A Novel Approach Based Design of Neural Network Classification Algorithms for Pattern Recognition

Madhumita Panda[1], Dillip Narayan Sahu [2*]

[1]Associate Professor, Department of Computer Science, Gangadhar Meher University, Odisha, India.

[2*]Lecturer, Department of MCA, Gangadhar Meher University, Odisha, India.

Abstract

One of the key methods utilised in the area of artificial intelligence is pattern recognition. The most sophisticated senses in the human body, vision and perception, served as the inspiration for the computer paradigm known as neural networks. All real-time applications revolve on pattern recognition. Human decision-making is influenced by our ability to recognise patterns. Neural network classifiers are created for pattern recognition applications in this research effort. For the purpose of evaluating the benefits and drawbacks of the suggested Neural Network Classifiers, the job of handwritten character recognition has been selected as the application domain. With the aim of achieving rapid training, the standard Backpropagation method for Feedforward Neural Networks is changed. The proposed adaptive backpropagation method dynamically modifies the learning constants based on adaption cycles and error. When compared to traditional methods, the error is observed to reduce more quickly. The categorization of the Iris data set and the identification of handwritten characters serve as proof of the algorithm's effectiveness. Learning and generalisation are greatly influenced by the network size. The ideal network size is chosen using adaptive genetic algorithms. Depending on how sensitive the edges are to mistake, duplicated connections that contribute little to nothing are cut.

Keywords: *Neural Network, Machine Intelligence, Pattern Recognition, Conventional Algorithms and Backpropagation*.

DOI Number: 10.14704/nq.2022.20.9.NQ44539 **Neuro Quantology 2022; 20(9):4663-4675**

1. Introduction

Recent research has focused on understanding natural intelligence and using that knowledge to construct artificial intelligence, which is both a fascinating and difficult effort. Engineers may learn a lot from the difficulties of studying biological processes in order to create new mathematical techniques and scientific instruments[1]. Vision, learning, smell, touch, and taste are examples of natural sensory phenomena that should be aggressively studied and analysed in order to create smart sensors and new, intelligent systems. This research and the desire to create robots that imitate human cognitive functions are what first sparked interest in neural networks (NNs). Known also as "Parallel Distributed Processing" and "Connectionist," neural networks have gained popularity in the disciplines of cognitive science, neurobiology, computer science, signal processing, optics, and physics because to its high performance. The network's ability for categorization and generalisation is a key component of hierarchical problem solving[2]. Artificial intelligence can also successfully do these tasks if classification criteria and generalisation rules are known, while NNs can learn from examples and generalise even when the rules are unknown [3]. Other fascinating benefits of NNs include parallelization, VLSI implementability, fault tolerance, evidential response, contextual information processing, and adaptability. The ability to see patterns has some bearing on how humans make decisions [4]. As a result, the majority of NN applications are in the field of pattern recognition. Neural networks have their origins in early 1940s multidisciplinary history and were first developed by Warren McCulloch and Walter Pitts. Following this research, several advances were made to explore the nervous system, ranging from basic perceptron to large neural networks [5]. Feedforward, feedback, and self-organizing networks are the three basic categories under which artificial neural networks (ANNs) are categorised. The three categories of learning algorithms are supervised, unsupervised, and reinforcement learning [6]. Applying NNs to information processing in industry that is determined to be too costly with traditional systems is an ongoing area of research. Getting a technology to detect human voice, handwriting, and signatures has proven to be a major issue in this field. An further crucial field for NN research is image processing [7]. The capacity of NNs to define intricate decision boundaries between clusters results in improved classification performance. A examination of the literature reveals that traditional NNs have a number of problems, including long training durations, becoming stuck at local minima, and sometimes being unstable during learning adaption [8]. It has been discovered that neural networks are effective at handling erroneous input and poorly specified tasks. However, they are unable to engage in subjective processes like perception and thinking. It has been discovered that fuzzy logic may simulate human thinking skills and be used for knowledge-based tasks. Additionally, it offers the ability to mathematically represent the uncertainties linked to human cognitive processes like thinking and reasoning [9]. A lot of effort has been put into combining fuzzy logic with neural networks to create the new discipline of "neuro-fuzzy systems." Neuro-Fuzzy Systems combine the fuzzy information processing capabilities of fuzzy logic with the learning capabilities of NNs.

The weights provided to the connectivity connections and the network topology of the neural network models serve as the decision boundaries. Natural selection and genetics are the foundation of genetic algorithms (GA). They mimic the biological process in which the more physically fit people are more likely to prevail in a competitive setting [10]. The genetic processes used to produce new children at every generation replicate the process of inheritance of genes. In order to produce a population from generation to generation, the evolution operation replicates the Darwinian evolution process. These axe algorithms are excellent at quickly identifying the best answer since they do a global search.

Based on their measurable attributes or features derived from these properties, complex patterns or objects are automatically classified or clustered by pattern recognition algorithms. NNs are reliable classifiers that may be used to hardware that runs in parallel. The adaptive, non-parametric NN classifiers perform admirably for a wide range of real-world issues, including voice, vision, robotics, and underlying distribution are accurate models of test data and a sufficient number of artificial intelligence applications [11-12]. Comparing these classifiers to traditional Bayesian methods, it is discovered that they provide lower error rates. According to [13-14], NNs are excellent at identifying the pattern's underlying regularities. The study of NNs and pattern recognition are related in this way. Even if it is not explicitly a pattern recognition issue, each application of NN may be seen as pattern recognition since it depends on the network's capacity to find and recognise underlying regularities in the input. The most adaptable, effective, and versatile pattern recognizers are those found in biological systems. The NNs with biological inspiration attempt to imitate this behaviour. The study in the traditional field of statistical pattern recognition has been revived because to the current revival of interest in NNs, leading to the development of new and potent approaches. The decision regions are formed differently by the various kinds of classifiers. The four main categories of classifiers are Probabilistic classifiers, Hyperplane classifiers, Kernel classifiers, and Exemplar classifiers [15]. Figure 1 shows how the decision areas of each of these classifiers are created. Statistical classifiers These classifiers are traditional Bayesian classifiers, which take into account an initial probability distribution for the input characteristics, such as a Gaussian or Gaussian mixed distribution [16,17]. When training data is provided for precise estimate of distribution parameters, these classifiers function at their best.

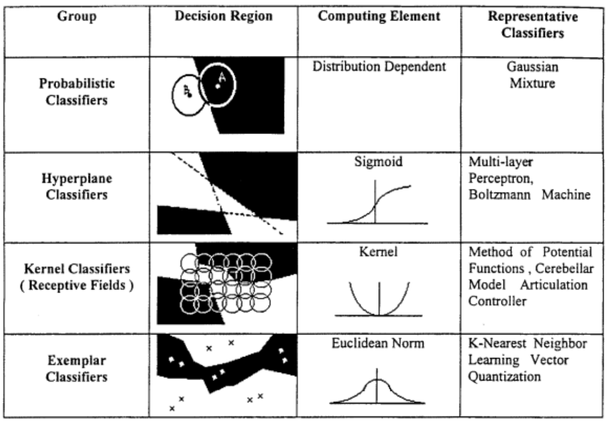


Figure.1. Types of classifiers

Using nodes that create hyperplane decision boundaries in the input space, hyperplane classifiers create complicated decision regions. The weighted total of the inputs is subjected to nonlinearity (such as a sigmoid function) by the computing nodes. Although these classifiers have low memory and processing needs, they have limitations such as lengthy training periods and complicated learning algorithms. Boltzmann Machines [19], Multilayer Perceptron (MLP) networks using the "Group Method of Data Handling" (GMDH) algorithm [18], and high order networks are a few examples of this class.

They are also known as classifiers of receptive fields. Using kernel-function nodes that create overlapping receptive fields, they create sophisticated decision regions. These are nodes that have maximum output when input is placed close to the node's centroid. The weighted total of the outputs from the kernel-function nodes is used by the high-level nodes to determine the classification. Traditional classifiers that calculate the probability density function using mixed distributions are examples of kernel classifiers [20]. Also included are classifiers based on the Cerebellar Model Articulation Controller (CMAC) [21]. Based on how closely the input and cluster centroid resemble one other, this sort of classifier does categorization. Based on the Euclidean distance between the input patterns and the cluster centroid, similarity is calculated. These classifiers learn quickly, albeit at the expense of high memory and processing demands. Adaptive Resonance Theory (ART) classifiers [23], Learning Vector Quantization (LVQ) classifiers [22], and Restricted Coulomb Energy (RCE) classifiers are examples of classifiers.

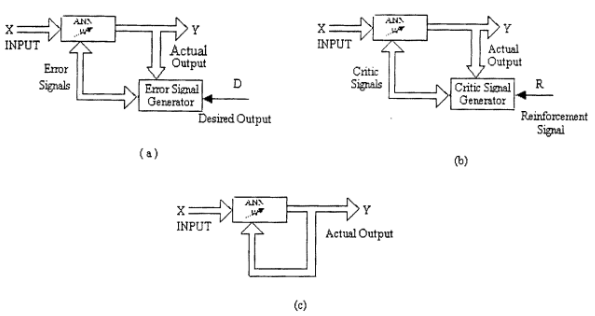
Through the categorization of Iris data and the identification of handwritten samples, the performance of the suggested models is proven. In many cases, the reference data set for pattern recognition is iris data. It comprises of 150 four-dimensional data points that reflect the petal, sepal, and sepal width lengths and widths of three different iris species: Iris sestosa, Iris versicolor, and Iris virginica. Additionally, 26-class English characters and 10-class handwritten I numerals are evaluated on the models. 4665

After the first adaption cycles, it has been discovered that using an alternative goal function speeds up convergence in feedforward networks. This requirement typically makes algorithms more computationally demanding while also making their applications more challenging. These algorithms were discovered to rely on an excessive number of variables. There have been efforts to update weights using the Kalman filter model. This is a recursive method that has been used for signal processing and system identification in control issues to estimate a system's state from a noisy environment. When compared to gradient descent approaches, this model accelerates network convergence at the expense of an increase in algorithm complexity.

2. Literature Survey

Even if learning happens without overt behaviour, its presence can only be deduced by observing the changes in behaviour. Non-associative and associative learning are typically categorised by psychologists into two categories. Non-associative learning involves exposing the organism to the stimulus and forcing it to learn about the stimulus's characteristics. In associative learning, the organism discovers the connection between a stimulus and its behaviour. The act of collecting information and encoding it in the synaptic weights is known as learning in neural networks (NNs). Three categories may be made of it: supervised, reinforced, and unsupervised (self-organized). Learning is classified as supervised or unsupervised depending on whether the network's goal output is present or absent [24]. Reward learning is learning using critical information, such as a "too high" output. In Fig.2, these categories are shown in pictures.

F



igure.2. Learning Techniques (a) Supervised (b) Reinforcement (c) Unsupendsed Learning

Application of a pattern classification technique is required for handwritten number recognition, which maps the input picture to an output space composed of the 10 potential numeral classes (0-9). We offer a unique method that makes use of a tailored CNN architecture to recognise handwritten Kannada numbers. A ten node terminal layer is used to infer the class value, and a softmax activation function is used to determine the prediction output. The advantages of the proposed CCNN architecture over other conventional approaches include: in-depth feature extraction that concentrates on both edges and the overall shape of the digit, accurate prediction of specific digits based on a softmax activation coupled with pertinent weight vectors derived using a deep architecture. The Dig-MNIST dataset [25] and our own custom-created dataset are used to evaluate the suggested architecture, and it is discovered that it outperforms the conventional approaches.

One of the key components of the machine learning field is pattern categorization. This area encompasses a broad range of real-world issues. The process of pattern categorization entails a thorough investigation and review of the provided dataset [26]. The dataset is used to generate the dataset's most important attributes. The precise identification of a pattern among these characteristics is subsequently achieved by use of certain algorithms. Correctly detecting these patterns, which may be used to a number of activities, is the primary goal of pattern categorization. Pattern categorization has emerged as a key area within the artificial intelligence research with the introduction of new algorithms and an increase in data storage and processing capability. For a long time, researchers in the machine learning field have focused on pattern classification in order to develop new algorithms and enhance current structures for improved performance. Multiple methods may be used to classify patterns: Unsupervised, semi-supervised, and supervised activities A labelled dataset—one that has an output label—is necessary for supervised pattern classification [27]. The next step is to identify similarities between input data characteristics and output labels, using these patterns to reliably predict the output label for a fresh set of test data.

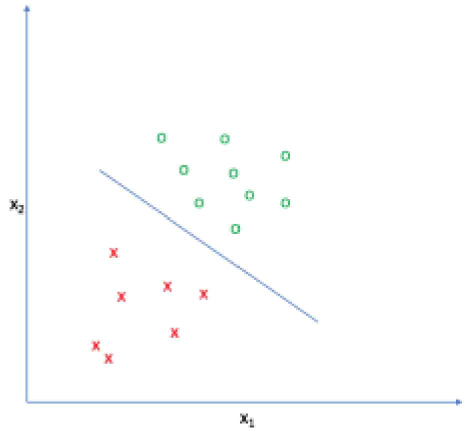


Figure.3. Supervised Pattern Classification

The feature representation of existing data, as shown in the image, enables us to construct a decision boundary by classifying each sample according to the labels that are already in use [5]. A new test case input is given the representation label depending on where it stands in relation to the determined decision boundary. Clustering of supplied data based on patterns discovered among the given characteristics is known as unsupervised pattern classification [28]. The main goal in this kind is to separate the data into different clusters or areas; there is no input to output mapping [6]. Test inputs are assigned to the patterns with which they correlate most strongly.

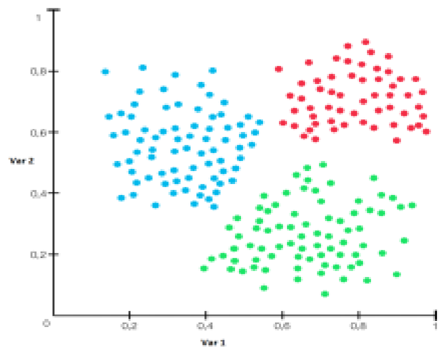


Figure.4. Unsupervised Pattern Classification

As can be observed, groups of data samples are formed based on comparable patterns in the values of the input characteristics rather than any division of data samples or decision boundary based on output labels. When just a portion of the data is labelled, semi-supervised pattern clustering is used [29]. Unlabelled data aids in a better grouping of data samples across the calculated border whereas labelled data is utilised to construct the decision boundary. We concentrate on supervised pattern categorization among these subgroups. Pattern matching and pattern detection are this sort of classification's two primary stages. Pattern matching is the testing phase, while pattern detection is the training phase. There has been a lot of study in this area, and several algorithms have been developed, including SVM, logistic regression, artificial neural networks, KNN, and others [30]. Each of these methods has certain benefits and drawbacks, making it suitable for use in particular contexts.

A basic supervised classification method, it forecasts the likelihood of the output variable given a certain class label. The output, which is given a value between 0 and 1 such that the total of all such probabilities equals 1, forecasts the likelihood that the input case belongs to a certain class. An inverted exponential function known as the sigmoid function is utilised to represent the binary dependence and output this probability. By feeding the values of the independent variables, or the inputs, to the logistic function, the likelihood of the dependent variable, or the output, is anticipated.

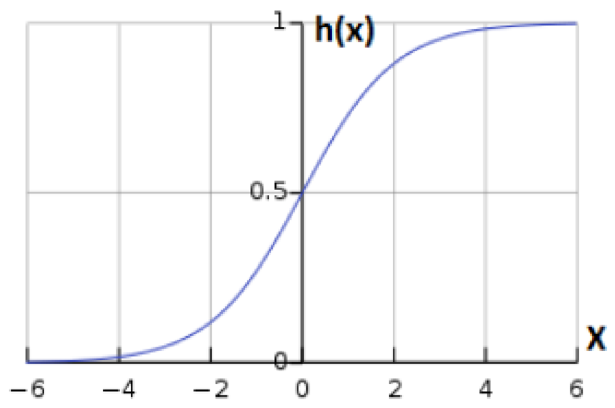


Figure .5. Distribution of Logistic Function Curve

As observed above, the predictions are constrained by the Logistic func's range of values, which is between zero and one. As can be seen, the value of y is more than 0.5 for positive values of h(x), and it is precisely 0.5 for zero values of h(x). The value of y is smaller than 0.5 in all other situations.

There are two different forms of logistic regression: multinomial and binomial. When the output variable may have three or more different values, the later is utilised. When there are just two possible outcomes, binary regression is utilised [27]. Predicting whether a certain email is spam or not is a typical example of binary logistic regression. In this scenario, the corresponding class labels are typically assigned the values 0 and 1, respectively. With a continuous character to the prediction, logistic regression is sometimes described as the constrained variation of the linear regression process. It should be highlighted that logistic regression predicts probabilities of certain outcomes rather than output values.

The input characteristics are grouped in a linear fashion using LDA [28] so that they may establish a dividing or deciding line between the output class labels or values. LDA has two applications: it may be used to solve classification issues, or it can be used to reduce the input dimensionality before sending the results to another classification model. Regression modelling and analysis of variance (ANOVA) [29], in which the output variable is stated as a mixture of many input factors, serve as the foundation for LDA. Although the output variable in LDA will have a categorical value, the independent variables may have continuous values. LDA presupposes multivariate normality as well as a normal distribution of probability density functions.

This method of categorization is linear in nature and non-probabilistic. The class separation issue is solved using support vector machines, which require generating a hyperplane. The optimal hyperplane is the one that can keep the two class labels at their greatest distance from one another [30]. Therefore, the goal of building an SVM model is to maximise the distance between the closest element of the class and the hyperplane. The maximum-margin hyperplane of optimum stability is the hyperplane that is ultimately attained after many iterations of the model. SVMs are helpful for modelling non-linear data using certain kernel processing techniques, in addition to determining linear decision boundaries. Artificial Neural Network (ANN): This method or architectural style is the fundamental component of deep learning and involves stacking several units that resemble logistic regression to create a deep structure [9]. The general design of these units, which are each referred to as neurons, is similar to that of the human nervous system. The nodes of one layer are mapped to the nodes of the one beneath it and the one above it. The structure of an ANN is shown below:

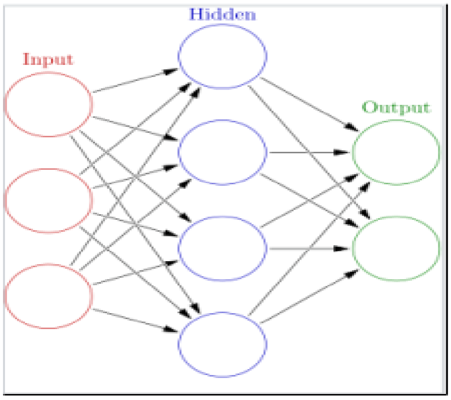


Figure.6.Sample ANN Architecture

Using propagation functions, which determine this value as a weighted total, the inputs of one layer are concatenated to get the value at the following layer. The implementation of backpropagation [3] is used for weight updates and error correction. An ANN structure may have many hidden levels. As a consequence, while the structure and training are more complicated, the feature extraction capacity is richer [9, 3]. In comparison to the earlier techniques, more computing power is needed for training these networks.

For training image-based data, particularly with high-resolution photos, a highly heavy neural architecture is needed even if ANNs are superior at extracting the relevant features. This is because each node can only keep one data value. Therefore, computer vision-based solutions were first the method of choice for solving image-based challenges. However, Alex Krizhovsky et al .'s 2012 achievement of the top result on the ImageNet challenge showed the value of utilising convolutional neural networks for picture data. Since then, feature extraction and modelling for image data have been done mostly using convolutional neural network-based designs.

2.1 CNN for Pattern Classification Tasks

Despite operating at an abstract level, CNN architecture is more efficient for feature extraction in pictures. In order to go through the picture and conduct convolution operations for each point to compute the values for the next layer, CNNs employ filters or kernels of a certain size [23]. Together, many of these filters are utilised, each of which extracts a different property. A representative iteration of the convolution technique is shown in Fig. 7.

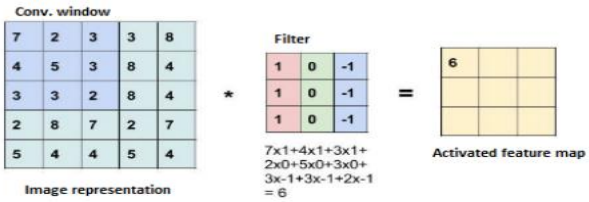


Figure .7. A Sample Convolution Operation

The filter settings are dynamically changed during the training period rather than being fixed [8]. A filter map is created after the data has been passed through such a convolution filter, and it is then altered or "activated." An activation function converts the concatenated weight total value to a defined range and determines whether the specific features are present in the map or not. The Rectified Linear Unit activation function, or ReLU, is used to address nonlinearity because it is more effective than the sigmoid function in solving the vanishing gradient issue. To create a CNN, many similar convolution filters are combined with additional layers. In Fig. 8, an example CNN architecture is shown. Convolutional, pooling, and fully connected layers are the three primary kinds of layers that make up a CNN.

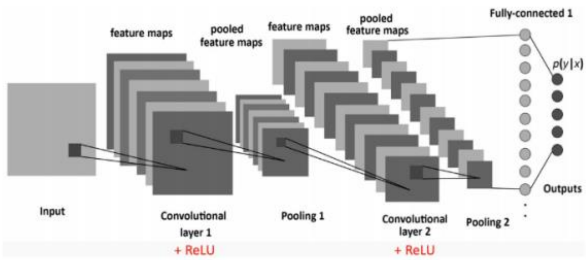


Figure.8. A Sample Convolutional Neural Network

Convolutional neural networks are able to extract detailed information from each layer, which aids in their ability to identify the proper patterns required for classification tasks. The convolution layers comprehend and pick up on characteristics that are more sophisticated as we continue to go deeper into the network. Within a deep convolutional network, an input picture propagates in the following order: The main or first-order feature map is extracted from the input picture using a first convolutional layer. Then, using a pooling layer, the most important features from regular subparts of this feature map are chosen. This relevance may be determined by looking at the minimum, maximum, or average values of pixels inside the segment. Then, using a pooling layer, the most important features from regular subparts of this feature map are chosen. This relevance may be determined by looking at the minimum, maximum, or average values of pixels inside the segment. A second convolutional layer and a subsequent pooling layer receive the now-pooled feature map. In the network, a number of these convolutional and pooling layers may be layered together to extract increasingly complex characteristics from the picture. Until the desired degree of rich features is not reached – neither too little nor too much – this procedure is iterated upon (leading to overfitting).

3. Dataset Distribution and Evaluation Methods

As previously mentioned, many datasets, including the DigMNIST dataset and our custom created dataset, are used to evaluate our technique. Of these, 3240 photos are used for the Dig-MNIST dataset. These are divided into 2000 photos, with 1600 images used for training and 400 images used for validation, with an internal split of 4:1. The remaining 1240 photos are utilised for testing. We utilise 10,000 photos from our own dataset for training and validation in the same ratio as before. 2000 photos serve as the validation data, while 8000 images are utilised for training. The testing dataset is composed of the remaining 5000 photos. It should be emphasised that we use a k-fold validation technique for both datasets as usual procedure, which means that in at least one training cycle, all photos from the non-testing set are included in the validation set.

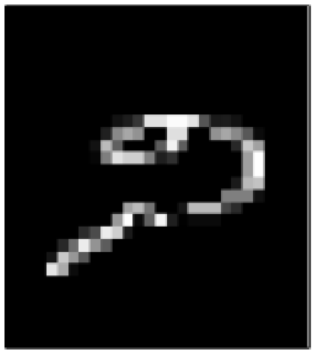


Figure.9. Sample Image from Created Dataset

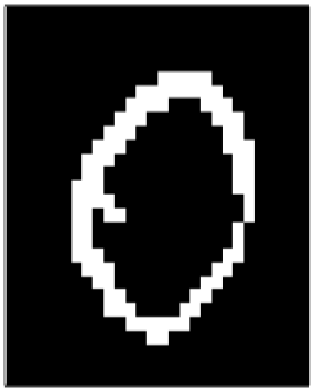


Figure.10. Sample Image from the Dig-MNIST Dataset

While Fig. 10 displays an example picture of the number 0 from the Dig-MNIST dataset, Fig. 9 displays a sample image of the number 8 from our own datasets. We demonstrate the validity of our suggested approach's hypothesis on both of these datasets. It is clear from the comparison that the two photos' pixel representation and quality vary. Therefore, while assessing the effectiveness of our strategy, its potential to generalise will also be put to the test.

4. Topology of the CCNN

We start with a fundamental CNN and expand it further to create what we refer to as customised CNN. We outline the three primary layers that make up our CCNN architecture's main part: Convolution is a mathematical process used in this layer to combine the input data with a filter to produce a feature map. The filter, also known as the kernel, is a mxm window that is moved over the whole picture, where m is the image's dimensions. The imposing elements are element-wise products operated on for each instance, and the total of these products is then computed to provide the final convolution value for that specific case [8]. The feature maps that are generated are classified according to the values on the filter. A convolution process would result in a decrease in the dimensions of the input data as an element-wise product and sum operation is carried out. Two factors govern this decrease or change in output size: • Step: Step specifies the value by which we want to shift the filter at each step. By default, the filter window advances every iteration by one unit stride. The size of the final feature map will be reduced if a larger stride value is utilised, however. Additionally, the resulting characteristics would have less overlap between them than they would if the stride value were lower. • Padding: If we want the output feature map to be a certain size bigger than anticipated, we may pad the output map by adding additional pixels to make the required dimension. Padding is what this is, and the pixels that are padded often have a value of zero or one. Although the result of the convolution operation is a feature map, non-linearity must be preserved in its values to keep the design effective. For this, we feed the values from the feature map via an activation function. For all convolution layers, the Rectified Linear Unit (ReLU) activation function is used. layering pools In general, pooling is used to reduce dimensionality while ensuring information retention following convolution processes. By lowering the number of trainable parameters in the architecture, pooling layers aid in combating the overfitting issue and shorten training time. Each output feature map's height and breadth are decreased as a result of the pooling procedure, which is done for each feature map. The window used in pooling is similar to the one used in convolution. Pooling will be employed in this instance, nevertheless, to determine the outcome from the feature map values that are present in the window frame at a certain instance. Thus, other than the window size and the stride, no further parameters are needed with regard to the pooling window. In our design, we employ maximum pooling, which allows the pooling window's largest element to be carried over to a later instance. Complete Layer Connectivity Each neuron in these layers is able to store a single value, and every node in a given layer is connected to every other node in the layer below it. Each node is coupled with a bias factor and weights for all previous nodes. These completely linked layers have more parameters than other layers as a consequence. The parameter values in these layers are likely to be overfitted. It introduces the idea of dropout to prevent overfitting. Dropout describes the haphazard termination of a specific group of neurons from a layer during the training phase, necessitating more iterations to reach convergence [7]. Dropout aids in the frequent level prevention of co-dependency growth among various neurons. With the exception of the top layer, this layer uses ReLU activation functions. We stack many of these layers to create our ultimate CCNN design. Fig.11 displays a schematic of the CCNN's design overall. We also stress the use of more convolutional layers in the CNN component than what is seen in the image.

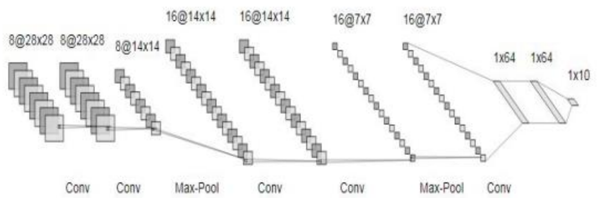


Figure 11. Architecture Diagram of the Proposed CCNN

Dimensions of the input picture are 28x28x1. It goes through two convolutional layers, each of which has eight filters with a 3x3 individual dimension. The intermediate output is then sent via a size 2-capacity max pooling layer. It then goes through two further convolutional layers, each of which has 16 filters with a 3x3 dimension. The next layer is a max pooling layer with a 2x2 pooling window. The intermediate feature map is reduced to a one-dimensional vector before a final convolution layer with 16 filters in size 3x3 is applied in the next post. The information is then passed on to two fully connected layers that each have 64 nodes and are followed by a dropout layer with a value of 0.5. Ten nodes make up the last output layer. The procedures taken for each operation needed to train the CCNN architecture are shown in Algorithm 1. This gradual training of the model is followed by evaluation of the outcomes.

Algorithm 1 CCNN(Customized Convolutional Neural Network)

Input: Image I(i_wxi_h), Filter f, Learning rate α, Number of channels n_c, Stride S, padding P, weights w, bias b
Output: Output Feature map

```
{Forward propagation}
Initialize weights and bias between 0 and 1
w_u = int((i_w − f)/S) + p
h_u = int((i_h − f)/S) + p
for each h in h_u pixel do
   for each wd in w_u pixel do
      for each c in n_c channels do
         layer_slc = prev_layer[v_s : v_e,h_s : h_e,:]
         orp = sum(layer_slc * w) + b {Output representation}
      end for
   end for
end for
orp[orp ≤ 0] = 0 {ReLU layer}
{Max Pooling}
for each h in h_u pixel do
   for each w in w_u pixel do
      for each c in n_c channels do
         layer_slc = prev_layer[v_s : v_e,h_s : h_e,:]
         orp = max(layer_slc)
      end for
   end for
end for
{Backward Propagation}
for each h in h_u pixel do
   for each w in w_u pixel do
      for each c in n_c channels do
         layer_slc = prev_layer[v_s : v_e,h_s : h_e,:]
         dw += layer_slc * orp[h,w,c]
         db += orp[h,w,c]
      end for
   end for
end for
w = w - α * dw
b = b - α * db
return Output weights
```

5. Experimental Results

For assessment reasons, the datasets were divided into training, validation, and test datasets as was previously mentioned. Accuracy is a perfect statistic to evaluate the effectiveness of our algorithm since the dataset is evenly distributed across all classes. The system is trained using the k-fold cross validation concept, as was previously discussed. In this method, the training data is divided into k parts, of which k-1 parts are utilised for training and one part is used for validation. This is repeated k times repeatedly to ensure that every sample of the training data is included in the validation set. As previously said, k=5 in this instance since the train to validation data ratio split is 4:1. The model is thus trained for 10 epochs across 5 folds, totaling 50 epochs. Additionally, we assess the system using additional measures including accuracy, recall, and F1 score as well as fluctuations in loss across the training period. For example photos from both datasets, the intermediate feature maps are also shown. First, we display all of these findings for our unique dataset, which are shown and summarised below:

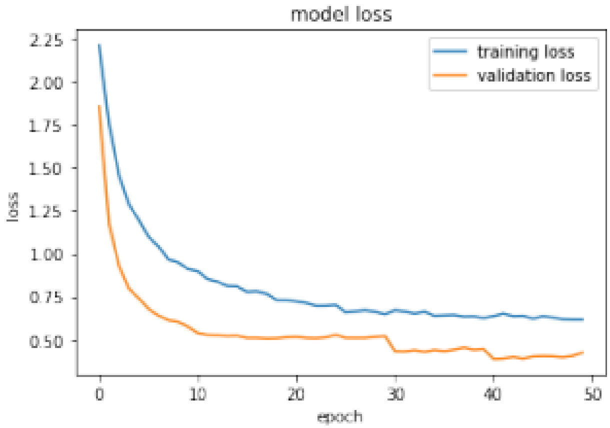


Figure.12. Variation in CCNN Model Loss over epochs for Created Dataset

Figure 12 displays the model loss for the training and validation sets of data. As can be observed, the losses gradually decrease and the model reaches convergence with a suitable flow. The accuracy acquired by our suggested technique for each class label is then tabulated, and the average accuracy for our dataset is also calculated. The acquired findings are contrasted with several other conventional implementations, which we test on our data with the same split, to further highlight the power of our model. Decision trees, multilayer perceptrons, linear and nonlinear SVMs, and the LeNet CNN architecture are examples of common techniques. It is evident that our suggested strategy has done noticeably better than any other way now in use. Additionally, we compute the model's class-wise performance on the three additional metrics of accuracy, recall, and F1 score. For a better understanding of the operation of the CCNN architecture and its capacity for pattern extraction, the intermediate visualisation of the CCNN at various convolution layers for Fig. 13 is displayed in the following images:

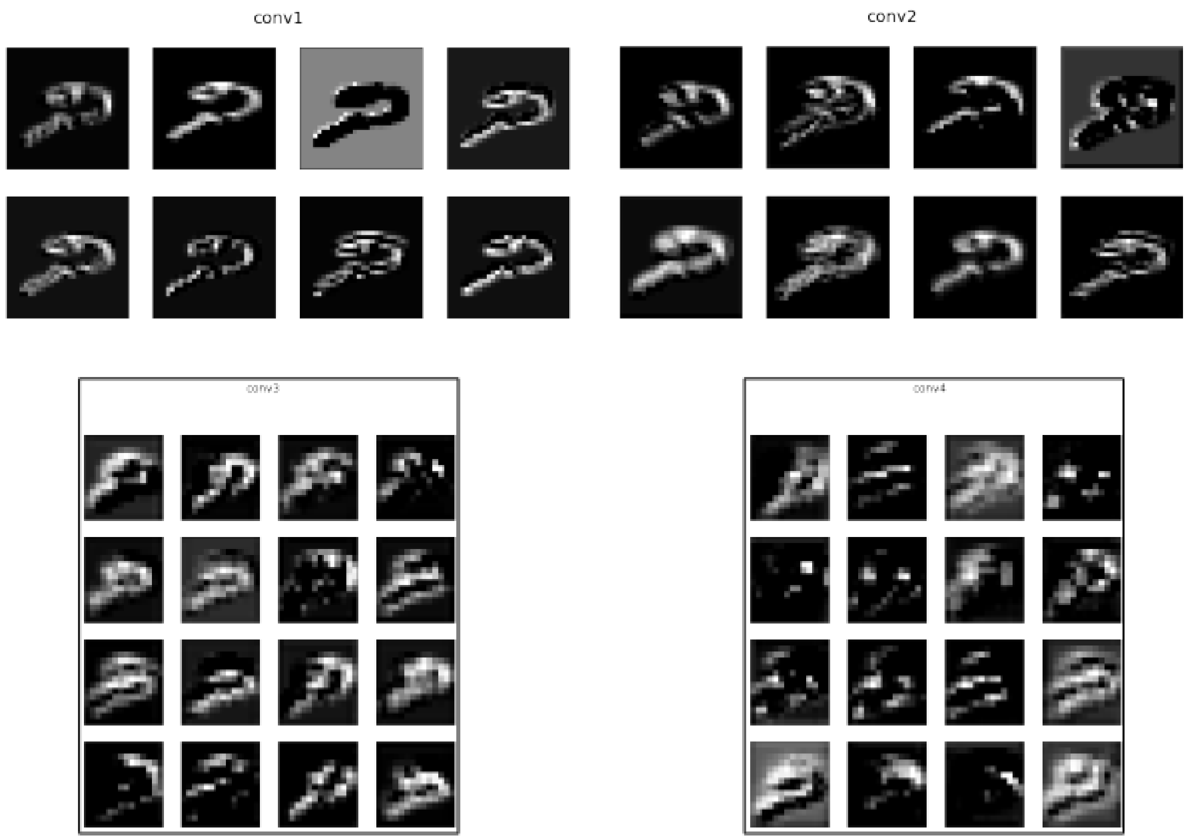


Figure .13. Visualization of Internal Feature Maps of CCNN for Sample Image from Our Dataset

In order to gain a statistical overview of the true positives and true negatives for each class label, we additionally tabulate the confusion matrix produced across the test set of 5000 pictures as part of the final ablation research.

429	14	4	9	4	2	5	5	2	3
5	414	11	13	7	4	4	2	0	3
6	11	456	8	5	2	5	4	2	7
5	7	7	442	11	13	9	9	3	5
4	9	8	6	470	8	7	8	6	5
5	8	10	9	10	439	2	6	7	8
2	2	9	4	7	4	395	9	8	14
2	3	8	13	12	4	26	465	5	8
6	7	7	5	9	5	9	8	468	8
6	0	10	3	9	3	11	4	5	425

Figure.14. Confusion Matrix of CCNN for Our Dataset

The standard Dig-MNIST dataset is used to assess the findings, and it contains 1240 samples for testing and 2000 samples for training and validation.

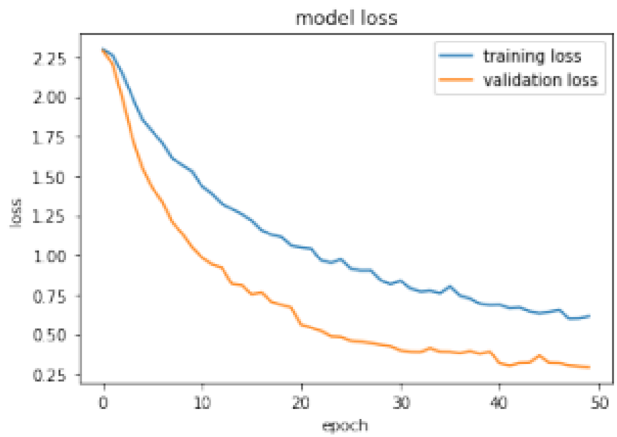


Figure.15. Variation in CCNN Model Loss over epochs for Dig-MNIST Dataset

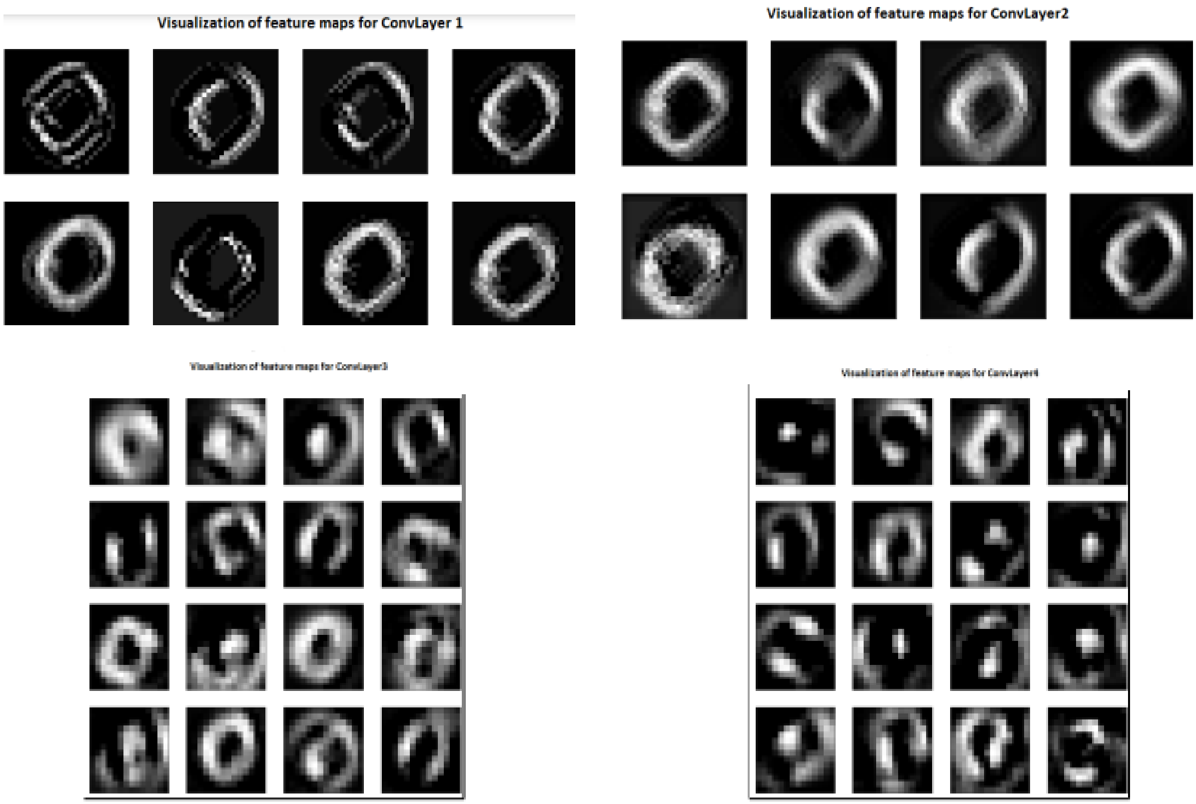


Figure .16. Visualization of Internal Feature Maps of CCNN for Sample Image from Dig-MNIST Dataset

The interpretability of the feature maps derived from the internal layers of the proposed architecture is exhibited by the intermediate visualisation of the CCNN at various convolution levels in Fig. 16. It is evident that several facets of the handwritten number were caught by the internal layers of the CCNN. We can see that our suggested CCNN model easily defeated other conventional techniques on a conventional external dataset. In the case of this Dig-MNIST dataset, the convergence for training and validation loss has likewise followed the anticipated trend.

6. Conclusion

For the purpose of handwritten Kannada number identification, we concentrated on creating a customised CNN in this article. The suggested design is more universal and does not simply adhere to training data. We can get a perfectly impartial picture of the system's performance with the use of cross-validation. On our produced dataset's testing data, the suggested model achieves an accuracy of around 88%, while on the Dig-MNIST dataset, it achieves an accuracy of about 87%. The method may be improved by establishing consistent accuracy across all numbers and further generalising the accuracy. It is also possible to further minimise the amount of incorrectly predicted samples in the confusion matrix, both positive and negative. The feature maps created with this CCNN architecture may also be used to improve the accuracy of other models and techniques. Instead of initialising weights in the CCNN architecture randomly, weights that are best suited to the model for this job may be initialised. A further development may include figuring out the optimal base weights for the CCNN architecture to boost performance. As a result, this method has shown the CCNN's capacity for feature extraction. On a number of performance indicators that were looked at, the model produced results that were adequate.

References

1. Duda, Richard O., Peter E. Hart, and David G. Stork, “Pattern Classification,” John Wiley Sons, 2012.
2. Papa, Joao P., Alexandre X. Falcao, and Celso TN Suzuki, “Supervised Pattern Classification Based on Optimum-path Forest,” International Journal of Imaging

Systems and Technology, 19, no. 2 (2009): 120-131.

3. C. F. Eick, N. Zeidat and Z. Zhao, "Supervised Clustering - Algorithms and Benefits," IEEE International Conference on Tools with Artificial Intelligence, pp. 774-776, 2004.
4. T. Finley and T. Joachims, "Supervised k-Means Clustering," July 2010.
5. Xiangyang Li and Nong Ye, "A Supervised Clustering and Classification Algorithm for Mining Data With Mixed Variables," IEEE Transactions on Systems Man Cybernetics—Part A: Systems and Humans, Vol. 36, No. 2, 2006.
6. L. Kaufman and Peter Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis," DOI: 10.2307/2532178, 1990.
7. Hong Zeng and Yiu-Ming Cheung, "Semi-Supervised Maximum Margin Clustering with Pairwise Constraints," IEEE Transactions on Knowledge and Data Engineering, Vol. 24, No. 5, 2012.
8. P. M. Patil, P. S. Dhabe, U. V. Kulkarni and T. R. Sontakke, "Recognition of Handwritten Characters Using Modified Fuzzy Hyperline Segment Neural Network,"12th IEEE International Conference on Fuzzy Systems, pp. 1418–1422, 2003.
9. G. E. Hinton and Salakhutdinov, "R. Reducing the Dimensionality of Data with Neural Networks," Science, Vol. 313, pp. 504–507, 2006.
10. Phan Khanh Minh, Cuong Tuan Nguyen, Anh Due Le and Masaki Nakagawa, "An Incremental Recognition Method for Online Handwritten Mathematical Expression," 3rd International Asian Conference on Pattern Recognition, 2015.
11. M. A. H. Akhand, Mahtab Ahmed and M. M. Hafizur Rahman, "Convolutional Neural Network Training with Artificial Pattern for Bangla Handwritten Numeral Recognition," 5th International Conference on Informatics Electronics and Vision, 2016.
12. Vinay Uday Prabhu, "Kannada-MNIST: A New Handwritten Digits Dataset for the Kannada Language," arXiv preprint, arXiv:1908.01242, 2019.
13. A. Ganesh, A. R. Jadhav and K.A. Cibi Pragadeesh, "Deep Learning Approach for Recognition of Handwritten Kannada Numerals,"Proceedings of the 8th International Conference on Soft Computing and Pattern Recognition, pp. 294–303, 2016.
14. G. Mukarambi, V. Dhandrab and M. Hangarge, "Recognition System for Handwritten and Printed Kannada Numerals and Vowels," Int. J. Mach. Intell. Vol. 3 Issue 4, pp. 259–262, 2011.
15. V. C. Hallur and R. S. Hegadi, "Offline Kannada Handwritten Numeral Recognition:Holistic Approach," Proceeding of Second International Conference on Emerging Research in Computing Information Communication and Applications, Vol. 3, pp. 632–637, 2014.
16. W. Bieniecki, S. Grabowski and W. Rozenberg, "Image Preprocessing for Improving OCR Accuracy,"International Conference on Perspective Technologies and Methods in MEMS Design, pp. 75-80, 2007.
17. Y. Fataicha, M. Cheriet, J. Y. Nie and C. Y. Suen, "Retrieving Poorly Degraded OCR Documents," International Journal on Document Analysis and Recognition, DOI: 10.1007/s10032-005-0147-6, 2005.
18. T. Kanungo and R. M. Haralick, "An Automatic Closed-loop Methodology for Generating Character Ground-truth for Scanned Documents," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 2, pp. 179-183, 1999.
19. J. Peng, L. An, X. Zhu, Y. Jin and D. Shen, "Structured Sparse Kernel Learning for Imaging Genetics Based Alzheimer's Disease Diagnosis," Medical Image Computing and Computer-Assisted Intervention, Springer Lecture Notes in Computer Science, Vol. 9901, 2016.
20. M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe and S. G. Mougiakakou, "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network,"IEEE Transactions on Medical Imaging, Vol. 35, No. 5, pp. 1207–1216, 2016.
21. D. K. Renuka, T. Hamsapriya, M. R. Chakkaravarthi and P. L. Surya, "Spam Classification Based on Supervised Learning Using Machine Learning Techniques," International Conference on Process Automation Control and Computing, pp. 1-7, 2011.
22. Steve Lawrence, et al., "Face recognition: A convolutional Neural-network Approach,"

IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 98-113, 1997.
23. Shawn McCann and Jim Reesman, "Object Detection using Convolutional Neural Networks," Stanford University, pp. 1-5, 2013.
24. J. Chung and K. Sohn, "Image-Based Learning to Measure Traffic Density Using a Deep Convolutional Neural Network," IEEE Transactions on Intelligent Transportation Systems, Vol. 19, pp. 1670-1675, 2018.
25. Christian Gerber and Mokdong Chung, "Number Plate Detection with a MultiConvolutional Neural Network Approach with Optical Character Recognition for Mobile Devices," Journal of Information Processing Systems, Vol. 12, pp. 100-108, 2016.
26. A. Graves, A. Mohamed and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," 2013 IEEE International Conference on Acoustics Speech and Signal Processing, pp. 6645-6649, 2013.
27. S. James Press and Sandra Wilson, "Choosing Between Logistic Regression and Discriminant Analysis," Journal of the American Statistical Association, Vol. 73, No. 364, pp. 699-705, 1978.
28. David M. Blei, Andrew Y. Ng and Michael I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning and Research, Vol. 3, pp. 993–1022, 2003.
29. "Analysis of Variance" Wikipedia, 17 Apr. 2020. Wikipedia, https://en.wikipedia.org/w/index.php?title =Analysis of variance&oldid=951505372.
30. X. Zhou, X. Zhang and B. Wang, "Online Support Vector Machine: A Survey," Springer Advances in Intelligent Systems and Computing, Vol. 382, pp. 269-278, 2016.