correlated features and is another alternative for calculating feature weights. Another popular statistical technique is the Analysis of Variance (ANOVA) [50]. This method works better when the class contains three or more levels. Hence, ANOVA is usually preferred over Pearson Correlation for multiclass datasets and regression tasks. ANOVA can also be used with DWTM for assigning weights to determine the importance of the features. Due to the introduction of DWTM, feature analysis may become essential for CNN applications on tabular data tasks. Combining various feature analysis techniques and using them with DWTM has the potential to produce state-of-the-art results in all kinds of tabular data tasks.

In the experiments, a maximum number of 30 epochs are used. The CNN models are pre-trained on the ImageNet dataset; thus, 30 epochs are sufficient in most cases. Increasing the number of epochs may produce even better results. Additionally, models like the Resnet-152, Inception-v4 and other CNN models show substantial learning capabilities. Using these models with DWTM can further increase the performance of CNNs for tabular data. Feature Selection is a vital part of tabular data analysis. In the future, DWTM can be upgraded further to produce the best subset of features for CNN applications on tabular datasets.

DWTM proves to be an effective tool for tabular data classification. Nevertheless, CNNs and the method itself are computationally expensive and thus for simple datasets it is better to use the cost-effective traditional classifiers. DWTM should only be used when these classifiers fail. The DWTM is also limited to only classification tasks. In the future, a viable regression package is required to make CNNs viable for tabular data regression tasks. Furthermore, using pixel intensities instead of integer values should theoretically give the CNNs more information to complete the prediction tasks effectively. Further analysis needs to be conducted to compare the performance of DWTM using embedded values and pixel intensities.

VI. CONCLUSION

In this paper, we have developed a feature embedding technique named DWTM that dynamically assigns feature weights for tabular dataset, while using different CNN architectures. We have applied DWTM over six benchmark datasets and compared the results with popular existing methods (i.e., SuperTML, IGTD, etc.) and three traditional classifiers. We have also observed that DWTM usually outperforms (an average accuracy of 98%) the results of traditional classifiers and the previously mentioned CNN-based methods on the benchmark datasets. Additionally, the method is robust for utilization in various types of datasets (e.g., multiclass, large/small). To the best of our knowledge, this is the first study for feature embedding, which computes the strength of the features for a tabular dataset and applies any CNN architectures for classification tasks. This study can be considered the new and novel benchmark for embedding techniques on tabular data.

SUPPLEMENTARY MATERIALS

The implementation of DWTM is available at: https://github.com/Ifraham/Dynamic-Weighted-Tabular-Method, The readme file contains the information on how to apply DWTM. The package is available in the directory labeled *DWTM*. The experiments on the benchmarked datasets is available in the directory labeled *Experiments*.

REFERENCES

[1] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big Data*, vol. 1, no. 1, pp. 51–59, 2013.

[2] W. S. Noble, "What is a support vector machine?" *Nature Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, 2006.

[3] D. W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.

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

[5] M. Iman, A. Giuntini, H. Reza Arabnia, and K. Rasheed, "A comparative study of machine learning models for tabular data through challenge of monitoring Parkinson's disease progression using voice recordings," 2020, *arXiv:2005.14257*.

[6] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Raheem, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[7] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2016.

[8] P. Kim, "Convolutional neural network," in *MATLAB Deep Learning*. Berkeley, CA, USA: Apress, 2017, pp. 121–147.

[9] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," 2015, *arXiv:1501.02876*.

[10] D. Mishra, B. Naik, R. M. Sahoo, and J. Nayak, "Deep recurrent neural network (deep-RNN) for classification of nonlinear data," in *Computational Intelligence in Pattern Recognition*. Singapore: Springer, 2020, pp. 207–215.

[11] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong, "SuperTML: Two-dimensional word embedding for the precognition on structured tabular data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–9.

[12] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Sci. Rep.*, vol. 9, no. 1, pp. 1–7, Aug. 2019.

[13] Y. Zhu, T. Brettin, F. Xia, A. Partin, M. Shukla, H. Yoo, Y. A. Evrard, J. H. Doroshow, and R. L. Stevens, "Converting tabular data into images for deep learning with convolutional neural networks," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, May 2021.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[17] M. N. Islam, N. I. Khan, A. Roy, M. M. Rahman, S. H. Mukta, and A. K. M. N. Islam, "Sentiment analysis of Bangladesh-specific COVID-19 tweets using deep neural network," in *Proc. 62nd Int. Sci. Conf. Inf. Technol. Manage. Sci. Riga Tech. Univ. (ITMS)*, Oct. 2021, pp. 1–6.

[18] E. M. Khan, M. S. H. Mukta, M. E. Ali, and J. Mahmud, "Predicting users' movie preference and rating behavior from personality and values," *ACM Trans. Interact. Intell. Syst.*, vol. 10, no. 3, pp. 1–25, Nov. 2020.

[19] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018. [Online]. Available: https://www.hindawi.com/journals/cin/2018/7068349/
[20] M. Rashid, H. K. Jahan, A. Huzzat, R. A. Rahul, T. B. Zakir, and F. Meem, "Text2chart: A multi-staged chart generator from natural language text," in *Pacific-Asia Conf. Knowl. Discovery Data Mining*. Springer, 2022, pp. 3–16.
[21] F. Ahmed, N. Akther, M. Hasan, K. Chowdhury, and M. S. H. Mukta, "Word embedding based news classification by using CNN," in *Proc. Int. Conf. Softw. Eng. Comput. Syst. 4th Int. Conf. Comput. Sci. Inf. Manage. (ICSECS-ICOCSIM)*, Aug. 2021, pp. 609–613.
[22] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Audio-visual speech recognition using deep learning," *Appl. Intell.*, vol. 42, no. 4, pp. 722–737, 2015.
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1106–1114.
[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[25] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[26] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7335–7345.
[27] L. Buturović and D. Miljković, "A novel method for classification of tabular data using convolutional neural networks," bioRxiv, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/05/03/2020.05.02.074203
[28] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong, and C. Young, "Super characters: A conversion from sentiment classification to image classification," 2018, *arXiv:1810.07653*.
[29] D. Luo, S. Zeng, and J. Chen, "A probabilistic linguistic multiple attribute decision making based on a new correlation coefficient method and its application in hospital assessment," *Mathematics*, vol. 8, no. 3, p. 340, Mar. 2020.
[30] R. J. Tallarida and R. B. Murray, "Chi-square test," in *Manual of Pharmacologic Calculations*. New York, NY, USA: Springer, 1987, pp. 140–142.
[31] R. K. Prematunga, "Correlational analysis," *Austral. Crit. Care*, vol. 25, no. 3, pp. 195–199, 2012.
[32] L. Rello and R. Baeza-Yates, "The effect of font type on screen readability by people with dyslexia," *ACM Trans. Accessible Comput.*, vol. 8, no. 4, pp. 1–33, May 2016.
[33] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision With the OpenCV Library*. Sebastopol, CA, USA: O'Reilly Media, 2008.
[34] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet, S. Arnoud, and L. Yatziv, "Ontological supervision for fine grained classification of street view storefronts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1693–1702.
[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[36] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," 2021, *arXiv:2110.01889*.
[37] R. R. Fletcher, A. Nakeshimana, and O. Olubeko, "Addressing fairness, bias, and appropriate use of artificial intelligence and machine learning in global health," *Frontiers Artif. Intell.*, vol. 3, p. 116, Apr. 2021.
[38] M. F. Islam, R. Ferdousi, S. Rahman, and H. Y. Bushra, "Likelihood prediction of diabetes at early stage using data mining techniques," in *Computer Vision and Machine Intelligence in Medical Image Analysis*. Singapore: Springer, 2020, pp. 113–125.
[39] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. Nat. Acad. Sci. USA*, vol. 87, no. 23, pp. 9193–9196, 1990.
[40] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 10665–10673.
[41] S. Kim, W. Chung, and S. Shin, "Acoustic full-waveform inversion using Adam optimizer," *Geophysics Geophys. Exploration*, vol. 22, no. 4, pp. 202–209, 2019.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[43] X. Han, Y. Zhong, L. Cao, and L. Zhang, "Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification," *Remote Sens.*, vol. 9, no. 8, p. 848, 2017.
[44] A. Kamilaris and F. X. Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Apr. 2018.
[45] R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande, and P. Singh, "Prediction of heart disease using a combination of machine learning and deep learning," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–11, Jul. 2021.
[46] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. ICLR*, 2016, pp. 1–4.
[47] Z. Zhang, "Improved Adam optimizer for deep neural networks," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–2.
[48] N. Shirish Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," 2017, *arXiv:1712.07628*.
[49] A. G. Asuero, A. Sayago, and A. González, "The correlation coefficient: An overview," *Crit. Rev. Anal. Chem.*, vol. 36, no. 1, pp. 41–59, 2006.
[50] J. Kaufmann and A. Schering, *Analysis of Variance ANOVA*. Wiley, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat06938

MD. IFRAHAM IQBAL received the B.Sc. degree in computer science and engineering from United International University, in 2021. He is currently pursuing the degree with the Data Science Department, Friedrich Alexander University of Erlangen, Germany. His research interests include computer vision, deep learning for tabular data, object detection, reinforcement learning, medical data analysis, and pattern recognition.

MD. SADDAM HOSSAIN MUKTA received the Ph.D. degree from the Data Science and Engineering Research Laboratory (DataLaboratory), BUET, in 2018. He is currently an Associate Professor with the Computer Science and Engineering Department. His research interests include deep learning, machine learning, data mining, and social computing. He has a number of quality publications in both national and international conferences and journals.

AHMED RAFI HASAN is currently pursuing the degree with the Computer Science and Engineering Department, United International University. He is also an Undergraduate Teaching Assistant with the Computer Science and Engineering Department. His research interests include deep learning, computer vision, pattern recognition, imbalanced data analysis, and scene understanding.

SALEKUL ISLAM (Senior Member, IEEE) received the Ph.D. degree from the Computer Science and Software Engineering Department, Concordia University, in June 2008. He is currently a Professor and the Head of the CSE Department, United International University, Bangladesh. Previously, he worked as an FQRNT Postdoctoral Fellow at the Énergie, Matériaux et Télécommunications (EMT) Centre, Institut National de la Recherche Scientifique (INRS), Montreal, Canada. His research interests include future internet architecture, blockchain, edge cloud, software-defined networks, multicast security, security protocol validation, and machine learning and AI. He is also serving as an Associate Editor for the IEEE Access and *Frontiers in High Performance Computing* journals.