ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 11, No. 1, 44-50 (2026)
www.astesj.com

ASTESJ
ISSN: 2415-6698

Efficient Pattern Recognition Resource Utilization Neural Network

Hanan Hassan Ali Adlan*[1], Elsadig Ahmed Mohamed Babiker[2]

[1]*Department of Computer Science, Princess Nourah Bint Abdulrahman University, Riyadh, 11761, Saudi Arabia*

[2]*Department of Computer Science, Prince Sattam Bin Abdulaziz University, Kharj, 16273, Saudi Arabia*

Email(s): hahassan@pnu.edu.sa (H. Adlan), e.babiker@psau.edu.sa (E. Babiker)

∗Corresponding Author: Hanan Hassan Ali Adlan, hahassan@pnu.edu.sa

ARTICLE INFO

Article history:
Received: 15 November, 2025
Revised: 6 January, 2026
Accepted: 8 January, 2026
Online: 16 January, 2026

Keywords:
Neural Networks
Backpropagation
Kernel
Feature Map
Receptive Fields
Euclidean Distance
K-Means

ABSTRACT

Neural Networks derives intelligent systems and modern autonomous applications. With the complexity introduced in today's systems, designing architectures remains open research problem in all fields. Despite many efforts to generalize, the resources required by neural architectures remain challenge. Robots design, Smart cities, IoTs ...etc. become the leading industry and driving the fourth industrial revolutions. All these challenges dictate the search for efficient utilization of resources. This paper demonstrates details of the architecture ElHa Net. The architecture finds the minimum resources required for pattern recognition domains. Composed of two stages, Extraction stage followed by classification stage. The extraction stage is a self-extraction, performed by convolutions like processes. The classification stage receives the extracted patterns and associates them through weighting to defined classes. The architecture is found to compete with many reported in literature.

1. Introduction

The industry and the fourth industrial revolution witness automation in all systems. With the realization of Artificial Intelligence, deep learning plays an important role in generalization of Neural Networks. Deep learning results in huge architectures and resources to perform computing tasks. Thus it becomes essential to introduce the issue of complexity in finding solutions towards efficient resources to address this problem.

In search for enhanced architectures in machine learning, convolution neural networks and deep learning play significant roles. Deep learning proves its efficiency in learning patterns using huge volume of inputs to train machines for self-efficient extraction utilizing convolution in the extraction of features. Recent architecture *ElHa Net* originally presented at ISAS symposium addresses deep learning pattern recognition neural networks architecture [1].

The research on Hand written digits recognition is a bench mark for neural network in pattern recognition. Many models were developed and tested on the MNIST data set. Originally, models based on feature extraction of LeCun in 1989 [2]. Then emergence of convolution nets become the state of the art. To Develop and train complex convolution architectures requires huge volume of data composed of thousands of neurons. LeCun ideas based on the concepts of local receptive fields, shared weights, and spatial/temporal sub sampling. Zip codes digits used LeNet as the first convolution neural network [2]. Then appear AlexNet in 2012 in computer vision, beside others. The Network architecture is similar to LeNet of LeCun, utilizing deeper, bigger, and featured Convolution Layers stacked on top of each other followed by a Pool layer [3] - [5]. Many developments took place across the years [6], [7]. In 2013 Matthew Zeiler and Rob Fergus developed ZF Net (short for Zeiler& Fergus Net). It was an improvement over AlexNet architecture hyper parameters, it expands the middle convolution layers and modify the stride and filter size on the first layer [8]. GoogLe Net 2014, from [9] developed an Inception Module that reduced the number of parameters in the network. The authors use Average Pooling instead of Fully Connected layers at the top of the ConvNet, which reduces parameters that do not contribute to the classification process [9]. In 2014 also appeared VGG Net. Karen Simonyan and Andrew Zisserman that became known as the VGG Net. Identifies good performance as a result of

the depth in the network [9]. 2016 witness the appearance of ResNet. Residual Network developed by [10]. The architecture uses heavy batch normalization, the fully connected layers at the end of the network disappeared. ResNet are currently the state-of-the-art Convolution Neural Network models [10]. Despite the success of these models, remain the challenge of huge architecture and the connectionist model among the cascaded layers of the networks. These result in demands for computing resources that beyond individual systems.

In this paper, we developed *ElHa Net* Neural Network utilizing the efficiency of deep learning and the convolution neural network of self-extraction. The architecture used the hand written digit characters of the MNIST data set. The MNIST data set is used as the bench mark across the literature. The approach is found to perform favorably with minimal computation and time resources. By computation resources we refer to the neurons architecture and the connections required, while time resource refers to the processor's time [11], [12].

The organization of the paper includes an Introduction. In section 2 we demonstrate the model in a top-down approach. We give the functions, then the structure of the components, followed by the overall architecture. The data flow demonstrated in section 3. Results discussions and conclusions of the implementation using MNIST data set is demonstrated in section 4.

2. *ElHa* Net Neural Network

2.1. ElHa Net Model

ElHa Net model is a hybrid architecture utilizing the advantages of the k-Means approach and the Convolution Neural Networks. The network is a self-extraction Neural Network. Composed of two stages, a feature extraction Network and a classification Network, Figure 1.

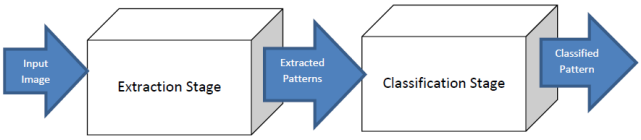

Figure 1: *ElHa Net* Block Diagram

Figure1 demonstrates the block diagram for the two stages of *ElHaNet* Neural Network composed of two stages, An input stage is the feature extraction network. Output of this stage is the input patterns to the classification stage. The classification network will perform on the extracted features and produced output. Training the network is a backpropagation algorithm.

2.2. ElHa Net Functions and Structure

We follow a Top-Down approach to define *ElHa Net*, first we start with the functions, then define the structure for the building blocks defined in Figure 1.

ElHa Net Functions:

- Extraction; features extraction measured based on Euclidean distance. The feature extraction is a self-process.
- Dimensionality reduction and preserve spatial invariance with pooling.
- Classification; classify the patterns detected as belong to particular classes, as probability of image.
- Pattern recognition; recognize patterns of different objects based to their present in the input. Through none linearity and training

ElHa Net Structure:

The main building blocks of Figure 1 structured in extraction stage, and a classification stage. The extraction stage is capable of self-extraction. Neurons in the first layer accepts inputs from an input image. The weights of the neural network store kernels that produce feature maps. The feature maps are distance-base operation using Euclidean distance as the square root of the quantity in the summation of equation (1)

$$D = \sum_{i=1}^{n}(x_i - y_i)^2 \quad (1)$$

In the above equation x, y are two points in Euclidean n-space

x_i, y_i Euclidean vectors, starting from the origin of the space (initial point).

n is the n-space

The extraction operation produces feature maps based on the minimal distance obtained for a particular kernel. Receptive fields in the image are then represented by the kernel that produces this minimal distance.

To reduce dimensionality, the minimum distance kernel is searched, the kernel representing an area is averaged and forms the input to the feature map produced from the pooling operation. The feature maps are passed through a ReLU function (2) to finalize the extraction stage.

$$f(x) = \begin{Bmatrix} 0 & x < 0 \\ 1 & x \geq 0 \end{Bmatrix} \quad (2)$$

The classification stage is a fully connected network. Composed of a flattened pattern layer, formed from the extraction stage, one hidden layer, and an output layer. Neurons are fired as a result of sigmoid activation, equation (3)

$$f(x) = \frac{1}{1+e^{-x}} \quad (3)$$

3. *ElHa* Net Architecture

ElHa Net Architecture Figure 2 shows the two stages of the network. The extraction pipeline network accepts *nXn* input image. The classification stage accepts flattened patterns detected by the extraction network.

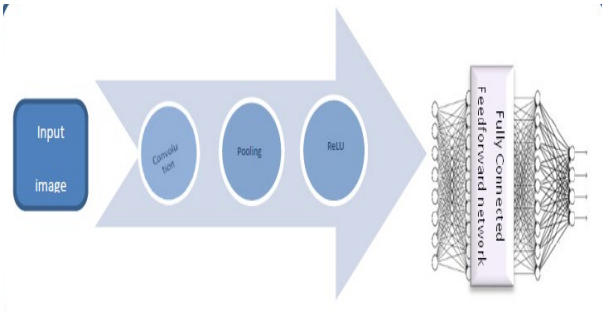

Figure 2: *ElHa Net* Architecture

3.1. ElHa Net Operations

The extraction stage of the network specifies the number of kernels -code words-, two-dimensional array of real values. Each has a square dimension of kSize X kSize. Typical kernel dimensions are 2X2, 3X3, or 5X5, Figure 3. Initialized at the beginning of the training and adjusted gradually.

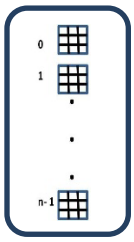

Figure 3: A set of n 3×3 kernels.

Figure 3 gives typical n 3X3 kernels that is designed specific. The size of such kernels is based on the application domain. Feature maps are computed for an input image according to the distance of a frame from the kernel, Figure 3. By frame we mean portion of the image that will be convolved with the kernel.

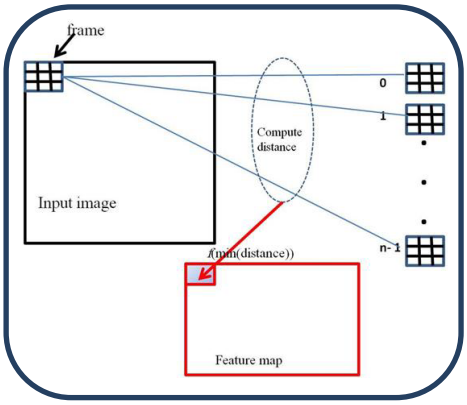

Figure 4: A Feature Map Entry Computation

Figure 4 gives the computation of one entry in the feature map. An input image is scanned from left to right, top to bottom partitioning it into frames. Each is of size, Ksize X Ksize. A frame will be represented by the nearest kernel. This will be the entry in the feature map, equation (4).

$$Fmapij = F(\min(\text{distance}(frameij, kernel0 \ldots \text{no. of kernels}))) \quad (4)$$

The feature map is then pooled for dimensionality reduction. This step also achieves spatial invariance. The pooling is done by averaging the kernels present in the feature map. Passes through a ReLU, then flattened to produce a one-dimensional array for the classification network.

3.2. ElHa Net Learning

The training algorithm is a backpropagation algorithm. Errors are backpropagated to update the kernels. Kernels are randomly initialized, and are updated according to the backpropagation in the extraction network, as well as weights in the fully connected network for enhanced recognition rate. The training continues to reach a preset target or the maximum number of the training epochs reached. Kernels are ksize X ksize stored as weights. Feature maps are produced based on the distance measure. The minimum distance kernel is the input to the feature map. Pooling is the dimensionality reduction process that produces an averaged kernels to represent the code word for the detected features. The rectified linear unit Figure 5, works on the code words to normalize the detected features. The output of the feature extraction is then flattened to compose the input to the classification stage.

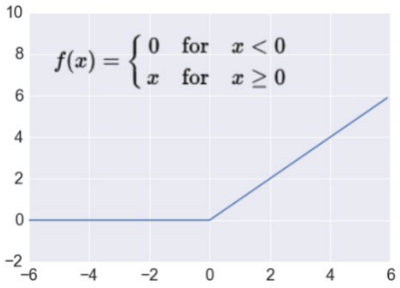

Figure 5: Rectified Linear Unit

The classification stage is a two-layer dense network. Neurons in the hidden layer uses sigmoid activation to fire. Figure 6. This process classify the detected patterns as one of the output classes.

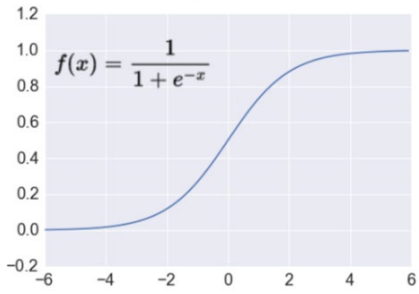

Figure 6: Sigmoid activation

Output of the network, Algorithm 1 specifies the process of pattern detection present in the input image.

Algorithm 1: Network Output Computation

Compute_feature_map (input image x) returns feature map fmap
Kerns ≡ array of kernels, each of dimension kSize×kSize
for i:= 0 … N - 1
for j := 0 … M – 1
set frame := x[i..i+kSize][j…j+kSize]

```
        set fmap[i][j] := min(distance(frame, kerns))
        netOutput(inputs: image x, weights matrices W1,
        W2)
        returns netStruct,  classIndex
        fmap:=Compute_feature_map (x)
        y := Flatten(fmap)   // y: input to the fully connected
        layer
        y :=relu(y)
        netStruct.y:= y
        //Full connected layers
        o1 := matrixMultiply(y, W1)
        netStruct.ot1 := o1
        o1 := sigmoid(o1)
        netStruct.o1 := o1
        o2 :=matrixMultiply(o1, W2)
        netStruct.ot2 := o2
        o2 := sigmoid(o2)
        netStruct.o2 := o2
        classIndex:=argMax(o2)
    end
end
```

Algorithm 1 gives the algorithm for computing the output. The feature extraction stage produces features for the classification stage. The dense network as mentioned is two fully connected layers that accept the flattened output of the extraction stage. Neurons activation uses sigmoid activation both in the hidden layer and the output. One of the output neurons will fire based on activation.

To produce the output, the network is trained to recognize the pattern present in the input image, Algorithm 2, *ElHa* Net learning algorithm uses backpropagation to recognize the input pattern. A set of training images is used for the training process.

Algorithm 2: Network Training

```
Initialize weights to small random values
Initialize kernels
Initialize training parameters
set N := feature map number of rows
set M := feature map number of columns
repeat
for each training sample tsmp do
// compute network output
netStruct, class :=netOutput(tsmp, W1, W2)
out1, out2, ot1, ot2, y :=netStruct.( out1, out2, ot1, ot2, y)
for k := 0 . . . length(out2)
diff := target – out2k
δ2k := α × diff × sigmoid(out1k)
for j := 0 . . . W2 rows
W2jk := W2jk + δ2k
```

$$sum := \sum_{j=0}^{W2rows} \delta_k \times W2_{jk}$$

```
        sd = 0.0
        for j := 0 . . . W1 cols
        for k := 0 . . . W1 rows
        δtj:= sum× sigmoid(o1tj)
        δ1kj :=δtj × inpk
        W1kj := W1kj + δ1kj
        sd:=sd + δtkj  ×  W1kj × reluD(inpk)
// update kernels
for  p:= 0 . . . noofKernels - 1
        kernMp:=mean(all frames belong to kernelp)
        for i, j := 0 . . . kSize
        kernpij:= 2×sd×(kernMpij - kernpij)
until either number of epoch = maxepoch or error < eps
```

Algorithm 2 used to train the network. The kernels are updated according to the propagation of the error. The successive iterations continue to meet a criterion, or a stated number of epochs expired.

3.3. MNIST Implementation

The MNIST is a world bench mark data set used globally to test novel architectures. The data set is used in order to recognize hand written digits. The data set composed of 60000 Hand written digit images each is 28X28. Data is divided into training and testing sets.

ElHa Net accepts MNIST images and apply kernels to filter the images. This process produces the feature maps. The feature maps consist of kernels that matches certain patterns in the presented image. Pooling produces a reduced feature plane by averaging each kernel present in the feature map. ReLU applied to the pooled plane. Output of the ReLU is then flattened to compose the input to the classification stage. This is the two layer fully connected network that classify the extracted features as one of the ten decimal digits.

4. Results, Discussions and Recommendations

The architecture *ElHa* Net applied MNIST data set. Results are demonstrated together with recommendations that can foster further research..

4.1. Results and Discussions

ElHa Net uses 2X2 kernels in the extraction stage of the network for the MNIST application, Figure 7. This size was approached after extensive experimentation with first 5x5 kernels, then a 3x3 kernels due to poor recognition rates. The size then reduced to 2x2 kernels.

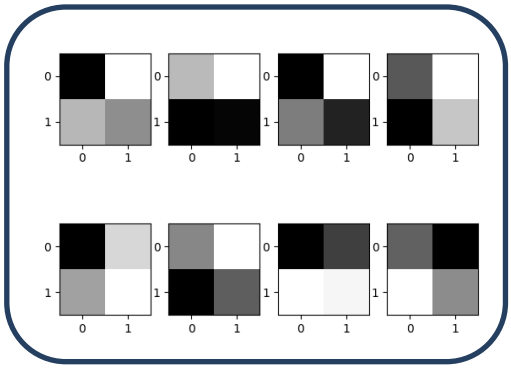

Figure 7: 2X2 Kernels in the extraction stage of the MNIST Application

Figure 7 visualizes 8 kernels used in the *ElHaNet*. Figure 8 visualizes sample feature maps produced in extraction stage.

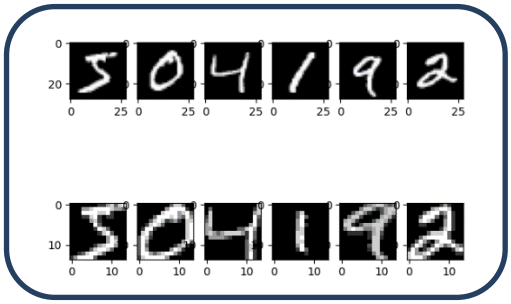

Figure 8: Sample Feature maps of the extraction stage , first row original digits, second row output.

As seen in the figure 8, the feature map reduces the 28X28 input image into half the size, which is 14X14, thus reduced dimensionality by 50%. This achieved through extraction and pooling explained earlier. These extracted patterns are then flattened to form the input to the classification stage. The classification network initializes the weights randomly. The backpropagation, propagates the errors across the network back and forth to meet a threshold of $1e^{-4}$, Figure 9.

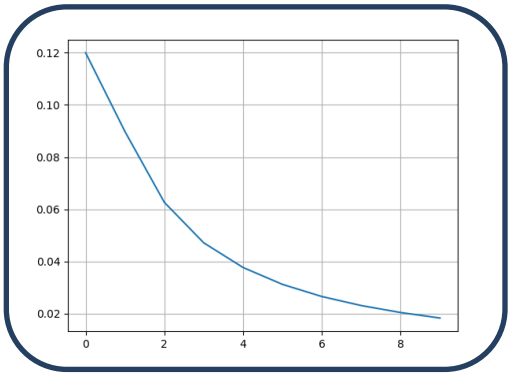

Figure 9: Visualizing the Training Error

Figure 9 shows the error during the training of the network. Due to random initialization, the curve shows an improvement as the time progress, and the network learns the pattern of interest.

Figure 10 visualizes the digits in the test set. The figure shows 969 for digits 0, 1127 for digits 1, 1015 for digit 2, 989 for digits 3, 961 for digit 4, 858 for digit 5, 933 for digit 6, 1005 for digit 7, 955 for digit 8, and 974 for digit 9.

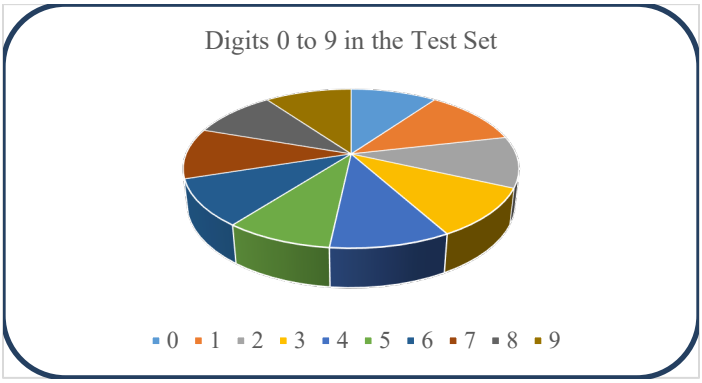

Figure 10: Digits 0 through 9 in the Test Set

Figure 11 shows the training and testing accuracy for the handwritten digit's application. *ElHa Net* achieved 99.96 accuracy during the training, and 99.53 in the testing.

```
Training accuracy = 99.96159690103688
Testing accuracy = 99.52994073165748
[5918. 6742. 5947. 6117. 5836. 5409. 5913. 6245. 5836. 5928.] 59891.0
[ 969. 1127. 1015.  989.  961.  858.  933. 1005.  955.  974.] 9786.0
[2. 0. 0. 5. 2. 2. 1. 3. 4. 4.] 23.0
[ 1.  0.  4.  7.  2. 1 0.  0.  5.  8.  9.] 46.0
[0 1 2 3 4 5 6 7 8 9]
```

Figure 11: Training and Testing Accuracy

The recognition for individual digits is shown in Figure 12. The network capability to recognize the ten digits varies considerably. Figure 13 detailed the frequency of the miss classified digits.

```
c = [ 99.89680083  100.    99.60591133  99.29221436  99.79188345
   98.83449883  100.    99.50248756  99.16230366  99.07597536]
  [0.10319917   0.     0.39408867   0.70778564   0.20811655
   1.16550117   0.     0.49751244   0.83769634   0.92402464]

PS =  0.9952449892041173
RS =  0.9951620753892669
F1S =  0.9951991801112723
 0.9952035305697337
```

Figure 12: Recognition of Individual Digits 0 Through 9

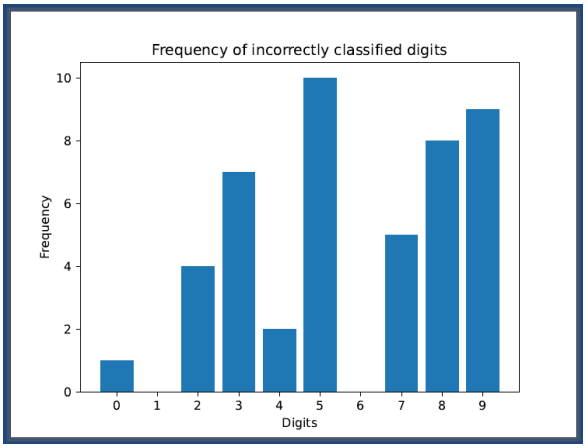

Figure 13: Incorrectly Classified Individual Digits 0 Through 9

As seen in the above figure, some digits are well recognized, produced 100% recognition rate, as in digits 1 and 6. Other digits vary. The most challenging digit to be recognized is digit 5, with 98.83% recognition rate.

```
[[ 968    0    0    0    0    0    1    0    0    0]
 [   0 1127    0    0    0    0    0    0    0    0]
 [   0    0 1011    2    0    0    0    0    2    0]
 [   1    0    0  982    0    2    0    1    2    1]
 [   0    0    0    0  959    0    0    1    0    1]
 [   2    0    0    6    0  848    1    0    1    0]
 [   0    0    0    0    0    0  933    0    0    0]
 [   0    0    3    1    1    0    0 1000    0    0]
 [   1    0    2    2    0    2    0    1  947    0]
 [   0    0    0    2    5    0    0    0    2  965]]
```

Figure 14: Confusion Matrix

Figure 14, confusion matrix, shows the performance of the network on individual digits. For digit 0, 968 out of 969 images are

successfully recognized. While one image missed and classified as 6. For digit 1, represented by 1127, all are successfully recognized. Digit 2, from 1015, 2 were classified as 3, and 2 as 8. For digit 3, 989 images were used, 982 are correctly recognized, 1 classified as 0, 2 as 5, 1 as 7, 2 as 8, and 1 as 9. For the digit 4, we have 961 images, 959 were well recognized, while 1 classified as 7 and another as 9. Digit 5 represented by 858 images, 848 were recognized, 2 classified as 0, 6 classified as 3, 1 as 6 and 1 as 8. Digit 6 images are 933 all recognized. Digit 7 images are 1005, 1000 were correctly recognized, 3 missed as 2, 1 as 3, 1 as 4. For digit 8, the total images are 955, 1 classified as 0, 2 as 2, 2 as 3, 2 as 5 and 1 7. For digit 9 from 974, 965 were recognized, 2 classified as 3, 5 as 4, and 2 as 8.

```
Precision = [99.89680083  100.  99.60591133  99.29221436
  99.79188345  98.83449883 100.   99.50248756
  99.16230366  99.07597536]  -1.4210854715202004e-14

Recall = [ 99.58847737  100.   99.50787402  98.69346734
 99.3782383  99.53051643 99.78609626 99.70089731
  99.26624738  99.79317477]

F-measure= [99.74240082 100. 99.55686854   98.99193548
  99.58463136  99.18128655  99.89293362
 99.60159363 99.2142483 99.43328181]  99.51991801112723
```

Figure 15: Precision and F Matrix

The Precision matrix, Figure 15 again shows the precision for individual digits, which is a measure of how well the network recognizes a certain pattern. Recall is a measure of how well the network remembers a pattern of interest; it gives the percentage of actual positive predictions correctly classified. The F-measure lay the foundation for comparison and useful to bench the performance of the algorithm with other produced on similar applications. It is a weighted average of the precision and recall. Figure 16 visualizes the accuracy of individual digits 0 through 9.

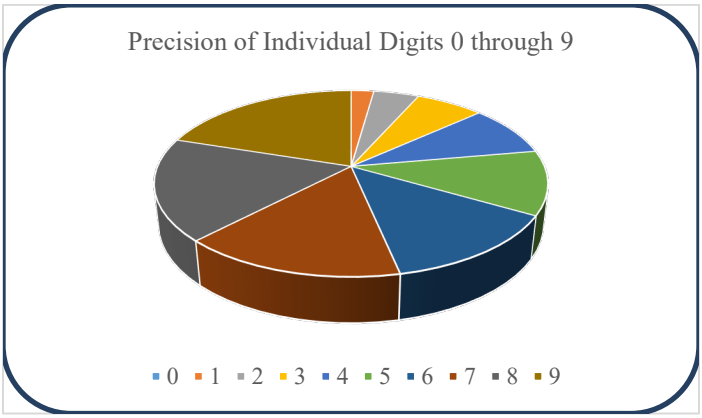

Figure 16: Precision of Individual Digits 0 Through 9

Table 1 reports performance of different architectures referenced in the literature, the first row reports the performance of *ElHa* Net architecture. The recognition rate based on the MNIST data is found to compare favorably with other architectures reported in the literature.

Table 1: Recognition of different Architectures on the MNIST Data Set

Architecture	Recognition Rate	Reference
ElHaNet	99.5	Current work
CNN	98.32	[2]
ResNet	99.16	[10]
CNN (Keras & Zeano)	98.7	[13]
DNN	98.08	[14]
LeNet-4-CNN	98.9	[15]
SVM	98.6	[16]
Robert Edge	99.01	[17]
Adaptive MLP	98.3	[18]
ANN	97.32	[19]
CNN	99.15	[20]
CNN	98	[21]
CNN and Multi-Level Fusion	>=98	[22]
DenseNet	99.37	[23]

ElHa Net developed on CPU Intel core i7 (8750H) @2.2 GH GPUINVIDIA Geforce GTX1050 with 16GB RAM, 128GM and 1 TB HHD, USB SSD (Ubuntu) 500 GB. Windows 11&Ubantu C++ Compiler: gcc, g++. These specifications contribute to the processing time spent on training and testing. Table 2 compare the performance of ElHa Net with reported architectures on the same data set [24]-[29].

Table 2: *ElHa* Net Architectural Training and Size Comparison

Model	Training Time (s)	Model Size (MB)
ElHa Net	600	10.8
GoogleLeNet	512	49.7
MobileNet V2	498	13.6
ResNet-50	510	97.8
ResNeXt-50	549	95.8
Wide ResNet-50	540	132

The time required for training can be attributed to the computational power of the machine. Table 2 reported the least model size for *ElHa* Net.

4.2. Recommendations

The architecture developed in this paper aims to provide an efficient off line recognition that can be implemented in many applications. Excellent recognition rate on the MNIST application demonstrated the efficiency of this approach. Utilizing code word approach enable reducing architecture [30], [31]. The architecture can be implemented on other scenarios for testing efficiency and robustness.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] E. Babiker, H. Adlan, "ElHa Net: A Pattern Recognition Neural Network", 2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS), 2024, DOI: 10.1109/ISAS64331.2024.10845600.

[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation, **1**(4):541-551, 1989, DOI: 10.1162/neco.1989.1.4.541.

[3] Y. LeCun, "LeNet-5, convolutional neural networks", Yann LeCun, 2013.

[4] Y. LeCun, Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series", The Handbook of Brain Theory and Neural Networks, 255-258, 2002.

[5] F. Lauer, C. Suen, G. Bloch, "A trainable feature extractor for handwritten digit recognition", Pattern Recognition, **40**(6):1816-1824, 2007, DOI: 10.1016/j.patcog.2006.10.011.

[6] Y. LeCun, J. Denker, S. Solla, "Optimal Brain Damage", Advances in Neural Information Processing Systems (NIPS), **2**, 598-605, 1990.

[7] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, **86**(11):2278-2324, 1998, DOI: 10.1109/5.726791.

[8] M. Zeiler, R. Fergus, "Visualizing and Understanding Convolutional Networks", European Conference on Computer Vision (ECCV), 818-833, 2014, DOI: 10.1007/978-3-319-10590-1_53.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going Deeper with Convolutions", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9, 2015, DOI: 10.1109/CVPR.2015.7298594.

[10] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778, 2016, DOI: 10.1109/CVPR.2016.90.

[11] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv, 2014.

[12] Y. Hou, H. Zhao, "Handwritten digit recognition based on depth neural network", International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), 35-38, 2017, DOI: 10.1109/ICIIBMS.2017.8279710.

[13] A. Dutt, A. Dutt, "Handwritten digit recognition using deep learning", International Journal of Advanced Research in Computer Engineering & Technology, **6**(7):990-997, 2017.

[14] M. Ghosh, A. Maghari, "A Comparative Study on Handwriting Digit Recognition Using Neural Networks", International Conference on Promising Electronic Technologies (ICPET), 77-81, 2017.

[15] W. Lee, "Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network", IEEE Transactions on Pattern Analysis and Machine Intelligence, **18**(6):648-652, 1996.

[16] K. Sahu, K. Dewangan, "A survey on handwritten character recognition", International Advanced Research Journal in Science, Engineering and Technology, **4**(1):120-120, 2017, DOI: 10.17148/IARJSET.2017.4120.

[17] S. España-Boquera, M. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martínez, "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models", IEEE Transactions on Pattern Analysis and Machine Intelligence, **33**(4):767-779, 2011, DOI: 10.1109/TPAMI.2010.141.

[18] G. Katiyar, S. Mehfuz, "A hybrid recognition system for offline handwritten characters", SpringerPlus, **5**, 357-357, 2016, DOI: 10.1186/s40064-016-1775-7.

[19] M. Siddique, M. Khan, R. Arif, Z. Ashrafi, "Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Neural Network Algorithm", arXiv, 2018.

[20] A. Hossain, M. Ali, "Recognition of Handwritten Digit using Convolutional Neural Network (CNN)", Global Journal of Computer Science and Technology, **19**(D2):27-33, 2019.

[21] A. Enriquez, N. Gordillo, L. Bergasa, E. Romera, C. Gómez Huélamo, "Convolutional Neural Network vs Traditional Methods for Offline Recognition of Handwritten Digits", Advances in Intelligent Systems and Computing (WAF 2018), **855**, 87-99, 2019, DOI: 10.1007/978-3-319-99885-5_7.

[22] H. Zhao, H. Liu, "Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition", Granular Computing, **5**(3):411-418, 2020, DOI: 10.1007/s41066-019-00158-6.

[23] A. Krizhevsky, I. Sutskever, E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems, **25**, 1097-1105, 2012.

[24] L. Seng, B. Chiang, Z. Salam, G. Tan, H. Chai, "MNIST handwritten digit recognition with different CNN architectures", Journal of Applied Technology and Innovation, **5**(1):7-10, 2021.

[25] Y. Chychkarov, A. Serhiienko, I. Syrmamiikh, A. Kargin, "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks", CEUR Workshop Proceedings (CMIS-2021), **2864**, 496-509, 2021.

[26] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, B. Yoon, "Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)", Sensors, **20**(12):3344-3344, 2020, DOI: 10.3390/s20123344.

[27] M. Soomro, M. Farooq, M. Raza, "Performance evaluation of advanced deep learning architectures for online handwritten character recognition", International Conference on Frontiers of Information Technology (FIT), 362-367, 2017.

[28] R. Ptucha, F. Such, S. Pillai, F. Brokler, V. Singh, P. Hutkowski, "Intelligent character recognition using fully convolutional neural networks", Pattern Recognition, **88**, 604-613, 2019, DOI: 10.1016/j.patcog.2018.12.017.

[29] F. Chen, N. Chen, H. Mao, H. Hu, "Assessing Four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)", arXiv, 2019.

[30] A. Shaukat, Z. Azeem, M. Sakhawat, Z. Mahmood, "An efficient and improved scheme for handwritten digit recognition based on convolutional neural network", SN Applied Sciences, **1,** 1125-1125, 2019, DOI: 10.1007/s42452-019-1161-5.

[31] E. Babiker, H. Adlan, A. Ramli, "A Combined Rough Set-K-Means Vector Quantization Model for Arabic Speech Recognizer", International Journal of Computer Science and Information Technologies (IJCSIT), **7(1)**:260-265, 2016.