Performance Comparison of Convolutional Neural Networks for Handwritten Digit Recognition Using Activation Functions and Optimization Methods

Abdul Razaque[1], Saule Amanzholova[1], Azhar Sagymbekova[1] and Aizhan Zaurbek[1]

[1]*International Information Technology University, Manas St. 34/1, Almaty, 050040, Kazakhstan*

Abstract

The industrial sector and large-scale data statistics, such as population censuses, checks, tax statements, and so on, rely significantly on handwritten digit recognition. To satisfy the needs of paperless workplaces and significantly increase labor efficiency, it is important to investigate and implement a high-accuracy handwritten digit recognition system. Several studies have found that Convolutional Neural Networks (CNN) excel at addressing various types of prediction problems, including those requiring visual data as an input. The CNN activation functions and optimization approaches allow neural networks to express themselves nonlinearly and with smaller loss functions, improving their capacity to match data reliably. However, different neural networks react differently to optimization and activation functions. The classification accuracy of CNN for handwritten digits is investigated in this paper utilizing various combinations of activation functions and optimization approaches. In this study, we compared the performance of the CNN model using RMSprop and Adam optimization approaches, as well as ReLU and PreLU activation functions. Additionally, we used the Dropout regularization method throughout the model training to increase the model's ability to generalize and decrease overfitting. The collected findings demonstrate that, when trained on the Kaggle handwritten digit dataset, the CNN model using the Adam optimization technique and PReLU activation function beats other models with a high accuracy of 98.60%.

Keywords

PReLU, ReLU, Adam, Root Mean Square Propagation, digit recognition, CNN, Softmax classifier

1. Introduction

Pattern recognition has been a key and ongoing requirement in natural language processing (NLP). Pattern recognition is used in many fields, such as those involving digit, facial, object, fingerprint, and number identification [1]. This subject has been continuously studied and advanced in this field by numerous experts and academics since the middle of the 20th century. One difficulty with great application value is the recognition of handwritten numerals. Since even a minor inaccuracy in number identification could result in a huge error that cannot be detected by context, everyone anticipates that the accuracy of number recognition should be improved. Consequently, it might lead to significant losses on occasion, such as when opening accounts and making cheques in the banking sector. The biggest difficulty in classifying handwritten characters is that different languages have diverse writing styles. When compared to other formats, it is more difficult to recognize handwritten digits because even when written by the same person, the characters vary in font, similarity, size, and shape. So, the main difficulty in identifying specific characters is the variety in their writing styles and this makes it more difficult to pinpoint the pattern recognition issue with character recognition [2].

Since the development of artificial intelligence technology, deep learning-based handwritten

DTESI 2023: Proceedings of the 8th International Conference on Digital Technologies in Education, Science and Industry, December 06–07, 2023, Almaty, Kazakhstan

a.razaque@iitu.edu.kz (A. Razaque); s.amanzholova@iitu.edu.kz (S. Amanzholova); a.sagymbekova@iitu.edu.kz (A. Sagymbekova; a.zaurbek@iitu.edu.kz (A. Zaurbek)

0000-0003-0409-3526 (A. Razaque); 0000-0002-6779-9393 (S. Amanzholova); 0000-0001-8878-3895 (A. Sagymbekova); 0000-0002-4475-2613 (A. Zaurbek)

digit identification algorithms have been able to outperform more conventional methods in terms of accuracy. The most popular classification techniques include SVM, closest neighbor algorithm, and others but these traditional approaches are obviously exceedingly difficult and ineffective. The growth of neural network theory has led to the emergence of numerous new, more effective techniques. The Convolutional Neural Network is now a hot topic in machine learning. Its network structure resembles that of the visual nerve's system receptive field, making it especially well-suited for activities requiring image processing [3-4]. Feature extraction is a key component of the handwritten digit recognition system. Convolution Neural Networks (CNN) automatically extract features from training datasets that are fixed and to some extent susceptible to character shifting and structural distortions. It is possible to repair and reconstruct features directly from initial images using the automatic feature extraction approach, whereas traditional feature extraction methods are labor-intensive and inefficient [5-6].

Deep learning approaches, such as multilayer CNN using Tensor flow and Keras, have the maximum accuracy when compared to the most common machine learning algorithms, such as k-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest Classifier (RFC) [7]. Because of its great accuracy, CNN is widely utilized in image classification, video analysis, and other applications. As a result, in this study, a network model of this type is developed using deep learning as a starting point to evaluate the performance of handwritten digit recognition.

1.1. CNN Architecture

The initial step in classifying handwritten digits is to extract features from the images. We can now easily extract features from images and classify them by using deep learning techniques. Convolutional neural networks (CNNs) are often chosen for pattern recognition challenges since they don't require manual selection of significant features from the images [8]. Without any human oversight, it is capable of automatically selecting an image's most significant features or patterns. Due to these factors, CNN is regarded as a top feature extractor and classifier. In this study, CNN architecture has been employed to recognize handwritten digits. The simplest structure of a neural network contains three layers: input, implicit, and output layer. Several neurons are present in every layer of the network. Through an activation function and matching weights between each neuron, the last layer neurons are translated to the neurons in the following layer, and the result is our categorization category. CNNs are the advancement of neural networks which have mainly four basic layers: the first is the Convolutional Layer, the second is Pooling Layer, the third is Flatten Layer and finally for the fourth, we have Fully connected layer. In CNN, to extract features we use convolutional and pooling layers. When features have been extracted, CNN can use a final layer which is fully connected to map the features into the final output. Below Figure 1 depicts the basic CNN structure.

A convolutional layer may be subjected to numerous filters for more precise feature extraction. In a CNN model, multiple convolutional layers are frequently stacked up and the outputs serve as inputs for subsequent layers. It aids in the extraction of more sophisticated and intricate information from the input layer [9]. The pooling layer is one more CNN component used to minimize feature map size. Additionally, it decreases the number of parameters that must be taught, which in turn helps in less computational time and energy. The final feature mapping is fed to the Flatten layer after convolutional and pooling layers to create a one-dimensional vector. After then, fully connected layers receive this one-dimensional variable as input. SoftMax combines the features of fully connected layers to train the computer to recognize them. The deep learning library used by Python is called Keras. It implements deep learning using a tensor flow backend. For the implementation of CNN, many researchers have employed keras . Figure 1 shows CNN architecture.

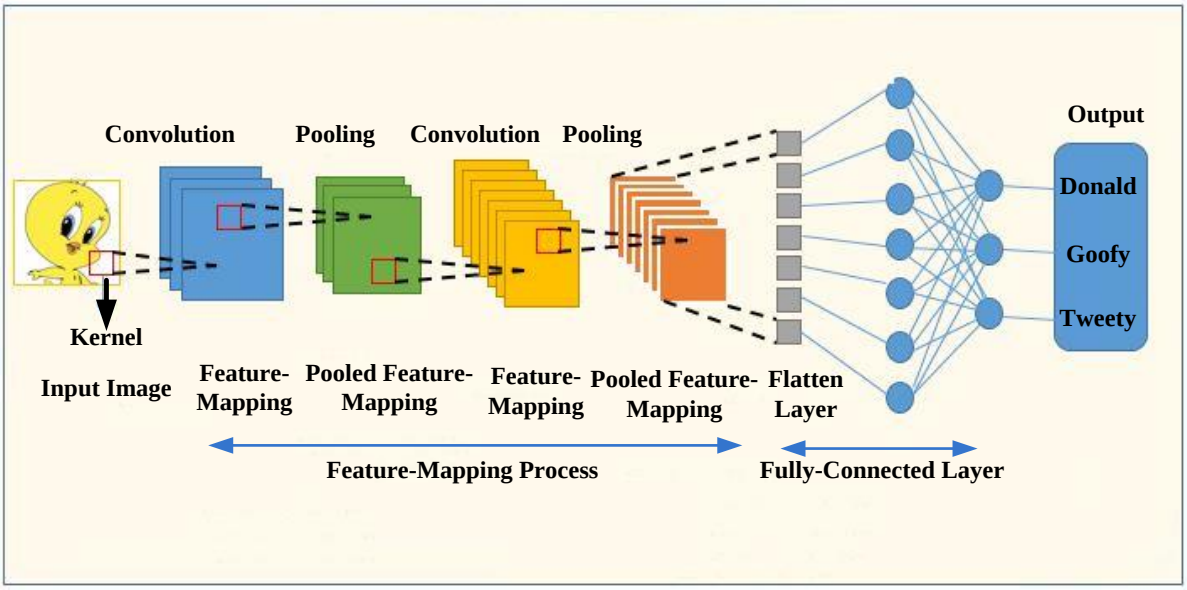

Figure 1: Convolutional neural network architecture

1.2. Contribution

The main contributions are summarized as follows:

- The classification accuracy of CNN for handwritten digits is investigated that utilizes various combinations of activation functions (ReLU and PreLU).
- The performance of the CNN model is compared using RMSprop and Adam optimization approaches. Furthermore, dropout regularization method is employed throughout the model training process to increase the model's ability to generalize and decrease overfitting.

1.3. Problem Identification

CNN is ideal for image recognition. However, a large amount of training data is required to construct a high-accuracy model. Not only is there a high need for data, but researchers' computational abilities are also challenged. Researchers are always looking for ways to strike a balance between accuracy and speed. They concentrated on obtaining more accurate models without boosting data collection. There are several techniques, such as improving data improvement algorithms, changing network architecture, improving activation functions, and so on. Different optimization strategies and activation functions behave differently in different neural networks [10]. As a result, we evaluate the impact of ReLU and PReLU activation functions with Adam and RMSprop optimizers on CNN model accuracy using the Kaggle dataset of handwritten digits in this study. After comparing the accuracy of the handwritten digit recognition methods with other literature, we determined that employing the PReLU activation function and the Adam optimizer successfully increases the rate at which handwritten digits are recognized.

1.4. Paper Organization

The remainder of the paper is organized as follows:

Section II provides a review of relevant work. Section III presents methods and materials. The experimental approach is described in Section IV. The experimental result is presented in Section V. Finally, the entire paper is concluded in Section VI.

2. Related work

This section describes salient features of existing approaches. The hand digitization recognition is becoming increasingly important. There has already been a significant amount of research that includes an in-depth examination and use of numerous well-known algorithms for recognizing handwritten digits. Handwritten digit recognition can be implemented using several deep learning and machine learning approaches. [11] investigated the effectiveness of SVM and KNN for handwritten digit recognition, discovering that these approaches perform better. There are various more challenges that must be addressed in order to obtain outstanding performance in terms of accuracy for detecting handwritten numbers using machine learning and deep learning algorithms, such as big input data, sluggish computation speed, and a few other aspects. Model information (weights) is spread across many levels in a neural network, and model information is dispersed in diverse neurons within each layer [12]. A large amount of study has already been conducted into strategies for improving neural network efficiency by harnessing the natural parallelism that exists within them. The majority of this research has concentrated on implementing neural networks on a shared memory multiprocessor parallel computer or on special-purpose hardware. [13] investigated the theoretical cost of each parallelization technique while keeping the number of processors and the size of the neural network in mind, which was then analyzed for performance. [14] contributed to creating and analyzing ideal parallel methods for CNN training based on digit recognition, with a particular emphasis on a parallelizing platform employing OpenMP technology on a traditional multi-core CPU. They looked at how rapidly CNN training progressed and offered advice for efficient OpenMP parallel modeling based on the dimensions of the input images. Recurrent neural networks can be trained efficiently using a simple algorithm. Thus, the longest sequence in a training can be used to calculate its computational complexity. In most typical datasets, recordings of different lengths are included for perceptual machine learning tasks. These recorded data's training set can be arranged using batch grouping techniques.

Deep neural network (DNN) is used on multiple devices to reduce the total training time of CNN [15]. A DNN's layers can be parallelized in a variety of ways. It would be impractical and time-consuming to thoroughly analyze this list to find the optimal parallelization strategy. Data parallelism is the preferred method due of its convenience. Data parallelism, on the other hand, typically falls short of system reliability and has a high memory requirement [16]. On a case-by-case basis, experts designed methods have been put advanced employing domain-specific information. These expert-made techniques are not always the best option and do not generalize well to DNNs other than the ones for which they were built. The objective is to provide the work for automatically determining effective parallelization techniques for DNNs from their execution graphs. The quick approach is offered that may be used in the real world to evaluate these techniques. On several DNNs, the performance is assessed for the proposed strategy. The effectiveness of various data parallelism-discovered and expert-designed solutions is compared utilizing cutting-edge methods as well as data parallelism. The findings show that, in every case, the solutions produced using this methodology outperform the typical data parallelism strategy [17].

3. Methods and materials

Convolutional neural network model parameter optimization frequently uses the gradient descent technique. The method of parameterization involves minimizing the loss function. A dataset with D training data, for instance, has the following loss function as shown in equation 1.

$$L(W) = \frac{1}{|D|} \sum_{i}^{|D|} fW(x^i) \; + \; \lambda r(W). \tag{1}$$

Where $fW(x^i)$ is a single sample (x^i) of the loss, the $r(W)$ is the canonical term, the λ is the weights.

Learning can be applied in a variety of ways using various optimization techniques. Amongst the most used stochastic methods for deep neural network training is Stochastic gradient descent (SGD). Although the standard SGD method with learning rate does converge, as it is challenging to select an appropriate learning rate, its empirical performance may nevertheless stagnate. Therefore, to further improve the empirical performance of SGD, a wide range of adaptive algorithms have been developed, including AdaGrad, RMSProp, Adam, etc., that exploit second-order moments of historical stochastic gradients to alter the learning rate automatically.

- Stochastic Gradient Descent: The complete dataset is trained using the standard gradient descent algorithm. Stochastic gradient descent is a variation of it that trains each data element separately.
- Adagrad: This approach selects the learning rate based on the circumstances. Because the real rate is based on parameters, learning rates are adaptive. The learning rate will be lower for parameters with a high gradient and higher for parameters with a short gradient.
- RMSProp: Adagrad is altered by RMSProp in terms of finding the gradient. The accumulation of gradients results in a weighted average exponentially. RMSProp keeps only the most recent gradient data and throws away the history. The rmsprop and its variations are covered in [18]. The study investigates adagrad with logarithmic regret bounds as well.
- Adam: Its name comes from "adaptive moments." It incorporates both momentum and rmsprop. A bias adjustment technique is also included in the update operation, which takes gradient's smooth type into account. The Adam method is discussed in [19].

3.1. Dropout regularization methods

As the availability of training data is limited, overfitting can quickly happen while training a big network [20]. When a network fits a training dataset well but performs poorly when that dataset is replaced with another, this is referred to as overfitting. A model with millions of parameters would substantially run the danger of overfitting the training set because in typical neural networks, each neuron is intimately coupled, causing each neuron to back-propagate to the subsequent neuron. This greatly increases the difficulty of training the network. Dropout regularization techniques enable the network to have a high learning rate, where some nodes in specific layers are arbitrarily ignored, while also accelerating convergence, controlling, and reducing overfitting. As a result, the network learns features in a distributed manner and discards a network property at random. The technique also enhances generalization, which successfully lessens overfitting. It is comparable to mixing multiple models to create the final model, which can effectively reduce overfitting look.

3.2. Activation function

The output of the higher node and the input of the lower node in a multilayer neural network are functionally related. This function is called the activation function. The following qualities should be included in the ideal activation function:

- It can stop the gradient from vanishing when the data is output to both ends.
- Select the symmetry center point as (0, 0) to avoid the gradient from working in a particular direction.
- The computational cost of the network should be very minimal because each layer needs to employ an activation function.
- The gradient descent method is employed by the neural network for iterative training, and each layer's activation function should be differentiable.

Some researchers focus more on selecting an effective activation function in their deep learning study. To give the neural network nonlinear capabilities, activation functions are included, and

various activation functions affect the model's ability to fit nonlinear functions in different ways. We have several activation mechanisms, including the ReLU and PReLU activation functions, among others.

ReLU has gained a lot of popularity as an activation function recently. It is described in equation 2 as shown below.

$$y = \begin{cases} 0, (x \leq 0) \\ x, (x > 0) \end{cases} \tag{2}$$

The corresponding image is shown in Figure 2:

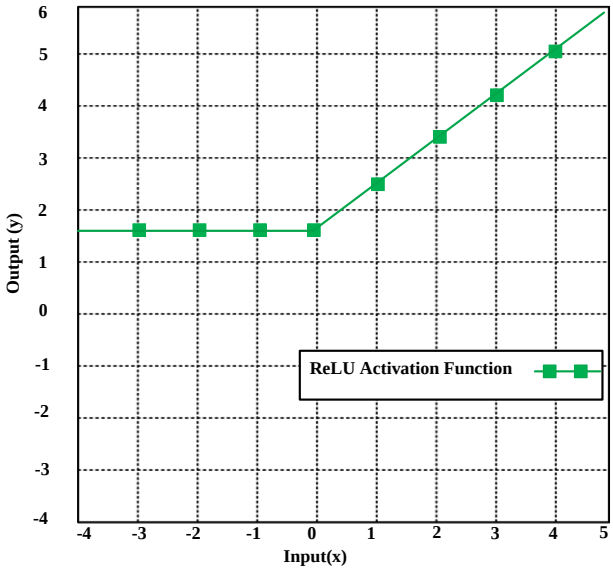

Figure 2: ReLU activation function image

ReLU is hard saturated when x is less than 0 and when x is greater than 0, there is no saturation issue. ReLU can prevent the gradient from declining at x greater than 0, solving the gradient disappearance problem, and enabling supervised direct training of deep neural networks without the need for unsupervised layer-by-layer pre-training. However, as training progresses, some inputs enter a hard saturation zone and the corresponding weights are not updated, which affects the network's convergence. Hence, the ReLU activation function has been enhanced, and the result is the PReLU activation function. It is defined as $y = \max(\alpha x, x)$ $(0 < \alpha < 1)$, and the corresponding image is shown in Figure 3.

In the negative region, the PReLU activation function has a small slope, which avoids the problem of the ReLU activation function losing its role. Although the slope is slight, the PReLU activation function is a linear operation in the negative region, and it does not converge to 0. This resolves the issue of ReLU's hard saturation at $x < 0$, which has no effect on the network's ability to converge with the training input's hard saturation area. In the PReLU activation function formula, the parameter α is typically assumed to be an integer between 0 and 1, and it is typically still small, such as zero point zero.

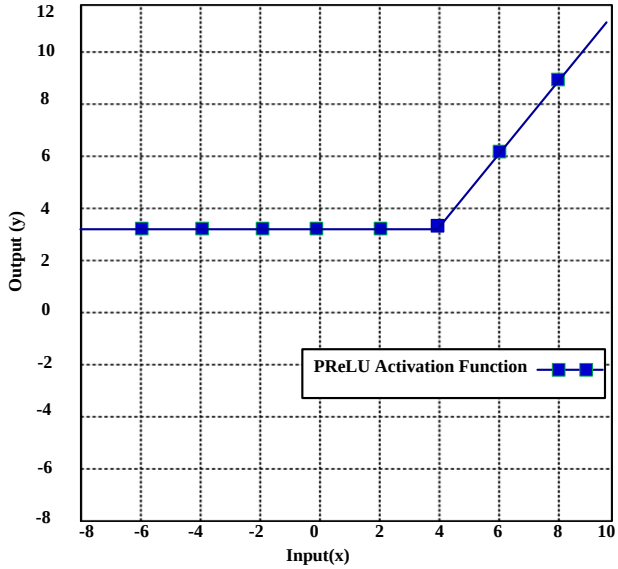

Figure 3: PReLU activation function image

A. Setting up the optimizer and annealing function

Once the network model has been successfully built, we will require an optimization algorithm, a scoring function, and a loss function. The model's performance on image datasets with known labels is measured using the loss function. Cross-entropy is the one that is most frequently employed loss function, and the optimizer is the most important function that iteratively improves the parameters to minimize the loss function.

B. Softmax regression classifier

The Softmax classifier, which uses a version of logistic regression, simplifies binary concepts like hinge loss [1]. Typically, Softmax is used for multi-classification issues. Through function action, it serves the purpose of mapping the output of many neurons to the range (0, 1). This procedure can finish jobs requiring several classifications since it can be thought of as probability. When an array V has an element i identified as Vi, its value after softmax regression [2] is shown in equation 3.

$$S_i = \frac{e^i}{\Sigma_j e^j} \tag{3}$$

For instance, if you design a neural network-based classifier using Softmax, there are 10 categories, ranging from category 1 to category 10 and 10 output neurons.

4. Experimental approach

We examined the performance of CNN models utilizing ReLU and PReLU activation functions with Adam and RMSprop optimizers on the Kaggle dataset of handwritten digits. The effectiveness of these models is evaluated in terms of recognition rates. The Jupyter Notebook platform was used to run the simulations. A training and testing set is created from the input images. Each image has 784 pixels, which stand in for the digit structures.

4.1. Dataset

This paper makes use of the Kaggle dataset that contains samples of handwritten digits. Machine learning models are used to recognize and develop systems based on handwritten digits.

The researchers frequently use the Kaggle handwritten digit dataset that has 42,000 sample images of handwritten digits. Different training and testing dataset (e.g. 70% training with 30% testing and 80% training with 20% testing) are used.

4.2. Pre-Processing

When developing a predictive model, we must first examine and change the data. This requires performing several operations to be pre-processed like importing images, scaling them, modifying their color, displaying the dataset, and transforming the images to vector form [19]. Exploratory Data Analysis is the umbrella term for all these actions taken collectively. We take these actions to speed up our computing process and simplify the model.

4.3. Layer-Construction Process

After the preprocessing phase is complete, we build the CNN model. On the handwritten digit dataset from Kaggle, convolutional neural networks are trained using the Keras API and Tensor flow as the backend. The convolutional neural network, as previously mentioned, consists of four layers. In our experimental approach there are nine layers overall, of which 1st, 3rd and 5th are convolutional layers, 2nd, 4th and 6th are pooling (MaxPool2D) layers, the then we have flattening layer, and the final two are fully connected layers that are simply an artificial neural networks (ANN) classifier. In our model, we employed learnable filters for Conv2D layers with sizes of 32 filters for 1st layer, 64 filters for 2nd layer, and 64 filters for the third layer. By specifying the kernel size, kernel filters transform a specific area of an image. All the convolutional layers are subjected to the kernel filter matrix, which is 3x3s in size. Filters can be thought of as feature map-based image transformations. The MaxPooling2D layer comes next, and it's responsible for segmentation and feature extraction. As implied by the function's name, the Max Pooling function is used to assess the max for each step. Here, we used Maxpool filters of the size 2*2. To give the network nonlinearity, this study uses the activation functions “ReLU” and “PReLU”. The Flatten layer receives the input from the Maxpool layer and converts it to a 1D vector. We employed a dropout regulator with a 50% dropout ratio prior to the 1st and 2nd fully connected layers. 128 neurons were employed in the 1st dense layer and 10 neurons in the 2nd dense layer. An activation function named softmax is added to the final output layer's probabilistic value based on 10 neurons for 10 classes.

4.4. Optimization and Loss Functions

After adding all our layers to the model, the next step is to define the loss function, as well as the optimization process, to test the performance of our model on labeled images. The difference between the expected and observed labels' error rates is the loss function. For categorical classifications with more than two classes, we employed a special form called "categorical cross-entropy." The optimizer is the most vital component. To reduce loss, this function iteratively modifies kernel values, weights, and biases. In this paper, we have chosen RMSprop and Adam optimizers.

4.5. Accuracy

We have evaluated the performance of our model using the metric function "accuracy". Unlike the loss function, the metric evaluation results are used only for evaluation and not to train the models.

5. Experimental results

CNN has been applied on the Kaggle dataset to observe the variation of accuracies for handwritten digits. The accuracy is achieved using Keras and Tensorflow. Utilizing configurations of activation functions and optimization techniques, training, and validation accuracy for 18 epochs are observed. Based on the experimental results, following combinations have been analyzed.

- ReLU activation function + RMSprop
- ReLU activation function + Adam
- PReLU activation function + RMSprop
- PReLU activation function + Adam
- Heat map of confusion matrix
- Handwritten digit recognition accuracy

A. ReLU activation function + RMSprop

The CNN model's effectiveness with the ReLU activation function and the RMSProp technique is determined. According to the results, the model achieved 94.63% training accuracy and 96.93% validation accuracy with 80% training data and 20% testing data, as shown in Figure 4(a). When the number of training data is reduced to 70% and the number of testing data is increased to 30%, the model's performance suffers marginally. As shown in Figure 4(b), training accuracy is 92.74% and validation accuracy is 96.28%.

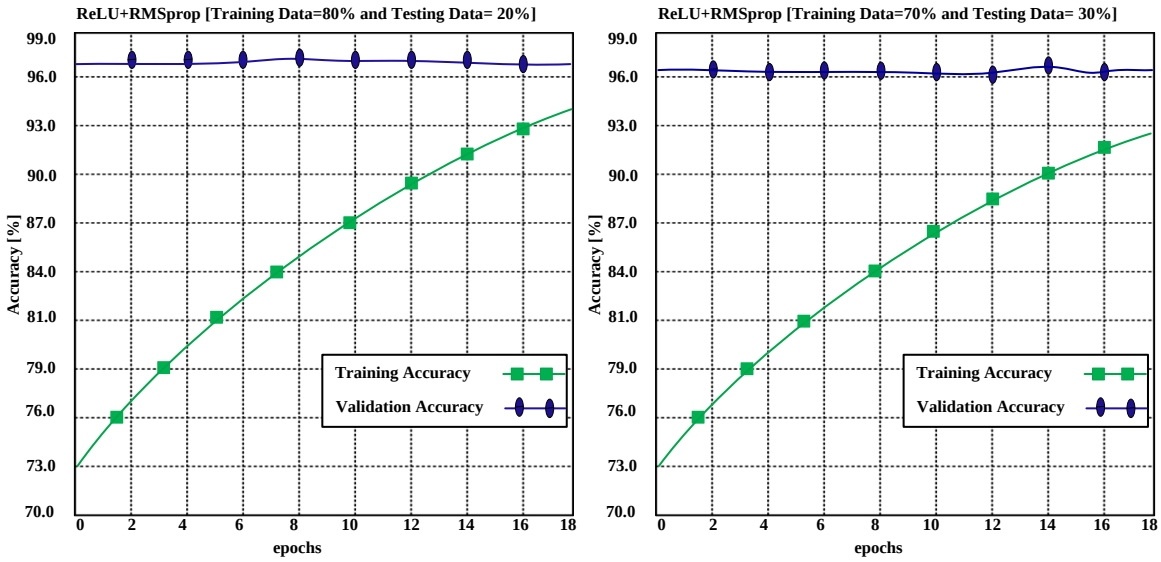

Figure 4 (a): CNN model accuracy with ReLU and RMSprop using 80% training data and 20% testing data; (b): CNN model accuracy with ReLU and RMSprop using 70% training data and 30% testing data

B. ReLU activation function + Adam

The effectiveness of the CNN model using the ReLU activation function and the Adam method is determined. As demonstrated in Figure 5(a), the model achieved 97.12% training accuracy and 98.14% validation accuracy with 80% training data and 20% testing data. The model's performance falls marginally when the amount of training data is reduced to 70% and the number of testing data is increased to 30%. Figure 5(b) shows that training accuracy is 95.89% and validation accuracy is 98.08%.

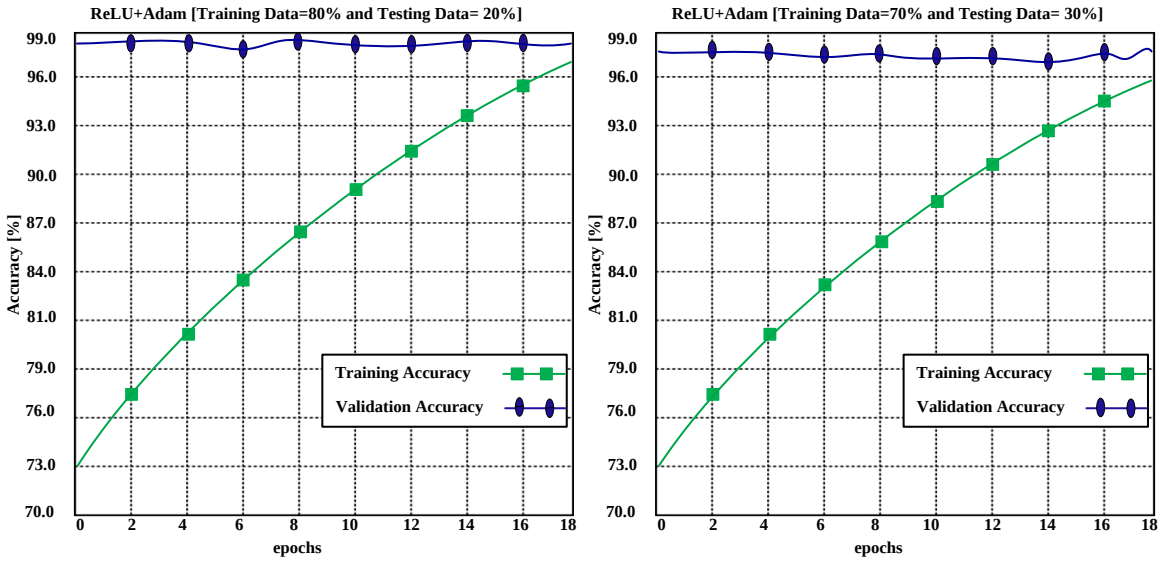

Figure 5 (a): CNN model accuracy with ReLU and Adam using 80% training data and 20% testing data; (b): CNN model accuracy with ReLU and Adam using 70% training data and 30% testing data

C. PReLU activation function + RMSprop

The effectiveness of the CNN model using the PReLU activation function and the RMSProp approach is determined. As shown in Figure 6(a), the model achieved 98.58% training accuracy and 98.45% validation accuracy with 80% training data and 20% testing data. The model's performance falls marginally when the amount of training data is reduced to 70% and the number of testing data is increased to 30%. Figure 6(b) shows that training accuracy is 98.04% and validation accuracy is 97.92%.

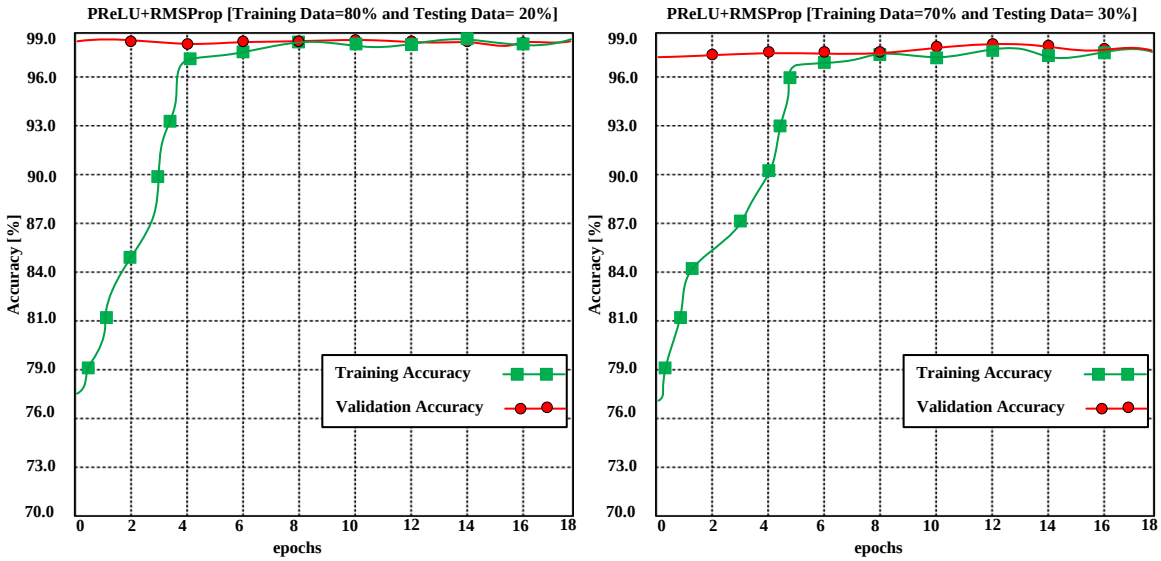

Figure 6 (a): CNN model accuracy with PReLU and RMSProp using 80% training data and 20% testing data; (b): CNN model accuracy with PReLU and RMSProp using 70% training data and 30% testing data

D. PReLU activation function + Adam

The efficiency of the CNN model is determined by applying the Adam technique and the PReLU activation function. As seen in Figure 7(a), the model obtained training accuracy of 98.73% and validation accuracy of 98.61% with 80% training data and 20% testing data. The model's

performance somewhat declines when the quantity of training data is reduced to 70% and the number of testing data is increased to 30%. Figure 7(b) illustrates that the training accuracy is 97.22% and the validation accuracy is 97.04%.

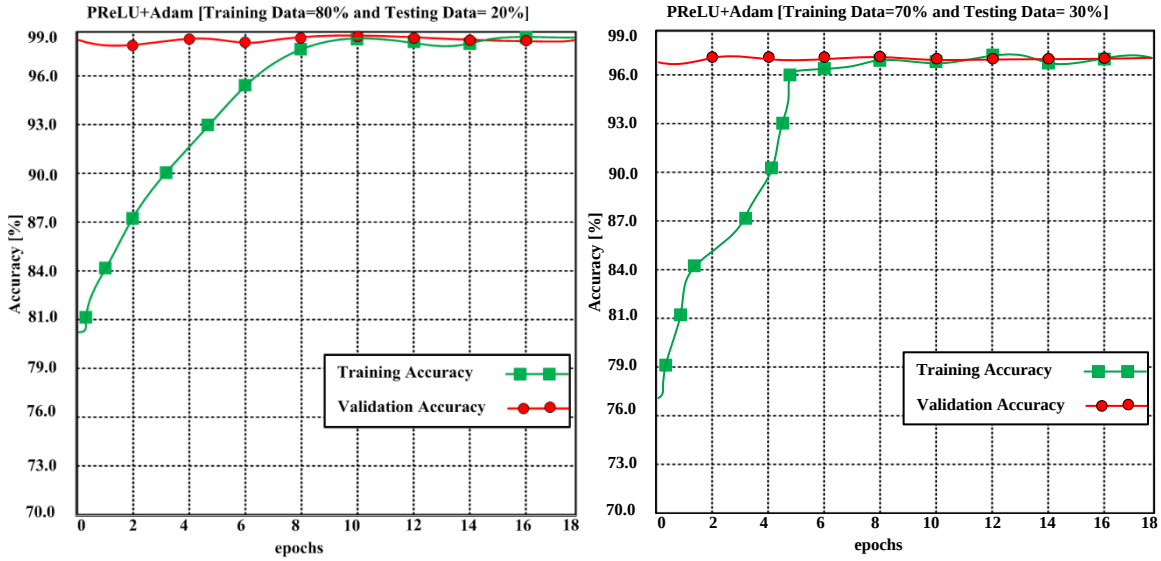

Figure 7 (a): CNN model accuracy with PReLU and Adam using 80% training data and 20% testing data; (b): CNN model accuracy with PReLU and Adam using 70% training data and 30% testing data

Table 1 shows the training and validation accuracies of CNN model for different activation functions and optimization method combinations using 80% training and 20% testing data. Table 2 shows the performance of the CNN model with 70% training and 30% testing data, as well as various activation functions and optimization methods.

Table 1
Performance of CNN for various activation function and optimization method combinations

Case	Activation Function	Optimization Methods	No of epochs	Training Accuracy (%)	Validation Accuracy (%)
1	ReLU	RMSprop	18	94.63	96.43
2	ReLU	Adam	18	97.12	98.14
3	PReLU	RMSprop	18	98.58	98.45
4	PReLU	Adam	18	98.73	98.60

Table 2
Performance of CNN for various activation function and optimization method combinations

Case	Activation Function	Optimization Methods	No of epochs	Training Accuracy (%)	Validation Accuracy (%)
1	ReLU	RMSprop	18	92.74	96.28
2	ReLU	Adam	18	95.89	98.08
3	PReLU	RMSprop	18	98.04	97.92
4	PReLU	Adam	18	97.22	97.04

E. Heat map of confusion matrix

Figure 8(a) depicts the confusion matrix heat map for the CNN model employing the Adam optimization method and the PReLU activation function. Figure 8(b) compares the accuracy of various models for handwritten digit recognition. PReLU+Adam has the highest accuracy for handwritten digit recognition.

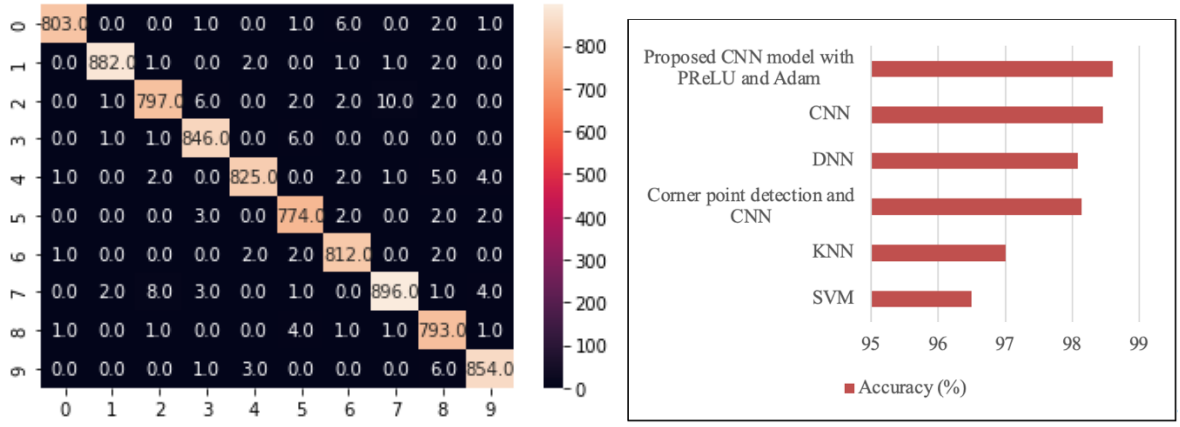

Figure 8 (a): Heat map of confusion matrix for the CNN model with PReLU activation function and Adam optimization method; (b): Comparison of Accuracy of different models for handwritten digit recognition

6. Conclusion and future work

The performance of CNN with different activation functions and optimization methods is evaluated. The main goal of this study is to identify and classify handwritten digits. The handwritten digit recognition necessitates better accuracy in some critical areas. As a result, deep learning techniques are employed with CNN to identify handwritten digits with great accuracy. The CNN is used with ReLU/PReLU activation function and RMSprop and Adm optimization methods. To conduct the experiment, the Kaggle handwritten digit dataset is used. The test results demonstrate that CNN with PReLU activation function with Adam optimization method produces the best validation accuracy of 98.60% for handwritten digit recognition when compared to other methods. The suggested method can be enhanced, utilized with larger datasets, and used to the classification of handwritten alphabets in the future. It is possible to use a three-step model with CNN as the first two classifiers and SVM as the third classifier. The current implementation can be expanded to support additional datasets and/or languages, such as Swedish church records for ARDIS (Arkiv Digital Sweden) or Arabic digits for MADbase (Modified Arabic Handwritten Digits). Techniques for feature selection can also be utilised to limit the training time and error rate.

7. References

[1] Raina, Vineet, Srinath Krishnamurthy, Vineet Raina, and Srinath Krishnamurthy. "Natural language processing." Building an Effective Data Science Practice: A Framework to Bootstrap and Manage a Successful Data Science Practice (2022): 63-73.

[2] Hamdan, Yasir Babiker, and A. Sathesh. "Construction of statistical SVM based recognition model for handwritten character recognition." Journal of Information Technology and Digital World 3, no. 2 (2021): 92-107.

[3] Almiani, Muder, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, and Abdul Razaque. "Deep recurrent neural network for IoT intrusion detection system." Simulation Modelling Practice and Theory 101 (2020): 102031.

[4] Almiani, Muder, Alia AbuGhazleh, Yaser Jararweh, and Abdul Razaque. "DDoS detection in 5G-enabled IoT networks using deep Kalman backpropagation neural network." International Journal of Machine Learning and Cybernetics 12 (2021): 3337-3349.

[5] Chen, Xu, Jianjun Li, Yanchao Zhang, Yu Lu, and Shaoyu Liu. "Automatic feature extraction in X-ray image based on deep learning approach for determination of bone age." Future Generation Computer Systems 110 (2020): 795-801.

[6] Chatzimparmpas, Angelos, Rafael M. Martins, Kostiantyn Kucher, and Andreas Kerren. "FeatureEnVi: Visual analytics for feature engineering using stepwise selection and semi-automatic extraction approaches." IEEE Transactions on Visualization and Computer Graphics 28, no. 4 (2022): 1773-1791.
[7] Hatuwal, Bijaya Kumar, Aman Shakya, and Basanta Joshi. "Plant Leaf Disease Recognition Using Random Forest, KNN, SVM and CNN." Polibits 62 (2020): 13-19.
[8] Su, Dan, Liangming Chen, Xiaohao Du, Mei Liu, and Long Jin. "Constructing convolutional neural network by utilizing nematode connectome: A brain-inspired method." Applied Soft Computing (2023): 110992.
[9] Apicella, Andrea, Francesco Isgrò, Andrea Pollastro, and Roberto Prevete. "Adaptive filters in graph convolutional neural networks." Pattern Recognition 144 (2023): 109867.
[10] Alkhouly, Asmaa A., Ammar Mohammed, and Hesham A. Hefny. "Improving the performance of deep neural networks using two proposed activation functions." IEEE Access 9 (2021): 82249-82271.
[11] Chychkarov, Yevhen, Anastasiia Serhiienko, Iryna Syrmamiikh, and Anatolii Kargin. "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks." CMIS 2864 (2021): 496-509.
[12] Amsaad, Fathi, P. L. Prasanna, T. Pravallika, G. Mamatha, B. Raviteja, M. Lakshmi, Nasser Alsaadi, Abdul Razaque, and Yahya Tashtoush. "Toward Secure and Efficient CNN Recognition with Different Activation and Optimization Functions." In International Conference on Advances in Computing Research, pp. 550-568. Cham: Springer Nature Switzerland, 2023.
[13] Teodoro, Arthur AM, Otávio SM Gomes, Muhammad Saadi, Bruno A. Silva, Renata L. Rosa, and Demóstenes Z. Rodríguez. "An FPGA-based performance evaluation of artificial neural network architecture algorithm for IoT." Wireless Personal Communications (2021): 1-32.
[14] Wang, Xudong, Changqing Miao, and Xiaoming Wang. "Prediction analysis of deflection in the construction of composite box-girder bridge with corrugated steel webs based on MEC-BP neural networks." In Structures, vol. 32, pp. 691-700. Elsevier, 2021.
[15] Gawlikowski, Jakob, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe et al. "A survey of uncertainty in deep neural networks." Artificial Intelligence Review 56, no. Suppl 1 (2023): 1513-1589.
[16] Hnamte, Vanlalruata, and Jamal Hussain. "Dependable intrusion detection system using deep convolutional neural network: A novel framework and performance evaluation approach." Telematics and Informatics Reports 11 (2023): 100077.
[17] Hosseininoorbin, Seyedehfaezeh, Siamak Layeghy, Brano Kusy, Raja Jurdak, and Marius Portmann. "Exploring Edge TPU for deep feed-forward neural networks." Internet of Things 22 (2023): 100749.
[18] Xu, Dongpo, Shengdong Zhang, Huisheng Zhang, and Danilo P. Mandic. "Convergence of the RMSProp deep learning method with penalty for nonconvex optimization." Neural Networks 139 (2021): 17-23.
[19] Shahade, Aniket K., K. H. Walse, V. M. Thakare, and Mohammad Atique. "Multi-lingual opinion mining for social media discourses: an approach using deep learning based hybrid fine-tuned smith algorithm with adam optimizer." International Journal of Information Management Data Insights 3, no. 2 (2023): 100182.
[20] Beltran-Royo, Cesar, Laura Llopis-Ibor, Juan J. Pantrigo, and Iván Ramírez. "DC Neural Networks avoid overfitting in one-dimensional nonlinear regression." Knowledge-Based Systems (2023): 111154.