Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 9, Issue 32, pages 9-16, 2023

Convolutional Neural Network-Based Model for Handwritten Digit Recognition

Emmanuel E. Alechenu
Department of Computer Engineering.
The Federal University of Technology, Akure
Akure, Nigeria
ealechenu05@gmail.com

Ibukunoluwa A. Olajide
Department of Electrical and Electronics Eng.
The Federal University of Technology, Akure
Akure, Nigeria
iaadebanjo@futa.edu.ng

***Abstract* – The topic of transcribed digit acknowledgment has been an open one in the field of pattern recognition for a long time. Several studies have shown that Neural Networks perform exceptionally well in data organization. The main goal of this study is to provide effective and reliable approaches for recognizing transcribed numerical data by examining various existing arrangement model. The Convolutional Neural Network (CNN) from Machine Learning can be used to recognize handwritten digits. The core structure of this research development is based on the MNIST (Modified National Institute of Standards and Technologies) database and CNN compilation. So, in order to run the model, libraries like NumPy, Pandas, TensorFlow, and Keras were employed. These are the fundamental pillars that the design is built upon. MNIST has roughly 60,000 photos of handwritten numbers ranging from 0 to 9. As a result, it is a classification model of class 10. The dataset is split into two parts: training and testing with Image representation as a 28*28 matrix with grayscale pixels in each cell. The result display of Convolutional Neural Networks (CNN) is the subject of this paper, which shows that the CNN classifier outperformed the Neural Network in terms of computing efficiency without sacrificing execution time. Notably, the combination of pre-processing and CNN reached the highest recognition rate of 99.16% in the experiment.**

Keywords-component; CNN, MNIST, Machine Learning, Neural Networks

I. INTRODUCTION

Identification of numbers from which to extract the best distinguishing features is one of the most important tasks in the digit recognition system. To locate such regions, different types of region sampling techniques are used in pattern recognition [1]. The challenge in handwritten character recognition is mainly caused by the wide variation of individual writing styles [2]. Therefore, robust feature extraction is very important to improve the performance of a handwritten character recognition system. Nowadays, handwritten number recognition has received much concentration in the field of pattern recognition system for its application in various fields. In the coming days, the character recognition system could serve as a cornerstone to create a paperless environment by digitizing and processing existing paper documents.

Intelligent image analysis is an appealing research area in Artificial Intelligence and also crucial for a variety of present open research difficulties. Handwritten digits recognition is a well-researched subarea within the field that is concerned with learning models to distinguish pre-segmented handwritten digits [3]. The main application of machine learning methods over the last decade has produced decisive systems which are far improved than manually written classical artificial intelligence systems used in the beginnings of optical character recognition technology [4].

Machine Learning encourages various methods in which human efforts can be reduced in recognizing and extracting manually written digits. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or solve problems. Machine learning algorithms instead allows computers to train on data inputs and use statistical analysis in order to produce values that falls in specific ranges. In view of this machine learning helps computers in building models from sample data in order to automate decision-making processes based on data input. Deep Learning is a machine learning method that trains computers to do what is easy for humans to do. It does this by learning through various examples. With the use of deep learning methods, human efforts can be reduced in perceiving, learning, recognizing, predicting and in several other regions. Deep learning teaches the computer how to perform classification work from various sources such as images, videos or content of any document. Deep Learning models are guaranteed to produce very high accuracy beyond human-level performance. The digit recognition model makes use of large data sets to recognize digits from distinctive fonts. Recognition of handwriting characters has been around since the 1980s. The task of handwritten digit recognition, using a classifier, has an extraordinary significance and uses such as – online digit recognition on PC tablets, recognizing postal codes on mail, processing bank check amounts, numerical sections in hand-filled structures (e.g. - tax forms) and so on. There are several challenges in trying to solve this problem. The handwritten numbers do not always have the same size, thickness or orientation and position relative to the margins [5].

Handwritten numerical data sets are vague in nature because there are not always sharp, perfectly straight lines. The basic purpose of digit recognition is to extract attributes in order to reduce data redundancy and generate a more efficient representation of the word picture using a collection of numeric attributes. It entails sifting through raw visual data to find the most important information Also, unlike written characters, the curves are not always smooth. Furthermore, the data set can be recovered from characters of various formats and orientations, which must always be written in a vertical or direct position in a guide. Consequently, an effective handwriting recognition system can be developed with these limitations in mind. It is quite tiring to identify handwritten characters at times, as most people cannot even recognize their own written scripts. Therefore, there is a limitation for a writer to write apparently for the recognition of handwritten documents.

In this paper a convolutional neural network is used to recognize handwritten digits through a webcam which will recognize and extract handwritten digits. This provides effective and reliable approaches for recognizing transcribed numerical data by examining various existing arrangement models.

II. Literature Review

A. Machine Learning

Machine learning is a subset of artificial intelligence that enables machines to make decisions based on their past experiences, grow and learn over time, and use without being explicitly programmed. Machine learning is concerned with creating computer programs that can access data and use it to learn on their own. Machine learning algorithms are frequently divided into three categories: supervised, unsupervised, and reinforcement learning[6][7].

B. Supervised Learning

The search for algorithms that reason from externally supplied instances to develop broad hypotheses, which subsequently make predictions about future instances is known as supervised machine learning [8]. it can use labeled examples to apply what it has learnt in the past to fresh data in order to anticipate future events. supervised learning techniques include classification methods like as decision trees, Naive Bayes, Neural Networks, k-Nearest Neighbors, and Support Vector Machines (SVM).

C. Unsupervised Learning

Unsupervised learning is a sort of machine learning technique in which no defined or labeled classes are used and the system makes inferences from datasets on its own [9]. Unsupervised learning methods include clustering algorithms such as K-means, mixture models, hierarchical clustering, anomaly detection, Hebbian Learning Networks, Generative Adversarial Networks, and others.

D. Reinforcement Learning

Reinforcement learning is a machine learning training approach that rewards desirable behaviors while punishing those that are undesired. In general, a reinforcement learning agent can observe and grasp its environment, act, and learn through trial and error.

III. Methodology

Convolutional Neural network (CNN) is a deep learning algorithm widely used for image recognition and classification, and it requires minimal pre-processing. It inputs the image in small chunks rather than a single pixel at a time, allowing the network to more efficiently detect uncertain patterns (edges) in the image.

The input image was taken from a webcam and after capture it was pre-processed (normalized) and then segmented to show how many numbers are written. Extraction to the CNN model for evaluation was carried out and the result is taken as the model prediction which is then stored on a spreadsheet. Figure 1 provides the stepwise design flow. The stages of implementation are image input, Pre-processing, segmentation, feature extraction, training and evaluation, then storage.

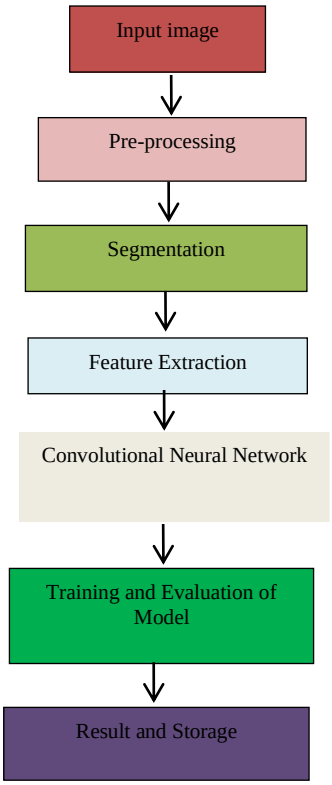

Figure 1: Design work flow

The handwritten digit recognition model was prepared via Convolution Neural Networks, using the 'Tensorflow' module from version 2.3.0 and all pre-processing was performed via 'Keras' version 2.4.3.

A. Segregation

Initially, all the necessary modules needed to build the model were imported. Keras provides many predefined datasets, among which the most important

part is importing the dataset used in the model, i.e. the "mnist" dataset of the handwritten numbers. A segregation of the imported dataset into training and test datasets for evaluation was done.
Random images (between 0 and 9) of the first 5 images in the dataset is provided in figure 2 to get an idea of what the images look like.

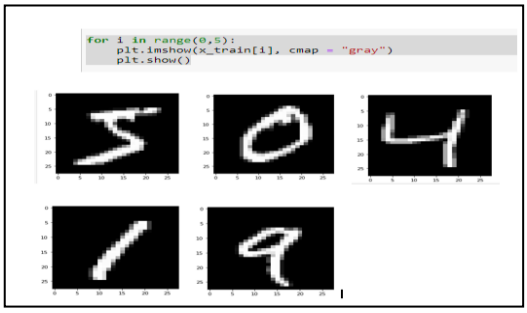

Figure 2: Plot showing first 5 images in the dataset

B. Reshaping of Data

The input data were reshaped to ensure complete uniformity before training the model. The image size I considered as 28x28x1, which means that the images are square with the same height and width and the 1 indicates that they are grayscale images.

C. Model Building

The Keras model type used to build the model is 'Sequential'. It helps to build the model, one layer after another, using the add () function to add the layers one by one. 'Sequential' can be imported using the " keras.models import Sequential"
The first two layers of the model are the convolution layers. Because the images to be shared are in the form of 2D matrices, the layers are 'Conv2D' layers. In the first layer there are 32 nodes and the second layer has 64 nodes, chosen according to the model. The size of the kernel used was set to 3, which means that a 3x3 filter matrix was used for convolution. The activation function used is ReLU. This function returns a 0 if the input is negative or returns the same values, that is, ReLU -> max (0, input).

After defining the Sequential () as a model, all the required layers of the CNN structure were added using the 'model.add ()' function. The first layer includes the two-dimensional convolution layer with core size 3 and activation function as 'relu'.
The input shape is (28,28,1), which means that each image has a pixel size of 28x28 and the '1' defines the image in grayscale.

In the following layers have 2x2 grouping layer with the same fill. This layer helped to extract key features from the convolution layer. This is followed by the second convolution layer, where 64 entities were extracted with the same kernel size of 3 and the same padding, followed by another layer of grouping. Table 1 provides the parameters outlay.

In other to avoid over-fitting, Drop-out regularization technique was used. During dropout, random neurons are randomly selected and turned off. This helps in reducing learning from the given data and makes the model more suitable for unseen data. If the data learning is done strictly, there is a possibility that the model will not be robust and will perform miserably when subjected to a new data set that is largely different from the training set.

The dropout layer is not only applied in the lower levels of the model, but can also be used in the intermediate levels to help build a highly regularized model. The contribution of the "failed" neurons is less or completely removed during the forward passing of the weights.

A flattened layer was added between the convolution layers and the dense layer to convert the data into a single-dimensional vector form and acts as a link between the layers. The flat layer was followed by another dropout layer and then finally the dense layer.

The dense layer is a fully connected layer that is used for the output of the deep learning model. Here each node is connected to each other and therefore the most important learning takes place in this layer before the output, as observed in the network architecture diagram in Fig. 3.

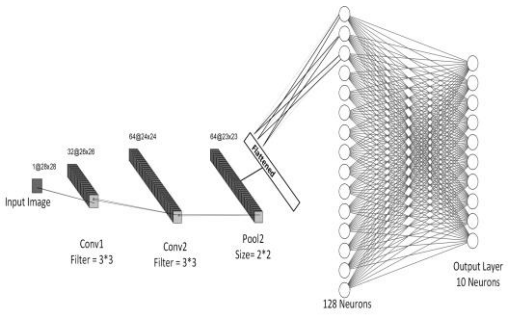

Figure 3: The Network Architecture

TABLE 1: THE NETWORK PARAMETERS

No	Layer	Output Size	Filter Size	Stride Size	Drop out
1	Input	28x28x1	-	-	-
2	Convolution 1	26x26x32	-		-
3	Relu	26x26x32	-	1x1	-
4	Convolution 1	24x24x64	-	-	-
5	Relu	24x24x64	-	-	-
6	MaxPooling	23x23x64	-	1x1	0.25
7	Flatten	33856	-	-	-
4	Dense	128	-	-	0.5
5	Softmax	10	-	-	-

D. Compilation

There is a trigger function in the dense layer called Softmax, which was used to get the output in the form of probabilities ranging from 0 to 1. Therefore, while producing an output, the algorithm tries to match the maximum probability with the input data and thereby gives the prediction. Next, the models were compiled together using the parameters "optimizer" set to *keras.optimizers.Adadelta()*, because it is a very good optimizer that uses a stochastic gradient descent method based on adaptive learning speed per dimension to the continuous decrease in learning speed during training and the need for manually selected global learning speed (shown in Fig.4).

The Loss function was conceived as "categorical_crossentropy" which is one of the most commonly used functions for classification problems. Metrics was chosen as "accuracy" to check the validation accuracy while training the model.

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

Figure 4: Compiling the Model

E. *Training*

After compiling, the training of the model with all the characteristics that were specified was done. The model was trained on the x_train, y_train data frames and validated on the x_test and y_test data frames.

IV. Image Processing

While the video was opened, the image was initially converted to grayscale to allow the frame for image processing. As with all other signals, images can also contain various types of noise, mainly due to the source (camera sensor). The Gaussian Filter was used because it has the properties of not overshooting with a step function input while minimizing the rise and fall time. In terms of image processing, any sharp edges in images are smoothed out while too much blur is minimized. Thresholding is a technique in OpenCV, the assignment of pixel values in relation to the specified threshold value. Threshold value compares each pixel value to the threshold value. If the pixel value is less than the threshold, it will be set to 0, otherwise it will be set to a maximum value (usually 255). Thresholding is a very popular segmentation technique used to separate an object considered foreground from the background.

A. *Noise Elimination*

To eliminate noise, the kernel, dilation and canny edge detection methods were used. Kernels in computer vision are matrices, which are used to make some kind of convolution in a given data. In dilation, a pixel element is '1' if at least one pixel below the kernel is '1'. Canny Edge Detection is used to detect the edges in an image. The code is provided in Fig. 5.

```
kernel = np.ones((5, 5), np.uint8)
dilation = cv2.dilate(thresh, kernel, iterations=1)
cv2.imshow("Frame dilation", thresh)

edged = cv2.Canny(dilation, 50, 250)
cv2.imshow("Frame edged", thresh)
```

Figure 5: Noise Elimination

B. *Segmentation and Prediction*

Contour is derived when all the points at the boundary of an object are joined. Typically, a specific outline refers to boundary pixels that have the same color and intensity. OpenCV makes it really easy to find and draw contour on images. The CHAIN_APPROX_SIMPLE algorithm compresses horizontal, vertical, and diagonal segments along the contour, leaving only their end point. This means that any of the points along the straight paths will be discarded and will be left with only the end points.

The processed image will be fed to the model for prediction and an in test will be added to the frame to enable user to see the predicted output before saving. This can be observed in Fig. 6.

```
        new_img = thresh[y:y + h, x:x + w]
        new_img2 = cv2.resize(new_img, (28, 28))
        im2arr = np.array(new_img2)
        im2arr = im2arr.reshape(1, 28, 28, 1)
        y_pred = model.predict(im2arr)

        num, per = get_numbers(y_pred)
        num_list.append(str(int(num)))
        num_str = '[' + str(str(int(num))) + ']'
        cv2.putText(img2, num_str, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 2, cv2.LINE_AA)

str1 = ' '.join(num_list)
if (str1 != ''):
    y_p = str('Predicted Value is ' + str(str1))
    print(y_p)
    cv2.putText(img2, y_p, (30, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
cv2.imshow("Capture Frame", img2)
cv2.imshow("Contours Frame", thresh)
```

Figure 6: Prediction Code

V. Results and Discussion

A. *Training Analysis*

Analysis after training and testing using the datasets showed an accuracy of the model after 12 epoch to be 0.9916 that is 99% and error of 2%. The analysis after each epoch is provided in Table 2.

Table 2: The Loss and Accuracy Values

Epoch Number	Value Loss (%)	Value Accuracy (%)
1	6.18	98.13
2	4.38	98.14
3	4.51	98.87
4	3.22	98.82
5	3.25	98.91
6	3.16	98.88
7	3.03	98.97
8	2.69	99.10
9	2.74	99.07
10	3.05	99.00
11	2.61	99.15
12	2.57	99.16

Fig. 7 displays the graphical analysis of the accuracy and each training step taken, the accuracy increases after each training step, the convolutional neural network produced a very high accuracy after the first training iteration and increases steadily afterwards.

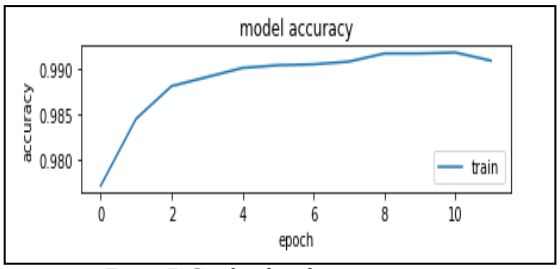

Figure 7: Graphical analysis on accuracy

The graphical analysis of the loss after each training step taken is observed in Fig. 8. The loss decreases after each training step, the convolutional neural network produced a very low error rate after the first training iteration and decreases steadily afterwards.

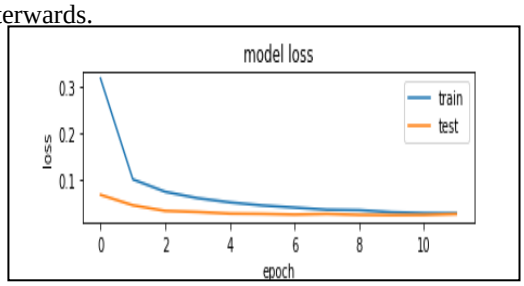

Figure 8: Graphical analysis on model loss (error)

B. Results

1) Single digit test

The model was tested on a single written digit to check the predicted output, the digit was written boldly with a marker on a plain piece of paper as shown in Fig. 9. Fig. 10 shows the processed image sample of the digit. The predicted output of the sample is shown in Fig. 11. The model's prediction was accurate.

Figure 9: Original picture captured via webcam a single digit

Figure 10: The processed image of the single digit sample

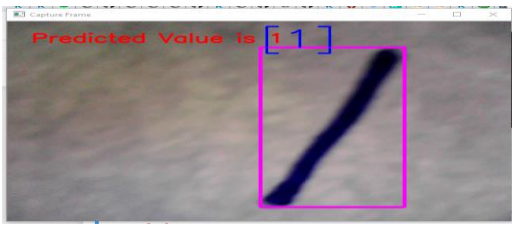

Figure 11: Predicted output of single digit sample

2) Double Digit Test

The model was tested with double digits to check the performance, using a white plain sheet of paper and written boldly with a marker. The model prediction was accurate as observed in figures 12 to Fig. 14.

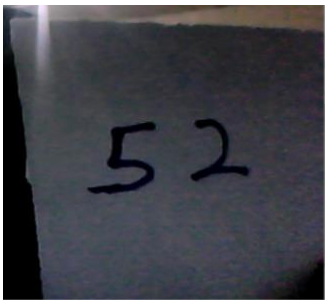

Figure 12: Original picture captured via webcam of double digit

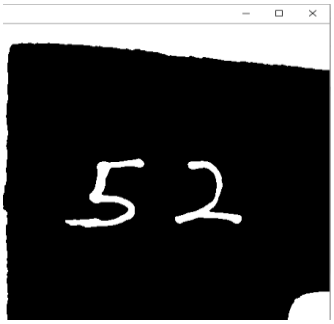

Figure 13: Processed image of the double digit sample

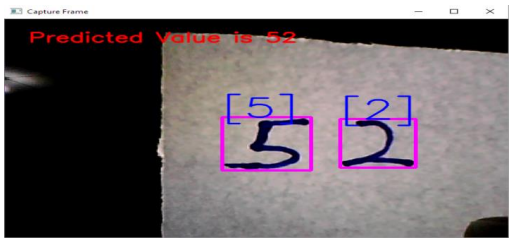

Figure 14: Predicted output of the double digit sample

3) Multiple digit test

Next the model is tested with multiple digit to check the performance, still using a white plain sheet of paper and written boldly with a marker. The model output a correct prediction. Fig. 15 gives the original picture of the written digits, Fig.16 shows the processed image, while Fig. 17 shows the predicted output.

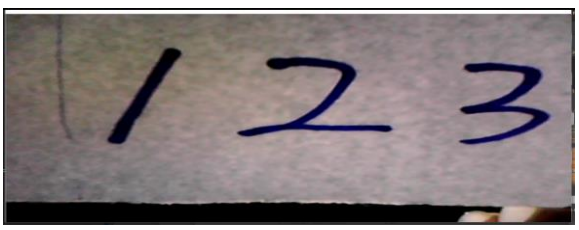

Figure 15: Original picture captured via webcam of multiple digits

Figure 16: Processed image of the Multiple digit sample

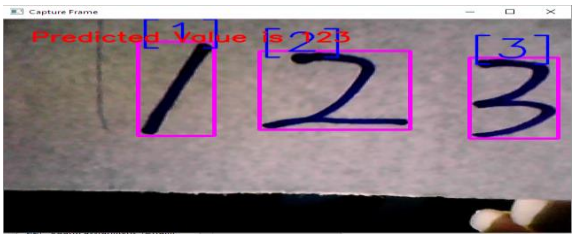

Figure 17: Predicted output of the multiple digit sample

4) Faint handwriting

Next the model is tested with multiple digits to check the performance, using a white plain sheet of paper and written with a pencil (HB). The model could not recognize the input image and there no output (Fig. 18 to Fig. 20).

Figure 18: Original picture captured via webcam of faint digits sample I

Figure 19: Processed image of faint digits sample I

Figure 20: No predicted output picture of faint digits sample I

Next the model was tested with multiple digits to check the performance, still using a white sheet of paper and written faintly with a pen (red). The model prediction was not accurate, as observed in Fig. 21 to Fig. 23.

Figure 21: original picture captured via webcam of faint digits sample II

Figure 22: Processed image of faint digits sample II

Figure 23: Wrong predicted output of faint digits sample II

VI. CONCLUSION

Handwritten digits recognition is a well-researched subarea within the field that is concerned with learning models to distinguish pre-segmented handwritten digits. Machine learning algorithms allow computers to train on data inputs and use statistical analysis.

The Handwritten digit recognition is a challenging topic because of different writing styles and forms. Also, the collected dataset is trained using an efficient CNN model that represents the current state of affairs for various applications. Therefore, the model was extensively analyzed by carefully selecting their parameters and demonstrating its robustness for dealing with our dataset. The result shows that the maximum accuracy 99.16% was obtained in MNIST dataset using deep learning technique and an appropriate learning rate at 12 EPOCHS and error of 2% was obtained. Each modification produced changes in the results in mostly improved accuracy and widely varying performance times.

The Model was designed to show the processed image in black and white showing how sharp the numbers. The major problem encountered during this implementation is finding the right environment for capturing the images because capturing the image in bad lightning conditions affects the accuracy of prediction.

Aside this, it was observed that once the handwriting is lightly written, there will not be a detection by the algorithm, thereby, bringing out no output

REFERENCES

[1] Das, N., Sarkar, R., Basu, S., Kundu, M., Nasipuri, M., & Basu, D. K. A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application. Applied Soft Computing, 12(5), 1592-1606, 2012.

[2] Plamondon, R., & Srihari, S. N. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), 63-84.

[3] Shamim, S.; Miah, M.B.A.; Angona Sarker, M.R.; Al Jobair, A.: Handwritten digit recognition using machine learning algorithms. Global J. Comput. Sci. Technol. 18(1), 1–8 (2018).

[4] Seewald, A. K. (2011). On the brittleness of handwritten digit recognition models. ISRN Machine Vision, 2012.

[5] Vijayalaxmi R Rudraswamimath, and Bhavanishankar K , Handwritten Digit Recognition using CNN , International Journal of Innovative Science and Research Technology, Volume 4, Issue 6, June – 2019

[6] Saleh Albahli, Marriam Nawaz , Ali Javed, and Aun Irtaza, An improved faster-RCNN model for handwritten character recognition, Arabian Journal for Science and Engineering march 2021.

[7] Ritik Dixit ,Rishika Kushwah and Samay Pashine , Handwritten Digit Recognition using Machine and Deep Learning Algorithms, International Journal of Computer Applications (0975 – 8887) Volume 176 – No. 42, July 2020.

[8] Dayan, P. "Unsupervised Learning," The MIT Encyclopedia of the Cognitive Sciences, Massachusetts, 1999.

[9] Kotsiantis,S.B, "Supervised Machine Learning: A Review of Classification Techniques," Department of Computer Science and Technology, University of Peloponnese, Peloponnese, Greece, 2007.