Enhancing the Odia Handwritten Character and Numeral Recognition System's Performance with an Ensemble of Deep Neural Networks

Mamatarani Das[1], Mrutyunjaya Panda[2], Soumya Sahoo[3]
Department of Computer Science and Applications, Utkal University, Bhubaneswar, Odisha, India[1, 3]
Department of Computer Science and Engineering, C.V. Raman Global University, Bhubaneswar, Odisha, India[1, 2]

***Abstract*—Offline handwritten character recognition (OHCR) is considered a challenging task in pattern recognition due to the inter-class similarity and intra-class variations among the symbols present in the alphabet set. In this work, a learning-based weighted average ensemble of deep neural network models (WEnDNN) is proposed to classify the 10 digits and 47 characters present in the alphabet set of Odia language, an official language of India. To build the base model for the ensemble network (EnDNN), three suitable convolutional neural networks (CNN), are designed and trained from scratch. The WEnDNN's accuracy is increased by using a grid search approach to determine the ideal weight allocations to give to the top-performing model. The performance of the WEnDNN model is compared with several standard machine learning models, which take the non-handcrafted features extracted from the finely tuned, pre-trained VGG16 model and a combination of Gabor and pixel intensity values to create handcrafted features. On several benchmark handwritten datasets, including NITR Odia characters (OHCS v1.0), ISI Kolkata Odia numerals, and IITBBS Odia numerals, the performance of the proposed WEnDNN model is assessed and compared. The experimental results demonstrate that, in terms of recognition accuracy, the proposed approach beats other state-of-the-art approaches.**

Keywords*—*Odia language; ensemble learning; machine learning; Gabor features; CNN; DNN

I. INTRODUCTION

It is possible to recognize a symbol easily with our naked eye, but hard for a handwritten character recognition (HCR) model. To reduce this recognition gap between humans and models and to achieve human-like accuracy, handwritten character and numeral recognition (HCNR) systems have made significant advancements in recent years, with various approaches developed in different languages. These systems play a crucial role in applications such as document digitization, automatic form processing, and handwriting analysis. While numerous methods have been proposed to tackle this challenging task, researchers are more inclined towards deep neural networks (DNN). Deep Convolutional Neural Networks have proven their advantage in getting high performance in different applications of pattern recognition tasks when handling large data sets to extract features automatically.

Acquisition of character images, pre-processing, feature extraction, and classification make up the three major steps of the conventional OHCR workflow, and much research in this paradigm has concentrated on enhancing each of these steps. For instance, the feature extraction stage has advanced to the point that many researchers aim to create potent feature descriptors or vectors referred to as handcrafted in the literature. The basic goal of feature engineering is to design features that maximize patterns' separation from other classes while placing patterns from the same class close to one another in the feature space.

From the literature, in the late 1990s, study into the recognition of Odia characters began. The research community has paid a lot of attention to the most popular Indian scripts, Devanagari, Bangla, and Telugu, compared to Odia scripts. Natives of the Indian state of Odisha as well as its neighboring states, including West Bengal, Chhattisgarh, and Jharkhand, are fluent in Odia, a popular and official language of India. The necessity to digitize historical documents available in Odia literature inspires researchers to create Odia HCRs that have advantages for both business and society. The advancement of Odia OHCR needs to be enhanced to meet the requirements of real-time recognition. Modern schemes use features that are manually designed (handcrafted), which requires a lot of work. Several researchers have designed CNN based classification model to obtain deep features (non-handcrafted features) for Odia OHCR [1], [2]. In Odia language, most letters have a perpendicular straight line on the right side, while the upper portions are mostly circular. The characteristics of similar characters present as well as the roundish structure and the randomness of its writing, bring great challenge to the recognition task, which motivates us to propose an Odia OHCR model that enhances the classification accuracy in this regard.

The right selection of feature descriptors still presents the biggest hurdle in these OHCR systems. Utilizing a method known as “transfer learning”, those architectures are being employed for numerous applications all around the world. In this transfer learning method, the weights of a model that has already been trained for a particular job are used for a variety of tasks. Such architectures include VGG16 [3], ResNet, Xception, DenseNet, MobileNet, InceptionNet, ResNeXt etc. These architectures differ from one another in terms of depth, complexity, and size of input data. Despite having been trained on ImageNet 1000 classes, they are successfully used in all applications of pattern recognition tasks. According to Odia OHCR's related work[4], [5], the majority of researchers

choose the model that performs better in terms of accuracy in classification. Although significant progress has been made in developing individual handwritten character and numeral recognition models, their accuracy levels often plateau or show diminishing returns with increased complexity. This limitation is primarily due to the inherent variability in handwriting styles, diverse character and numeral shapes, and the presence of noise and distortions in handwritten samples. Therefore, there is a need to explore alternative approaches that can enhance when solving a classical classification problem using various trained machine learning or deep learning models, the model that produces the best results is maintained and the other models are discarded. If all of the trained models are put together for classification, that will be a better option, as some models are good for extracting certain features while some other models are good for extraction of other kinds of features. An ensemble of different trained models can be used for this purpose. In the case of handwritten character recognition using Convolutional Neural Networks (CNNs), an ensemble of different CNNs often performs better than a single CNN for several reasons:

1) Each CNN model in the ensemble is trained independently on a different subset of the data or with different initialization weights. By combining their predictions, the ensemble can help to reduce bias and overfitting.

2) Ensemble learning can help reduce the impact of errors made by individual models. If one model misclassifies a particular handwritten character, other models in the ensemble may still correctly classify it. Through the combination of predictions, the ensemble can reduce the overall error rate and improve the final classification result.

3) Ensemble learning involves averaging the predictions of individual models. This averaging process helps to smooth out noisy predictions and reduce the effects of outliers. By leveraging the collective wisdom of multiple models, the ensemble can provide a more confident and accurate prediction.

4) Different CNN models may excel at capturing different types of features or have different strengths in recognizing certain patterns. By combining the strengths of multiple models, the ensemble can achieve a more comprehensive and discriminative representation of the handwritten characters, leading to improved classification performance.

Now, there is always the option to employ an ensemble learning method that boosts efficiency by using several CNN models for the same tasks. This has inspired researchers that utilize CNNs for the task of character recognition and to develop methods for ensembles of networks to enhance CNN performance. It has been observed from related work that although many efforts have been made in the area of Odia OHCR to improve the performance of the model, no work has presented the ensemble of various CNNs. These models even result in a further improvement in accuracy of about 1 to 2% across the models when combined with the ensemble of different CNN's methodologies outlined in the research. This motivation led us to the development of EnDNN and WEnDNN.

This paper presents a novel approach for offline Odia handwritten character and numeral recognition (OHCNR) by using an ensemble of deep neural networks and a weighted average ensemble of deep neural networks. Our main contributions to this research are as follows:

- Three CNN models are designed from scratch, from a simple to a slightly complex model, by varying the feature maps and number of layers, and these CNNs are combined to create the base model of the ensemble network (EnDNN).
- A grid search method is used to get the right combination of weights to be assigned to the best-performing model to construct a weighted average ensemble of deep neural network models (WEnDNN), to boost the ensemble networks' accuracy.
- Traditional ML models (Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbour (kNN), and Extreme Gradient Boosting (XG-Boost) are used, which are trained on non-handcrafted features obtained from fine-tuned, pre-trained VGG16 model and handcrafted features extracted by Gabor filters, combined with pixel intensity values to create feature descriptor.
- The performance of the WEnDNN model is compared with the individual CNN, EnDNN and ML models to show the effectiveness of the proposed work and the models are verified using a set of benchmark Odia databases, namely ISI Image database, NITROHCSv1.0 and IITBBS numeral database.

Here is a summary of the remaining portions of the paper: Some of the most significant studies on deep learning for Odia and other language OHCNR tasks currently published in the literature is highlighted in Section II. Section III discusses the materials and methodology, which covers the description of DCNN models and their components. The datasets used for the proposed work are covered in Section IV and the proposed model architecture is covered in Section V. Section VI reports the results and discussion, and Section VII provides the conclusion.

II. Related Work

In the Odia script, like every other script vowels, consonants, and composite characters (combinations of characters with other characters) are present. A total of 10 numerals and 47 alphabets (vowels and alphabets) are present in the Odia script, as shown in Fig. 3(a), 3(b), and 3(c). With different handwriting styles and high similarity between different characters, it's challenging for any system model to get human-like accuracy. Several works on Odia OHCR were reported in [6], [7] based on handcrafted feature extraction. In [8], authors have used curvature features and reduced the feature set by PCA and with quadratic classifier got a classification accuracy of 94.6%. In [9] Binary External Symmetry Axis Constellation (BESAC), features are used with an accuracy of 95.01 by the k-NN classifier. The authors of

[10] used zone centroid distance and standard deviation to extract features and got 94% accuracy by back propagation NN with a genetic algorithm approach. [11], [12], [13]–[15], [16], [17][18] had contributed their work on handwritten Odia handwritten numeral recognition (Odia OHNR). The same BESAC features are used for numeral classification, on the IITBBS numeral dataset [9]. In [19], the authors achieved an accuracy of 95% by SVM with directional features by zoning method. In a work of [20], authors used Gradient, curvature feature, and Feature reduction using PCA fed to low complexity neural classifier for recognition with an accuracy of 98% by gradient feature and 94% by curvature feature. In [16], the DCT and DWT coefficients are used by the BPNN classifier. Several studies have been reported on the ISI numeral dataset [21][22], [23] with a promising accuracy of over 90% for handwritten Odia numeral recognition.

The goal of researchers is to increase the optical character recognition (OCR) model's accuracy, so they are more focused on deep neural networks and ensembles of networks. To the best of our knowledge, almost little efforts on deep neural networks were contributed to the field of Odia OHCR and OHNR. In [24], the authors proposed RNN and CNN-based classification techniques for Odia compound characters. To improve the classification accuracy, different augmentation techniques were used by [1] to expand the dataset, and different CNNs were used for the classification of Odia handwritten numerals and characters. Different deep learning-based classification models proposed for Bangla OHCR, a sister language of Odia. In [25], the authors used mobilenet v1 architecture, whereas the authors [26] proposed a hybrid Bangla OHCR model that is a combination of stacked Bi-directional Long Short-Term Memory (Bi-LSTM) applied on the features extracted from CNN. A deep analysis was carried out by [27] for Bangla OHCR by different deep networks i.e. InceptionResNetV2, DenseNet121, InceptionNetV3, NASNet, VGG16, VGG19 and authors claimed InceptionResNetV2 as the best performing model. An improved CNN based digit recognition on MNIST dataset with an accuracy of 99.87% by [28].

Combining CNN models into an "ensemble" is one strategy for improving the handwritten character recognition system's accuracy. For the Odia OHCR, very few works based on ensemble networks were published. Three ensemble learning methods (AdaBoost, Bagging, and Random Subspace) are utilized in the study[29], for improved sentiment analysis and in [30]different features were selected and classified using Random Forest, which is an ensemble of several decision trees for Odia vowel recognition. The authors of [31] reported an offline Tai Le OHCR using ensemble deep learning, with a DCNN serving as the primary or base classifier. An ensemble deep learning model is created by stacking several different base classifiers, and the model achieves an accuracy of more than 98% on the Devanagari handwritten characters and MNIST handwritten digits datasets. The base classifiers' parameter combinations are optimized using a grid search technique.

Apart from OCR applications, ensemble networks were used in different fields [23]-[27], and some of the applications are described below. A deep ensemble network by using LSTM-B was proposed by [32] to obtain the accurate results of exchange rates forecasting and to improve the profit of exchange rates trading. The authors of [33] proposed a deep ensemble learning algorithm on a variety of datasets, including those for letter recognition, cancer, diabetes, heart disease, thyroid, etc., which determines the ensemble size, the number of hidden nodes in a neural network, etc. In [34] the authors used CNN as an ensemble model for object detection by selecting the region from each CNN model is combined, classified, and finally voted. Automated audio classification is proposed by [35] that fuses different types of features extracted from audio files and uses different pre-trained CNN models AlexNet, GoogleNet, VGG16, VGG19, ResNet50, InceptionV3 as ensemble and got the maximum accuracy of 99.3% by using ensemble DL and handcrafted features. In [36], the authors used an ensemble model for crash prediction model using road geometric alignments (CPM-GA) with three traditional models NB model and IHSDM-China and IHSDM-US models, and CART+SVM, RF + SVM, CART + BPNN, RF + BPNN as base models of the ensemble by selecting the model by model prediction test and model's sensitivity test. The results of the ensemble learning CPM-GAs using the IHSDM + China model and CART + SVM model are promising. Due to their improved accuracy, increased robustness, and scalability in model design for character recognition, ensemble models are becoming more and more popular nowadays. Utilizing ensemble data mining techniques for the classification of skin diseases is reported in [42][43]. It explores methods to enhance accuracy and reliability in diagnosing skin conditions through ensemble data mining techniques. To improve optical character recognition (OCR) performance by employing an ensemble of Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and Extra Trees classifiers is shown in [44]. An ensemble model composed of Convolutional Neural Networks (CNNs) for classifying cloud image patches, particularly on small datasets addresses the challenge of achieving accurate classification results with limited data and is proposed in [45].

III. MATERIALS USED

This section discusses all the materials and methodologies that are utilized to construct the proposed ensemble model of deep neural networks.

A. Deep CNN Models

Due to the non-linear behavior of neural network models, CNNs can learn the complex nonlinear relationships in the given input data. Convolutional layers, pooling layers, and fully connected dense layers are the three fundamental layers that make up the conventional CNN structure. These layers are repeated to make an NN to a deep CNN, and it is shown in Fig. 1.

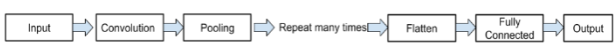


Fig. 1. Basic structure of a deep convolutional neural network.

1) Convolutional layer: These layers identify patterns in images by sliding a filter over the input image to produce a feature space or feature map. If the input image is directly

connected to the fully connected layer for classification, we may get the result, but the complexity increases when the input image size is large and the number of images is greater. The expensive computation and the cost reduction can be achieved by including the convolution and pooling layers. The basic convolution operation in convolutional layer is represented mathematically in Eq. (1), where f (x, y) is the input image, c (x, y) is the convolved image and h (x, y) is the filter or kernel.

$$\boldsymbol{c(x,y) = h(x,y) * f(x,y)} \quad (1)$$

The advantage of CNN can be taken to extract important features by reducing the image dimension and keeping important features for better prediction. It learns images by applying a filter of a certain size while maintaining translation invariance, in addition to learning the features from the data. The convolutional layer has several learnable filters, each of which can be thought of as a matrix. The convolutional layer produces numerous feature maps (also known as activation maps), and these feature maps corresponding to distinct filters are layered together along the depth dimension. Each member of the matrix or filter serves as a parameter (weight and bias) of neural networks. The convolutional layer's operational structure is shown in Fig. 2(a).

The basic component of a convolution operation in a convolutional layer is the kernel. The features or significant patterns in an image are extracted from the image using a filter called a kernel. It is a matrix $\boldsymbol{h(x,y)}$ that traverses the input image $\boldsymbol{f(x,y)}$, performs a dot product with the sub-region of the input data, and produces the matrix $\boldsymbol{c(x,y)}$ of values from the dot product. To obtain another value in the feature map, the kernel moves the input image by a stride value.

2) Pooling layers: This layer down-samples the features in the feature map by reducing their dimension. It also introduces translation invariance, i.e., even if the CNN input image is translated, the CNN will still be able to recognize the features, which reduces the CNN model's tendency to overfit data. The quantity of network computation and the number of parameters to learn are both decreased by the pooling layer. The two most common pooling methods are max-pooling and average pooling, as shown in Fig. 2(b). The most prominent patterns of the feature map are retained as a result of max pooling, and the resulting image is sharper than the original. Max pooling operates by choosing the maximum value from each pool. By averaging the pool, the average pooling layer operates and it smooths the image by maintaining the image feature's essential qualities.

3) Fully connected dense layer: A dense layer is one whose interior neurons are connected to every neuron in the layer preceding to it. Finally, it is connected with the number of units, the same as the number of classes, and produces output. A CNN model employs one or more FC layers following a series of convolutional, ReLU, or pooling layers to produce the output. The way FC layer works is similar to how classic neural networks work in that it combines all of the features that the earlier layers have acquired in order to find more important patterns. The main issue with the fully connected layer is that it has a lot of trainable parameters and requires a lot of computation to train. Therefore, current research efforts are concentrated on either lowering these layers or substituting methods that may accomplish the same purpose with less computational effort for the layers. A soft-max function is utilized to determine the class label by giving each class a probability distribution after the final FC layer. The operational structure of a fully connected layer is shown in Fig. 2(c).

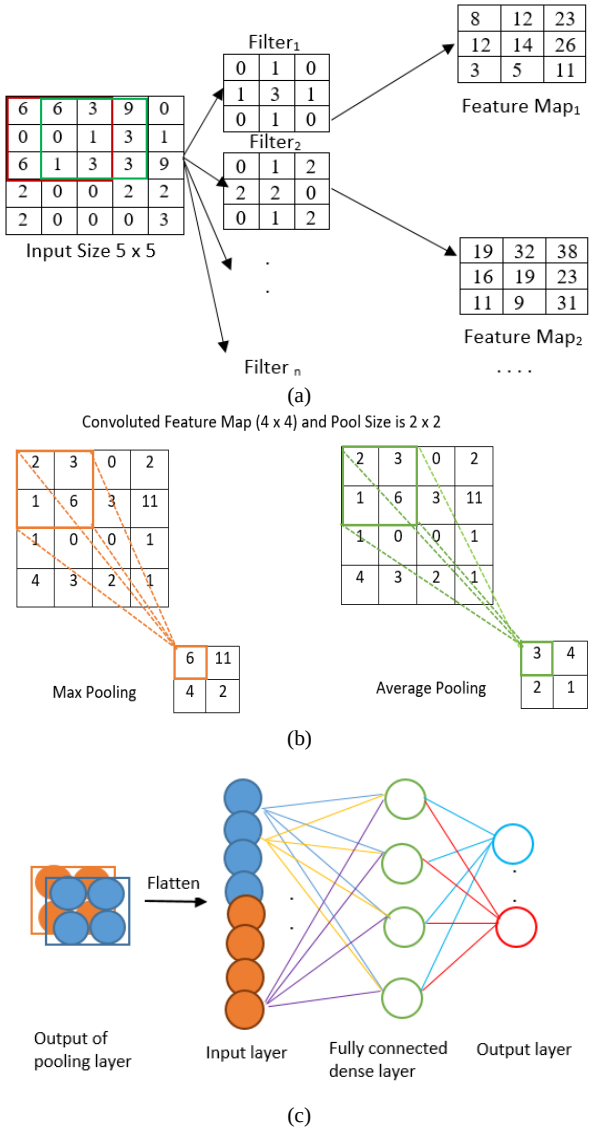


Fig. 2. (a) The basic operational structure of the convolutional operation. (b) The basic operational structure of pooling. (c) The basic operational structure of fully connected layer.

4) Rectified Linear Unit (ReLU) Activation: After the convolutional layer, the ReLU layer is frequently used, which introduces non-linearity to the output. All negative input values are mapped to zero in this layer, R (I) = max (0, I), and its operation is denoted by the following Eq. (2):

$$R(I) = \begin{matrix} I & \text{if } I \geq 0 \\ 0 & \text{otherwise} \end{matrix} \quad (2)$$

ReLU activation function has several advantages, including computing efficiency, quicker convergence than non-linear functions like sigmoid and tanh, and protection against vanishing gradient issues.

5) Softmax activation: The activation function known as softmax, scales numbers into probabilities that generate a vector V with probabilities for each class. The sum of all output values in the V adds up to 1. It is defined in Eq. (3), where y is the vector of possible outcomes of n elements for n classes, is input to the softmax function and y_j is the j^{th} element of vector y.

$$\text{softmax}(y)_j = \frac{e^{y_j}}{\sum_{k=1}^{n} e^{y_k}} \quad (3)$$

6) Cross-entropy as loss function: The cross-entropy loss quantifies the dissimilarity between the predicted class probabilities and the actual class labels. It penalizes the model for assigning low probabilities to the correct class and assigning high probabilities to incorrect classes. The loss value is larger when the model's predicted probabilities deviate further from the true expected values. During the training process, the model's weights are adjusted to minimize the cross-entropy loss. By iteratively updating the weights using techniques like gradient descent, the model learns to improve its predictions and reduce the loss. As the model gets better at classifying the handwritten characters, the loss decreases. Cross entropy is defined in Eq. (4), where t_j is the true label and p_j is the class probability value computed by the softmax activation function for class j.

$$CE = -\sum_{j=1}^{n} t_j \log(p_j) \quad (4)$$

IV. DATABASES

A benchmark database is necessary for any text recognition research to be successful. For efficient classifier or recognizer training, large databases are needed. The accuracy of recognition is entirely dependent on the type of feature extractor employed and the number of training samples taken from the database because of cursive scripts and various handwriting styles. The databases used for our study are shown in Table I, and sample images from the databases are shown in Fig. 3. The NITROHCSv1.0 data set is publicly available on the NIT Rourkela website, IITBBS numeral, and ISI image database will be available on request. These three databases are only available to the research community on the handwritten character recognition of the Odia language.

TABLE I. HANDWRITTEN ODIA CHARACTER AND NUMERAL DATASETS

Database	Training Size	Testing Size
ISI Image Database	4,970	1,000
IITBBS Numeral Database	4,000	1,000
NITROHCSv1.0	10,528	4,512

1) ISI image database: An isolated database of handwritten Odia numerals was created in 2005 by [37] at ISI Kolkata, India. There were precisely 356 participants in the data collection procedure. It has 5,970 samples that were gathered via 166 application forms, and 105 pieces of mail, and the remaining samples were personally collected. The data set is then split into a training set and a test set, consisting of 4,970 samples and 1,000 samples, respectively. Sample numeral images of the ISI Image Database are shown in Fig. 3(a).

2) IITBBS numeral database: A new database for Odia numerals has been discussed by the authors [38] at the IIT in Bhubaneswar. At 300 and 600 dpi, the images were scanned. The IITBBS numeral database now has 5,000 handwritten examples of Odia numbers, and the database contains 10 classes and the sample numeral images are displayed in Fig. 3(b).

3) NITR OHCSv1.0 character database: Databases are also created and defined at NIT Rourkela by [39], which contains an Odia alphabet with 47 classes. There are 15,040 samples of atomic characters from the Odia language in the OHCSv1.0 database, each class contains 320 images. Data collection, picture enhancement, and size normalization are the procedures used in the construction of the database using the Odia character set. The database is split into 70:30 ratios for train and test sets. The total number of images in the train and test set is 10,528 and 4,512. Fig. 3(a) represents Odia character images of the NITR OHCSv1.0 database.

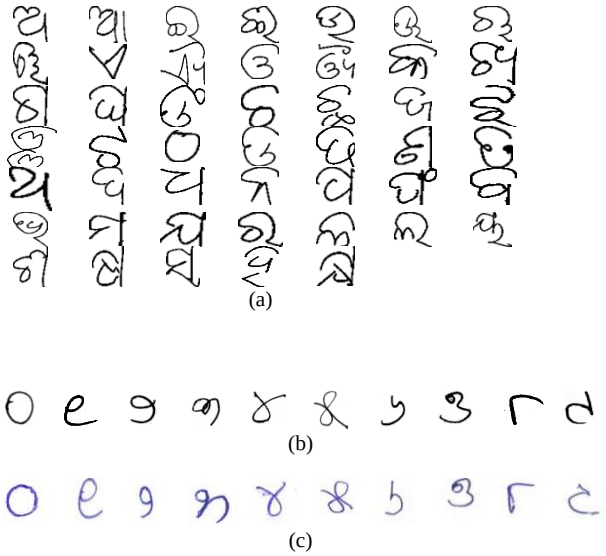


Fig. 3. (a) Sample characters of NITROHCSv1.0 database. (b) Sample numerals of ISI Image numeral database. (c) Sample numerals of IITBBS numeral database.

V. METHODOLOGY

The following are the steps of the experimental environment for the Odia handwritten character and numeral recognition model (OHCNR), which is shown in Fig. 4.

- Load handwritten images from the training and test sets.

- Convert the images to grayscale.
- Normalize the pixel values of the grayscale images to a range of 0 to 1. This enhances the training of the neural network (NN) model.
- Design three CNNs as base models for the Ensemble Deep Neural Network (EnDNN).
- Construct the EnDNN by integrating the three designed CNNs.

Select the right combination of weights by the grid search method to be assigned to the best-performing model to construct WEnDNN.

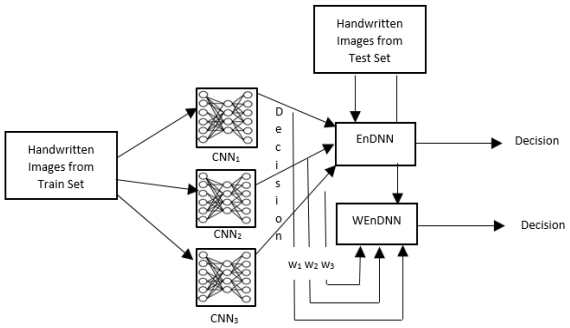


Fig. 4. Proposed OHCNR model.

1) Designed CNN models as base model for EnDNN: To achieve high recognition accuracy, a character and numeral classifier based on a convolutional neural network (CNN) is used. Convolutional, max-pooling, fully-connected, and softmax layers are used in the construction of three different CNN-based handwritten digit classifiers. Additionally, the training is carried out utilizing the back-propagation method with mini-batches of size 28 and the adam optimization methodology. When an image is processed for a character recognition task, the crucial features are retained in the convolution layers, intensified, and maintained throughout the network, while the irrelevant information is eliminated by the pooling operation. Fig. 5 lists the parameters utilized in all three created CNN classifiers, each of which has a distinct number of convolutional layers, kernel sizes, filters, and strides.

For instance, the CNN1 depicted in Fig. 5(a) contains one output layers of 10 classes, 2 max pooling layers, 3 convolutional layers, and 1 fully-connected layers. The size of kernel, stride value, and number of filters in the first convolutional layer are 3 x 3, 1, and 32 with an activation function ReLU. For down sampling, pool of size (2,2) is applied in max-pooling layer, next to convolutional layer to reduce the dimensions with a dropout value of 20%. The second convolutional layer of filter size (3,3) and 64 number of filters are used with a dropout value of 20%. 128 filters with a (3,3) filter size are put in the third layer. Next the feature map is flattened to create one dimensional feature vector and one fully-connected dense layers are used, which are connected with 10 output classes. To calculate the class probabilities for three CNN models, ReLU activation function is utilized in the hidden layer and softmax activation function is used in the output layer. Categorical cross entropy is used as the loss function, and iteratively updating network weights based on training data is done using the adaptive moment estimation (adam) optimization method. The input images are fed to the network taking 28 images as a batch at a time and epoch size is 10. In Fig. 5(b) and 5(c), the other two classifiers' convolutional, max pooling, and fully connected layer counts with activation functions are displayed.

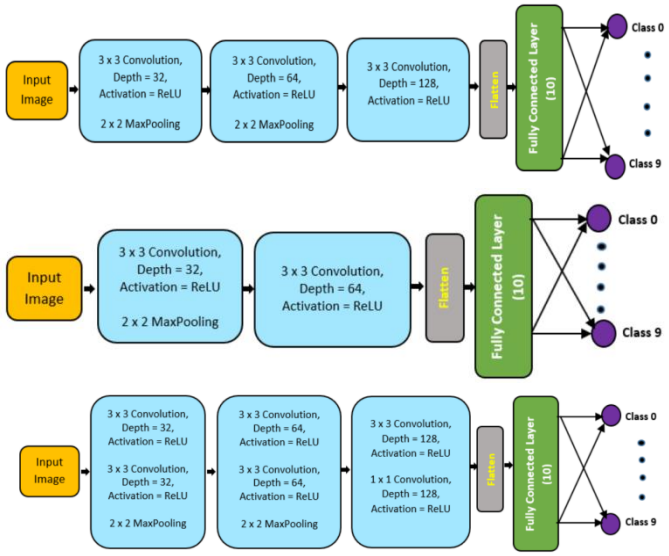


Fig. 5. (a) CNN1 architecture. (b) CNN2 architecture. (c) CNN3 architecture.

a) Steps of the recognition process by CNN: The handwritten characters and numeral recognition process, using a Convolutional Neural Network (CNN), typically involves the following steps:

i) Data Acquisition: Collect Odia handwritten dataset

ii) Data Split: Creating training and test sets from the dataset. The model is trained using the training set, and its performance is assessed using the test set.

iii) Pre-processing: Applying pre-processing techniques to the images in both the training and test datasets like resizing the images to a consistent size, applying image enhancement techniques, and normalization.

iv) Data Normalization: Normalizing the pixel values of the images so that they range from 0 to 1. This step helps in improving the convergence of the neural network during training and ensures that all features have a similar scale.

v) Batch Training: Dividing the training dataset into batches of a suitable size. Batch training involves feeding a subset of the training data to the network at a time instead of using the entire dataset in one go. This approach facilitates efficient computation and allows the network to update its weights based on smaller subsets of data at each iteration.

vi) Model Training: Training the CNN model and its variants using the labelled training data. This step involves feeding the batches of training images to the network, performing forward and backward propagation, and adjusting the network's weights using optimization techniques like

gradient descent. The training process aims to minimize the difference between the predicted output and the actual labels.

*vii)*Classification: Using a trained model to classify new, unseen images. This involves passing the test images through the trained network and obtaining predictions for each image. The predicted labels are compared to the true labels to evaluate the model's accuracy.

viii) Performance Analysis: Analysing the recognition accuracy and processing time for all the variants of the trained model. This step includes calculating metrics such as accuracy, precision, recall, and F1 score to assess the model's performance. Processing time can be measured during both training and classification phases to evaluate the efficiency of different model architectures and training strategies.

2) Ensemble of Deep Neural Networks (EnDNN): Ensemble is the process of combining several learning algorithms to improve the performance of existing models by combining different models into a single reliable model. There is now always the choice to use an ensemble learning approach, which increases efficiency by applying a number of CNN models to the same tasks. By training numerous models instead of just one and combining their predictions, neural network models can successfully reduce their variance. So, the ensemble learning method, not only lowers the variance of predictions but also has the potential to produce predictions that are superior to those produced by a single model.

In a CNN, the produced output probabilities are o1, o2, o3 .. on, where $\sum o_i = 1$, for an unseen image x of n-class classification, the CNN determines the unseen image x belongs to the class i with the greatest likelihood probability oi. In our study, the proposed CNNs should provide a probability value to each unseen test image that it was labelled by either of the 10 numerals for numeral recognition or 47-character classes for character recognition. Class probabilities for each image, derived by the individual CNN's, will be the input of our ensemble of networks and the ensemble algorithm is shown in Algorithm 1. Every individual model will make a prediction based on the test data. The ensemble approach combines the predictions of the three CNN models by summing their predicted probabilities and selecting the class with the highest summed probability as the final prediction.

Algorithm 1: EnDNN: The algorithm evaluates the performance of three designed CNN models individually and an ensemble of the three models by comparing their predicted labels with the true labels.

1. Load the dataset of Odia character or numeral images along with their corresponding class labels.
2. For each image in the dataset:
   - Apply pre-processing techniques such as resizing, RGB to Gray conversion and normalization.
3. Define three sets of convolutional neural network (CNN) models as the base models for the ensemble. (The architecture of each model is defined in Section 5)
4. Split the pre-processed dataset into training and test sets.
   - For each base model in the ensemble:
     - Train the model on the training set using adam as an optimizer and categorical – cross entropy to compute loss.
     - Evaluate the model's performance on the test set to measure its individual recognition accuracy.
     - Save these models as CNN1, CNN2, CNN3.
5. Load these pre-trained models: CNN1, CNN2, CNN3.
6. For each model in the list of models, do the following:
   - Predict the output for the test data and store the predictions in the prediction list.
7. Sum the prediction probabilities of each test image for each class obtained from different models of ensemble DNNs.
8. Determine each test image's maximum class recognition accuracy from the summed prediction values of an ensemble of DNN as ensemble accuracy:
   - For each test image in the dataset:
     - Determine the class or category with the highest summed prediction value.
     - Compare this prediction with the ground truth label of the image.
     - Calculate the ensemble accuracy by measuring the percentage of correctly recognized test images

3) Weighted ensemble of DNNs (WEnDNN): Since deep learning models differ in architecture and complexity, not all of them produce the same outputs; some produce superior output than others. To get the maximum output from any model, it would be beneficial if we gave larger weights to the better-performing models. Weighted ensemble learning is a variation of ensemble learning where different models in the ensemble are assigned different weights to determine their contribution to the final prediction. In the case of handwritten character recognition using a weighted ensemble of different CNNs, it can perform better because assigning different weights to individual CNN models allows the ensemble to emphasize the strengths of each model. Certain CNN models may be particularly effective at recognizing specific types of handwritten characters or capturing certain features. Models that consistently produce more accurate predictions can be assigned higher weights, while models with lower accuracy can be assigned lower weights. By assigning higher weights to these specialized models, the ensemble can benefit from their expertise and improve the classification accuracy for the corresponding classes.

Finding the ideal mixture of model weights is the issue in this situation, and the grid search method is employed to achieve this. To determine the best weight, various weight combinations were tested. The search procedure will continue until it has checked every combination, at which point the algorithm will give us the ideal weight combination that maximizes accuracy. We multiply the output probability values $output_{ij}$ of CNN_j (i =1,2 and 3) by $\omega eight_j$ (j = 1,2, and 3) after determining the appropriate weights for all the individual CNNs, and the class probabilities are calculatedwe using the weighted output probability values $\omega eight_j output_{ij}$ instead of

the original $output_{ij}$ ones. The weighted ensemble algorithm is shown in Algorithm 2.

Algorithm 2: WEnDNN: This algorithm outlines the steps involved in generating predictions using an ensemble of DNNs with different weighting schemes and evaluating the accuracy of the weighted ensemble predictions on the test data.

1. Create a list for an ensemble of models, called models, and add the models (CNN1, CNN2, and CNN3) to it.
2. Initialize equal weights for each base model in the ensemble.
3. For each model in models, do the following:
   - Predict the output for each image in test set and store the predictions in the predictions list, where each base model's prediction should be weighted equally at the start.
     predictions ← *[prediction1, prediction2, prediction3]*
4. Generate different combinations of weights for the base models in the ensemble.
   weights ← [weight1, weight2, weight3]
5. For each possible weight combination, multiply each prediction by its corresponding weight.
   $weighted\_predictions \leftarrow prediction_i * weight_i$, where i = 1,2,3
6. For each test data instance, determine the weighted ensemble prediction by selecting the maximum value among the weighted predictions.
   - Combine the predictions from different models and choose the prediction with the highest weighted value. *weighted_ensemble_prediction←maximum (weighted_prediction)*
7. Compare the weighted ensemble predictions to the ground truth labels of the test data.
   - Calculate the accuracy of the weighted ensemble by measuring the percentage of correctly predicted instances.

4) OHCNR model with deep and hand-crafted features: In order to evaluate the performance of the proposed OHCNR model, two more experiments were carried out by extracting deep features using a pre-trained VGG16 model and handcrafted features from Gabor filter that captures texture features and pixel-level features from the images and use these features to train and test machine learning models for recognition purpose.

a) Extraction of deep features from pre-trained VGG16 model: The researchers use a variety of strategies to extract the pertinent features, whether handcrafted or non-handcrafted. Automatic feature extraction techniques have grown in popularity in recent years for solving character recognition problems due to their capacity to extract robust features. For non-handcrafted feature extraction, transfer learning techniques have recently been applied. A learned model for one problem is used for solving another problem, a process known as transfer learning. Diverse pre-trained models, including VGG16 (Visual Geometry Group), VGG19, InceptionV3, MobileNetV2, Resnet50, ResNetV2, Xception, DenseNet, etc., are used in transfer learning. The weights of the pre-trained models are used for the training process for the new problem. These pre-trained models are used for classification tasks, stand-alone or integrated feature extraction processes, and weight initialization. These non-handcrafted features are fed to RF, SVM, kNN, and XG Boost to train these models and compare the results with the proposed state-of-the-art model.

Data from a subset of the ImageNet dataset, which consists of over 14 million photos organized into 22,000 classes, was used to train a DCNN variation called VGG16 [3]. The VGG16 Model has 16 convolutional layers and, 5 max pooling layers connected to convolutional layers of 5 different blocks, 3 dense layers for the fully-connected layer, and an output layer with 1,000 nodes. The model architecture of VGG16 is shown in Fig. 6(a). To extract the deep features from the handwritten character image can be possible by removing the last few layers (fully connected layers) from the VGG16 model, as they are specific to classification, and retaining the convolutional layers, and its architecture is shown in Fig. 6(b). The filters of size (3,3) is used at different layers to extract deep features automatically. The filters and extracted features after layers Block1-Conv1, Block1-Pool, Block3-Conv2, Block4-Conv1 and Block5-Conv1 by VGG16 model for the Odia digit 3 is shown in Fig. 6(c).

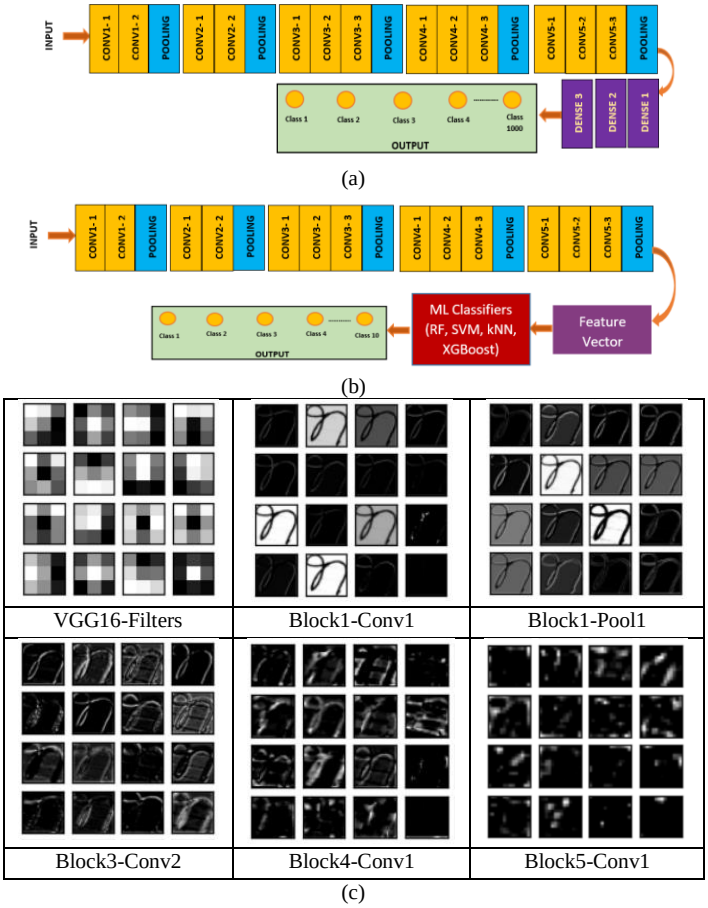


Fig. 6. (a) VGG16 Model architecture. (b) Deep feature extraction from VGG16 model. (c) Extracted features at different layers of VGG16 model.

b) Extraction of hand-crafted features using Gabor filter bank, pixel intensity values: According to the literature, the feature extraction stage of OCR is the one that most heavily influences any system's accuracy among all other OCR stages. Different hand-crafted features that can be extracted from an image are structural or geometrical features. Either the entire

image or the features that were taken from it serve as the input to any OCR. An image is made up of high-frequency components that originate from the edges, or the sudden changes in intensity values, and low-frequency components that make up the image's smooth sections. Any image must be transformed into a specific domain to be analyzed. In [40], to extract the discriminant features from a picture, image transformation is an essential step. Gabor filters are typically employed in texture analysis, edge detection, feature extraction, and other aspects of image processing and computer vision since they are independent of light, rotation, scale, and translation. They also infer optimal localization, making them a strong candidate for feature extraction issues.

In our study, 2D Gabor filter bank [41] is used. It is a sinusoidal plane with a certain frequency and orientation that is modulated by a Gaussian envelope is known as a 2D Gabor filter. The Gaussian component provides the weight, and the sine component provides the directionality. By using the Gabor filter, a bank of filters that can be used to detect and extract textures present in an image are created. The Gabor filter has a real and an imaginary component, as shown in Eq. (5), (6), and (7).

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = exp\left(-\frac{x'^2+y'^2}{2\sigma^2}\right) exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right) \quad (5)$$

The real part and imaginary parts are represented as:

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = exp\left(-\frac{x'^2+y'^2}{2\sigma^2}\right) cos\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right) \quad (6)$$

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = exp\left(-\frac{x'^2+y'^2}{2\sigma^2}\right) sin\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right) \quad (7)$$

Where,

$$x' = xcos\theta + ysin\theta \; and \; y' = -xsin\theta + ycos\theta$$

The parameters are (x, y) is size of the kernel, σ is the standard deviation or sigma of the Gaussian envelope, ψ is the phase offset, θ is the orientation of the Gabor function, γ is spatial aspect ratio and λ is the wavelength of sinusoidal component. These five parameters determine the magnitude and shape of the Gabor function shown in Table II. By adjusting these parameters, a variety of Gabor filters can be applied to extract relevant characteristics from an image.

In our study, a bank of Gabor filters with a constant frequency for various standard deviations (1.5, 2, 2.5) and orientation values ($\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3\pi}{4}$), are created to extract features. For every handwritten image, nine Gabor filters (GF) are created which are convoluted on original image to get the filtered image. The largest response occurs at edges and locations where texture changes when a Gabor filter is applied to an image. Also, the pixel intensity values are included to the handcrafted feature descriptor (HFD) as shown below:

$$HFD = Pixel_{value}, GF_1, GF_2, GF_3, GF_4, GF_5, GF_6, GF_7, GF_8, GF_9$$

The sample two numeral images (0, 3) and character images (ma, pa) and their transformation following the application of the filter are shown in Fig. 7.

TABLE II. GABOR KERNEL PARAMETERS

Gabor Kernel Parameters	Purpose
Sigma (σ)	Determines the total size of the Gabor envelope. The envelope grows to allow more stripes when the bandwidth is bigger, and it shrinks when the bandwidth is smaller.
Aspect ratio (γ)	Determines the height of the Gabor function, height increases at very low gamma values and falls at very high aspect ratios.
Theta (θ)	Determines the Gabor function's orientation. Theta at $\frac{\pi}{2}$ and 0 degree represents the horizontal and vertical positions of the Gabor function.
Wavelength (λ)	It controls the strips' width. When the wavelength is lowered, stripes are thinner; when the wavelength is increased, stripes are thicker.
Phase offset (ψ)	Varying the phase offset can help in detecting edges or texture patterns in different orientations and locations within an image.

c) Classification by machine learning algorithms: Utilizing the Support Vector Machine (SVM), Random Forest (RF), XG Boost, and k-Nearest Neighbor (kNN) algorithms, the performance of the proposed OHCNR model is compared with these techniques.

i) Random Forest: Random Forest is a bunch of decision trees (DT), a supervised learning methodology that can be applied to classification problems based on the idea of ensemble learning (EL). The method of integrating various classifiers to address complex problems and improve model performance is known as EL. Random forest takes a random subset from the training dataset, and adds some duplicate instances to make the same size as the training set. This is called a bootstrapped dataset, and its working procedure is shown in Fig. 8(a). So many DT's are trained on these various subsets of the training set, and it takes the average value to make the decision. Instead of depending on a single decision tree, the random forest uses decisions from all of the trees to anticipate the outcome based on majority voting. The root of RF takes a random subset of features available and picks the one that gives the best split in data based on Gini impurity, as shown in Eq. (8), where C is the number of classes and p(i) is the probability of randomly picking an element of class i.

$$G = \sum_{i=1}^{C} p(i) + (1 - p(i)) \quad (8)$$

ii) Support Vector Machine: A supervised machine learning technique called SVM requires labelled data. The goal of the SVM algorithm is to find the best decision boundary or line that divides the data into "n" classes so that following data points can be promptly classified into the appropriate class category. This ideal decision boundary is known as the "hyperplane". Mathematically, any hyperplane can be represented as in Eq. (9), where xi is the feature value.

$$\sum_{i=1}^{k} w_i . x_i + b = 0 \quad (9)$$

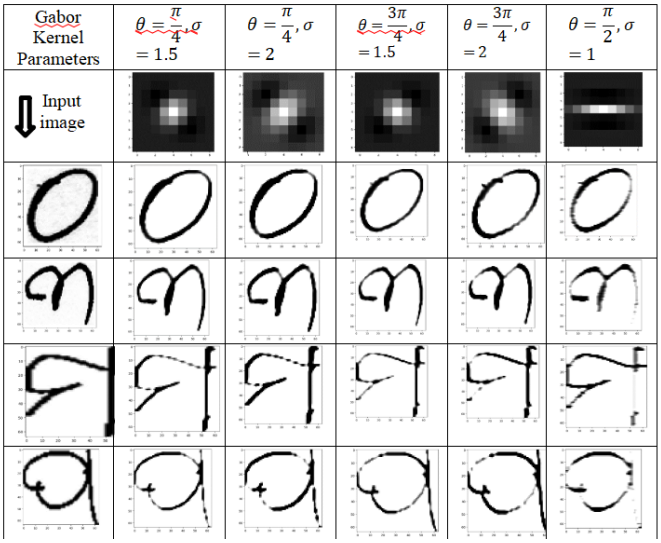
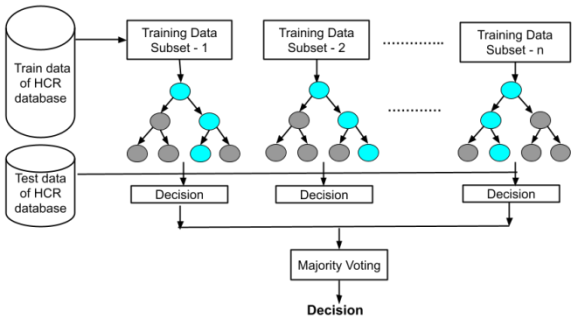
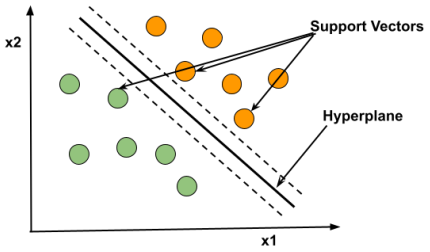


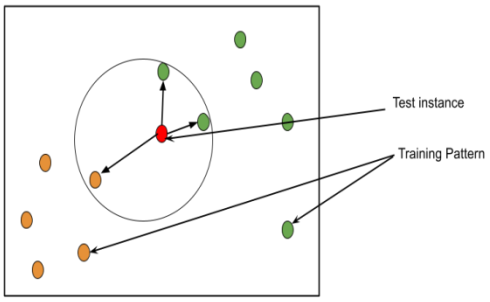
Fig. 7. Input, Gabor filter and filtered image.



(a)



(b)



(c)

Fig. 8. (a) Working procedure of random forest. (b) Working procedure of support vector machine. (c) Working procedure of k – Nearest neighbor.

Support vectors are the data points or vectors that are closest to the hyperplane and have the biggest impact on where the hyperplane is located, as seen in Fig. 8(b).

iii) Extreme Gradient Boosting (XG Boost): Gradient boosting is the technique of "boosting" or strengthening a single weak model by combining it with a number of additional weak models to produce a more reliable model all together. The XG Boost is a gradient boosting solution that pushes the limits of processing power for boosted tree algorithms. It is scalable and incredibly accurate. It was developed mainly to improve the efficiency and performance of machine learning models. In addition to building trees, XG Boost also evaluates the quality of splits at each potential split in the training set by scanning through gradient values level-wise and using these partial sums.

iv) K-Nearest Neighbor (kNN): One of the fundamental classification algorithms, the k-nearest neighbor algorithm, is well-liked for its effectiveness and simplicity. It stores the training dataset rather than learning from it immediately, which makes it a lazy learner algorithm. The algorithm selects k, the number of the nearest data points or neighbors, then calculates the Euclidean distance of the k number of neighbors, and its working process is shown in Fig. 8(c). The most popular classes are selected and given to the test pattern in the k-NN algorithm after searching for the closest training patterns for each test pattern. Some of the method's limitations are that it stores the complete training set for testing purposes, searches the entire training set in order to categorize a certain pattern, and that classification performance suffers in the presence of noisy data.

VI. Result Analysis

A PC with a 2.81 GHz processor and 16 GB of RAM was used to implement both the proposed character recognition model and the state-of-the-art models. The EnDNN model is implemented in Python to evaluate its recognition accuracy of handwritten characters and numerals. Training and validation accuracy are computed for the three different CNN, EnDNN, and WEnDNN models. Three standard benchmark datasets of handwritten Odia characters are used to evaluate the performance of the proposed model. Table I lists the number of samples used in training and testing for each dataset. The accuracy is calculated by using a confusion matrix and is defined as the number of correct predictions by the classifier based on the total number of predictions. For our study, in a random forest model, 50 decision trees are ensembled. The handcrafted features for our study were a combination of pixel intensity values (PV) as well as features extracted from Gabor filters (GF). The Gabor filters are applied to original images with the following parameters: Image size (x, y) = (9,9) $\sigma = 1.5,\ 2,\ 2.5$ $\theta = 45^0, 90^0, 135^0, \lambda = 0.79$, $\gamma = 0.5$ and $\psi = 0$. The classification result obtained from different experiments by these models with handcrafted and non-handcrafted feature descriptors is shown in Table III.

TABLE III. PERFORMANCE ANALYSIS OF THE PROPOSED MODEL

Approach	Descriptor	Datasets		
		ISI Image – numeral (10 classes)	IITBBS-numeral (10 classes)	NITROHCS – character (47 classes)
1. Handcrafted features with ML algorithms	Pixel value +Gabor + RF	95.80	90.34	91.93
	Pixel value + Gabor + SVM	94.81	89.52	90.62
	Pixel value +Gabor + XG Boost	94.21	91.21	90.01
	Pixel value +Gabor + kNN	92.51	88.78	89.85
2.Non-handcrafted feature with VGG16 + ML algorithms	VGG_{16} + RF	98.61	93.37	93.59
	VGG_{16} + SVM	98.21	92.54	88.58
	VGG_{16} + XG Boost	98.67	95.03	93.35
	VGG_{16} + k NN	98.50	92.63	88.36
3.Non-handcrafted feature with NN	CNN_1	97.92	96.31	96.11
	CNN_2	98.12	95.40	94.73
	CNN_3	96.33	95.73	95.19
4.Ensemble of Deep learning models	EnDNN	98.33	96.13	96.31
5.Weighted Ensemble of Deep learning models	WEnDNN	98.78	96.81	96.45

Table III and Fig. 9 investigate the performance of the WEnDNN model under different datasets. The proposed state-of-the-art model's performance was also compared with other approaches (1-5). The experimental results highlighted that the proposed WEnDNN model has gained maximum recognition performance for all handwritten characters as well as numeral recognition. This is because each model in the ensemble might focus on different aspects of the input data, capturing distinct patterns and characteristics of handwritten characters. Each CNN model in an ensemble learns different representations or features from the input data. Combining these diverse representations allows the ensemble to capture a broader range of patterns and variations in handwritten characters. By combining the strengths of multiple models, the ensemble can achieve a more comprehensive and discriminative representation of the handwritten characters, leading to improved classification performance in EnDNN. In case of WEnDNN, the weights assigned to CNN models can be dynamically adjusted based on their performance on validation data or during training. By continuously adjusting the weights, the WEnDNN optimizes its performance, leading to improved classification accuracy.

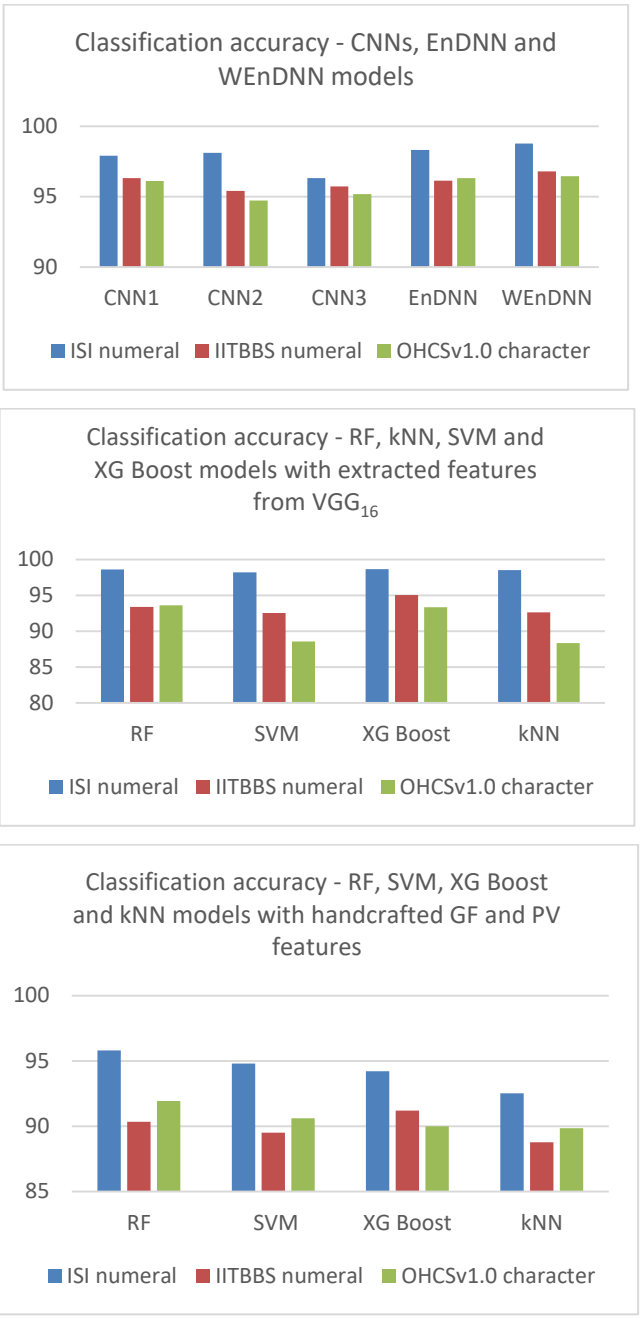


Fig. 9. Result analysis of proposed ensemble model and different ML models with hand-crafted and non-handcrafted features.

The training and validation accuracy as well as training and validation loss of CNN1, CNN2, and CNN3 for the ISI image database are shown in Fig. 11. The confusion matrix and fraction of incorrect predictions of the proposed WEnDNN for the ISI Image numeral database are shown in Fig. 12(a). Structural difference between the numerals present in Odia language is shown in Fig. 10, which leads to more misclassification results. From Fig. 12(b), it is clear that the incorrect prediction of the numeral six( ) is more compared to other numerals, as six( ) is predicted as nine ( ), three( ) or seven( ).

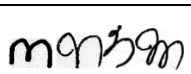
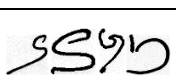
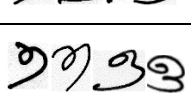
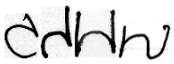
One (eka)	
Three (tini)	
Six (chha)	
Seven (sata)	
Nine (na)	

Fig. 10. Structurally different numerals in ISI Kolkata Image database.

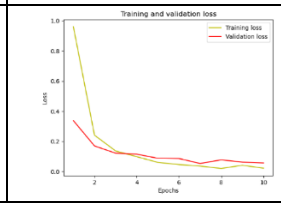
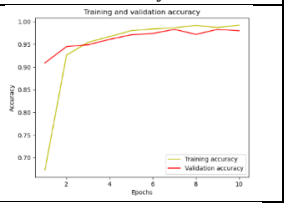
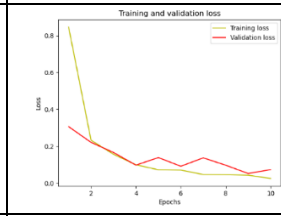
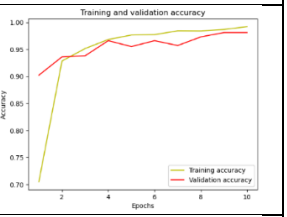
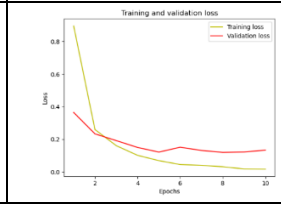
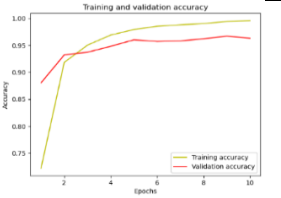
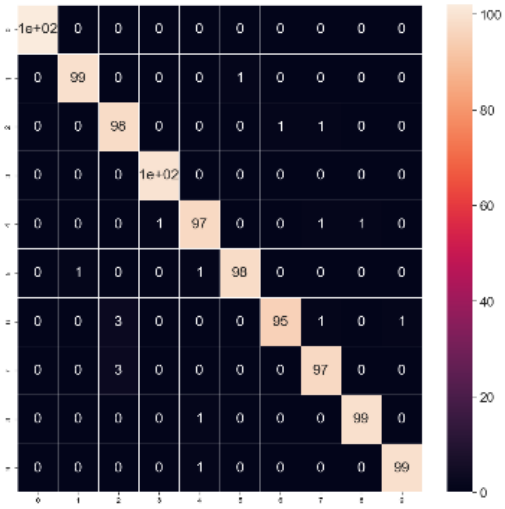
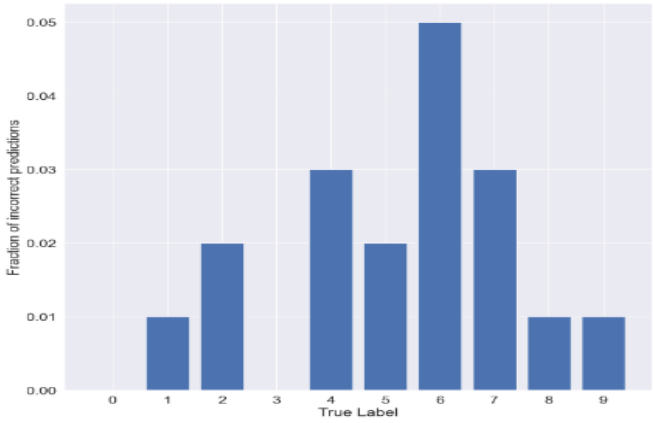
CNN#	Training and validation loss	Training and validation accuracy
CNN_1		
CNN_2		
CNN_3		

Fig. 11. The training and validation accuracy and loss at each epoch of ISI numeral database.



(a)



(b)

Fig. 12. (a) Confusion matrix of WEnDNN. (b) Fraction of incorrect predictions of WEnDNN.

WEnDNN can also handle the noise in a more effective way. When noise is present in the data, different models may have different levels of sensitivity to noise. By combining their predictions with appropriate weights, the ensemble can reduce the impact of noise by relying more on the predictions of models that are less affected by the noise. If a particular CNN model is more susceptible to noise and tends to produce incorrect predictions for noisy samples, its weight can be reduced. Other models that are more accurate in the presence of noise can be assigned higher weights. By giving more importance to the predictions of robust models, the ensemble can mitigate the effects of noise and make more reliable classifications.

A comparison among the performance of ensemble model applied on applications is shown in Table IV.

TABLE IV. DIFFERENT ENSEMBLE LEARNING APPLICATIONS

Reference	Application field	Method	# of classes	Accuracy
[42]	skin disease	Ensemble CART, SVM, DT, RF, GBDT	6	95.9%
[43]	skin disease	Ensemble using Bagging, AdaBoost and Gradient Boosting classifier techniques; PAC, LDA, RNC, BNB, NB, ETC	6	98.56% - Bagging 99.25% - AdaBoost 99.68% - Gradient Boosting
[44]	OCR	Ensemble of Decision Trees, Random Forest, Extra Trees Classifier, MLP, and SVM for the detection of printed regions in an invoice	-	94.53%
[31]	OCR	Ensemble of 30 deep convolutional neural network model was constructed using a stacking method	35	98.85%
[45]	Cloud image patches	Ensemble of 10 CNN's (4 CONV	5	99.40%

		and POOL layers and 3FC layers)		
[46]	IoT Cyber Attacks	An Ensemble of Deep RNN for the detection of IoT Cyber Attacks by 6 LSTM models	2	99.41%
[47]	Cardiovascular Disease	An ensemble-based approach of machine learning and deep learning models	2	88.70%
[48]	Plant disease	An ensemble of Random Forest and K-Nearest Neighbor (KNN)	3	96.00%

VII. Conclusion

This study makes an effort to identify the handwritten atomic Odia character and numeral. The handwritten characters causes the biggest issue in the character recognition procedure due to the freestyle writings of the individual and varies from person to person. There are many different shapes and orientations that a letter can take. Firstly, CNN models are designed from scratch, then a learning-based weighted average ensemble of deep neural network models (WEnDNN) is proposed to classify 10 digits and 47 characters present in alphabet set of Odia language and to enhance the accuracy. The performance of proposed WEnDNN model with EnDNN and machine learning models, namely, like RF, SVM, k NN and XG Boost trained on hand-crafted extracted features by using Gabor filter and pixel intensity values, non-handcrafted extracted features from pre-trained VGG16 neural network are compared. The proposed WEnDNN OHCNR model's overall accuracy recorded as 98.78% on ISI image numeral database. In comparison to state-of-the-art techniques, it has been found that the suggested method offers superior recognition accuracy. These ensemble models can be extended to continuously learn and adapt to changing data patterns over time. This could involve online learning approaches where the ensemble is updated incrementally as new data becomes available, allowing it to stay relevant and effective in dynamic environments.

References

[1] M. Das, M. Panda, and S. Dash, "Enhancing the Power of CNN Using Data Augmentation Techniques for Odia Handwritten Character Recognition," *Advances in Multimedia*, vol. 2022, 2022, doi: 10.1155/2022/6180701.

[2] Das A, Patra G A, and Mohanty M N, "LSTM based Odia Handwritten Numeral Recognition," in *International Conference on Communication and Signal Processing, July 28 - 30, 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020. doi: 10.1109/ICECCT.2017.8117879.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.

[4] K. S. Dash, N. B. Puhan, and G. Panda, "Odia character recognition: a directional review," *Artif Intell Rev*, vol. 48, no. 4, pp. 473–497, Dec. 2017, doi: 10.1007/s10462-016-9507-5.

[5] R. K. Mohapatra, B. Majhi, and S. K. Jena, "Printed Odia digit recognition using finite automaton," *Smart Innovation, Systems and Technologies*, vol. 43, pp. 643–650, 2016, doi: 10.1007/978-81-322-2538-6_66.

[6] D. Basa and S. Meher, "Handwritten Odia Character Recognition," no. July 2015, pp. 5–8, 2011.

[7] I. Rushiraj, S. Kundu, and B. Ray, "Handwritten character recognition of Odia script," *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, pp. 764–767, 2017, doi: 10.1109/SCOPES.2016.7955542.

[8] U. Pal, T. Wakabayashi, N. Sharma, and F. Kimura, "Handwritten numeral recognition of six popular Indian scripts," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2, pp. 749–753, 2007, doi: 10.1109/ICDAR.2007.4377015.

[9] K. S. Dash, N. B. Puhan, and G. Panda, "BESAC: Binary External Symmetry Axis Constellation for unconstrained handwritten character recognition," *Pattern Recognit Lett*, vol. 83, pp. 413–422, 2016, doi: 10.1016/j.patrec.2016.05.031.

[10] D. Padhi, "A Novel Hybrid approach for Odiya Handwritten Character recognition System," *IJARCSSE*, vol. 2, no. 5, pp. 150–157, 2012.

[11] T. K. Mishra, B. Majhi, P. K. Sa, and S. Panda, "Model based odia numeral recognition using fuzzy aggregated features," *Front Comput Sci*, vol. 8, no. 6, pp. 916–922, 2014, doi: 10.1007/s11704-014-3354-9.

[12] P. G. Dash Kalyan S, Puhan N.B., "Non Redundant Stockwell Transform Based Faeture Extraction For Handwritten Digit Recognition," *IEEE International Conference in Signal Processing and Communications*, 2014, [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84911969452&partnerID=tZOtx3y1

[13] P. KSarangi, A. K Sahoo, and P. Ahmed, "Recognition of Isolated Handwritten Oriya Numerals using Hopfield Neural Network," *Int J Comput Appl*, vol. 40, no. 8, pp. 36–42, 2012, doi: 10.5120/4986-7250.

[14] P. K. Sarangi, P. Ahmed, and K. K. Ravulakollu, "Naïve Bayes Classifier with LU Factorization for Recognition of Handwritten Odia Numerals," *International Journal of Science and Technology*, vol. 7, no. January, pp. 35–38, 2014.

[15] M. Das, M. Panda, and S. Dash, "4 A Comparative Analysis of Machine Learning Techniques for Odia Character Recognition," *Machine Learning Applications*, pp. 65–90, Apr. 2020, doi: 10.1515/9783110610987-006.

[16] T. K. Mishra, B. Majhi, and S. Panda, "A comparative analysis of image transformations for handwritten Odia numeral recognition," *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, pp. 790–793, 2013, doi: 10.1109/ICACCI.2013.6637276.

[17] M. K. Mahato, A. Kumari, and S. Panigrahi, "A System For Oriya Handwritten Numeral Recognition For Indian Postal Automation," *IJASTRE*, pp. 1–15, 2014.

[18] N. Tripathy, M. Panda, and U. Pal, "System for Oriya handwritten numeral recognition," *SPIE Proceedings*, vol. 5296, pp. 174–181, 2004.

[19] C. Mitra and A. K. Pujari, "Directional Decomposition for Odia Character Recognition," Springer, Cham, 2013, pp. 270–278. doi: 10.1007/978-3-319-03844-5_28.

[20] B. Majhi, J. Satpathy, and M. Rout, "Efficient recognition of Odiya numerals using low complexity neural classifier," *Proceedings - 2011 International Conference on Energy, Automation and Signal, ICEAS - 2011*, pp. 140–143, 2011, doi: 10.1109/ICEAS.2011.6147094.

[21] K. S. Dash, N. B. Puhan, and G. Panda, "On extraction of features for handwritten Odia numeral recognition in transformed domain," *ICAPR 2015 - 2015 8th International Conference on Advances in Pattern Recognition*, pp. 0–5, 2015, doi: 10.1109/ICAPR.2015.7050694.

[22] P. K. Sarangi and P.Ahemad, "Recognition of Handwritten Odia Numerals Using Artificial Intelligence Techniques," *International Journal of Computer Science and Applications*, vol. 2, no. 02, pp. 41–48, 2013.

[23] U. Pal, T. Wakabayashi, and F. Kimura, "A system for off-line oriya handwritten character recognition using curvature feature," *Proceedings - 10th International Conference on Information Technology, ICIT 2007*, pp. 227–229, 2007, doi: 10.1109/ICOIT.2007.4418301.

[24] R. Panda, S. Das, S. Padhy, S. Palo, and P. Suman, "Complex Odia Handwritten Character Recognition using Deep Learning Model," in *Proceedings of 2022 IEEE International Conference of Electron Devices Society Kolkata Chapter, EDKCON 2022*, Institute of Electrical and

Electronics Engineers Inc., 2022, pp. 479–485. doi: 10.1109/EDKCON56221.2022.10032934.

[25] T. Ghosh *et al.*, "Bangla handwritten character recognition using mobilenet v1 architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2547–2554, Dec. 2020, doi: 10.11591/eei.v9i6.2234.

[26] J. Fairiz Raisa, M. Ulfat, A. Al Mueed, and M. Abu Yousuf, "Handwritten bangla character recognition using convolutional neural network and bidirectional long short-term memory," in *Advances in Intelligent Systems and Computing*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 89–101. doi: 10.1007/978-981-33-4673-4_8.

[27] Tapotosh Ghosh, M. H. Z. Abedin, H. Al Banna, N. Mumenin, and M. Abu Yousuf, "Performance Analysis of State of the Art Convolutional Neural Network Architectures in Bangla Handwritten Character Recognition," *Pattern Recognition and Image Analysis*, vol. 31, no. 1, pp. 60–71, Jan. 2021, doi: 10.1134/S1054661821010089.

[28] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (Cnn)," *Sensors (Switzerland)*, vol. 20, no. 12, pp. 1–18, Jun. 2020, doi: 10.3390/s20123344.

[29] Onan, Aytug. "Ensemble of classifiers and term weighting schemes for sentiment analysis in Turkish." *Scientific Research Communications* 1, no. 1 (2021).

[30] M. Das and M. Panda, "An ensemble method of feature selection and classification of Odia characters," *1st Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology, ODICON 2021*, 2021, doi: 10.1109/ODICON50556.2021.9428979.

[31] H. Guo, Y. Liu, D. Yang, and J. Zhao, "Offline handwritten Tai Le character recognition using ensemble deep learning," *Visual Computer*, vol. 38, no. 11, pp. 3897–3910, 2022, doi: 10.1007/s00371-021-02230-2.

[32] S. Sun, S. Wang, and Y. Wei, "A new ensemble deep learning approach for exchange rates forecasting and trading," *Advanced Engineering Informatics*, vol. 46, no. July, p. 101160, 2020, doi: 10.1016/j.aei.2020.101160.

[33] K. M. R. Alam, N. Siddique, and H. Adeli, "A dynamic ensemble learning algorithm for neural networks,"*Neural Comput Appl*, vol. 32, no. 12, pp. 8675–8690, 2020, doi: 10.1007/s00521-019-04359-7.

[34] J. Lee, S. K. Lee, and S. il Yang, "An Ensemble Method of CNN Models for Object Detection," *9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018*, pp. 898–901, 2018, doi: 10.1109/ICTC.2018.8539396.

[35] L. Nanni, Y. M. G. Costa, R. L. Aguiar, R. B. Mangolin, S. Brahnam, and C. N. Silla, "Ensemble of convolutional neural networks to improve animal audio classification," *EURASIP J Audio Speech Music Process*, vol. 2020, no. 1, 2020, doi: 10.1186/s13636-020-00175-3.

[36] P. Wu, X. Meng, and L. Song, "A novel ensemble learning method for crash prediction using road geometric alignments and traffic data," *Journal of Transportation Safety and Security*, vol. 12, no. 9, pp. 1128–1146, 2020, doi: 10.1080/19439962.2019.1579288.

[37] U. Bhattacharya and B. B. Chaudhuri, "Databases for research on recognition of handwritten characters of Indian scripts," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2005, pp. 789–793, 2005, doi: 10.1109/ICDAR.2005.84.

[38] K. S. Dash, N. B. Puhan, and G. Panda, "Odia character recognition: a directional review," *Artif Intell Rev*, vol. 48, no. 4, pp. 473–497, 2017, doi: 10.1007/s10462-016-9507-5.

[39] R. K. Mohapatra, T. K. Mishra, S. Panda, and B. Majhi, "OHCS: A database for handwritten atomic Odia Character Recognition," *2015 5th National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, NCVPRIPG 2015*, 2016, doi: 10.1109/NCVPRIPG.2015.7490020.

[40] R. K. Mohapatra, "Handwritten Character Recognition of a Vernacular Language : The Odia Script Handwritten Character Recognition of a Vernacular Language : The Odia Script".

[41] D.Gabor, "Theory_of_communication_Part_1_The_analy-1," 1946.

[42] A. K. Verma, S. Pal, and S. Kumar, "Classification of skin disease using ensemble data mining techniques," *Asian Pacific Journal of Cancer Prevention*, vol. 20, no. 6, pp. 1887–1894, 2019, doi: 10.31557/APJCP.2019.20.6.1887.

[43] A. K. Verma, S. Pal, and S. Kumar, "Comparison of skin disease prediction by feature selection using ensemble data mining techniques," *Inform Med Unlocked*, vol. 16, no. April, p. 100202, 2019, doi: 10.1016/j.imu.2019.100202.

[44] L. Abhishek, "Optical character recognition using ensemble of SVM, MLP and extra trees classifier," *2020 International Conference for Emerging Technology, INCET 2020*, pp. 7–10, 2020, doi: 10.1109/INCET49848.2020.9154050.

[45] V. H. Phung and E. J. Rhee, "A High-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences (Switzerland)*, vol. 9, no. 21, 2019, doi: 10.3390/app9214500.

[46] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K. K. R. Choo, and R. M. Parizi, "An Ensemble of Deep Recurrent Neural Networks for Detecting IoT Cyber Attacks Using Network Traffic," *IEEE Internet Things J*, vol. 7, no. 9, pp. 8852–8859, 2020, doi: 10.1109/JIOT.2020.2996425.

[47] A. Alqahtani, S. Alsubai, M. Sha, L. Vilcekova, and T. Javed, "Cardiovascular Disease Detection using Ensemble Learning," *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/5267498.

[48] M. Peker, "Multi-channel capsule network ensemble for plant disease detection," *SN Applied Sciences*, vol. 3, no. 7. 2021. doi: 10.1007/s42452-021-04694-2.