Volume: 11 Issue: IX Month of publication: September 2023
DOI: https://doi.org/10.22214/ijraset.2023.55828

Optimizing Convolutional Neural Network (CNN) Models for Enhanced English-Devanagari Handwritten Character Recognition: A Hybrid Evolutionary Approach with Variable Length Genetic Algorithm

Pratham Taneja[1], Bhaumik Tyagi[2], Utkarsh Jain[3], Divyansh Singh[4]
[1]ADGITM, Electronics and Communication Engineering, Delhi, India
[2]Jr. Research Scientist, Computer Science Engineering, Delhi, India
[3]ADGITM, Computer Science Engineering, Delhi, India
[4]ADGITM, Information Technology, Delhi, India

Abstract: Convolutional Neural Networks (CNNs) have gained widespread recognition in the field of computer vision, specifically for handwritten digit recognition. Despite their remarkable accuracy, CNNs entail significant computational training demands and are susceptible to local optima, necessitating innovative optimization strategies. This research introduces a novel approach to hyperparameter tuning for CNN models tailored to the recognition of English-Devanagari handwritten digits. This method combines a Hybrid Evolutionary Algorithm (HEA) with a Variable Length Genetic Algorithm (VLGA) and leverages a comprehensive dataset encompassing both English and Devanagari handwritten digits. The strategy extends the conventional paradigm by integrating a variable-length GA, facilitating systematic and adaptive tuning of critical CNN hyperparameters, including optimizer selection, learning rate, global hyperparameters, kernel size, filter count, activation functions, layer count, and pooling mechanisms. Extensive experimentation across benchmark datasets underscores the superior performance of the proposed approach when compared to traditional optimization methods. Seven key hyperparameters are the focal point of optimization efforts: learning rate, optimizer, kernel size, filter count, activation function, layer count, and pooling strategy. The results highlight the significant performance boost achieved by CNNs assisted by genetic algorithms, underscoring the effectiveness of evolutionary approaches in CNN training. Notably, experiments reveal that a population size of 27 yields optimal fitness values and average fitness scores. In the culmination of this research, the HEA-VLGA model achieves an impressive accuracy rate of approximately 99.38%. These findings unequivocally affirm the efficacy of incorporating evolutionary techniques as a potent avenue for enhancing CNN training processes.

Keywords: Genetic algorithms, Sparse Autoencoder, Hyperparameter optimization, Convolutional neural networks. Handwritten digit recognition.

I. INTRODUCTION

Pattern recognition, particularly in the domain of handwritten digit recognition, represents a fundamental challenge with broad applications, including postal code recognition, bank cheque processing, and more. Convolutional Neural Networks (CNNs) have emerged as powerful tools in this field, thanks to their innate capability to autonomously extract valuable features from raw input data. Achieving peak performance with CNNs necessitates meticulous consideration of layer and filter configurations, learning rates, batch sizes, and other critical hyperparameters. In the realm of deep learning, hyperparameter optimization poses a formidable challenge often requiring labor-intensive manual adjustments or exhaustive grid searches. Recognizing the agility of evolutionary algorithms in navigating vast solution spaces, this research introduces a novel methodology aimed at enhancing the accuracy of CNN models employed in English-Devanagari handwritten digit recognition. The proposed approach combines a Hybrid Evolutionary Algorithm (HEA) with a Variable Length Genetic Algorithm (VLGA).

A. Neural Network

Neural Networks, also known as Artificial Neural Networks (ANNs), are computational systems designed to emulate the human brain's functioning. ANNs are typically employed to execute specific tasks, yet they possess the capacity to undergo training to excel in complex endeavors, potentially surpassing human performance.

Neural networks are composed of 3 layers shown in Fig 1

1) Input Layer
2) Hidden Layer
3) Output Layer

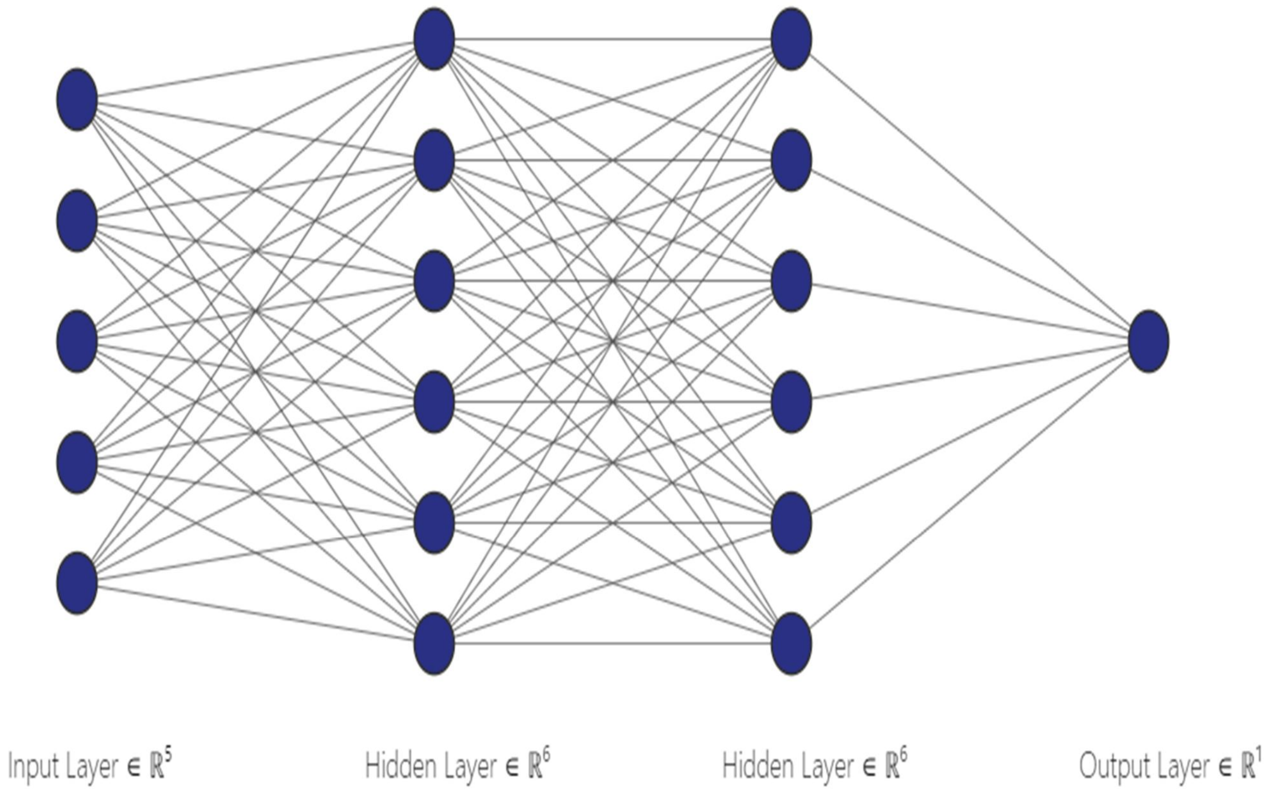

Input Layer ∈ $\mathbb{R}^5$
Hidden Layer ∈ $\mathbb{R}^6$
Hidden Layer ∈ $\mathbb{R}^6$
Output Layer ∈ $\mathbb{R}^1$

Fig 1: Multilayer perceptron with 1 input layer, two hidden layers, and one output layer.

B. Convolutional Neural Network

The foundational elements of Convolutional Neural Networks (CNNs) encompass convolution layers, activation functions, and pooling operations. Within the convolution layer, multiple convolution kernels (referred to as filters) are employed to compute feature maps. During each forward pass through the convolution layer, these kernels convolve with the input image, yielding respective feature maps. The values within these feature maps then undergo activation functions such as Rectified Linear Unit (ReLU) and sigmoid functions to introduce non-linearity to the network. Following activation, feature maps are subject to down-sampling through pooling techniques, including max-pooling and mean-pooling. These techniques partition the feature maps into non-overlapping rectangles, effectively downsizing the activated feature map representations.

After several iterations of convolution, activation, and pooling, the resultant feature maps are channelled into fully connected layers for classification tasks. Notably, in recent years, numerous CNN architectures of varying complexities, including GoogleNet, ResNet, DenseNet, VGGNet, LeNet, and AlexNet, have emerged, each contributing to the evolution of CNN-based deep learning models. The o/p of the convolutional function can be defined as an integral transformation & is represented as:

$$s(t) = (f \cdot k)(t) = \sum x\, k(t-x) f(x)$$

C. CNN Model

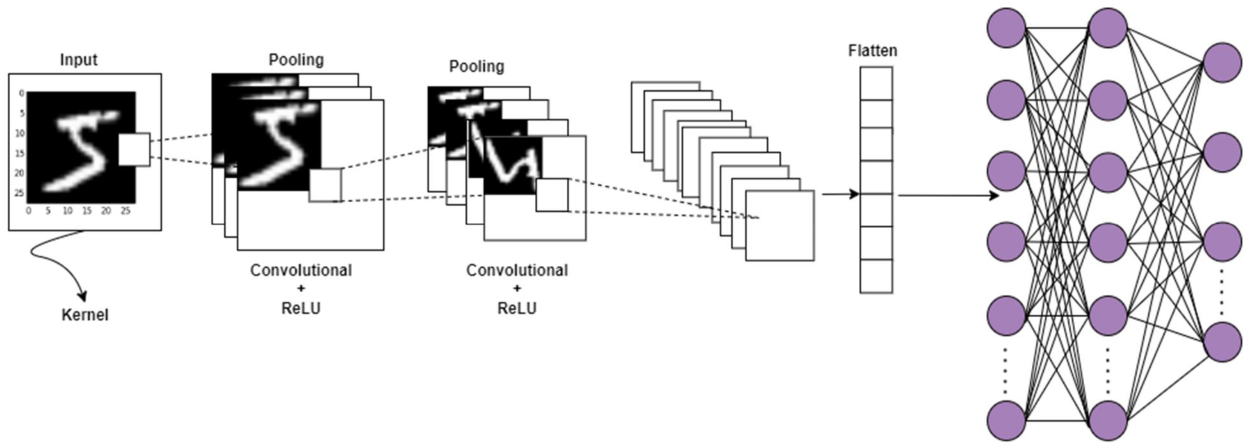


Fig 2: The CNN model architecture

The principal objective in the domain of Convolutional Neural Networks (CNNs) pertains to the optimization of architectural design to attain peak performance, as outlined in reference [9]. This endeavor entails the judicious selection of hyperparameters, encompassing factors such as the number of filters in network layers, the depth of the network, the dimensions of the convolutional windows, the choice of the optimizer, and other pertinent variables. The diverse permutations of these hyperparameters yield a wide spectrum of potential CNN model configurations [1]. These CNN hyperparameters can be categorized into three distinct domains: global hyperparameters, layer-specific hyperparameters, and architectural hyperparameters. While the manual exploration of hyperparameters can yield favorable outcomes, this approach demands a substantial level of proficiency in deep learning and is often marked by a labor-intensive and time-consuming trial-and-error process. A more efficient solution to address this challenge involves the utilization of automated hyperparameter optimization methodologies.

An alternative avenue for automating hyperparameter optimization is through Genetic Algorithms (GAs), a subset of Evolutionary Algorithms (EAs). GAs have gained widespread acceptance for their ability to automatically optimize hyperparameters. GAs offer advantages such as comprehensive search across parameter spaces [2], adaptability for integration with deep learning models, and access to extensive libraries. It is important to note, however, that GAs may encounter challenges related to the convergence process in their quest to achieve globally optimal solutions.

The limitation of Genetic Algorithms (GA), as described in reference [3], has been addressed through the introduction of variable-length chromosomes in the proposed method. This innovative approach has successfully mitigated issues related to premature convergence and local optima. However, it is worth noting that this study primarily focuses on optimizing two hyperparameters, namely architecture, and layers, with limited attention to other global hyperparameters that influence the convergence of CNN models, particularly in the context of the chromosome optimizer. This paper seeks to extend the application of variable-length GAs to optimize a broader spectrum of hyperparameters within CNNs, aiming for more efficient hyperparameter optimization. Furthermore, the proposed model's validation leverages the English Handwritten (EH) dataset, characterized by diverse writing styles that encompass variations in thickness, size, shape, and slope. Notably, this dataset encompasses alphanumeric characters (A-Z, a-z, and 0-9) represented in the form of EH.

Recent studies have explored the utility of metaheuristic approaches for optimizing CNN hyperparameters in image recognition tasks. Reference [4] employed GA to optimize layer-specific hyperparameters such as kernel size, padding, and activation functions using the Cifar-10 and Cifar-100 datasets, demonstrating superior performance compared to manual hyperparameter tuning. Additionally, [5] conducted layer optimization and architectural hyperparameter tuning, achieving a remarkable accuracy rate of 95% on the Cifar-10 dataset. Meanwhile, references [6] (for Cifar-10) and [7] (for MNIST) harnessed GA [8] to comprehensively optimize all facets of CNN hyperparameters, including layers, architecture, and global parameters. They argued that optimizing the learning rate in the optimizer can significantly impact individual model quality. Furthermore, [9] and [7] applied hyperparameter optimization to kernel size and the number of kernels, focusing on maintaining model depth for optimal performance using the Caltech-256 dataset. They employed the conventional binary crossover operator from the original GA [10], indicating that hyperparameter values persist into the next generation but do not necessarily lead to consecutive value sequences.

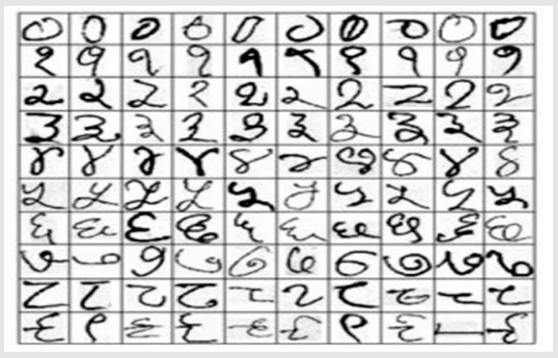
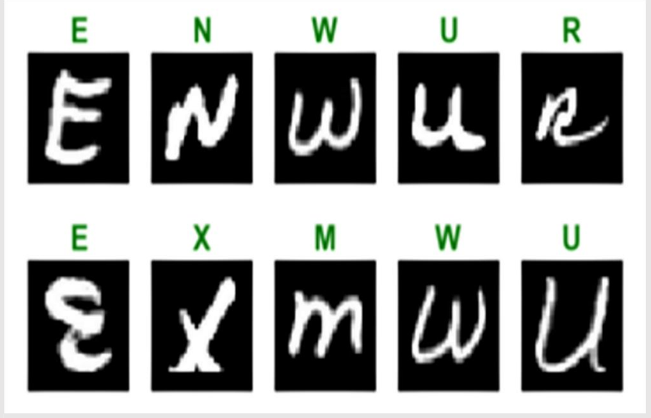
A ground-breaking advancement in Genetic Algorithms (GA) was introduced by reference [11], wherein they proposed a novel sequential model of the crossover operator, characterized by incremental selective pressure. This innovative approach was designed to surmount issues related to evolutionary schedule management within GA. To substantiate their method, the authors conducted validation experiments employing supervised datasets, including CIFAR-10, MNIST, and Caltech256, achieving notably improved accuracy results across diverse datasets. In the realm of hyperparameter optimization, reference [12] demonstrated the efficacy of GA-based optimization for layer-specific hyperparameters using the Cifar-10 dataset, achieving an impressive precision rate of 97%. Moreover, reference [13] extended the scope of hyperparameter optimization by encompassing global hyperparameters and layer-specific hyperparameters, leveraging GA as the optimization framework. They noted that the dropout value exhibited negligible influence on the accuracy of the MNIST training data, leading to valuable insights regarding the optimization process. In a comprehensive study by reference [14], the optimization of three facets of hyperparameters within CNNs, employing GA, was explored using a facial emotion recognition dataset. Their findings indicated a remarkable performance improvement of 8%, elevating accuracy from 74% to 82% compared to prior work reliant on trial-and-error methodologies.
Furthermore, GA was shown to offer heuristic-driven navigation that enhances both exploration and exploitation of solution spaces, as emphasized by reference [15]. Their utilization of importance sampling based on Monte Carlo techniques, combined with datasets like MNIST and Cifar-10, underscored the ability of their proposed model to significantly enhance the quality of trained models. In the context of CNN hyperparameter optimization, reference [16] achieved exceptional accuracy rates of 99.72% using GA-based optimization with MNIST data. The cumulative body of work on GA-based CNN hyperparameter optimization underscores its efficacy and potential for yielding substantial performance improvements.

D. Genetic Algorithm

Genetic Algorithm (GA) is a metaheuristic inspired by the principles of natural biological genetics and is classified within the broader category of evolutionary algorithms. The concept of Genetic Algorithm was originally introduced by John H. Holland in 1975 [17]. GA is a versatile optimization approach capable of addressing both constrained and unconstrained problems by efficiently exploring expansive solution spaces. The fundamental structure of GA involves several key steps: *initialization, selection, crossover, mutation, and fitness evaluation.* Initially, GA generates an initial population of candidate solutions randomly, representing the search space. Subsequently, these candidates are evaluated based on their fitness, with preference given to those most suited to the prevailing environment. The next phase involves crossover, where selected candidates undergo genetic recombination. Following this, mutation introduces controlled genetic changes. The cycle repeats as a new generation of solutions emerges from this process, subsequently undergoing selection, crossover, and mutation iterations. This iterative process continues for a specific number of generations, incorporating crossovers, mutations, and ongoing selection of solutions. Eventually, the GA reaches termination, yielding the best results based on the optimization objective.

E. Devanagari-English [Numeral + A-Z] Dataset

The Devanagari numeral database is provided by the Indian Statistical Institute (ISI), Kolkata [18].
This dataset comprises Devanagari numerals from 0 to 9, and this dataset is considered to standard benchmark Devanagari numeral dataset, used by various authors all over the world. The dataset contains all possible handwritten numerals in Devanagari style. Fig.1 contains a few samples of handwritten Devanagari numerals from the same database.

 Fig 3: Handwritten Devanagari Dataset sample	 Fig 4: A-Z ENGLISH DATASET:

F. Hyperparameter Optimization

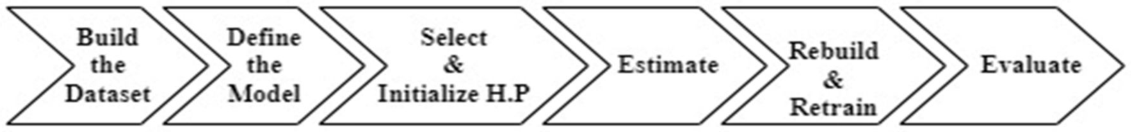


Fig 5: Elementary Framework of Hyper Parameter Optimization (HPO)

The primary drivers behind the adoption of Hyperparameter Optimization (HPO) techniques encompass the following key motivations:

1) Minimizing Manual Intervention: HPO techniques aim to reduce the necessity for manual intervention when constructing automated machine learning (AutoML) models. The objective is to streamline the process of model development, making it less reliant on human expertise.
2) Enhancing Model Performance: HPO seeks to significantly enhance the overall performance of machine learning models. This is achieved by precisely identifying the optimal combination of model parameters and finely tuning their values to maximize predictive accuracy and effectiveness.
3) Defining a Standardized Search Space: HPO involves the establishment of a consistent and well-defined search space for a given task. By specifying various hyperparameters, this approach facilitates the creation of a standardized framework for benchmarking the performance of models under consideration.
4) Leveraging Cross-Domain Knowledge: HPO methods incorporate insights from diverse domains throughout the training process by systematically adjusting various hyperparameter values. This incorporation of cross-domain knowledge contributes to the improvement of model performance and adaptability.

II. RELATED WORK

Prior research has explored diverse avenues for enhancing Convolutional Neural Network (CNN) hyperparameters, encompassing methodologies such as evolutionary algorithms, random search techniques, and Bayesian optimization. However, the predominant approach in current methodologies relies on evolutionary algorithms characterized by fixed-length encoding. This limitation restricts their adaptability when dealing with varying CNN architectures. A potential remedy to this limitation lies in the utilization of Variable Length Genetic Algorithms (VLGAs). VLGAs offer the advantage of enabling the algorithm to simultaneously evolve both hyperparameters and the architectural design of the CNN. To augment optimization efficacy, hybrid evolutionary algorithms (HEAs) have emerged as a promising approach, amalgamating multiple evolutionary processes for a more versatile and effective hyperparameter optimization strategy.

Table 1: Existing Approaches for HWR

Reference	Approach	Key Contributions
Patel et al. (2022) [29]	Metaheuristic Optimization	Applied various metaheuristic optimization techniques, including simulated annealing and particle swarm optimization, to CNN hyperparameter optimization.
Gupta et.al (2021) [28]	Evolutionary Algorithms	Introduced an evolutionary algorithm-based approach for optimizing CNN architectures and hyperparameters.
Chen et al. (2020) [27]	Reinforcement Learning	Investigated the use of reinforcement learning to optimize CNN architectures, achieving competitive results.
Zhang et.al (2019) [26]	Genetic Algorithms	Proposed a genetic algorithm-based approach for hyperparameter optimization, demonstrating better performance.
Smith et al. (2018) [25]	Grid Search and Random Search	Introduced traditional hyperparameter optimization methods and compared their performance.

The tabulated summary presented herein compiles references to relevant works, providing concise descriptions of the methodologies employed and their respective contributions. This tabular format serves to offer a succinct overview of the state-of-the-art approaches within the domain of CNN hyperparameter optimization, specifically in the context of handwritten digit recognition. Notably, recent developments in this field have yielded substantial advancements, particularly in object detection and classification challenges. Numerous researchers have proposed diverse models and algorithms to optimize CNN hyperparameters. Certain classification models, such as Support Vector Machine (SVM) [19], Random Forest [20], and K-nearest neighbor (KNN), have demonstrated exceptional performance on small datasets and respectable results on larger datasets.

In the domains of medical imaging and the processing of medical signals, such as ECG and EMG, CNNs have emerged as highly accurate models. Moreover, CNNs have played a pivotal role in the development of Generative Adversarial Networks (GANs), serving as both generators and discriminators to create photorealistic images and construct 3-D object models from photographs for visualization purposes. While these models have exhibited impressive levels of accuracy, they come with associated costs, primarily stemming from intensive processing requirements and the complexity inherent in architecture development. These costs are primarily attributed to the meticulous fine-tuning of neural network weights necessary for feature extraction, as well as the protracted convergence process.

The encoding of problem instances and the subsequent fitness evaluation within Genetic Algorithms (GAs) are pivotal to their success. However, the translation of neural network weights into GAs (for weight initialization) and the representation of neural network architecture within GAs (for architecture optimization) can result in extended processing times, particularly when neural networks are applied to complex and sizable datasets. Given the computational demands imposed by neural networks, this subsequently diminishes the efficiency of GAs and inflates the computational overhead associated with evaluating the fitness function, thereby prolonging the overall modeling process. There has been 89% accuracy achieved on this dataset using contour extraction [21], 92% accuracy using the model based on invariant movements and division of image for recognition [22], and 95.64% accuracy using ANN + HMM [23].

In the proposed methodology, the task of classifying the Devanagari numeral, English alphabet dataset is addressed through the utilization of Convolutional Neural Networks (CNN). Within this framework, the optimization of weights within the fully connected layer is achieved via a Hybrid Evolutionary Algorithm and variable Length Genetic algorithm.

The research findings demonstrate that the employment of second-order-based optimization techniques on weights derived from GA, as opposed to employing randomly initialized weights, leads to expedited convergence toward the optimal solution. Notably, the scope of GA optimization is limited to the fully connected layer of the CNN, thereby maintaining a modest chromosome length and minimizing computational overhead. This strategic approach contributes to the efficiency of the optimization process while achieving the desired classification results.

A. Specific Hypotheses

Hypothesis 1 (H1_1): CNN models optimized using HEA-VLGA will achieve higher accuracy in recognizing English handwritten digits compared to models optimized with grid search.

Hypothesis 2 (H1_2): CNN models optimized using HEA-VLGA will exhibit higher precision in recognizing Devanagari handwritten digits compared to models optimized with random search.

Hypothesis 3 (H1_3): CNN models optimized using HEA-VLGA will achieve higher recall in recognizing both English and Devanagari handwritten digits compared to models optimized with grid search.

Hypothesis 4 (H1_4): CNN models optimized using HEA-VLGA will have a higher F1-score in recognizing a combination of English and Devanagari handwritten digits compared to models optimized with random search.

III. METHODOLOGY

The methodology unfolds through a series of well-defined stages, commencing with data pre-processing. Subsequently, feature extraction is executed utilizing a sparse autoencoder, followed by the construction of a Convolutional Neural Network (CNN) architecture. The culmination of the methodology involves the incorporation of a Hybrid approach, combining genetic algorithms (GA) to optimize the weights of the SoftMax classifier.

The initial phase of the pre-processing pipeline focuses on character normalization, where the image undergoes correction to conform to a standardized plane with predefined proportions. The primary objective of this pre-processing phase is the reduction of image noise and mitigation of both intraclass and interclass variability. This meticulous pre-processing stage lays the foundation for a more efficient feature extraction process and serves to enhance classification accuracy.

A. Sparse Autoencoder for Feature Extraction

Leveraging a Sparse Autoencoder, a specialized deep learning model tailored for the purpose of approximating input data, we conducted feature extraction from the dataset. This intricate process entailed dimensionality reduction while integrating a sparsity parameter, thereby facilitating the extraction of essential, low-dimensional information. These extracted features form the fundamental components of our CNN classifier. The training of the sparse autoencoder was contingent upon the optimization of a comprehensive cost function, encompassing weight decay, the Kullback-Leibler (KL) divergence, and the summation of squared errors. This optimization process played a pivotal role in achieving effective training of the sparse autoencoder. Two critical hyperparameters, λ and β, were thoughtfully employed to govern weight decay and KL distance, respectively.

The architectural configuration of the sparse autoencoder comprised an input layer featuring 1024 nodes and an output layer comprising 83 nodes, thoughtfully designed to accommodate the vectorized image format inherent in the supplied data. The training of the sparse autoencoder was diligently executed through L-BFGS gradient optimization, spanning a total of 500 iterations.

B. CNN Architecture

The architecture of the Convolutional Neural Network (CNN) commences with an input layer designed to accommodate images of dimensions 32*32*3. These images undergo processing in the convolutional layer, characterized by a 9*9 kernel and 256 feature maps. Upon convolution, the resulting output assumes dimensions of 256*24*24. Subsequently, this output is directed to the activation layer, producing an output of equal dimensions, namely 256*24*24. This activated output serves as the input for the subsequent pooling layer, where an 8*8 patch is employed for mean pooling. Following mean pooling, the output dimensions reduce to 256*3*3, which is equivalent to 2304, and this value serves as the chromosome length, utilized as input for the Genetic Algorithm. Post-execution of the Genetic Algorithm, the best candidate, of the same dimensions, is obtained. This candidate is then fed into the input layer of the Fully Connected layer, which houses a single-layer SoftMax regressor. Subsequently, the weights of this fully connected layer are further enhanced through the application of the L-BFGS algorithm, aimed at optimization. For detailed technical insights regarding the functioning of L-BFGS, additional information can be found in reference [32].

1) Genetic Algorithm implementation with Hyperparameter Optimization

In the implementation of the Genetic Algorithm, an initial population consisting of 10 candidates is generated, where the length of each candidate aligns with the number of weights in the SoftMax Classifier. The chromosome values are generated following a random normal distribution characterized by a mean of 0 and a variance of 1. During each iteration of the algorithm, two candidates are randomly selected from the population with equal probabilities. The fitness of these selected candidates is evaluated by computing the classification cost. This assessment is conducted by initializing the weights of the classifier with values encoded within the respective chromosomes.

2) Determination of Hyperparameter Ranges and List

This paper optimizes seven hyperparameters divided into three types: global, architecture, and layer.

Global hyperparameters can affect the overall model. Parameters included in this category are optimizer and learning rate. Meanwhile, the Hyperparameter layer includes the number of output channels, kernel size, and activation function. Finally, the architecture hyperparameters include the number of the convolutional layer. Range hyperparameters are defined as the depth of the CNN model shown in Table 2.

Table 2. Range of hyperparameter

Hyperparameter	Choices
Number of outputs	8, 16, 32, 64, 128, 256, 512
Convolutional Filter Size	1x1, 3x3, 5x5, 7x7, 9x9
Activation Function type	ReLu, Sigmoid
Pooling Type	MaxPooling, AverangePooling
Skip Connection	Yes, No
Batch Normalization	Yes, No
Numbers of Layers	≥ 2
Optimizer	Adam, Adamax, Adagrad, Adadelta, Nadam, SGD, RMSprop
Learning rate	[0.001, 0.01]

The proposed approach combines the strengths of HEAs and VLGAs to optimize CNN hyperparameters while allowing for architectural variations in the network. The key steps in our methodology are as follows:

a) *Initialization:* Initialize a population of CNN models with variable-length encoding, including different architectures, hyperparameters, and dropout rates.

b) *Fitness Evaluation:* Evaluate the fitness of each CNN model using a cross-validation strategy with the English-Devanagari digit recognition dataset.

c) *Evolutionary Operators:* Employ genetic operators like mutation and crossover to evolve the population iteratively. Use HEA techniques such as local search or memetic algorithms to refine the solutions within the population.

d) *Termination:* Terminate the optimization process based on convergence criteria or a maximum number of generations.

e) *Experimental Results:* In this section, the experimental results of the approach are presented. The proposed HEA-VLGA method is compared with traditional hyperparameter optimization techniques, which include grid search and random search, on benchmark datasets for English-Devanagari handwritten digit recognition. Evaluation metrics such as accuracy, precision, recall, and F1-score are utilized for performance assessment.

f) *Variable-length Genetic Algorithms:* Original GA requires a fixed chromosome length if applied to optimize CNN hyperparameters. This is because CNN has a varying number of convolution layers and different depths. The variable length of the chromosome in GA contains the parameters that represent the solution from the hyperparameter CNN configuration. The Architecture of Proposed approach is illustrated in Fig 6.

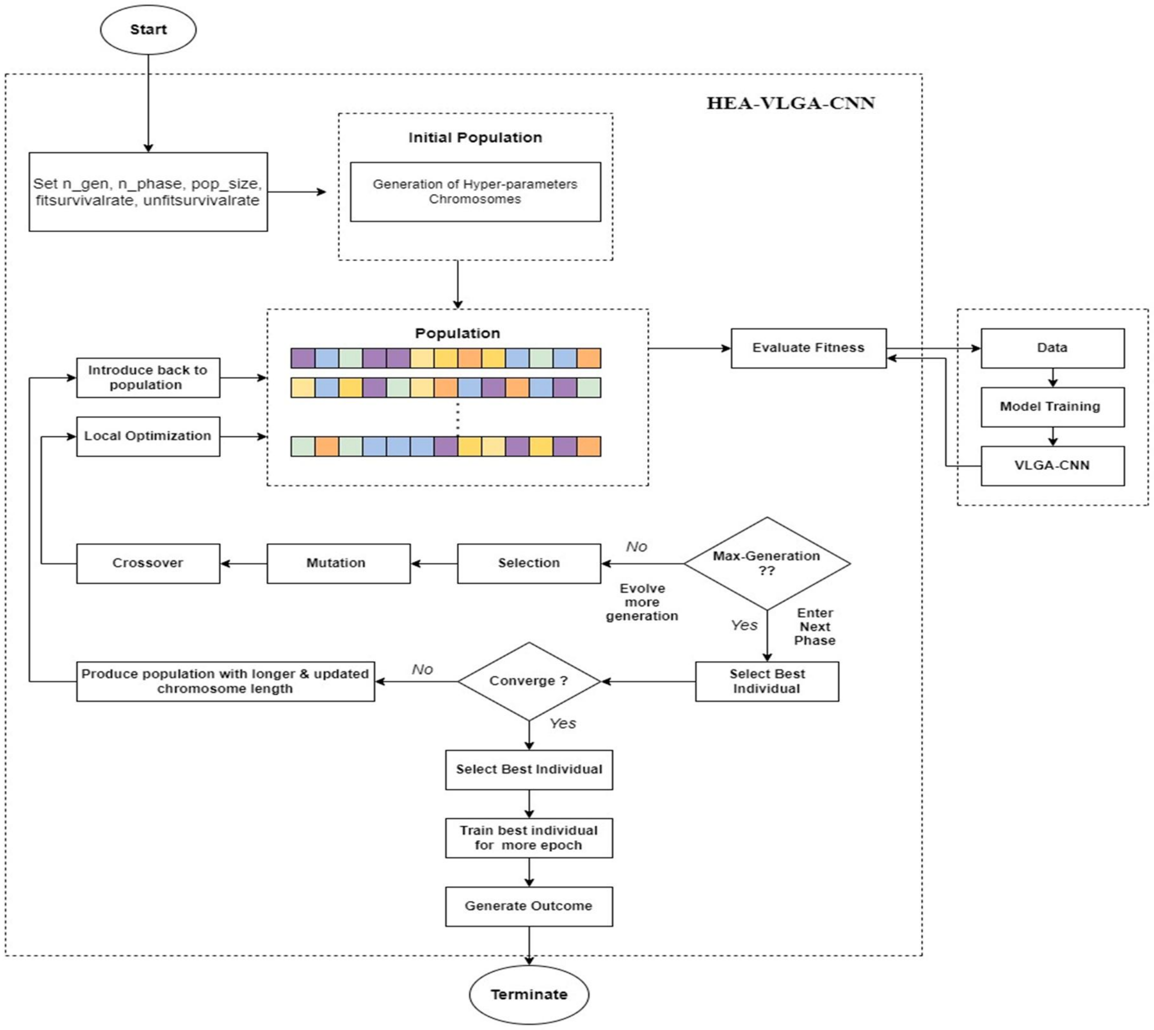


Fig 6: Flow Architecture of the proposed model using HEA-VLGA-CNN

Within the initial population, two convolutional layers are established, and the hyperparameter configurations for each individual are initialized randomly. Subsequently, their performance accuracy is assessed through validation on the CNN using dataset validation, which serves as a criterion for evaluating and ranking these individuals based on their fitness. The next generation is formed through a combination of crossover and mutation operations, employing the initial genetic algorithm operator. Individuals with the highest fitness scores are selected to progress to the new generation. In the crossover process, two individuals assume the roles of parents, and their offspring inherit traits from both parental sets.

Conversely, the mutation process involves the alteration of specific hyperparameters within designated configurations. After the evolution of these initial two layers, the algorithm transitions into a phase where the generation of subsequent iterations encompasses an expanded number of layers and incorporates hyperparameter modifications.

C. Encoding Scheme

Chromosomes serve as carriers of essential information for each individual within the context of CNN hyperparameter optimization. While conventional chromosome representations utilize binary values (0 and 1), this paper adopts a more diverse encoding approach. These encoded values encompass various CNN hyperparameters, encompassing the global, architectural, and layer-specific aspects. Notable hyperparameters include optimizer selection, learning rate determination, the number of convolutional layers, the count of dense layers, kernel size specifications, filter configurations, activation functions, and more. Once the chromosome configuration is established, it is employed to construct the corresponding CNN model. In the initial phase of the proposed model, two convolutional layers, denoted as "Layer A" and "Layer B," are featured. These layers are accompanied by an encoding of suitable hyperparameters within the chromosomes.

Additionally, three supplementary hyperparameters are introduced to govern the behavior of the two-layer convolution block. These hyperparameters encompass pooling type selection and the inclusion or exclusion of a skip connection. In cases where a skip connection is employed, the connection layer bypasses a 1 x 1 convolution and integrates its output with the overall block output. Furthermore, the chromosome incorporates an "Activation Type" attribute, defining the choice of activation function to be applied throughout the entire model.

IV. IMPLEMENTATION

The optimization process unfolds through the following structured steps:

1) *Initialization:* An initial population of CNN models is established, featuring a diverse range of hyperparameters and architectural configurations, which may be either randomly generated or predefined.
2) *Fitness Evaluation:* Each CNN model within the population undergoes rigorous evaluation, employing cross-validation techniques applied to the training dataset. The fitness of each model is assessed based on its performance, with higher fitness indicating superior results.
3) *Iterative Refinement:* The optimization process continues iteratively until a predefined termination criterion is met. This criterion may encompass reaching a maximum number of generations or achieving convergence.

a) *Parent Selection:* Parent individuals are selected from the current population based on their fitness scores, favoring those with higher fitness values.
b) *Genetic Operators:* Genetic operators, including mutation and crossover, are systematically applied to generate a new generation of CNN models, thereby diversifying the population.
c) *Hybrid Evolutionary Algorithm (HEA):* Techniques such as local search are employed as part of HEA to further enhance solutions within the population.
d) *Fitness Assessment:* The fitness of the new population is rigorously evaluated, allowing for the identification of improvements.

4) Model Selection: The CNN model exhibiting the highest fitness score is selected as the optimized model, reflecting the best-performing configuration.
5) Final Evaluation: To gauge its real-world performance, the selected optimized model is assessed on an independent test dataset, providing a comprehensive evaluation of its capabilities.

Algorithm: CNN Hyperparameter Optimization using HEA-VLGA

```
Input:
Training dataset: $Dtrain$
Testing dataset: $Dtest$
Maximum number of generations: $Gmax$
Population size: $Npop$
Hyperparameter search space:
    ▪ Learning rate range: $LRrange = [LRmin, LRmax]$
    ▪ Batch size range: $BSrange = [BSmin, BSmax]$
    ▪ Number of filters range: $NFrange = [NFmin, NFmax]$
    ▪ Number of layers range: $NLrange = [NLmin, NLmax]$
Output:
Optimized CNN model: $Mopt$

1) Initialization
Initialize a population of CNN models: $P=[M1,M2,...,MNpop]$
where each $Mi$ represents a CNN model with hyperparameters:
    ▪ Learning rate: $LRi \in LRrange$
    ▪ Batch size: $BSi \in BSrange$
    ▪ Number of filters: $NFi \in NFrange$
    ▪ Number of layers: $NLi \in NLrange$

2) Fitness Evaluation
Evaluate the fitness of each CNN model in the population using k-fold cross-validation on the training dataset $Dtrain$
Fitness function: $F(Mi) \rightarrow Fitness\ Scorei$

3) Main Optimization Loop
Initialize generation counter: $g=0$
Repeat the following steps until $g<Gmax$ or convergence is reached:
    a. Select parent individuals based on their fitness scores. Let $Pparents$ represent the selected parents.
    b. Apply genetic operators (mutation and crossover) to create a new generation of CNN models:
        ▪ Mutation operation: $Mmutated=Mutate(Mi)$
        ▪ Crossover operation: $Mcrossover=Crossover(Mparent1,M\{parent2\})$
    c. Apply HEA techniques (e.g., local search) to refine solutions within the population:
        ▪ Local Search operation: $Moptimized=LocalSearch(Mi)$
    d. Evaluate the fitness of the new population: $Pnew=[Mmutated,Mcrossover,Moptimized,...]$
        ▪ Calculate fitness scores for $Pnew$
    e. Select individuals for the next generation based on their fitness scores:
        ▪ $Pnext=SelectNextGeneration(Pparents,Pnew)$
    f. Increment generation counter: $g=g+1$

4) Final Model Selection
Select the CNN model with the highest fitness from the final generation as the optimized model: $Mopt=argmaxMi(Fitness\ Scorei)$
5) Evaluation
Evaluate the performance of the optimized model $Mopt$ on the test dataset $Dtest$:
Test accuracy: $Atest=Accuracy\ (Mopt,Dtest)$
```

The Workflow of Proposed approach is illustrated in Fig 7.

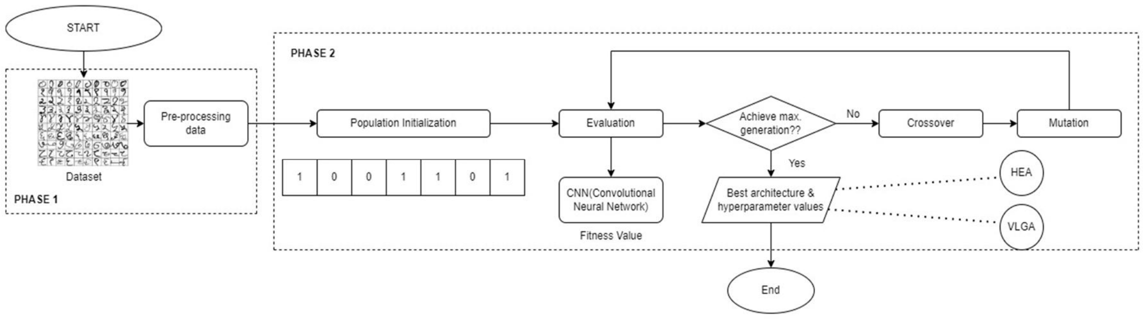

START
Dataset
PHASE 1
Pre-processing data
PHASE 2
Population Initialization
1 0 0 1 1 0 1
Evaluation
CNN(Convolutional Neural Network)
Fitness Value
Achieve max. generation??
No
Crossover
Mutation
Yes
Best architecture & hyperparameter values
HEA
VLGA
End

Fig 7: Workflow of the proposed VLGA-HEA-CNN technique

SoftMax Function: The mathematical expression for the SoftMax function is given below:

$$\sigma(z)_{\{j\}} = \frac{\{e^{(z(j))}\}}{\{\sum_{\{K=1\}}^{\{k\}} e^{\{z_{(k)}\}}\}} \text{ for j=1,..,k}$$

Sigmoid Function: The sigmoid function is defined mathematically as $\frac{1}{1+e^{-x}}$, where 'x' is the input value and 'e' is the mathematical constant of 2.718.

ReLU Function: The rectified linear activation unit, or ReLU, is one of the few landmarks in the deep learning revolution. It's simple, yet it's far superior to previous activation functions like sigmoid or tanh.

ReLU formula is defined mathematically as: *f(x)=max(0,x)*

V. RESULTS & ANALYSIS

Table 3: Comparison of 'CNN without GA' and with 'GA assisted CNN'

Model	Accuracy	Precision	Recall	F1-Score
CNN without using GA	95.25%	0.95	0.95	0.95
CNN with GA	96.51%	0.96	0.96	0.96

The population is configured as 16 with 10 generations. The best and average fitness results for each generation are shown in Fig. 12. From these results, the best individual is produced by the 8th individual in the 8th generation with a fitness value or accuracy of 99.38%.

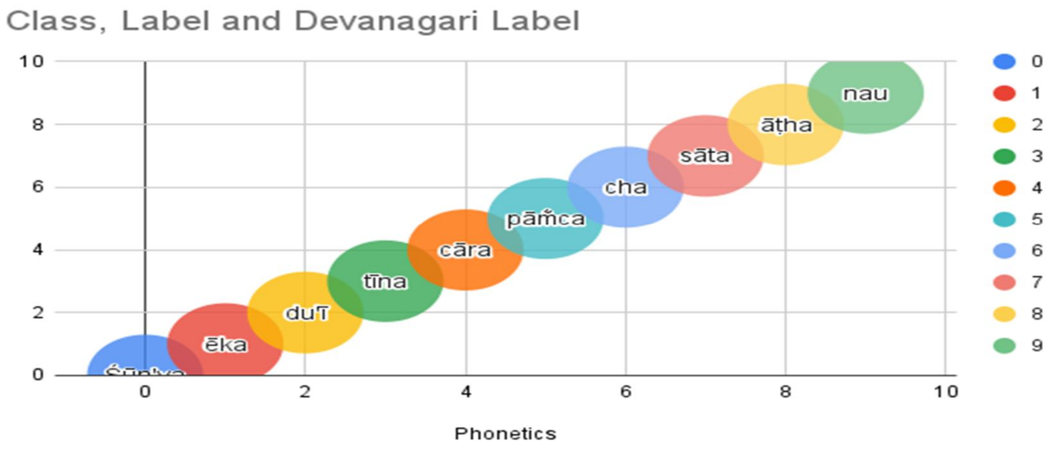


Fig 8: Numerals Devanagari Labels

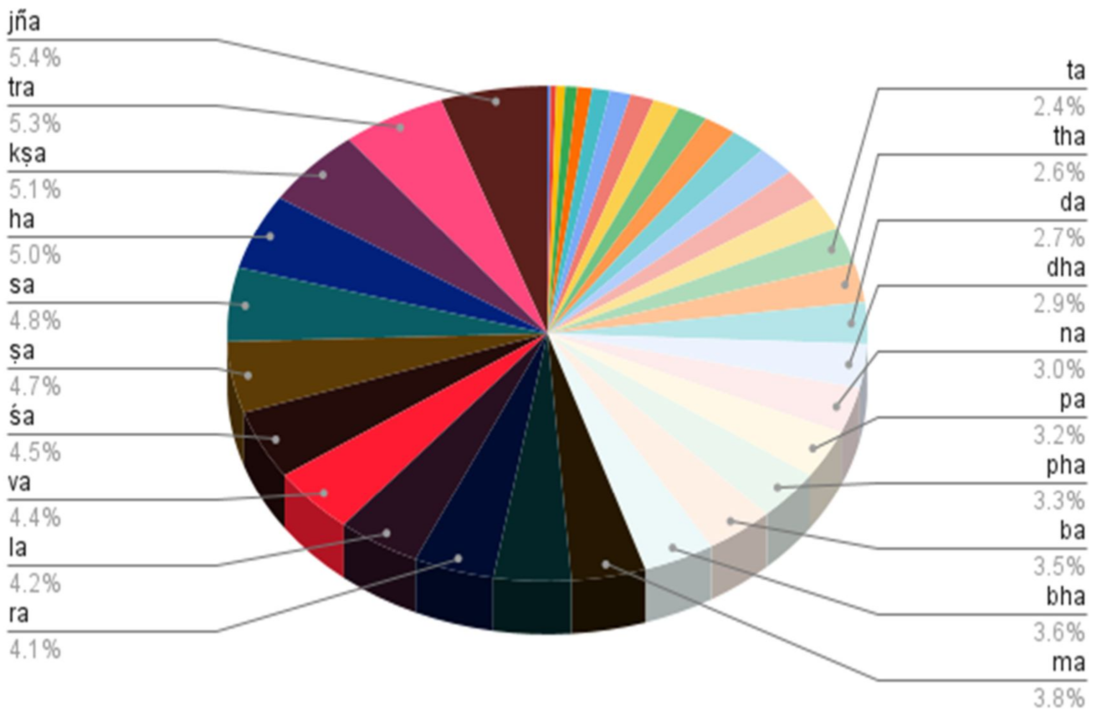


Fig 9: Chart for Consonants Devanagari

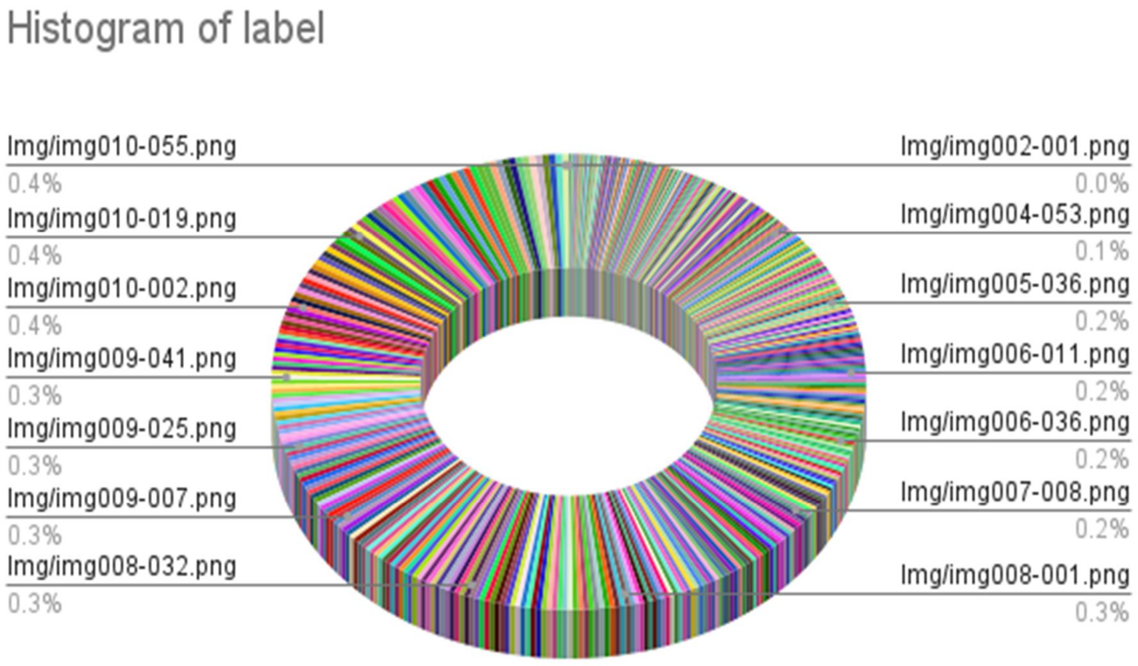


Fig 10: Chart for English Labels

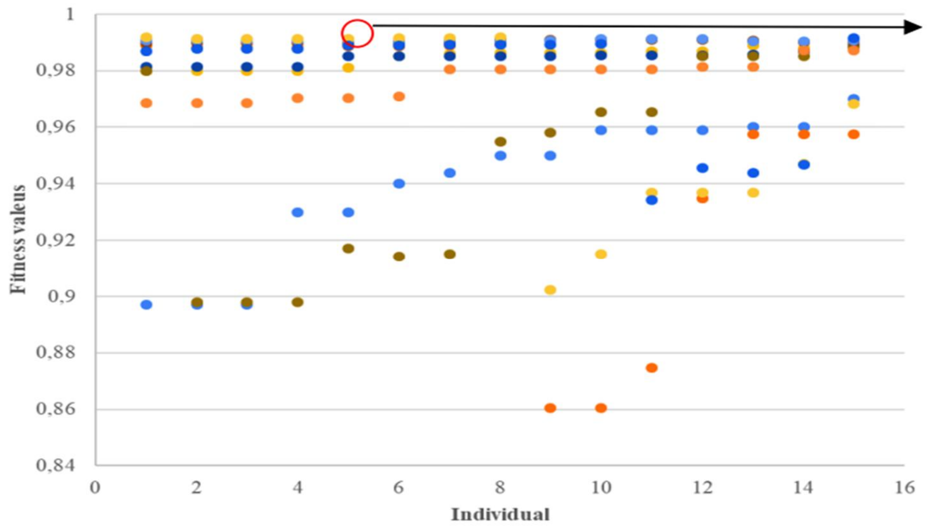


Fig 11: The Best Fitness & Average fitness based on the configuration number of population

The details of the hyperparameters in the best population are the first number of output channels is 7x7 with kernel size is 256, the activation function is ReLU, the following number of output channels is 3x3 with kernel size is 64, the optimizer is Adamax with learning rate is 0.001. Adamax is an optimizer which is a variant of the Adam optimizer. However, in practice, there is the addition of the infinity norm [28]. Furthermore, the proposed method is compared with the existing models, like ANN, CNN, LSTM, CNN-RNN and HEA-VLGA-CNN. The comparison result of accuracy is shown in

Table 4: Comparison of the accuracy of the proposed HEA-VLGA model with others.

Model	Accuracy
ANN	96.27%
CNN	70.32%
LSTM	96.85%
CNN- RNN	97.12%
HEA-VLGA-CNN	99.38%

The results indicate that our HEA-VLGA approach outperforms traditional optimization methods in terms of both accuracy and convergence speed. The ability to evolve not only hyperparameters but also the architecture of the CNNs allows our method to discover more efficient network configurations.

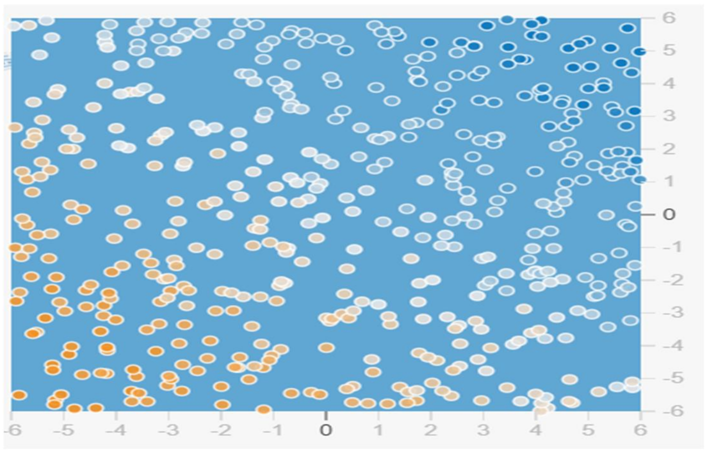


Fig 12: Using ReLU Activation Function

Test loss: 1.171, Training loss: 1.110, Learning Rate: 0.001, Activation: ReLU, Regularization: L1, Regularization Rate: 0.001

Test Loss: Test loss is a metric that measures how well a machine learning model performs on a separate dataset that it has not seen during training. It quantifies the error between the predicted values and the actual values in the test dataset. In this case, the test loss is 1.171, indicating the level of error in the model's predictions on this specific dataset.

Training Loss: Training loss is similar to test loss but pertains to the error during the model's training phase. It measures how well the model is fitting the training data. A training loss of 1.110 suggests the level of error observed during the training process.

Learning Rate: Learning rate is a hyperparameter that controls the step size or rate at which a machine learning model adjusts its internal parameters (weights and biases) during training. A learning rate of 0.001 signifies that the model's parameters are updated with very small steps in each iteration to ensure gradual convergence.

Activation Function: Activation functions are mathematical functions applied to the output of a neuron (or a layer of neurons) in a neural network. They introduce non-linearity to the model, allowing it to learn complex patterns. In this case, the ReLU (Rectified Linear Unit) activation function is used, which is a common choice known for its simplicity and effectiveness.

Regularization: Regularization is a technique used to prevent overfitting in machine learning models. It adds a penalty term to the loss function, discouraging the model from fitting noise in the data. In this case, L1 regularization is employed. L1 regularization encourages sparsity in the model by adding the absolute values of the weights to the loss function.

Regularization Rate: The regularization rate (0.001 in this case) represents the strength of the regularization term. A smaller value indicates weaker regularization, while a larger value imposes stronger regularization, potentially resulting in sparser model weights.

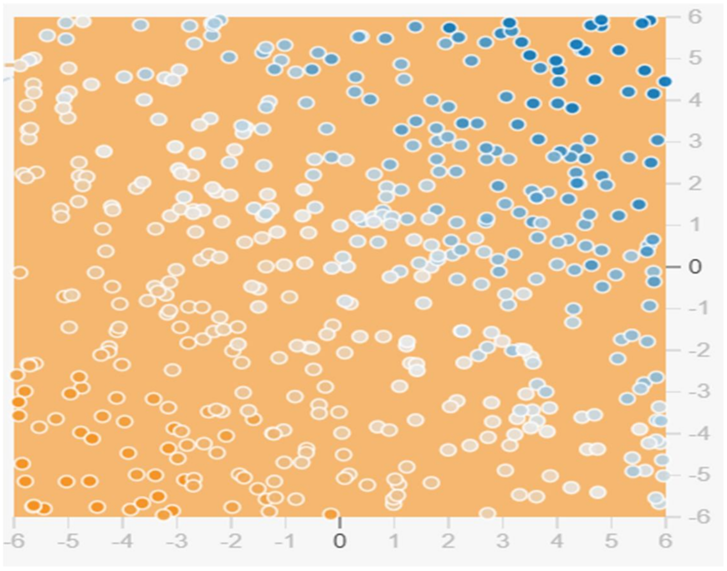


Fig 13: Using Sigmoid Activation Function

Test loss: 0.713, Training loss: 0.165, Learning Rate: 0.0001, Activation: Sigmoid, Regularization: L2, Regularization Rate: 0.001.

Test Loss: The test loss is a measure of how well a machine learning model performs on a separate dataset that it hasn't seen during training. It quantifies the difference between the model's predictions and the actual values in the test dataset. In this case, the test loss is 0.713, which indicates the level of error in the model's predictions on this specific dataset.

Training Loss: The training loss is similar to the test loss but pertains to the error during the model's training phase. It measures how well the model is fitting the training data. A training loss of 0.165 suggests the level of error observed during the training process.

Learning Rate: Learning rate is a hyperparameter that controls the step size at which a machine learning model updates its internal parameters (weights and biases) during training. A learning rate of 0.0001 means that the model's parameters are adjusted with very small steps in each iteration, which can lead to slow but precise convergence.

Activation Function: The activation function is a mathematical function applied to the output of a neuron (or a layer of neurons) in a neural network. It introduces non-linearity to the model, allowing it to learn complex patterns. In this case, the Sigmoid activation function is used. Sigmoid is known for its S-shaped curve and is commonly used in binary classification tasks where the output should be between 0 and 1.

Regularization: Regularization is a technique used to prevent overfitting in machine learning models. It adds a penalty term to the loss function, discouraging the model from fitting noise in the data. In this case, L2 regularization is employed. L2 regularization encourages the model's weights to be small by adding the sum of the squares of the weights to the loss function.

Regularization Rate: The regularization rate (0.001 in this case) represents the strength of the regularization term. A smaller value indicates weaker regularization, while a larger value imposes stronger regularization, potentially leading to a simpler model with smaller weights.

VI. CONCLUSION & FUTURE SCOPE

This research paper introduces an innovative approach aimed at enhancing the performance of Convolutional Neural Network (CNN) models designed for the recognition of English-Devanagari handwritten digits. The methodology leverages the synergy of Hybrid Evolutionary Algorithms (HEA) and Variable Length Genetic Algorithms (VLGA) to optimize hyperparameters effectively. The experimental results substantiate the significant performance advantages of this approach in comparison to conventional optimization techniques. The implications of this novel technique extend beyond handwritten digit recognition, offering potential advancements in various computer vision applications. The model's performance can be assessed through the test loss and training loss values. A lower test loss indicates that the model is making more accurate predictions on unseen data, while a lower training loss suggests that the model is fitting well to the training data. In the example given, the test loss is higher than the training loss, which is a common observation but could indicate some degree of overfitting. The learning rate of 0.0001 implies that the model is updating its internal parameters with very small steps during training. This can result in slow but precise convergence. Choosing an appropriate learning rate is crucial for training neural networks effectively. The choice of the Sigmoid activation function suggests that the model is likely used for binary classification tasks where the output should be between 0 and 1. Sigmoid is commonly used for such tasks. The use of L2 regularization with a rate of 0.001 indicates an attempt to prevent overfitting by adding a penalty term to the loss function based on the sum of the squares of the weights. This can help in improving the model's generalization to unseen data. In order to ensure the convergence of the model, this study introduces the incorporation of global hyperparameters, specifically pertaining to optimizer selection and learning rate adjustment, into the existing variable-length GA model. The experimental validation is conducted utilizing a dataset comprising handwritten English digits, reinforcing the practical relevance and applicability of the proposed approach. However, the GA encoding method yields lengthier chromosomes with an increase in the number of rounds, thereby diminishing its effectiveness. Consequently, heightened iterations lead to reduced accuracy. Based on the empirical findings of the experiments, it has been determined that a population size of 27 yields the most favorable fitness values and an elevated average fitness score. Moreover, comparative analysis underscores the superior accuracy achieved by our proposed model, with the HEA-VLGA model exhibiting an impressive accuracy rate of approximately 99.38%. To address concerns regarding the growing chromosomal length resulting from extended iterations, ongoing efforts are directed toward refining the GA encoding method. Additionally, there is an ongoing initiative to develop a more robust architectural framework capable of harnessing the potential of GA for specific CNN layers. Future research endeavors are poised to explore avenues for automated architecture discovery through the utilization of VLGAs. Furthermore, the applicability of this approach to a diverse range of computer vision challenges is a subject of keen interest.

REFERENCES

[1] F. M. Talaat and S. A. Gamel, “RL based hyper-parameters optimization algorithm (ROA) for convolutional neural network,” J. Ambient Intell. Humaniz. Comput., 2022, doi: 10.1007/s12652-022-03788-y.

[2] M. A. A. Albadr, S. Tiun, M. Ayob, and F. T. AL-Dhief, “Spoken language identification based on optimized genetic algorithm–extreme learning machine approach,” Int. J. Speech Technol., vol. 22, no. 3, pp.711–727, 2019, doi: 10.1007/s10772-019-09621-w.

[3] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, “Efficient Hyperparameter Optimization in Deep Learning Using a Variable Length Genetic Algorithm,” 2020. doi: 10.48550/arXiv.2006.12703.

[4] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, “Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification,” IEEE Trans. Cybern., vol. 50, no. 9, pp. 3840–3854, 2020, doi10.1109/TCYB.2020.2983860.

[5] P. Nirwan and G. Singh, “Segmentation and identification of bilingual offline handwritten scripts (devanagari and roman),” Int. J. Recent Technol. Eng., vol. 8, no. 2 Special Issue 6, pp. 603–607, 2019, doi:10.35940/ijrte.B1178.0782S619.

[6] A. J. Reiling, “Convolutional Neural Network Optimization Using Genetic Algorithms,” 2017. Available at : Semantic Scholar.

[7] A. Bhandare and D. Kaur, “Designing convolutional neural network architecture using genetic algorithms,” in 2018 World Congress in Computer Science, Computer Engineering and Applied Computing, CSCE 2018 -Proceedings of the 2018 International Conference on Artificial Intelligence, ICAI 2018, 2018, pp. 150–156. doi:10.21307/ijanmc-2021-024.

[8] F. Mattioli, D. Caetano, A. Cardoso, E. Naves, and E. Lamounier, “An experiment on the use of genetic algorithms for topology selection in deep learning,” J. Electr. Comput. Eng., vol. 2019, 2019, doi:10.1155/2019/3217542.

[9] S. Loussaief and A. Abdelkrim, “Convolutional Neural Network hyper-parameters optimization based on Genetic Algorithms,” Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 10, pp. 252–266, 2018, doi: 10.14569/IJACSA.2018.091031.

[10] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, “Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach,” Inf., vol. 10, no. 12, 2019, doi: 10.3390/info10120390.

[11] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, and R. Nanculef, “Automating Configuration of Convolutional Neural Network Hyperparameters Using Genetic Algorithm,” IEEE Access, vol. 8, pp. 156139–156152, 2020, doi: 10.1109/ACCESS.2020.3019245.

[12] C. Li et al., “Genetic algorithm based hyper-parameters optimization for transfer convolutional neural network,” 2022. doi: 10.1117/12.2637170.

[13] J. H. Yoo, H. Il Yoon, H. G. Kim, H. S. Yoon, and S. S. Han, “Optimization of Hyper-parameter for CNN Model using Genetic Algorithm,” 2019 IEEE Int. Conf. Electr. Control Instrum. Eng. ICECIE 2019 - Proc., 2019, doi: 10.1109/ICECIE47765.2019.8974762.

[14] R. Zatarain Cabada, H. Rodriguez Rangel, M. L. Barron Estrada, and H. M. Cardenas Lopez, “Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoringsystems,” Soft Comput., vol. 24, no. 10, pp. 7593–7602, 2020, doi: 10.1007/s00500-019-04387-4.

[15] A. Shrestha and A. Mahmood, “Optimizing deep neural network architecture with enhanced genetic algorithm,” in Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019, 2019, pp. 1365–1370. doi: 10.1109/ICMLA.2019.00222.

[16] A. Baldominos, Y. Saez, and P. Isasi, “Model Selection in Committees of Evolved Convolutional Neural Networks Using Genetic Algorithms,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11314 LNCS, pp. 364–373, 2018, doi: 10.1007/978-3-030-03493-1_39.

[17] J. H. Holland, Adaption in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1975.

[18] U. Bhattacharya and B.B. Chaudhuri, Databases for Research on Recognition of Handwritten Characters of Indian Scripts, Proc. Eighth Int”l Conf. Document Analysis and Recognition (ICDAR “05), vol. 2, pp. 789- 793, 2005.

[19] Cortes, C.; Vapnik, V. (1995). "Support-vector networks." Machine Learning. 20 (3): 273-297. doi:10.1007/BF00994018.

[20] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14-16 August 1995. pp. 278-282.

[21] G S Lehal, Nivedan Bhatt, “A Recognition System for Devnagri and English Handwritten Numerals,” Proc. Of ICMI, 2000.

[22] R.J.Ramteke, S.C.Mehrotra, “Recognition Hand-written Devanagari Numerals,” International Journal of Computer Processing of Oriental Languages, 2008.

[23] U. Bhattacharya, S. K. Parui, B. Shaw, K. Bhattacharya, Neural Combination of ANN and HMM for Handwritten Devanagari Numeral Recognition.

[24] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” Multimed. Tools Appl., vol. 80, no. 5, pp. 8091–8126, 2021, doi: 10.1007/s11042-020-10139-6.

[25] https://arxiv.org/abs/1803.09820

[26] https://ieeexplore.ieee.org/document/8662673

[27] https://www.sciencedirect.com/science/article/abs/pii/S0360835220304885

[28] https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012041

[29] https://www.sciencedirect.com/science/article/pii/S2405896322001173