ORIGINAL RESEARCH PAPER

Machine Learning

Comparative Performance Analysis Between K-Nearest Neighbors and Feed-Forward Neural Networks in Classifying Dimensionality-Reduced Image Data of Hand-Written Digits.

KEY WORDS: Hand-written Digit Classification, Image Classification, K-Nearest Neighbors, Feedforward Neural Network

Krish S. Kamath

ABSTRACT

Image classification is an essential tool used across various industries in our technologically enabled world. The process utilises the power of machine learning algorithms to transform industries and solve complex real-world problems. In machine learning, how does a streamlined and straightforward algorithm like the k-Nearest Neighbors algorithm perform, compared to a more intricate Backpropagation multilayer neural network like Feedforward Neural Network, especially in the case of image classification? This research aims to examine this comparative performance analysis, illuminating the intriguing complexities and efficiencies inherent in these contrasting approaches. Drawing from Yuchun Lee's 1991 findings, which revealed that both the backpropagation algorithm and the k-nearest neighbours algorithm had closely matched error rates of approximately 5.14% and 5.15%, respectively, on regular, non-dimensionally reduced data, this study hypothesises that a backpropagation algorithm, like the FeedForward Neural Network, and K-Nearest Neighbors might demonstrate comparable performance on such data. To perform this experimental analysis, however, various mediums are used. Python and essential libraries, including numpy, matplotlib, sklearn, and keras, are used to evaluate the research hypothesis. The experiment employs a Backpropagation Neural Network and K-Nearest Neighbors model, trained and tested on dimensionally reduced data. keras is employed for data acquisition, sklearn for creating the two separate models, and matplotlib for data and result visualization. This approach facilitates acquiring a more nuanced comparison of the two models' performance under varying data conditions. This research reveals distinctive/similar performance characteristics of both Backpropagation Neural Networks and K-Nearest Neighbors models when processing dimensionally reduced image data. The results are essential, for it enhances our comprehension of these models' behaviour in this particular context, thus providing a foundation for informed decisions in subsequent research and practical applications in image classification.

INTRODUCTION

Machine learning has been making significant strides, with its applications now reaching the field of image recognition. Specifically, neural networks have exhibited impressive results in the domain of handwritten digit recognition, progressively inching towards levels of performance that rival human capabilities (LeCun, Bengio & Hinton, 2015)[11]. The remarkable advancement of these technologies points to their potential to decode complex patterns within images and sets the stage for this paper's exploration. Herein, we focus on the comparative performance of K-Nearest Neighbors (KNN) and Feed-Forward Neural Networks (FFNN), two machine learning algorithms, in classifying image data of hand-written digits

The recognition of pattern recognition, or specifically, digit recognition, has seen a substantial surge in interest over the past decade, owing to its broad applicability in a multitude of areas. Technologies driven by such algorithms are transforming sectors as diverse as banking, postal services, healthcare, and even robotics. These applications range from automating check processing and mobile payments in finance to enhancing number recognition in vehicle license plates, sorting mail in postal services, improving medical imaging, and even refining Captcha systems for improved online security. It also plays a pivotal role in Optical Character Recognition (OCR) [5]. Even in the expanding field of robotics, it aids in enabling sophisticated computer vision capabilities. Our specific focus is on handwriting recognition, a subset of these applications that presents its own unique challenges. The goal of this study is to offer insights into the relative strengths of two well-established machine learning algorithms – K-Nearest Neighbors and Feed-Forward Neural Networks. By doing so, it aims to guide the selection of the most effective algorithm for similar image classification tasks. This has significant practical implications, potentially enabling more accurate and efficient digit recognition and propelling advancements across these diverse application areas.

However, In the field of machine learning, algorithms like KNN, which rely on Euclidean distances, encounter a significant challenge when dealing with high-dimensional data. This issue is often referred to as the "curse of dimensionality" in the machine learning society, and it results from the sparsity of data in high-dimensional spaces, where the concept of 'nearness' loses its meaning as all data points appear equally distant[10]. Consequently, KNN, with its principle of classifying inputs based on their proximity to examples, needs help finding meaningful patterns. This problem increases as the dimensionality increases, requiring an exponentially more significant number of training examples to maintain algorithm performance [10] [5]. This surge in training data imposes substantial computational demands and underlines the complexities associated with using distance-based algorithms in high-dimensional scenarios.

There are speculations based on theoretical and empirical results[10] that many "real-world" problems are much "simpler" than those problems that have been proven to be problematic in high dimensions. Theoretical challenges often highlight the complexities of high dimensionality, while real-world tasks like digit recognition might require fewer examples.

Guided by this, dimensionality reduction is implemented on the training image dataset using one of the more recognised ways called Principal Component Analysis (PCA) to create a more comparable, potentially realistic and "simpler" framework for examining and contrasting the effectiveness of two distinct algorithms, K-Nearest Neighbors and Feed-Forward Neural Networks, on identifying handwritten digits. The dataset comprises a sizable training set of 60,000 images, supplemented by a robust testing set of 10,000 images, enabling a detailed investigation into the efficiency and accuracy of these two prominent machine learning algorithms.

The underlying purpose of this study is to undertake a comparative performance analysis of K-Nearest Neighbors and Feed-Forward Neural Networks in classifying dimensionality-reduced image data of hand-written digits, emphasising their individual and comparative performances.

Before diving into the implementation, detailed explanation,

and visualisation of our experiment and its results, it is crucial first to understand the fundamental algorithms utilised in this study and the concept of dimensionality reduction.

K-Nearest Neighbors

Overview

The K-nearest neighbour (KNN) classification method was introduced to analyse features in situations where traditional mathematical models were not straightforward or feasible. A pivotal reference to this approach can be found in a 1951 report by Fix and Hodges for the US Air Force School of Aviation Medicine[14]. In this report, they presented what can be considered one of the earliest versions of the KNN algorithm for pattern classification. This groundbreaking work by Fix and Hodges laid the foundation for the KNN method we recognize today. Their innovative approach provided an alternative to traditional methods, especially when dealing with complex data patterns. The report's significance is underscored by its lasting impact on the field, with the KNN algorithm now being a staple in modern data analysis and machine learning.

The K-Nearest Neighbors algorithm is simple yet efficient in machine learning. It serves as a powerful tool for both classification and regression tasks. However, it is more widely used for classification predictions [6]. The KNN algorithm organises data into cohesive clusters or subsets, using similarities with previously trained data to classify new input. The input is categorised into the class to which it has the greatest number of nearest neighbours, reflecting its similarity with existing class members [6]. The KNN algorithm, a supervised learning algorithm, operates as a classification technique without imposing specific assumptions regarding the underlying structure of the dataset, categorising it as a non-parametric approach, and it is known for the simplicity and effectiveness of its operation and its low computational demands. It uses a training dataset wherein data points are systematically classified into distinct categories, and the algorithm is adept at predicting the class affiliations of new, unlabeled data[6]. The K-Nearest Neighbors algorithm functions by employing a labelled training set where data points are assigned to specific classes, thereby facilitating the accurate prediction of class labels for previously unlabeled instances[6].

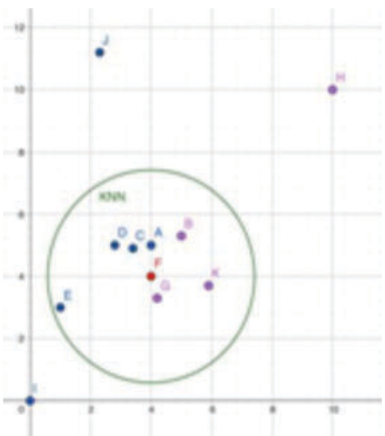


Fig 1.0 Visualization of the KNN algorithm in action: The red point represents the new data point, while the blue and purple points denote existing data from two distinct classes. KNN identifies the seven nearest neighbours to classify the red point.

Working

KNN is a categorisation algorithm involving two primary phases:

1) Training Phase: A classifier is built using the provided training data.

2) Evaluation Phase: The performance of the classifier is assessed.

Utilising the nearest neighbour method, unlabeled data is categorised based on the classifications of its adjacent neighbours. The K-Nearest Neighbors (KNN) algorithm incorporates this principle in its computations. Within the KNN framework, a specific 'K' value is designated, aiding in classifying unfamiliar data points.

When the dataset presents a new unlabeled data point, the KNN algorithm executes two primary steps: Initially, it examines the 'K' data points most proximate to the new entry, referred to as the K-nearest neighbours, and then, subsequently, drawing from the classifications of these neighbours, the KNN algorithm deduces the appropriate class for the new data point .

The algorithm works best for datasets that naturally cluster in some areas of the graph, making it easier to sort data accurately into clear groups. KNN determines where the new data fits by seeing which group has the most points close to it. This involves measuring the Euclidean distance between the new data and existing data points. Once we identify the closest groups, or "K-Nearest Neighbours", we pick the group with the most members as the category for the new data. After saving the training set, all parameters should be normalised, which means to adjust all the data to a standard scale so that no single type of data overshadows the others. The classification outcome is influenced by the value of 'K'.The algorithm's effectiveness depends on factors like the chosen number for 'K', the Euclidean distance, and the extent of normalisation of the parameters.[6]

Determining the Value of K

Selecting the optimal K value in the KNN algorithm is done by analysing and cross-validating the data [6]. Larger K values in the KNN algorithm result in more precision by reducing noise, smoother class boundaries, and potentially including points from other classes, mainly when data points are scattered. If we take smaller K values to implement the algorithm, it becomes highly sensitive to anomalies and tends to perform less accurately. For example, When K=1, data is categorised based on its closest neighbour, leading to zero error in training data since each point's nearest neighbour is itself. However, this can result in overfitting. Another example - we consider the two closest neighbours when K=2, which results in ambiguity if both neighbours are from differing classes. Increasing the number of neighbours, for example, 6-nearest neighbours can offer clarity and result in more accurate classifications. However, larger K values can blur class boundaries, potentially including other class points. Moreover, when data are scattered in both cases, determining the K value becomes more arduous and complicated. There are various methods present to determine an "ideal" K value; A potential way to accurately ascertain the most optimal value of K for the K-Nearest Neighbors algorithm is to divide the initial dataset into two distinct subsets: one for training the algorithm and another for testing its performance, ensuring that the chosen K value generalises well to new, unseen data, and redefining the value of K in an iterative process [6].

Odd Values for K in KNN Algorithm

In the K-Nearest Neighbors (KNN) algorithm, the choice of the 'k' value, which represents the number of neighbours to be considered, plays a pivotal role in determining the classification of a new data point. When 'k' is an even number, there is a potential risk of encountering a tie, especially when there are equal numbers of opposing class points among the nearest neighbours. This can lead to ambiguity in classifying the new data point.

To reduce this issue, often, it is recommended to use odd values for 'k'. By doing so, the algorithm ensures that there will always be a majority class among the 'k' nearest neighbours,

thus eliminating the possibility of a tie. This simple yet effective strategy can enhance the reliability and robustness of the KNN algorithm, especially in datasets where class distributions are nearly balanced, if not equal.

Feedforward Neural Network

Overview

Drawing inspiration from their biological counterparts, artificial neural networks, such as feedforward neural networks, emulate the human brain and the broader nervous system. Unlike traditional digital computers, the biological brain boasts a unique structure and distinct information processing methods. The human brain, representing the pinnacle of biological intelligence, often surpasses conventional computing systems in various capacities. While standard computers rely on predefined software or instructions, the biological brain's hallmark is its innate ability to adapt and learn. In the realm of artificial neural networks, feedforward neural networks stand out, aiming to mimic this dynamic learning process through structured layers[7].

At the heart of neural networks lies the neuron, serving as a computational unit. These neurons, interconnected through "synaptic weights" or concisely termed "weights," play a pivotal role. As each neuron receives weighted information, it can originate from external sources or the outputs of neighbouring neurons. Subsequently, by channelling this aggregated data through an activation function (When data moves through the network, each neuron calculates the weighted sum of its inputs, adds a bias, and uses an activation function for its output), the neuron produces a distinct output [7].

Expanding on the foundational concepts of neurons and their interconnected architectures, our experiment primarily aims to harness the capabilities of feed-forward neural networks. These networks can be broadly categorized alongside recurrent neural networks. The distinguishing feature of a feed-forward neural network is its lack of feedback loops from neuron outputs back to inputs. On the other hand, recurrent neural networks incorporate such feedback mechanisms, with synaptic connections looping from outputs to inputs, be it of the same neuron or others [7]. Neural networks are typically organized into layers. Based on the number of these layers, feed-forward neural networks can be delineated as single or multi-layer. For the purposes of our study, the emphasis will be on utilising a feed-forward neural network to achieve our experimental objectives.

Working

Feedforward Neural Networks (FFNNs) are a foundational class of artificial neural networks characterised by a unidirectional flow of information. The architecture of FFNNs consists of multiple layers, starting with the Input Layer that receives the initial data for processing. As data progresses through the network, each neuron in subsequent layers calculates the Weighted Sum of its inputs, which is a combination of the data it receives and the weights associated with its connections.[1][3]

An Activation Function is applied upon computing the weighted sum to introduce non-linearity into the model. Commonly used activation functions include sigmoid, tanh, and ReLU. These functions ensure that the network can learn and represent complex patterns in the data. The processed data then reaches the Output Layer, which produces the final predictions or classifications based on the transformed input. [1][7]

A distinguishing feature of FFNNs is their ability for Learning & Adaptation. Using a method called backpropagation, FFNNs adjust the weights of their connections based on the discrepancies between their predictions and the actual labels. Optimization techniques, such as gradient descent, are employed to minimize these errors, refining the network's performance over time.[7][3]

In practical applications, FFNNs have shown significant prowess in tasks like handwritten image recognition. Given handwritten characters' diverse styles and forms, traditional algorithms often must catch up. However, when considering FFNNs, with their layered architecture and adaptive learning capabilities, can effectively recognise and classify these characters. When a handwritten image is fed into the network, it undergoes a series of transformations, with the final output layer providing a probability distribution over potential characters. The character with the highest probability is then chosen as the recognised output, showcasing the network's ability to discern intricate patterns in handwritten data.[1][3]

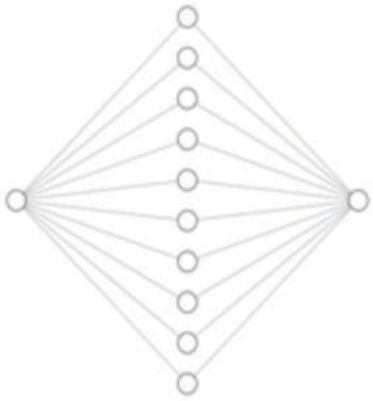


Fig 2.1 Single-layered Feedforward Neural Network with one layer containing ten neurons

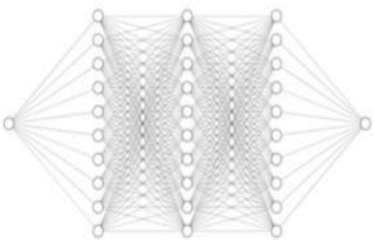


Fig 2.2 Multi-layered Feedforward Neural Network with three layers, each containing ten neurons

Dimensionality Reduction

Overview

In data analysis, the surge of high-dimensional data has presented both opportunities and challenges. As datasets grow in complexity, discerning their inherent structure becomes increasingly intricate, akin to discerning intricate patterns within a dense forest—moreover, the management, transmission, and computation of such voluminous data strain computational resources[4]. Consequently, there is a pressing need to condense the data's dimensionality while preserving its intrinsic characteristics.[8][9]

Principal Component Analysis (PCA)

Historically, numerous dimensionality reduction techniques have been introduced, with principal component analysis (PCA) being a notable method. PCA transforms the original data variables into a set of uncorrelated variables. These transformed variables, derived from linear combinations of the original set, offer a compact representation of the data. This condensed form facilitates efficient storage and transmission and enhances data modelling and pattern recognition capabilities. Though PCA has been a visualisation tool for high-dimensional datasets, its application was limited to datasets of specific dimensions due to computational constraints.

Principal Component Analysis (PCA) is primarily

approached in two ways. The matrix method uses the entire dataset to compute its variance-covariance structure, which is then decomposed to reveal principal data variances, often through techniques like singular value decomposition. On the other hand, the data method works directly with data points, adjusting principal component directions adaptively as new data is introduced. This method is efficient for real-time applications or high-dimensional data. While neural networks have been proposed for adaptive PCA, their effectiveness often hinges on the precise tuning of learning parameters.[8][2]

The Curse Of Dimensionality

As previously alluded to in this paper, the challenge of the "curse of dimensionality" is a significant concern in pattern recognition. To clarify this topic, the following is a detailed explanation of its implications and challenges. In pattern recognition, dealing with high-dimensional data spaces introduces a phenomenon known as the "curse of dimensionality." For instance, when examining measurements from a pipeline containing a mixture of substances, the data is represented in a 12-dimensional space. Visualizing just a fraction of these dimensions reveals the intricacies of classifying new data points based on their proximity to existing ones. A naive solution might involve segmenting the space into uniform cells and classifying based on the predominant class in each cell. However, as the dimensionality increases, the number of these cells grows exponentially. This exponential growth means that to populate these cells with data adequately, one would need an exponentially larger dataset. This rapid increase in required data points as dimensions grow highlights the challenges of the curse of dimensionality, emphasizing the need for more sophisticated classification techniques in high-dimensional spaces. [8]

Experimental Procedure

Overview of Experiment

In the following section, I will guide you through implementing my experiment. This will encompass a detailed exposition of how the training and testing datasets were procured, the specific machine learning models employed, and, ultimately, the results derived from the output. It is noteworthy to mention the paper previously mentioned in the introduction (Yuchun Lee, 1991), highlighting intriguing findings. Lee's research demonstrated that both the backpropagation algorithm and the K-Nearest Neighbors algorithm exhibited strikingly similar error rates, with values of approximately at 5.14% and 5.15%, respectively. This prior work provides a compelling backdrop against which our current findings can be compared.

Implementation of the Models

sklearn is utilized to implement the two algorithms and measure their accuracy using the mean squared error function. For visual representation of the results, matplotlib is employed. The dataset is sourced through keras, which also aids in the execution of the Feedforward Neural Network.

```
#modules and methods needed for a Comparative Performance Analysis Between
#K-Nearest Neighbors and Feed-Forward Neural Networks in Classifying
#Dimensionality-Reduced Image Data of Hand-Written Digits

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

from keras.models import Sequential
from keras.layers import Dense
from keras.datasets import mnist

from tensorflow.keras.utils import to_categorical
```

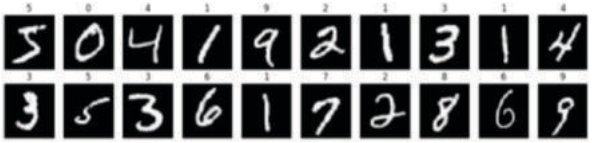
Loading in the MNIST Dataset of Images

The code begins by importing the training dataset into the "data" variable. Subsequently, each image from this dataset is transformed into a grid-like matrix, mirroring the structure of the 28x28 image. This grid representation ensures that every individual cell corresponds to a specific image pixel. The training set comprises 60,000 images, while a separate testing set introduces 10,000 images. Within this grid, each cell holds an integer value ranging from 0 to 255. Here, a value of 0 denotes black, 255 represents white, and any intermediate value between 0 and 255 captures varying shades of grey. The following lines of code shows information regarding the datasets.

```
Image shape (28, 28)
X_train shape (60000, 28, 28)
y_train shape (60000,)
X_test shape (10000, 28, 28)
y_test shape (10000,)
```

Sample Data Values Of Handwritten Digits

we cycle through a subset of 20 randomly selected dataset values. By plotting these values, we visually present a sample set of 20 hand-drawn digits, offering a glimpse into the nature of the data we are working with. Samples are shown below.



Creating A Flattened Image Dataset

Utilising the ".reshape()" method from keras, the matrix representations of the images, originally of size 28x28, are transformed into one-dimensional vectors with a length of 784 (which is the product of 28 multiplied by 28). This transformation is essential to facilitate the feeding of the dataset into the backpropagation neural network. The following lines of code shows information regarding the datasets.

```
After reshaping
Image shape (784,)
X_train shape (60000, 784)
y_train shape (60000,)
X_test shape (10000, 784)
y_test shape (10000,)
```

Application Of Dimensionality Reduction On The Image Dataset

We then utilise the Principal Component Analysis from the sklearn library to reduce our dataset to its 50 most vital components. After training the PCA model on our training set, it provides the variance explained by each component. These values, when visualised, highlight the importance of each component. The graph typically starts with higher values, indicating more significant components, and decreases to the right. This visualisation aids in understanding the essential components of the dataset. By applying PCA, we condensed the dataset's complexity for trainingand testing while retaining its fundamental information. The following lines of output show information regarding the datasets.

```
(60000, 50)
(10000, 50)
```

Testing the K-Nearest Neighbors Algorithm

The provided code lines detail the necessary libraries required to execute the KNN algorithm. These libraries facilitate the training, testing, and accuracy evaluation of the KNN model on the dimensionally reduced dataset. We use K = 7. The following lines of code train the dataset The following lines of code test the trained KNN algorithm

```
k = 7   #choose the number of neighbors
knn_model = KNeighborsClassifier(n_neighbors=k)   #creating the model
knn_model.fit(X_train_red, y_train_cat)   #fit ==> implementing the supervised training onn the reduced data
```

Testing the Feedforward Neural Network Algorithm

The provided code lines detail the necessary libraries required to execute the FFNN algorithm. These libraries facilitate the training, testing, and accuracy evaluation of the FFNN model on the dimensionally reduced dataset. The following lines of code train the dataset.

```
#build the model ...
model = Sequential()
model.add(Dense(10, input_dim = X_train_red.shape[1], activation= 'tanh'))
model.add(Dense(10, activation= 'tanh'))   #vary # of neurons
model.add(Dense(10, activation = 'softmax'))  #Softmax means output probability of each possible digit

#Loss function
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
#training the model
model.fit(X_train_red, y_train, epochs = 10, batch_size=100)
```

RESULTS

Feedforward Neural Network (%)	K-Nearest Neighbors (%)
91.96	97.39

The percentage accuracy rate of specified algorithms in identifying dimensionally reduced hand-drawn digits:

CONCLUSION

In the research titled "Comparative Performance Analysis Between K-Nearest Neighbors and Feed-Forward Neural Networks in Classifying Dimensionality-Reduced Image Data of Hand-Written Digits," an exploration was undertaken to validate the findings of Yuchun Lee from 1991. Lee's study indicated nearly identical error rates for the backpropagation algorithm and the k-nearest neighbours algorithm, with rates of approximately 5.14% and 5.15%, respectively, on standard data. Drawing from these findings, the hypothesis posited that the Feed Forward Neural Network (a backpropagation algorithm) and K-Nearest Neighbors would exhibit similar performance on dimensionally reduced data.

However, after rigorous experimentation using a training set of 60,000 images and a testing set of 10,000 images, the results presented a deviation from the initial hypothesis and Lee's findings. The performance metrics revealed distinct differences in accuracy between the two algorithms on dimensionally reduced data. This divergence underscores the importance of considering data characteristics, especially dimensionality when selecting an appropriate machine-learning model. It also highlights the evolving nature of algorithmic performance as data processing techniques advance.

Interestingly, the inherent simplicity of the K-Nearest Neighbors algorithm might have played a role in its comparative performance against the more complex FeedForward Neural Network. This observation suggests that, at times, simpler models can outperform or match more intricate architectures, especially when dealing with specific data types or structures.

Looking ahead, intrigue surrounds how this topic will evolve in future research. Understanding the nuances of these algorithms and their performance on various datasets can pave the way for more informed model selections in image classification tasks, and further testing and analysis should be verified to fine-tune results and improve accuracy. Such endeavours would enhance the understanding of why the K-Nearest Neighbors algorithm exhibited superior performance over the Feedforward Neural Network and guide future research in selecting and optimising algorithms for image classification tasks. The journey of discovery in this domain promises to be both challenging and enlightening. However, the research community is eager to see where it leads, and how it could be used to further innovate other domains of technology.

Acknowledgements

I extend my heartfelt gratitude to Professor Erik Sakk from Morgan State University for his invaluable guidance and mentorship throughout the development of my paper.

REFERENCES

1. Shrestha, A., & Mahmood, A. 2019. Review of Deep Learning Algorithms and Architectures.
2. Duda, R. O., Hart, P. E., & Stork, D. G. (2001). Pattern Classification (2nd ed.). Wiley-Interscience.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Adaptive et al. series). MIT Press.
4. Pattern Recognition and Machine Learning (Information et al.) by Christopher M. Bishop (Author)
6. Koutroumbas, K., & Theodoridis, S. (2008). Pattern Recognition (4th ed.). Academic Press.
7. Taunk, Kashvi & De, Sanjukta & Verma, Srishti & Swetapadma, Aleena. (2019). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification.
8. Sazlı, M. H. (2006). A brief review of feed-forward neural networks. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 50 (01), 0–0.
9. Partridge, Matthew and Calvo, Rafael A. 'Fast Dimensionality Reduction and Simple PCA'. 1 Jan. 1998 : 203 – 214.
10. Shereena, V. B., & David, J. M. (2015). Significance of dimensionality reduction in image processing. Signal & Image Processing: An International Journal (SIPIJ), 6(3), 27-42.
11. Y. Lee, "Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks," in Neural Computation, vol. 3, no. 3, pp. 440-449, Sept. 1991.
12. Fix, E. & Hodges, J.L. (1951). US Air Force School of Aviation Medicine. Unpublished report.