Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Kalken Mukhtar

Automatic Handwritten Digit Recognition on Images Using Machine Learning Methods

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computer Science
(degree code: 7M06102)
Department of Computer Science
Faculty of Engineering and Natural Science

Supervisor: **Ruslan Jantayev**

Kaskelen, 2022

Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Kalken Mukhtar

Automatic Handwritten Digit Recognition on Images Using Machine Learning Methods

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computer Science
(degree code: 7M06102)
Department of Computer Science
Faculty of Engineering and Natural Science

Supervisor: **Ruslan Jantayev**

Kaskelen, 2022

Abstract

With the transformation to digital information exchange, many papers, including invoices, taxes, notes and surveys, historical data, and test results, still require handwriting. In this context, handwritten digit recognition is required, which is a computer-automated method for decoding records. To achieve this purpose, machine learning mostly employs neural networks, with the convolution neural network being the most common. The primary objective of this dissertation is to develop an automatic technique for detecting handwritten digits by determining the optimal hyperparameter of a convolutional neural network and analyzing the impact of various hyperparameters on network performance. To do this, several values of each hyperparameter were chosen within a specified range using the MNIST database for training and testing. As a result, a large amount of data was collected in which a very high influence of hyperparameter selection on the performance of the neural network was determined, and a 99.2% accurate pure CNN design was achieved.

Аңдатпа

Сандық ақпарат алмасуға көшкенде, көптеген құжаттар, соның ішінде шот-фактуралар, салықтар, жазбалар мен сауалнамалар, Тарихи деректер мен тест нәтижелері әлі де қолжазбаны қажет етеді. Бұл тұрғыда жазбаларды декодтаудың автоматтандырылған компьютерлік әдісі болып табылатын қолмен жазылған сандарды тану қажет. Осы мақсатқа жету үшін машиналық оқытуда негізінен нейрондық желілер қолданылады, олардың ең көп тарағаны конвульсиялық нейрондық желі болып табылады. Бұл диссертацияның негізгі мақсаты-конвульсиялық нейрондық желінің оңтайлы гиперпараметрін анықтау және әртүрлі гиперпараметрлердің желінің жұмысына әсерін талдау арқылы қолмен жазылған сандарды анықтаудың Автоматты әдісін жасау. Ол үшін оқу және тестілеу үшін MNIST дерекқорын қолдана отырып, берілген диапазондағы әр гиперпараметрдің бірнеше мәні таңдалды. Нәтижесінде деректердің үлкен көлемі жиналды, онда гиперпараметрлерді таңдаудың нейрондық желінің жұмысына өте жоғары әсері анықталды және 99,2% CNN-дің дәл таза дизайнына қол жеткізілді.

Аннотация

С переходом на цифровой обмен информацией многие документы, включая счета-фактуры, налоги, заметки и опросы, исторические данные и результаты тестов, по-прежнему требуют рукописного ввода. В этом контексте требуется распознавание рукописных цифр, которое представляет собой автоматизированный компьютерный метод декодирования записей. Для достижения этой цели в машинном обучении в основном используются нейронные сети, причем наиболее распространенной является сверточная нейронная сеть. Основной целью этой диссертации является разработка автоматического метода обнаружения рукописных цифр путем определения оптимального гиперпараметра сверточной нейронной сети и анализа влияния различных гиперпараметров на производительность сети. Для этого было выбрано несколько значений каждого гиперпараметра в пределах заданного диапазона с использованием базы данных MNIST для обучения и тестирования. В результате был собран большой объем данных, в которых было определено очень высокое влияние выбора гиперпараметров на производительность нейронной сети, и был достигнут 99,2% точный чистый дизайн CNN.

Acknowledgements

I thank my supervisor, Ruslan Jantayev, for his remarkable supervision. I thank him for his relentless support, patience, and encouragement. He always supported and helped when difficulties arose. And I thank my family and friends for their unconditional love and continuous support.

Contents

Nomenclature

CNN Convolutional Neural Network

$CRNN$ Convolutional Recurrent Neural Network

CTC Connectionist Temporal Classification

DL Deep Learning

ML Machine Learning

MLP Multi-Layer Perceptron

$MNIST$ Modified National Institute of Standards and Technology

$NIST$ National Institute of Standards and Technology

NN Neural Network

$ResNet$ Residual Network

RNN Recurrent Neural Network

VGG Visual Geometry Group

1. Introduction

1.1 Motivation and Objective

Handwriting recognition is one of the most relevant topics in our time. This is due to the growing urgent need to translate handwritten documents into digital format. In the age of digital technologies, storing information in digital format is the most effective option. The effectiveness and convenience of this can be proved by the fact that the documents will be easily accessible from any necessary point, will be securely stored with the possibility of a quick copy, and it also becomes possible to search and quickly find information among a large number of files.[15] All these advantages are absent when storing information in paper form. At the time of writing the thesis, the most effective translation of documents into a digital version is the use of machine learning algorithms and models in conjunction with computer vision. [18]

This thesis is a solution to classification problems of handwritten recognition by evaluating possible variants of machine learning models with comparison their effectiveness. The effectiveness, in turn, is compared by changing the key parameters that affect the result. Before going deeper, we should familiarize ourselves with the principles of machine learning, the methods of which are the most effective in solving problems of handwritten classification.[2]

Machine learning is a set of algorithms and methods as part of architectures designed to train a machine in order to work independently without human intervention. In the modern field, machine learning is used to train a computer machine for prediction purposes under certain conditions. [45] Conclusions of information as a forecast and is accepted as a result. Machine learning, in turn, is divided into several sub-branches, which in turn are deepened in a certain area. The most popular of these areas is considered to be deep learning, which has

become popular due to the strong development of neural networks. Thus, constructed neural network architectures are used to solve a wide range of tasks, even in some cases surpassing human abilities. [30]

Unlike a machine, it is much easier for a person to recognize and determine the classification of certain things. Many methods and technologies are required to implement these abilities in a machine. Despite the good achievements in this area, there is still a need for improvements as well as for closing gaps. For example, we can take two branches of handwriting recognition: online and offline [21]. Online handwriting recognition is achieved by tracking the stroke in a dynamic order. In offline recognition, the main focus lays on a pre-trained model using a pre-prepared database. Online recognition thanks to the tracking of the stroke, thereby dynamically processing the data obtained, shows more accurate results in comparison with offline recognition. In this regard, at the moment there is a need for the development of this area.

To recognize handwritten numbers from images and documents, one common algorithm is used in which the image is first segmented and then recognition methods are used. In recent years, many segmentation options and recognition algorithms have been proposed. As segmentation algorithms, one can note the implementation of either heuristics, foreground information, or a combination of them[22] . On the one hand, excessive segmentation allows you to find the right points with a higher probability and efficiency, on the other hand, it increases the cost of the necessary calculations in the quality of equipment as well as in the quality of time. Considering all of the above, segmentation algorithms currently have performance limitations related to overlapping handwritten digits in images and documents. And as algorithms and recognition architectures, support vector machine (SVM), artificial neural network (ANN) and convolutional neural network (CNN) are currently considered the most effective. But the best result is shown by the CNN algorithm, which in turn have different implementations that differ in connection as well as in the number of layers. Changes in these parameters have an impact on the overall result. [43]

The first step is to submit a database for training. But for more effective training, preprocessing of the database is required. As a processing, the image transfers various operations. Noise reduction, tilt correction, normalization, smoothing and skeletonization are mainly used [31]. The result of these manipulations can be

used in the future to train the expected neural network. The images are mainly served as a common page, which is why it is necessary to divide the images by words and symbols. To do this, segmentation is used, which in turn uses a certain algorithm to divide the image into a subimage.

In the next stage, mainly convolutional neural networks use the methods of image generation in vector format [36]. To improve the efficiency of the learning algorithm, as well as to reduce the time, filters are used by which image reduction in volume is achieved. In order not to lose the clarity of the image without any noise, an image noise reduction algorithm is implemented. Next comes the classification process. Depending on the number of layers and other parameters of the neural network, many classifiers are used that reduce performance with each step of classification. Classifiers come in different types, including such types as classifiers for recognition, structural, as well as combined. In each step, a certain classifier is used, which, depending on the choice and parameter, affects the overall network performance, which affects the recognition result [46].

The main purpose of this work is a thorough study of the parameters of the CNN neural network by using a pure architecture. The MNIST database was chosen as the database for recognition and training. In parallel with this, a goal arises which is to study the influence of various hyperparameters with subsequent tuning to improve the efficiency of CNN architecture training. The results of this study are not insignificant. At the beginning, a thorough analysis of the parameters of the neural network is carried out, such as the size of the training step, the size of the image, fields, the size of the core, the number of layers, etc. In the next steps, there is possibility to obtain the best recognition result using the MNIST database, which is the most frequently used database in the research of handwritten numbers.

1.2 Problem Definition

The accuracy and efficiency of CNN are largely affected by the hyperparameters used, which are established via trial and error. Iteration number, learning step, hidden layers, number of layers, hidden units, activation function, and so on are examples of hyperparameters. These factors are given a weighting in training since they have a direct impact on the algorithm and the outcome. The model's

parameters are distinct from hyperparameters, which must be defined before the model can be trained. The CNN GoogLeNet, VGG, AlexNet, and ResNet variations have roughly 78, 57, 27, and 150 hyperparameters, respectively, in the known models. Incorrect CNN hyperparameter selection can result in high costs, low efficiency, and erroneous recognition. Under some situations, the researcher's choice of hyperparameter configuration has a significant impact on the results of model training. This raises a variety of concerns about CNN's architecture for activities involving handwriting recognition.

1.3 Research Question

The following research questions are created to meet the aims of this study in the context of examining the hyperparameters of the CNN architecture in the recognition of handwritten digits:

- RQ1: What is the impact of different hyperparameters on CNN performance?
- RQ2: In which cases does the CNN neural network more efficiently extract the necessary distinctive features from the image of handwritten characters?
- RQ3: Is it possible to achieve high results using only pure CNN?

1.4 Thesis Outline

The goal of the thesis, as well as its motivation and contributions to the area of study, were presented in the Introduction chapter. The Chapter 2 explains Background and Related work. Describes the Literature review and provides context for the themes employed. This chapter provides a comprehensive assessment of the literature on handwritten digits recognition and the approaches used in the handwritten digits recognition process. The Methods and Datasets used are described in depth in Chapter 3. The outcomes and analysis of the experiment are detailed with a discussion in Chapter 4. The outcomes of the experiments are then given. The conclusion and future work of this thesis are briefly summarized in Chapter 6.

2. Background And Related Work

2.1 Background

2.1.1 Machine learning

Machine learning is among the technologies that Artificial Intelligence is based on. Machine learning is a continuous data analysis that enables a system to fine-tune its operations in answer to fresh data. Machine learning has sparked a lot of attention, owing to the allure of a computer learning for itself. It alluded to the idea of self-awareness, and although this isn't currently attainable, machine learning demands our attention. The procedure starts with the input of training data into the chosen algorithms. The final machine learning algorithm is shaped by the training data, which might be known or unknown information. To ensure that the machine is working correctly, it is fed test data from the learning algorithm. This is determined by comparing predicted and actual outcomes. If they don't line up, the algorithm will be trained again and again until the desired result is achieved. This is how the machine learning algorithm is able to learn and improve its accuracy over time. [38]

There are two types of machine learning: supervised and unsupervised learning. About 75% of machine learning is supervised, with the remaining 10%-15% being unsupervised. The rest is made up of reinforcement learning: [28]

Supervised Learning:

This is accomplished by training the algorithm on pre-defined or labeled data. Because the data has been tagged and is therefore known, it is referred to as 'supervised'. It is a guided approach that aids in the model's education. After

the model has been trained with known data, it may be supplied with unknown data, which will clearly result in a different answer. For instance, once taught to detect a picture of an orange, it may be given photographs of various fruits to improve its accuracy.

Unsupervised Learning:

This is accomplished by giving the algorithm with unlabeled and unknown training data. Due to the algorithm's unknown state, it gets no direction, which is why this technique is referred as unsupervised. While the machine learning system is training on the data, it looks for trends and makes an effort to react appropriately. Naturally, this may be challenging. Using the orange example again, incorporating images of nectarines, which might seem identical at times, can be tough for the system, but the more it practices, the more proficient it becomes at distinguishing them.

Reinforcement Learning:

The program learns data via trial and error and then determines which activities result in the biggest rewards. The three components of reinforcement learning are the agent, the environment, and the actions, with the agent serving as the decision-maker learner, the environment referring to the environment with which the agent interacts, and the actions referring to what the agent does. When an agent chooses activities that increase the expected reward over a certain time period, this is called reinforcement learning. This is most readily accomplished when the agent operates inside a framework of side policies. [41]

2.1.2 Convolutional Neural Network

Convolutional neural networks (CNNs) are a kind of multi-layer perceptron (MLP) networks that were initially utilized in 1981 [23]. CNN's computations are inspired by the human brain. Visually, humans see and recognize things. We humans teach our children to identify items by exposing them to hundreds of images. This enables a youngster to recognize or make predictions regarding previously unseen items. A CNN operates in a similar manner and is widely used for visual image analysis. A CNN combines the phases of feature extraction and classification, requiring little pre-processing. A CNN can automatically extract affluent and related elements from photos. Additionally, a CNN may achieve a high level of recognition accuracy even when just a little amount of training data is given. No

longer are design details or prior knowledge of features necessary to be acquired. The primary advantage of adopting a CNN model is that it makes use of the topological information included in the input, resulting in great recognition results. A CNN model's recognition scores are also insensitive to the rotation and translation of input pictures. In contrast, MLP models do not make extensive use of input topological information. Additionally, MLP is not shown to be effective for complicated problems, and they do not scale well for higher resolution pictures. [19]

The CNN model has been widely used in recent years to recognize handwritten digits from the MNIST benchmark collection. Certain studies have claimed handwritten digit recognition accuracy of up to 98 percent or 99 percent. An ensemble model was created by combining numerous CNN models. MNIST digits were used in the recognition experiment, and an accuracy of 99.73 percent was recorded [33]. However, thorough examination reveals that the great identification accuracy of the MNIST dataset pictures is accomplished only using ensemble approaches. While ensemble approaches improve classification accuracy, they do so at the expense of greater testing complexity and computing cost in real-world applications Three layers compose a convolutional neural network: the convolu-

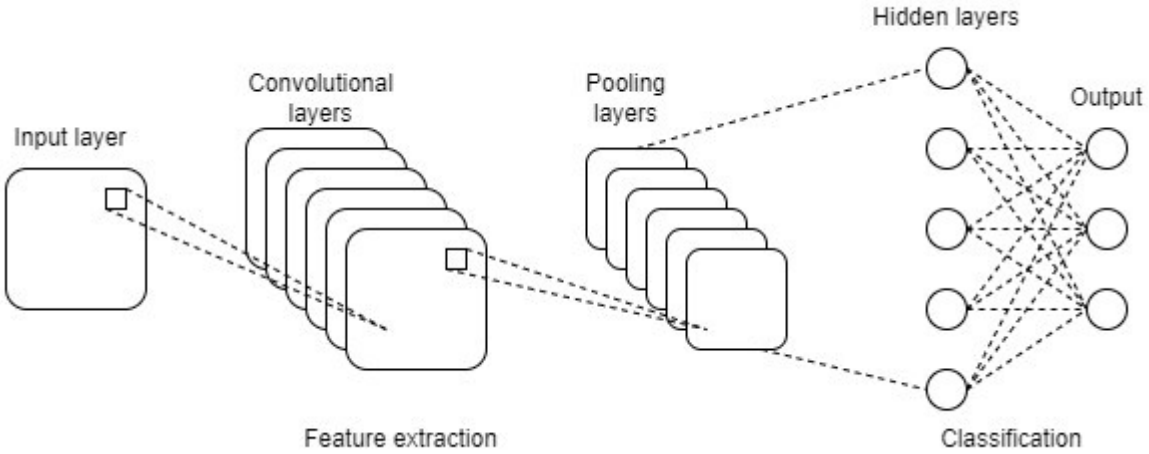

Figure 2.1: Convolutional neural network

tion layers, the pooling layers, and the output layers. Usually, the pooling layer gets omitted. As seen in Figure 2.1, the standard convolutional neural network design with 3 convolutional layers is highly suited for classifying handwritten pictures. It comprises of an input layer, many hidden layers, a fully connected layers, and an output layer. Neurons inside one layer communicate with neurons in the

following layer, which facilitates scaling for greater quality pictures. Sharing or subsampling may be used to compress the input's measurements. The input picture is seen as a group of tiny subspaces known as "receptive fields" in a CNN architecture. On the input layer, a convolutional mathematical operation is performed, simulating the answer to the subsequent layer. It is essentially a visual input that triggers the reaction. [27]

Input Layer

The input layer contains the data for the whole CNN. It typically represents the picture's pixel matrix in neural networks used for image processing. The CNN's input layer must include picture data. Three-dimensional matrixes are used to describe image data. We'll need to redesign it as a separate column. For instance, if we have a picture with a size of $28 \times 28 = 784$, then we must convert it to 784 x 1 before putting it into the input. If there are "n" train samples, the input dimensions will be (784, n). [12]

Convolutional Layer

A convolutional layer is a kind of layer that is used to compress and extract information from images according on the properties of the picture being processed. Placed in front of classification neural networks to improve their performance. It is the RGB values of each pixel in the picture that are utilized as input data, and these values are processed by input neurons with the size of an a * a. Only black and white colors are utilized in handwriting recognition, and hence the values of each pixel are set to the value between black and white, which is a value somewhere between 0 and 1. When a specific filter of size b * b is used, all of the values from the preceding layer are lowered by solving the equation of form (a - b + 1) * (a - b + 1), which is then applied to the prior layer. For example, for a 28 * 28 image and 3 * 3 filter, the output resulting after convolution operation would be of size 26 * 26. The following characteristics, such as step, fill, and field, have the most significant impact on the neural layers in this situation. [29] The matrix that consists of the color value digitally shown in each pixel is multiplied by the filter matrix thereby reducing the overall image size. This method of convolution is similar to the method of image formulation in the visual cortex of the brain of living beings as well as in humans. Figure 2.2 shows an example where

the image matrix stored in the initial neuron having a size of 28 * 28 using a 5 * 5 filter is transformed into a convolution layer with a size of 24 *24. There is also a parameter called stride. Which is responsible for the step of the filter movement along the image matrix. For example, if the step is 1, then the filter will move through the layer one step to the end of the layer. Stride cannot be equal to a non-integer number and the number depends on the size of the filter and the image. Basically, stride is equal to 1, but if necessary, compress the layer of a large image to a small size, the step can be increased with the subsequent appearance of distortion. When the filter moves through the layer, there is a

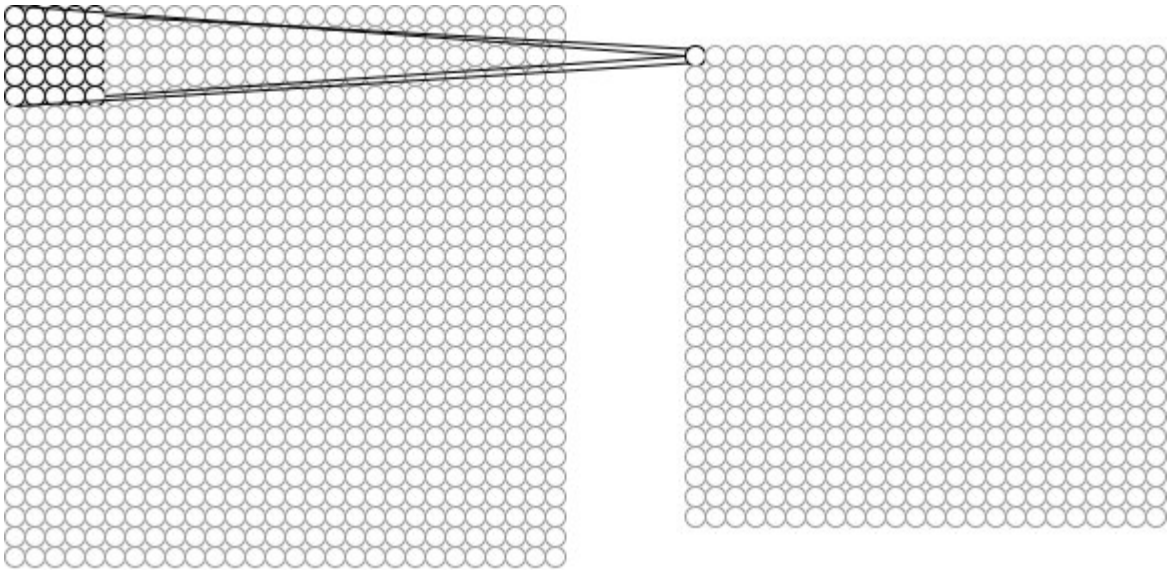

Figure 2.2: Visualization of input neuron 28*28 and convolutional layer 24*24 with kernel size 5*5.

problem with insufficient processing of pixels that are located in the corners of the layer. In this connection, there is a flaw in the information provided. A filter with a certain stride value moves through the layer covering each pixel with a certain frequency at the same time the frequency of coverage of the extreme pixels is underestimated. In this connection, there is a need for a parameter called padding. When the view is enabled, this parameter adds zero values at the edges of the matrix, which leads to the fact that the filter begins to cover the extreme pixels to a greater extent. Two types of padding are mainly used: valid and same. When valid, pixels with a zero value are added at the edges. And with the same, no values are added and the neuron remains in the same form. An example is given in Figure 2.3. [14] The output from the input to the convolutional layer is a feature map, that is smaller than the input layer and can be described by

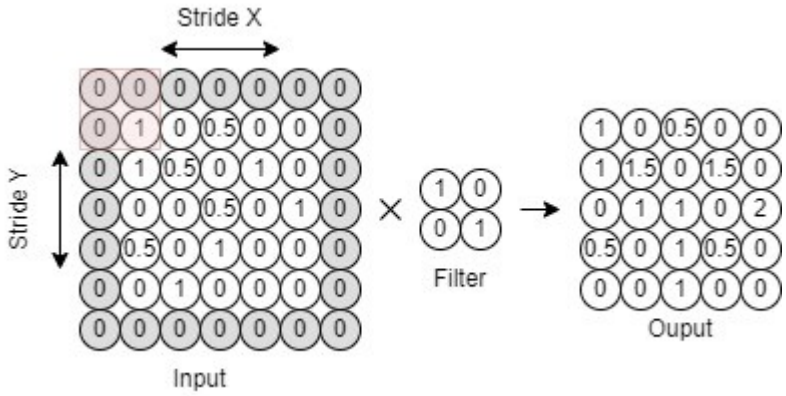

Figure 2.3: Convolutional layer with same padding.

the equations. This output of equations depends on the relationship between the input feature map, filter size and stride. [5]

$$W_{mx} = W_{m-1x} - F_{mx}S_{mx} + 1$$

$$W_{my} = W_{m-1y} - F_{my}S_{my} + 1$$

where W_{mx}, W_{my} are the sizes of output feature map, F_{mx}, F_{my} are the filter sizes, S_{mx}, S_{my} are the stride size and 'm' is the index of layer. Following the convolution of the picture, there is an issue with the distortion of some minor portions of the image following recovery from the convolution neurons. Numerous solutions to this problem have been offered, with dilated convolutions being one of the most successful. This approach enlarges the kernel by inserting "holes" between the filter pixels. Thus, the post-convolution picture better maintains the object's features. Downsampling is used to minimize the size of the image, convolution is used to recover the image, and upsampling is used to restore the image. However, when dilation is used, the process completely replaces the preceding steps and also better maintains the features of the object or picture. For instance, Figure 2.4 illustrates a vegetable picture produced using the two approaches explained before. [44]

$$y[i] = \sum_{k=1}^{K} x[i + r * k]w[k]$$

The above equation (3) describes the main process of dilated convolution. Where y[i] is the i-th element of the resultant feature map, x[i+r*k] is the initial data of the image, w[k] is the values of the convolution map, r in x[i+r*k] is the skip

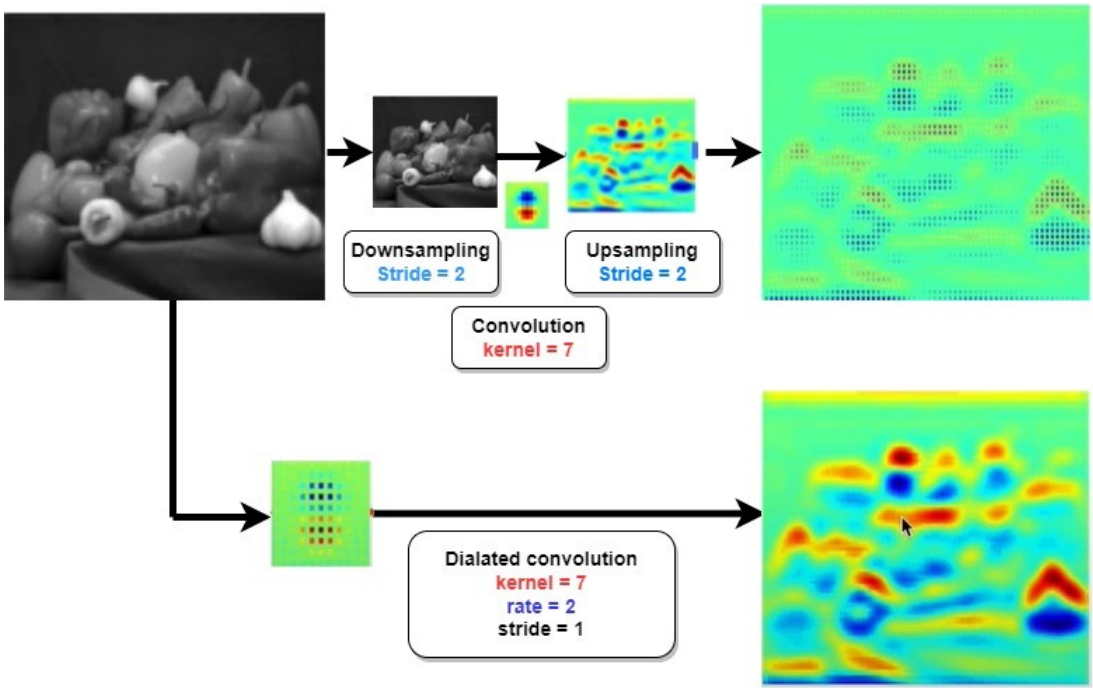

Figure 2.4: Dilated convolution with kernel = 7, rate = 2, stride = 1.

step, that is, the number of zero missed elements. If r is equal to 0, the equation will give the same result as with ordinary convolution.

Hidden Layer

Between the algorithm's outputs and inputs are hidden layers where the function adds weight to the inputs and sends it via an activation function as the result. In a nutshell, the hidden layers change the inputs to the network in a nonlinear manner. The hidden layers change according to the neural network's purpose, and equally, the layers differ according to their respect to the weights.

Pooling Layer

With a large image size, it becomes necessary to use many CNN layers. For example, if the image has a size of 1000x1000, then in this case it will be necessary to use about 500 convolutional layers to achieve a certain result in the network. Such a number of layers is very costly on the part of the computing power of the equipment as well as on time.[3] The pooling method is used to solve this problem. There are two types of pooling: average and max pooling:

1. Average pooling: The selection is based on the average values of the elements

$$F(x) = 1/k^2 \sum_{i=i_i...i_i+k} \sum_{j=j_i...j_i+k} x_{ij}$$

2. Max pooling: The selection is based on the maximum values of the elements

$$F(x) = max(x_{[i,i+k],[j,j+k]})$$

In max pooling, a feature map from convolutional layer is used as input. Depending on the step size, that is, stride and the size of the filter, there is a selection by pixels with the maximum value. For example, in Figure 2.5 on a feature map of size 4x4, max pooling was applied with a filter size of 2x2 and stride equal to 2. As a result, the size of the feature map decreased by two, saving the necessary data for further manipulation. Using the same max pooling on the above

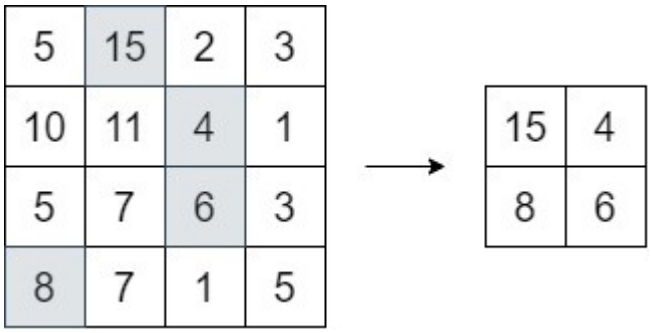

Figure 2.5: Max pooling with filter 2x2 and stride 2.

discussed 1000*1000 image, it will be possible to reduce the number of layers to 17: 8 convolutional layers, 8 max pooling and 1 fully connected layer. Average pooling works in exactly the same way, but instead of sampling by the maximum value, pixels of the average value are selected. Max pooling is mainly used due to its comparative efficiency. [11]

Activation Functions

In biological neural networks, the action potential in a neuron determines the state of a neuron, either it is activated or not. In artificial neural networks, activation functions are used to bring a neuron to state 1 or 0, that is, activated or not. The activation function takes as input the feature map value of each neuron, depending on the selected calculation function, outputs the calculated

value to the neural network. Thus, the calculation of neurons is reduced instead of a set of values, occurs only needed values. Convolutional networks mainly use nonlinear activation functions due to their ability to solve non-trivial problems using a small number of nodes. Let's take a closer look at the three most popular types: tanh(x), ReLU and Leaky ReLU. [6] The tanh(x) function, which is a kind

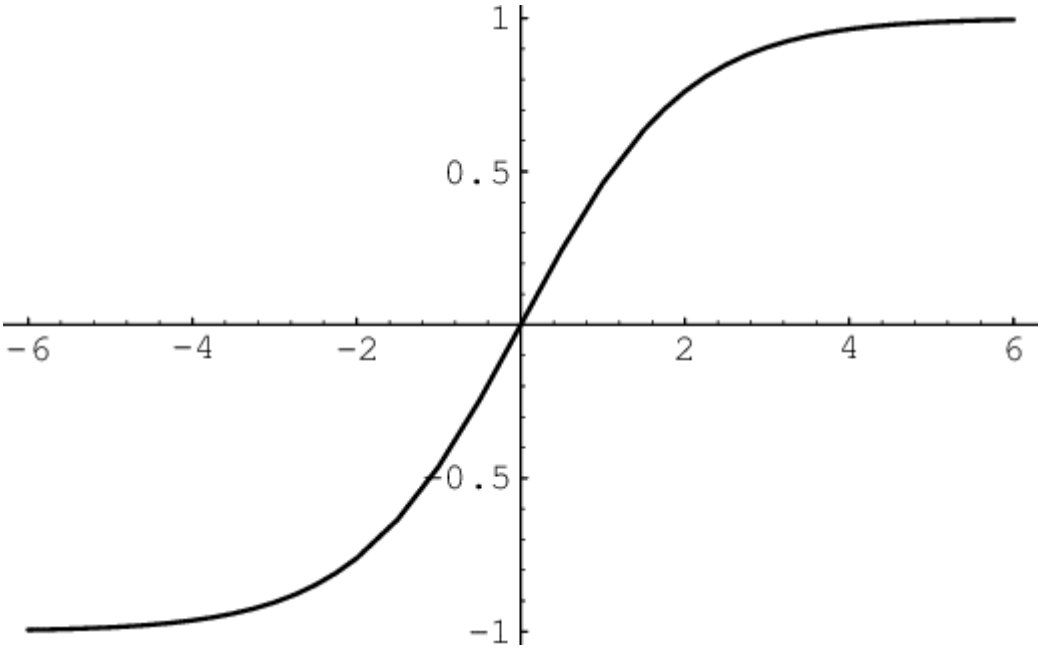

Figure 2.6: tanh(x) function.

of activation function that aims to solve the mean value problem:

$$tanh(x) = 2\alpha(2x) - 1$$

As shown in Figure 2.6, the gradients in the tanh function fluctuate in the interval [-1; 1]. Depending on the input value, the output value will also have a negative or positive result. Also, one of the advantages of this function is that it is centralized at 0. But this advantage is overlapped by the fact that the gradients begin to saturate at a higher value. This is noticeable on the chart in the form of large tails. The rectified linear activation function (ReLU) is a piecewise linear function (Figure 2.7) which is defined as:

$$f(x) = max(0, x)$$

where, $f(x) = 0$, if x <= 0 and $f(x) = x$, if x > 0.
Figure 2.7 shows a graph of the ReLu function where you can notice gradients

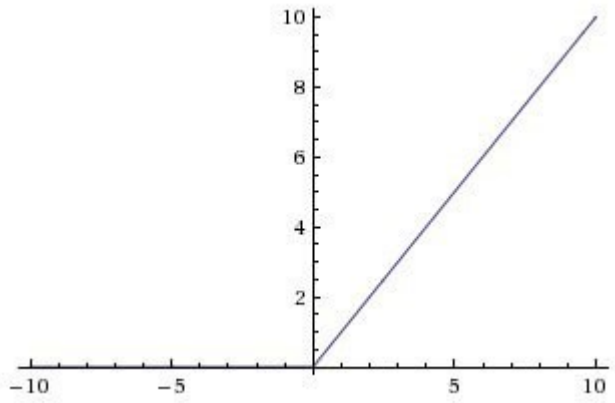

Figure 2.7: ReLu function.

that have a value below 0 are converted to 0, and those that have positive values remain unchanged. Also, the ReLu function, unlike the tanh function, has some advantages. Firstly, thanks to saving the value, the gradients do not saturate. Secondly, the function is very efficient from the point of view of machine computing. Thirdly, the calculation takes less time, 6 times less compared to tanh. Of the disadvantages, it can be noted that the function is not centralized under 0. And there is also disadvantage in the calculation of negative gradients. In connection with the latter, a solution was invented in the form of the Leaky ReLU function, which is often used in neural networks. [9] The equation of the Leaky

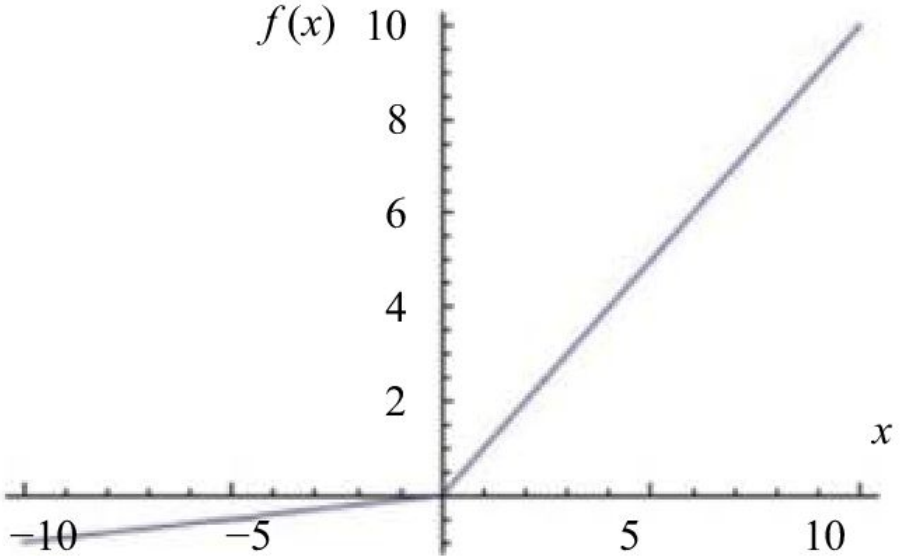

Figure 2.8: Leaky ReLu function.

ReLU function is described below:

$$f(x) = max(0.01x, x)$$

where, $f(x) = 0.01x$, if x < 0 and $f(x) = x$, if x > 0.
The Leaky ReLu activation function is completely similar to ReLu, there is a difference in that it copes with the disadvantages associated with the "dying" of gradients at a value below zero. Figure 2.8 shows a graph where we can notice that gradients below zero multiplied by 0.01, which lowers the value but at the same time retain their significance. There is also a Random ReLu activation function where, instead of a static 0.01, any value is selected under certain conditions. But mostly Leaky ReLu is widely used.

Recurrent neural network

Recurrent neural networks (RNN) are a kind of neural network that is utilized in the sequential data modeling of sequences of events. There are many similarities between the way it works and how a human brain function. For the most part, recurrent neural networks are capable of producing predicting findings with sequential data that other algorithms are unable of producing. In feed-forward

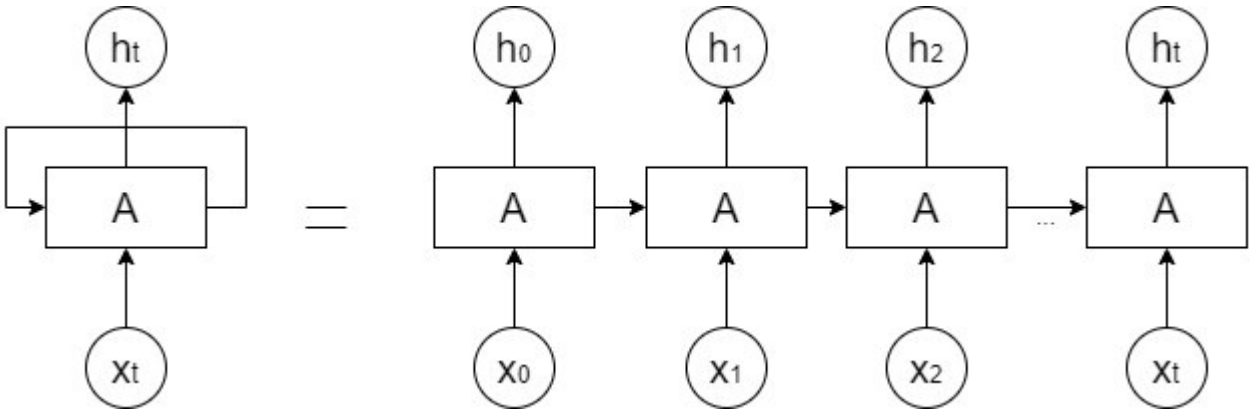

Figure 2.9: Recurrent Neural Network.

neural networks, information is sent from the input to the output in a single direction. It is hard to preserve knowledge about prior inputs in such networks, which is why there is a problem with forecasting what may happen next. And in RNN, as shown in figure 9, the data is looped over and, owing to the hierarchical memory, the prior input data is saved. For instance, if the input data includes the word "hello" then the sequence of letters "h", "e", and "l" is lost in a feed-forward neural network, making it difficult to anticipate the following letter. And with RNN, the sequence is retained in order to facilitate the prediction process.[24]

Connectionist Temporal Classification

Connectionist Time Classification (CTC) is a technique for processing the output of a neural network in tasks such as handwriting recognition or voice recognition that include time. CTC does not need alignment of data sets, which simplifies the learning process. When constructing a dataset from an image of text or symbols, the matching picture symbol is assigned a particular time step. As seen in Figure 2.10, a number of issues occur with this method, including calculating the time step in which numerous characters may fit, or vice versa, finding the time step in which the character does not fit entirely, resulting in the creation of a duplicate. [25] When CTC is applied to a picture, just the necessary check is performed,

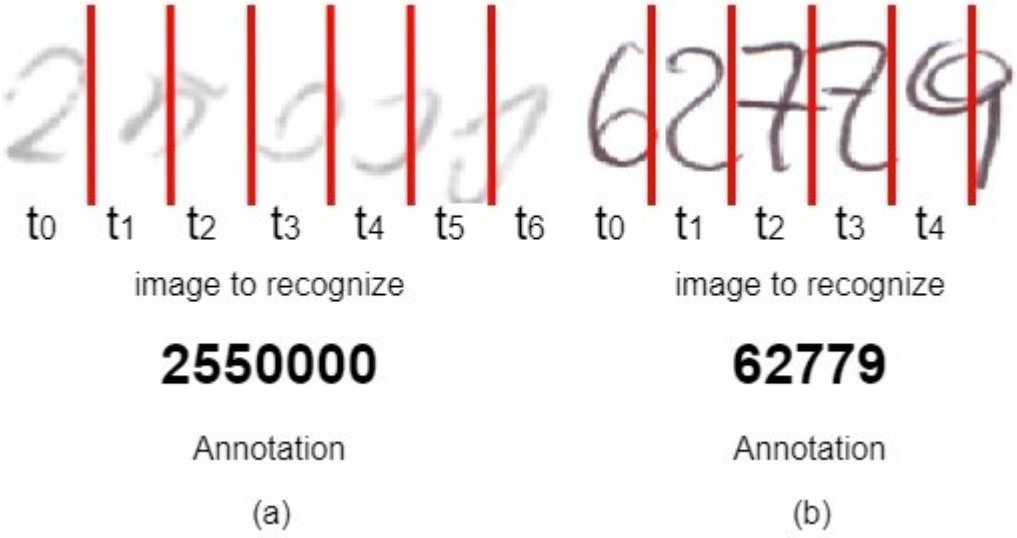

Figure 2.10: (a) The annotation for a situation in which few characters consume more than one-time step (b) The annotation for a situation in which each character use one-time step.

regardless of the width or location of the letters in the image. Following that, there is no need to process the CTC operation's output. It is now feasible to get the outcome of a network activity immediately through decoding techniques.
CTC is used mainly for three purposes: loss calculation, text encoding, and decoding.
Loss calculation
To train a neural network, feedback is required to determine the accuracy of the weight selection. This is accomplished by calculating loss and using the neural network's output and the truth table as parameters. Given instance in figure 2.11, for a pair of characters (1, 2), a set of three characters (1, 2, -) is chosen. Three

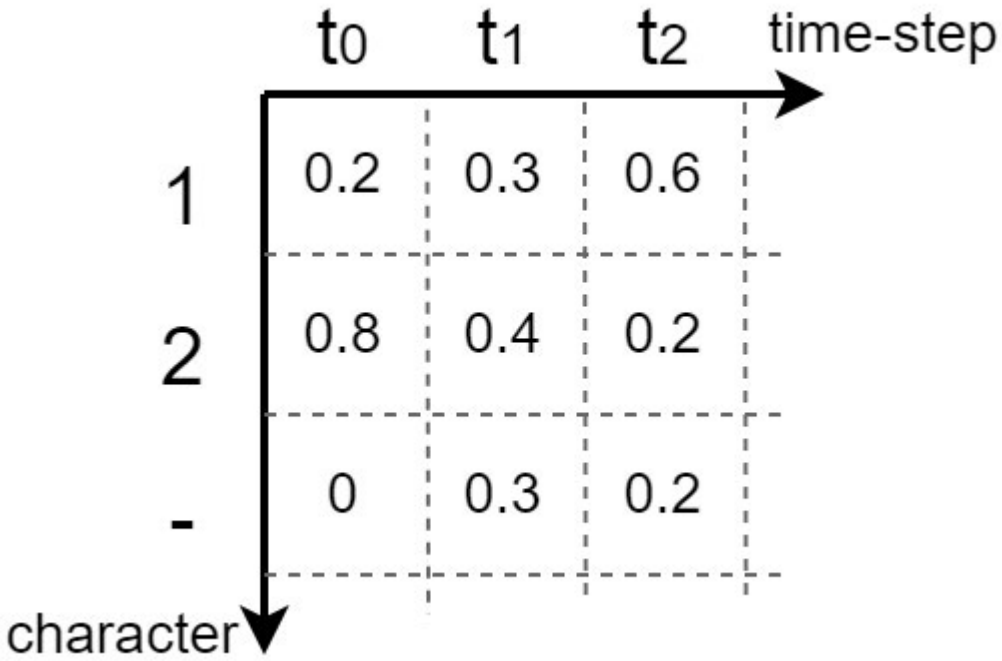

Figure 2.11: Output from NN with probabilities.

time-steps result in a 3x3 matrix. Each cell's value represents the probability of this symbol occurring in this time-step. The sum of all probabilities in each time-step equals one. The estimated probability of symbols is multiplied by each probability in each time-step. For example, for the path "1–" is 0.2x0.3x0.2 = 0.012, and for the path "111" is 0.2x0.3x0.6 = 0.036. To produce an estimate of a symbol's likelihood, the estimates of all possible pathways to the associated text are added together.

For instance, if value of ground truth is equal to "1", then all potential ways for "1" to occur via Figure 11 are as follows: "111", "1–", "-1-", "11-", "-11", "-1". By adding the probabilities of each point on a distinct route, we get 0.036 + 0.012 + 0.06 + 0.012 + 0.18 + 0.18 = 0.48. This number represents the probability of a sign occurring. Following that, we can compute the negative logarithm of this figure in order to determine the value of loss. To train a neural network, the network is fed back the loss value. [8] Text encoding

Without CTC, an issue with duplication occurs when the same character is repeated in the time step. CTC prevents this issue by concatenating repeated characters into a single character. For instance, if the value of the characters is "123," and "1" is repeated three times, advancing the next result to "11123," CTC resolves this issue by combining the letters to produce "123." Following

that, an issue emerges about what to do if the symbol in the truth table originally contains repeats. To overcome this issue, CTC employs a pseudo-character known as a blank "-". If there are repeated characters during encoding, a blank sign is added between them. For instance, if the number "1224" is translated to "1-2-2-4," the following choices may occur: "111-22-2-4","11-22-222-44", which are both valid. Variants of the type "11-22-4" are already incorrect since they will be decoded as "124". Decoding
After the neural network has been trained, previously unknown types of pictures are provided. In these circumstances, the best path method is primarily used to decode the output values of a neural network. The approach is based on sampling the largest value in each time step, and it also allows for the deletion of blanks and duplicate characters. For instance, in the matrix shown in Figure 11, this approach selects "2" in the first time-step, "2" in the second, and "1" in the third. So, the decoded value is "221". [26]

2.2 Related Work

Using various variations of neural networks handwriting recognition has already achieved impressive results. First of all, it all started with the work of Simard et al. in 2003, where the use of convolution networks as neural networks for document recognition was proposed. The network was trained and tested using the MNIST database, where as a result, the convolution network showed relatively the best result in the form of an error rate value of 0.4. After that, various variations and improvements of this network began to appear, one of which is the symbiosis of CNN and RNN proposed by Shi et al. and named by the authors of convolutional recurrent neural network (CRNN). The architecture has a much smaller weight and number of layers in contrast to other architectures and has been tested on scene text recognition showing one of the best results. Also, the CNN architecture was tested on different databases of manuscripts of languages in one of which the authors Boufenar C. et al. successfully trained the model using an Arabic manuscript and achieved a classification rate of 93.25%. Ahmed et al., and Hussain M. et al. successfully conducted a study in CNN training using a database of Urdu language manuscripts.
Tien Huu Nguyen et al.[39] proposed improved CNN-based events detectors by

introducing the non-consecutive convolution. They used ACE 2005 dataset to train network and reached best performance on the source domain and on the target domains.

Work of ZahraZiran et. al. [48] describes a method of transcript alignment in early printed books using deep models called Faster R-CNN. The paper uses two object detection models trained to find words.

Hinz et al. [16] performed aprocahes to optimize the architecture of CNNs using multiple algorithms. It was primarily utilized for increasing-resolution picture inputs. They use different input dimensions with the same input data to determine the hyperparameters' importance. Following that, they discovered that the same hyperparameters are critical for each dataset, regardless of the picture resolution.

Albelwi et al. [1] have suggested a strategy for improving CNN hyperparameters via preprocessing stages. Their approach is centered on minimizing the collection of instances used to train the neural network using instance selection.

Wei-Chang Yeh et al. suggested optimizing the simplest CNN model's hyperparameters. They optimize the LeNet model as the CNN model using simplified swarm optimization. As a consequence, studies on the MNIST, Fashion MNIST, and Cifar10 datasets revealed that the improved LeNet model outperformed the original LeNet model.

Taniya Seth et al. [34] suggested optimizing CNN hyperparameters for time series prediction. They accomplished this by solving the hyperparameter optimization issue using bilevel programming.

Yifeng Wang et al. [42] established an approach for optimizing CNN hyperparameters using the perceptual hash algorithm. At the conclusion of the experiment, it was determined that this strategy considerably enhanced the accuracy of the CNN model's picture categorization.

Adrian Vulpe-Grigoras et al. investigated the optimization of CNN hyperparameters for face emotion identification. The improved model was trained using the HER 2013 database and had an accuracy of 72.16 percent.

K. Uyar et al. [40] developed a Fuzzy Tree Model for optimizing CNN hyperparameters. They trained the model using the Cifar10 and Cifar100 datasets and produced a variety of outcomes by combining several CNN hyperparameters. The top model achieved an accuracy of 97.70 percent in recognition.

WooYoung Lee et al. [20] suggested a technique for enhancing the performance

of CNNs using hyperparameter adjustment during the feature extraction process. They accomplished this by using a parameter-free harmony search approach and training the model on CifarNet and Cifar-10 datasets.

As the works discussed above demonstrate, there are several versions and studies aimed towards enhancing CNN, each of which is focused on a certain field of study. Modern CNN architectures improve with time, and as a result, the number of hyperparameters rises in direct proportion. Several of these hyperparameters include the number of layers, the size of the filter, the number of feature maps, and the step. The number of hyperparameters in a single neural network job might range from 10 to hundreds of different variants susceptible to modification. Not only the selection of static and dynamic hyperparameter values, but also the selection of potential values for optimization is a challenging problem in such instances.

3. Methodology

3.1 Dataset

In this work, the MNIST database is used, which consists of separately segmented handwritten digits. Each experiment will split the database into two sections: one for training neural network models and another for validation.
The MNIST database is a widely utilized resource for scholars working in the area of handwritten number recognition. It contains 60,000 and 10,000 images of handwritten Arabic numerals for model training and for model validation. Each image is 28x28 pixels in size and has been transformed to a black and white tone. The database was constructed using data from the NIST database, which is divided into two sections: a special database 1 and a special database 3. A part for training a neural network was composed of 30,000 images from a special database 3 and 30,000 images from a special database 1. And to create a subset of the database for testing purposes, 5000 images from each section were chosen. Over 500 writers contributed to the creation of each section of the NIST database. Black-and-white images from the NIST database were first converted to the 20x20 format while maintaining image centralization, and then by calculating the centered pixel mass were converted to the 28x28 format and processed by normalization and smoothing, which led to grayscale.
This data set has established itself as the main database of handwritten numbers and will be used in our experiments. To improve the image quality and in situations of image resizing, image preprocessing methods will be applied.

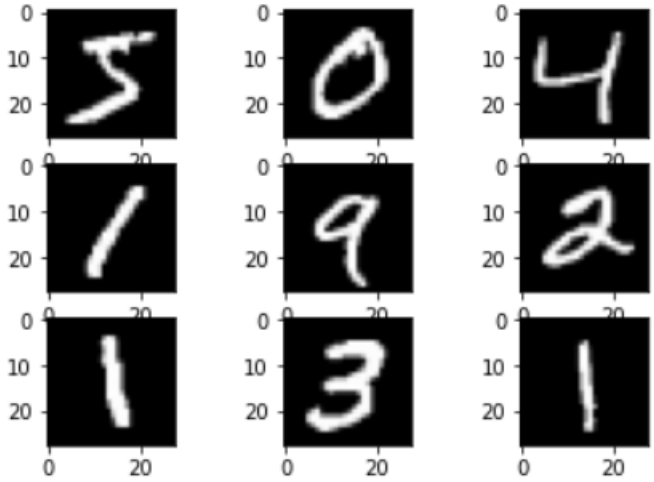

Figure 3.1: Images from MNIST database.

3.2 Preprocessing

3.2.1 Scaling

Scaling an image is an important factor of image preprocessing, which entails zooming in or out of a picture for various purposes.
To convert an image with a resolution of w × h to an image with a changed resolution of new_w × new_h, the scaling factors s_w and s_h must be specified:

$$s\_w = new\_w/w$$

$$s\_h = new\_h/h$$

If the scale factor is less than one, compression occurs; if the scale factor is more than one, stretching occurs. Nearest neighbor interpolation is one of the most frequently used scaling algorithms. Each source pixel is adjusted by its adjoining input pixel in this manner. A pixel with x, y coordinates at the input transforms the coordinates to x/s_x, y/s_y at the output. Due to the fact that coordinates of this kind have no place to be, the answer is to round them to the closest integer. As a result, this strategy is called like that since the number is rounded to the closest neighbor.

3.2.2 Noise reduction

The technique of reducing noise from images is called image noise reduction. Noise can be present in any image and is a hindrance to the effective training of the neural network, as it creates extra values in the pixel matrix. These values give an incorrect and accurate picture of a certain figure.
The autoencoder approach is employed in this paper. Autoencoders are a special kind of uncontrolled artificial neural networks. When input data is in the form of an image, the autoencoder reduces it to smaller code representations, which are then decoded back to the visible picture. So, the encoder is responsible for downsampling the data to a smaller size, while the decoder is responsible for reassembling the picture into its original format.

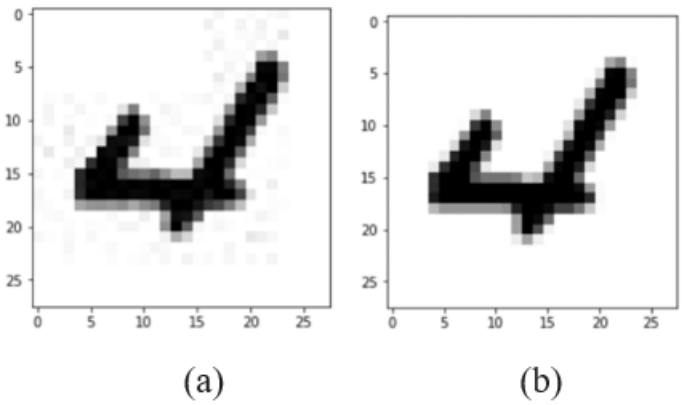

Figure 3.2: Before (a) and after (b) noise reduction.

3.2.3 Zero centering

Zero centering is a frequently used preprocessing technique. It operates by removing the average value from each data point in order to produce centered data at zero. For instance, if the input signals are positive or negative, the gradient computed along the back-propagation line will be positive or negative. As a result, it's difficult to observe that some parameter changes are direction-dependent. This complicates convergence. Following each one, the gradients become excessive, resulting in a decrease in the neural network's optimization. The neural network performs well due to the symmetry of the data, that is, when all pictures are equally centered. To carry out this preprocessing , the following equation is used:

$$\overline{x_l} = x_l - E_x[X]$$

Where, (x_l) is the features of image and $E_x[X]$ is the feature-wise mean. This procedure, if required, is typically reversible and is independent of the connection between the two samples and their constituents.

3.2.4 Slanting

Slant correction is the process of adjusting the text's slant in order to enhance recognition via general normalization. The text's slope is one of the distinguishing characteristics of a handwritten text, since each author's handwriting has a more or less unique slope. This might result in significant visual changes in the look of the handwritten text. The slant is corrected using the following method:

$$\widehat{\alpha} = argmax f(l, \alpha)$$

where, $\alpha \in [-45; 45]$, the function $f(l, \alpha)$ is used to determine the variance of the projection of the image I that is bounded by the angle .

3.2.5 Skew correction

Scanned images may be somewhat skewed. To extract information more precisely from an image, it is required to identify and compensate for skew. The projection profile approach is one of the simplest and most often used methods for correcting skew. The picture is initially projected horizontally using this approach, by picking the sums of pixels along the rows of the image matrix. Following that, the image is rotated at various angles, with the corners rotated in modest steps with a constant value. Finally, the difference between the peaks is determined. As a result, the ideal value of the skew rotation is the angle at which there is a significant variation in the slope value. After determining this value, the picture is rotated to conform to it.

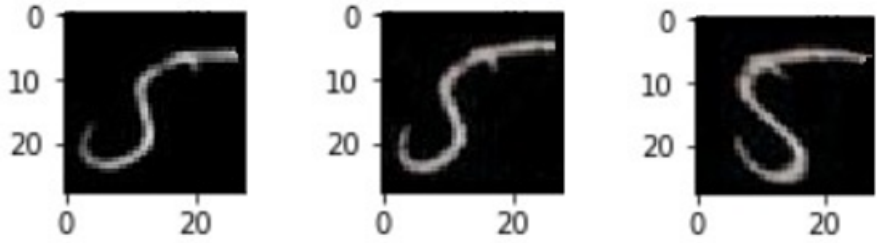

Figure 3.3: Sample of image (a) after skew correction (b) and slanting (c).

3.3 Experiment setup

3.3.1 Layer hyperparameters

Before making changes to the hyperparameters of the neural network, it is required to modify the convolution layer itself by adjusting and comparing the data acquired. Hyperparameters such as kernel size (k/s), stride (s), padding (p), dilation (d), input size (i/s) and output size (o/s) are considered as hyperparameters of the convolution layer. According to the hyperparameter of the neural network, average standard hyperparameters such as learning rate 0.01 and number of epochs are picked as 4, it is enough for data extraction. From MNIST database images with a size of 28x28 are supplied to the input of the neural network. Also, two variations were chosen to evaluate the impact of the number of layers: three and four convolution layers. The number of feature maps were picked by selecting the average values for three feasible alternatives for each layer values: for three layers 6-12-24, 8-16-32, 12-24-32; for four layers 6-12-24-32, 12-24-28-32, 6-12-24-24, respectively. Hyperparameters in 5 distinct cases were picked for each layer. The settings of all architectural parameters were determined in such a manner that the function of each parameter was considered. Hyperparameters for all convolutional layers include kernel size (1-5), stride (1-3), padding (1-2), and dilation (1-2).

3.3.2 Batch size and number of epochs

The neural network is trained via backpropagation, which returns a gradient after a single pass. Due to the iterative nature of learning, the neural network is taught in a cycle manner. The method is repeated up to a specified number

Neural Networks	Layers	k/s	s	p	d	i/s	o/s
CNN1	Layer 1	3	2	1	1	28	9
	Layer 2	3	2	1	1	9	3
	Layer 3	3	2	1	1	3	1
CNN2	Layer 1	5	2	1	1	28	12
	Layer 2	3	3	1	1	12	4
	Layer 3	3	3	1	1	4	1
CNN3	Layer 1	5	3	2	1	28	10
	Layer 2	5	3	2	1	10	4
	Layer 3	5	3	2	1	4	2
CNN4	Layer 1	5	2	2	1	28	14
	Layer 2	5	2	2	1	14	7
	Layer 3	5	2	2	1	7	4
CNN5	Layer 1	3	1	1	1	28	14
	Layer 2	3	1	1	1	14	6
	Layer 3	3	1	1	1	6	2

Table 3.1: Details of configuration of three-layer convolutional neural network.

Neural Networks	Layers	k/s	s	p	d	i/s	o/s
CNN1	Layer 1	3	2	2	1	28	14
	Layer 2	3	2	2	1	14	7
	Layer 3	3	2	1	1	7	4
	Layer 4	3	2	1	1	4	2
CNN2	Layer 1	5	2	2	1	28	14
	Layer 2	3	2	2	1	14	7
	Layer 3	3	2	1	1	7	4
	Layer 4	3	2	1	1	4	2
CNN3	Layer 1	3	1	2	2	28	14
	Layer 2	3	1	2	2	14	7
	Layer 3	3	2	1	1	7	4
	Layer 4	3	2	1	1	4	2
CNN4	Layer 1	5	1	2	2	28	14
	Layer 2	5	1	2	2	14	7
	Layer 3	3	2	1	1	7	4
	Layer 4	3	2	1	1	4	2
CNN5	Layer 1	5	2	2	1	28	14
	Layer 2	5	2	2	1	14	7
	Layer 3	5	2	2	1	7	4
	Layer 4	3	2	1	1	4	2

Table 3.2: Details of configuration of four-layer convolutional neural network.

of times during each iteration. The value of the epoch hyperparameter controls this quantitative repetition. Each iteration contains a sample of data on a neural network. The term "sample" refers to a single column of data. While the batch size hyperparameter specifies the number of such samples that should be passed through the neural network before the configuration is updated. To establish the batch size, the value is primarily divided by the number of images in the database. Essentially, the value is taken to be between 1 and 100. [13] Otherwise, if the value is not evenly divided by the number of images in the database, the last batch will contain a gap. To avoid this, a limited number of images from the database are frequently removed. The epoch, on the other hand, specifies the number of such iterations. It can be equal to or greater than huge values, depending on the situation. Both hyperparameters have an effect on the training effectiveness of neural networks. With a small batch size, there may be time and machine resource expenses; with a big batch size, there may be increase in losses. Epoch must also be regulated, as the model may not complete its research or incur additional costs by adopting a single static position. Following the trial using convolution layers, the most effective variant will be determined. Additionally, it will be used to test the neural network with different hyperparameters such as epoch and batch size. As parameters for batch size, the range from 10-200: [10, 20, 30, 50, 100, 200]. And as parameters for epoch, the range from 10-30: [10, 20, 30]. The overall performance of the neural network after training is compared with each experiment. The results can be as follows:

1) Accuracy will increase with increasing epoch and batch size;
2) Accuracy will increase as epoch and batch size decrease;
3) Accuracy will increase with increasing epoch and decreasing batch size;
4) Accuracy will increase with decreasing epoch and increasing batch size;
5) Accuracy will take the maximum value with the average value of epoch and batch size;

3.3.3 Optimization algorithm

The primary purpose of optimization algorithms is to optimize a neural network by modifying network parameters. The selected optimizer is responsible for the changes in the network's weight or learning rate. All of this is done to cut costs and boost production. Optimization algorithms or methods are in charge of minimiz-

ing losses and producing the most precise results. There are numerous algorithms, each of which is distinguished by the manner of calculation performed. One of the most popular is SGD, Adagrad, AdaDelta, Adam.
Stochastic Gradient Descent (SGD) is an algorithm that often updates the model parameters after calculating the losses in each training case. For instance, datasets typically comprise 100 rows. SGD, like gradient descent, updates the parameters of the neural network 100 times during each revolution of the data set.

$$f = f - \alpha * \bigtriangledown J(f; x(i); y(i))$$

where, $x(i)$ and $y(i)$ are the training examples.
Adagrad overcomes the disadvantage of other algorithms in that the pace of learning is constant with each revolution. At the selected time step, the optimizer modifies the learning rate.

$$g_{t,i} = \bigtriangledown_f J(F_{t,i})$$

AdaDelta is an extension of AdaGrad that, in general, solves the issue of learning rate slowing down by removing the need to aggregate all previously squared gradients. By constraining the gradients collected in the past, the method fixes the size x. In this scenario, rather than the sum of all slopes, an exponential moving average is employed.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2$$

Adam (Adaptive Moment Estimation) operates in such a manner that it does not roll as quickly as possible due to the possibility of jumping over the minimum; as a result, the speed required for a full search is reduced. Adam, like AdaDelta, saves exponentially decreasing mean of previous squared gradients. Adam also stores exponentially decreasing mean of previous gradients M(t).
After the batch size and epoch search results are acquired, tests with the optimizers SGD, RMSProp, Adagrad, Adadelta, Adam, Adamax, and Nadam employing these parameters will be conducted. By choosing, the effect of the optimizer on the neural network's overall performance will be investigated.

3.3.4 Learning rate and momentum

To reduce losses during training, a descent to the function's global minimum occurs. However, this is not always achievable on the first attempt, as the function may differ from the complexity and have multiple depressions where the search can become trapped and the only way out is to lift. The momentum is one method for determining the desired value. [47] It is utilized in a large number of networks. The following function can be used to describe the momentum:

$$\triangle\omega_{ij}(t) = \mu_i\delta_i\gamma_i + m \triangle \omega_{ij}(t-1)$$

where, m is momentum described by given t time. If 0<m<1 it means global parameter that must be determined try and error.
Momentum is calculated by adding a fraction of m to the last weight update. If the gradient continues to point in the same direction, the momentum increases the step. As a result, the learning rate must be reduced, as a big impulse is used. If the speed does not drop, or even increases, the steps become so huge that it becomes fairly possible to jump over the minimum.
The learning rate of a neural network is a parameter that controls the step size at which the weights are updated. The learning rate is modified within a narrow range that accepts only positive values, often between 0.00 and 1.00. The rate at which a neural network model adapts is determined by the value of the learning rate. A smaller number of periods necessitates a higher value for the learning rate, as weight changes should occur in a short interval. While large values in periods indicate that high learning rate values are unnecessary. It is vital to fine-tune the value of the learning rate, as a value that is too large can result in a suboptimal solution, while a value that is too little can result in excessive costs and learning process stagnation.
When training a neural network, it is occasionally preferable to vary the learning rate over time. It is vital to be able to identify the needed values at the appropriate time, as kinetic energy is significant at high speeds. This value results in a chaotic displacement of parameters, which cannot be located in the function's deeper and narrower regions. Knowing what value to assign to a particular segment of the training time is extremely difficult and frequently needs a trial-and-error technique, involving a slight reduction in value. There are three

techniques to perform a time decay:
Exponential decline. It takes the mathematical form $\alpha = \alpha_0 e^{(} - kt)$, where α_0 and k are hyperparameters and t is the number of iterations.
Attenuation. At the start of training, this parameter is set to a value that reduces the value of the learning rate after a specified amount of time. The numbers may range from a fall of a fraction of a percent over five periods to a decrease of half. The value should be chosen in accordance with the nature of the problem being addressed. There is one advantage to employing this strategy as a search for the optimal value with a constant loss value by varying the learning rate by 0.1 in each session.
The decay of 1/t. The mathematical expression for the decay of 1/t is $\alpha = \alpha_0/(1 + kt)$, where α_0, k are hyperparameters and t is the iteration number.
Step-by-step attenuation is more practical since it has the largest effect on the neural network's variability and enables you to assess the status and effect of the learning rate over time.
As the trials progress and the optimal optimization algorithm's effects are analyzed, a choice will be made based on the learning step and the momentum. The learning step is chosen between five values [0,001; 0,005; 0,01; 0,05; 0,1] and the momentum for each value is chosen between 0.0 and 0.9.

3.3.5 Weight initialization

Weight initialization is one of the most critical elements to consider while training a neural network.
Each connection in a neural network is made up of particular values of parameters that are conveyed through the network, which are referred to as weights. These weights represent the weight of the total of the data that is fed into the neuron via the connection. The optimization algorithm [10] that was previously selected progressively alters the status of these weights as time progresses. A loss function is imitated in order to achieve the desired mode of operation for a neural network, which may then be achieved by combining two or more neural networks. In order to make changes to these parameters, the optimization algorithm must first determine the first feasible value of the weights before proceeding with the procedure.
Weight initialization is a method that must be performed before the weights of a

neural network can be determined. The values should be tiny and unpredictable in order to be effective. During training, these variables set the starting point for optimizing the neural network model, which is determined by the values.
For each new initialization of the collection of weights, the beginning point for the optimization process is a different moment in time. It is possible that a different final set of weights with different performance characteristics will occur as a result of this. Because it is impossible to start all weights with a value of 0.0, the initial value must be assigned some significance to it. Consequently, there is some asymmetry in the error gradient for the optimization algorithm, which results in the method failing to optimize. Over time, it became clear that researchers in this discipline had developed a certain standard range for initial weight selection. The ranges of these small random numbers are [-0.3, 0.3], [0, 1], and [-1, 1]. Essentially, this random pick is sufficient. However, in order to get more precise outcomes, more customized methodologies have been developed throughout the last decade. These criteria may result in a significantly more efficient optimization procedure during model training.
These recent weight initialization approaches may be classified according to the activation function utilized, such as "Sigmoid and Tanh" or "ReLU." There is a normalized Xavier initialization approach that is utilized in the Tanh and Sigmoid functions. It is computed using a random integer with the same probability distribution (U) as the ranges:

$$weight = U[-(sqrt(6)/sqrt(n+m)), sqrt(6)/sqrt(n+m)]$$

where, n denotes the node's inputs and m denotes the layer's outputs. As seen in Figure 3.4, the range begins at around -0.3 to 0.3 with a limited number of inputs. Subsequently, as the number of inputs rises, the values fall to around -0.1 to 0.1. When the ReLU activation function is used, it was discovered that initializing the "xavier" weight is ineffective for weight initialization. The initialization of "he" is a well-known technique for initializing the weights of layers and nodes in a neural network through the ReLU function. A random integer with a Gaussian probability distribution is used as the initialization value in this technique

$$weight = G(0.0, sqrt(2/n))$$

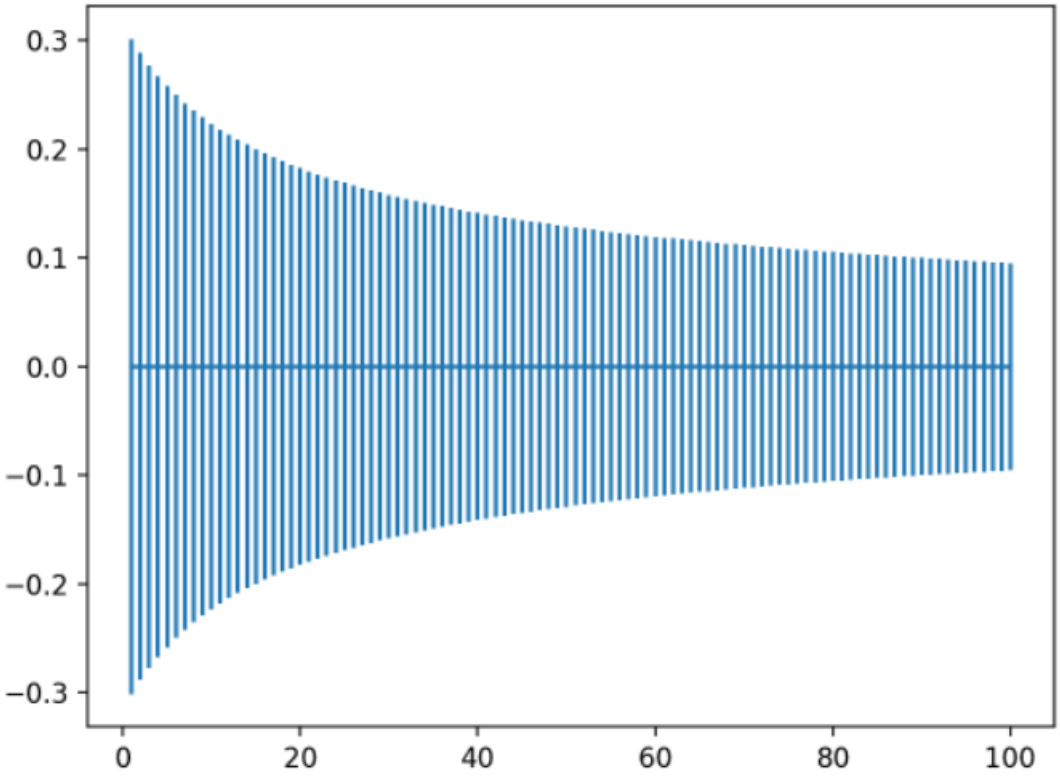

Figure 3.4: Range of Normalized Xavier Weight Initialization.

where, n is the node's number of inputs.

3.3.6 Activation function

Although it is often referred to as a "transfer function" the activation function may also be referred to as a "compression function" when the output range is restricted. Activation functions must, in general, be nonlinear in nature. Non-linearity in the network structure may also be referred to as non-linearity in the network.

A neural network's capabilities and performance are greatly influenced by the activation function that is used to solve issues[4]. The choice of activation function is critical in this process. The significance of this decision also demonstrates that in certain networks, numerous distinct activation functions may be utilized in various regions of the network, which demonstrates the flexibility of the network. The activation function may be used both within and outside of the computer, and it can be used before and after internal processing. However, in general, and to the extent that the network enables it, the same activation function is utilized by all nodes.

As previously stated, there are three different sorts of layers in the network. The first layer is comprised of the input layers, which are responsible for receiving raw input data. Afterwards, there are hidden levels, for the input of which there is data from a previous level of the hierarchy. The output layers, which are designed for use in forecasting, are added afterwards.

The activation functions that are utilized in layers of the hidden type and layers of the output type may vary from one another. The activation function for indoor layers is often the same as for outside layers. In addition, the output layers may make use of the activation function, which is distinct from the hidden layers in this regard. The kind of prediction determines which option is selected and how much of a difference there is. It is also possible to differentiate between activation functions. As a result, for a given input value, the first-order derivative is computed. As a result of the error back propagation technique, neural networks are trained, and this is a need. It is difficult to update the weights without having a derivative of the prediction error, which is required.

Furthermore, in addition to the above described forms of activation functions, there are several additional types of activation functions that are employed in neural networks. In practice, however, only a tiny number of functions are used by researchers.

The values of hidden layers are not directly related to the data that was used to generate them. This, in turn, prevents the model from producing any output data.

Furthermore, the number of hidden layers may be zero, depending on the neural network used.

In general, layers in a neural network use a differentiable nonlinear activation function in order to operate [7]. As a result, the model starts to investigate more complicated functions, which is a positive development. A network trained with a linear activation function, on the other hand, will not have this possibility.

As an experiment, it will be required to examine the three primary activation functions that are utilized in hidden layers:

- ReLU
- Sigmoid
- Tanh

The output layers in the neural network model reflect the level in the model where

the majority of the computations for the prediction are performed. An output layer is included in all neural network models.
It is also vital to take into account the three primary functions that are employed in the output layers:
• Softmax
• Sigmoid
• Softmax
Following the examination of the initial weights and selection of the ideal choice, the impacts of activation function on the neural network will be examined via iteration as a continuation of the experiment. Softmax, ReLU, Sigmoid, Tanh, Softplus, Softsign, Linear, and Hard Sigmoid are all functions.

3.3.7 Dropout regularization

When training a model using neural networks, numerous samples often overflow the training data. Due to the fact that these neural networks are often trained on short datasets, they are capable of overloading the training data.
As a result of these issues, the model incorporates statistical noise seen in the training data. As a consequence, when the model is evaluated on fresh data, its performance degrades. This is most apparent in the test dataset. Overfitting may amplify the generalization error. An ensemble of neural networks is one approach to solving such challenges. These neural networks use a variety of model configurations, which reduces the need for retraining. It entails combining all conceivable neural network configurations into a single data set. [35] They are merged via the use of an average of projections from each model, which is not practical in reality. Because the model may be approximated, it can be used with a minimal data set. Another significant downside of this strategy is the increased training expenses and support required to run many models concurrently.
There is now a mechanism to drop nodes randomly during training. Additionally, this enables the simulation of a vast variety of network designs inside a single model. This dropout is a low-cost, high-effective method of regularizing the neural network during retraining. It is applicable to all classes of neural networks and significantly reduces generalization mistakes.

During training, this approach ignores certain neurons in the output layer at

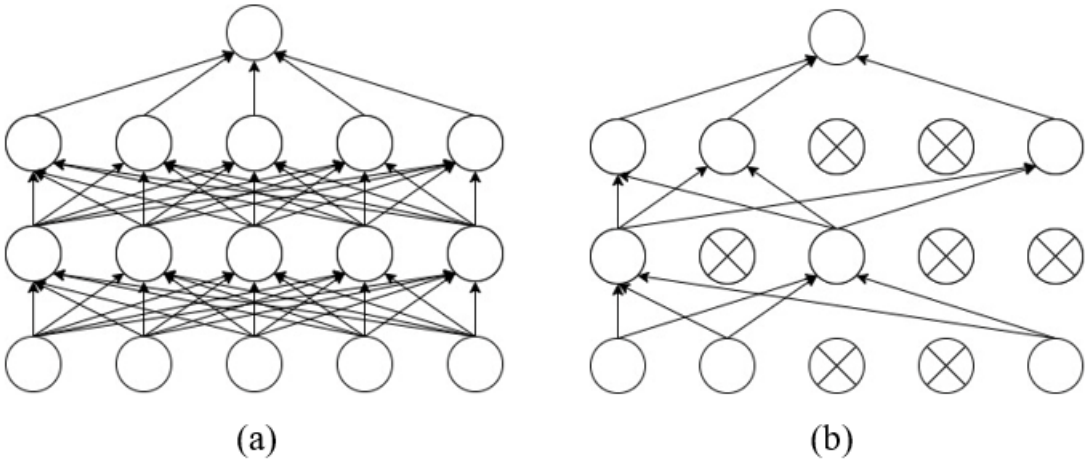

Figure 3.5: Standard neural network (a) and after applying dropout (b).

random. It seems as if the layers are treated as layers with varying node counts and that each layer update during training is conducted using a different configured layer.

This screening is carried out independently for each level of the neural network. Additionally, it is employed with the majority of layer types. Several of these layers are densely linked, convolutional, or repetitive. For instance, a network layer equipped with both long-term and short-term memory.

Additionally, it may be applied in all network's hidden levels. Additionally, on a visible or input level. However, it is omitted from the output layer.

The default dropout hyperparameter accepts values between 0.0 and 1.0, where 1.0 indicates no dropout and 0.0 indicates layer attenuation and no output data from the layer.

In the buried layer, the standard value is between 0.5 and 0.8.

However, there is a risk that the network's weights will grow in size as a result of the probabilistic removal of active levels. [17] This may result in a big weight, which may indicate an unstable network.

To address such issues, you might apply a weight restriction. This results in the magnitude of all weights in the layer being less than the specified value. For instance, the weight constraint's upper limit with a value between 3 and 4 is often employed.

After examining and choosing the activation function for a more extensive examination of the dropout, all eligible values between 0.0 and 1.0 with a step of 0.1 will be picked for future experimentation. Additionally, the aforementioned

weight limitation must be considered, which will be chosen from a range of 1 to 5 with a step of 1 for each dropout value.

3.3.8 Number of neurons in hidden layer

To identify the number of neurons in the hidden layer, it is required to consider the training data's structure. Essentially, the number of neurons in this layer equals the number of columns in the data. In rare circumstances, an extra node is inserted to offset the NN arrangement.
And the number of neurons in the output layer is entirely governed by the model's setup. With regards to the hidden layers, you must also consider the data. If the data is linearly separable, no hidden layers are required.
Essentially, for the majority of activities, standard guidelines are employed to achieve a desired performance. To do this, the hidden layer's settings are established using just two criteria. The initial concealed layer has a depth of one. The second number of neurons is equal to the sum of the input and output layers' neurons.
However, for more exact tuning, formulae are employed to determine the minimum number of hidden neurons required to avoid overfitting:

$$N_h = N_s(\alpha * (N_i + N_0))$$

where, N_s number of samples, an arbitrary scaling, N_i number of input, N_o number of output neurons. Additionally, there are techniques for adjusting alpha to a number between 3 and 8. However, you may essentially restrict yourself to a value of 2, which will not cause the model to retrain. There is a need to keep the number of free parameters in a neural network to a minimum by using a small fraction of the degree of data freedom. [32] The degrees of freedom of the data are calculated by multiplying the number of samples by the degree of freedom within each sample. Additionally, it may be written as Ns* (Ni + No). All values must be self-contained. From this, we may deduce that is a proxy for the model's generality. Additionally, it implies avoiding retraining to the greatest extent feasible. There are various established standards for calculating the optimal number of neurons to employ in hidden layers, including the following:
1. The hidden neuron count should be between the input and output layer counts.

2. Input layer + output layer = 2/3 of hidden neurons.
3. The total number of hidden neurons should not exceed twice the size of the input layer.
These three guidelines serve as a guide for choosing the appropriate quantity and the most effective result. However, in the end, it is entirely dependent on individual demands, and quantity selection is a matter of trial and error. Following the determination of the network's dropout value, the network's number of neurons will be chosen from 1 to 50 with varying step intervals.

3.3.9 Evaluation metrics

The first metric is a measure of learning losses. It's a measure of how well the deep learning model is being taught. This is accomplished by analyzing the training error between epochs. These training losses are determined by calculating the total number of mistakes for each example in the training set. There is also a note on the calculation of training loss after each batch. The acquired data is then shown by showing training losses.
The second metric likewise measures losses, but in the validation phase. It evaluates the performance of a deep learning model using a validation dataset. This data set contains a representation of the portion of the data used to evaluate the neural network's performance. Calculating verification losses is analogous to calculating training losses. Each is computed by adding the validation errors for each unique validation set. Here, losses are also assessed independently for each period. This information is used to assess whether or not the model requires adjustment. Typically, a graph showing the structure of the learning curve for the loss of verification is employed to depict the losses.
Both metrics are used in the CTC decoding stage and are measured by comparing the recognized value and the value in the truth table:

$$p(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X)$$

where, $p(Y|X)$ the CTC conditional probability, $p_t(a_t|X)$ probability for a single alignment.
The following measure is accuracy. It is a measure of the degree of correctness of

certain data. It is used in several classification tasks during neural network training. Calculated by dividing the number of accurate predictions by the number of total predictions.

$$Accuracy = Number\,of\,correct\,predictions/Total\,number\,of\,predictions$$

In the case of binary classification, the accuracy may be expressed as true/false positive/negative values.

$$Accuracy = TP + TN/TP + FP + TN + FN$$

where, TP – true positives, FP – false positives, TN – true negatives, FN – false negatives.

3.3.10 Tools used

It was determined that the most advanced and widely used programming language, Python, would be utilized to complete the tasks and analyze the neural network. Specifically, the Keras library is used to construct neural networks and conduct further training and testing. Figure 4.6 shows the Sequential approach used to build the convolution neural network. [37]

```
from keras.models import Sequential
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from keras.callbacks import TensorBoard
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier


model_1 = Sequential([
    Conv2D(6, kernel_size=5, strides = 2, padding = 'same', dilation_rate = 1, activation='relu',
           input_shape=image_shape, kernel_initializer='he_normal', name='Conv2D-1'),
    Conv2D(12, kernel_size=5, strides = 2, padding = 'same', dilation_rate = 1, activation='relu',
    Conv2D(24, kernel_size=5, strides = 2, padding = 'same', dilation_rate = 1, activation='relu',
    Flatten(name='flatten'),
    Dense(128, activation='relu', name='Dense'),
    Dense(10, activation='softmax', name='Output')
], name = '3_Layer_CNN_4')
```

Figure 3.6: Implementing Keras library.

4. Results and Discussion

4.1 Performance of Layer Hyperparameters

The convolution network layers were configured and tested in accordance with Table 3.1 and 3.2 shown in Chapter 3. During training and testing, three attempts were used to determine the average results in terms of accuracy and losses. Table 1 shows the data for a three-layer network. Below are the accuracy and loss graphs for this table.

In both graphs, we can see 3-layer convolution neural networks in certain different

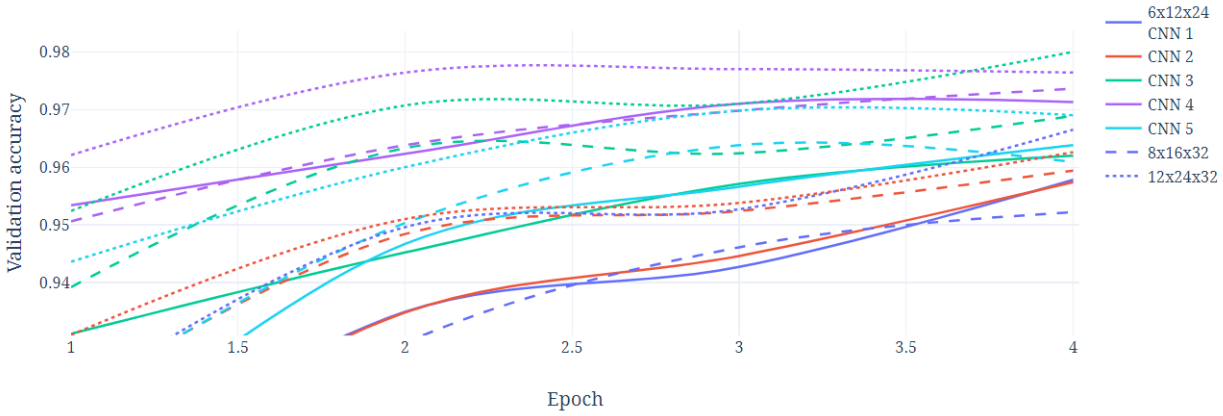

Figure 4.1: Accuracy of 3-layer CNNs.

colors. Solid, split, and dotted lines show the same networks only with different values in the feature map. Analyzing the graphs, we can notice a very strong influence of network hyperparameters on overall performance. In particular, network number 3 with a feature map value of 12x24x32 has the best indicators. The worst result compared to others in the number 1 network with a feature map value of 8x16x32. Network number 3, unlike network number 1, has a large value in the kernel size, which is increased by 2 in each layer. Steps and padding also have a larger value by 1 unit, which indicates an increase in the size of the feature

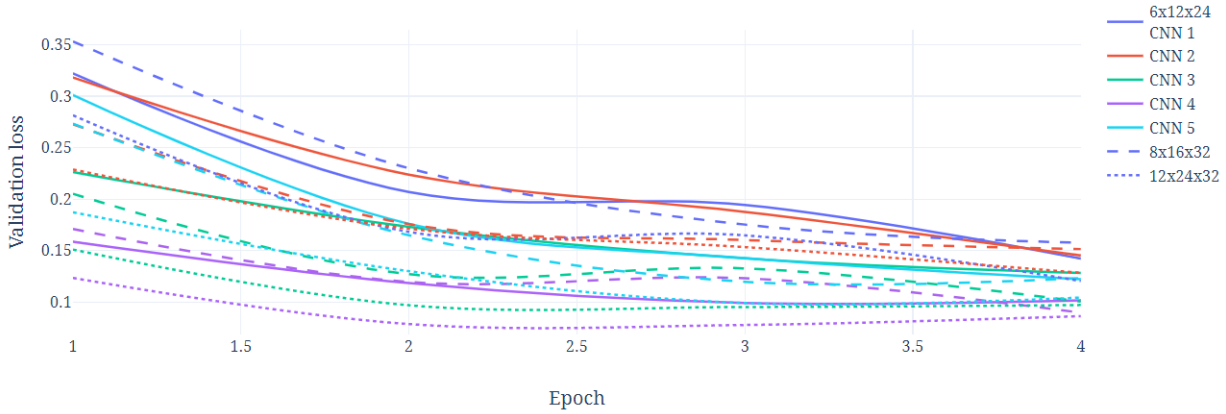

Figure 4.2: Loss of 3-layer CNNs.

map, but at the same time compensation by increasing the steps with a larger filter size. Feature map also plays an important role as it has a difference in both cases.

There are also, in both graphs, 4-layer convolution neural networks in certain

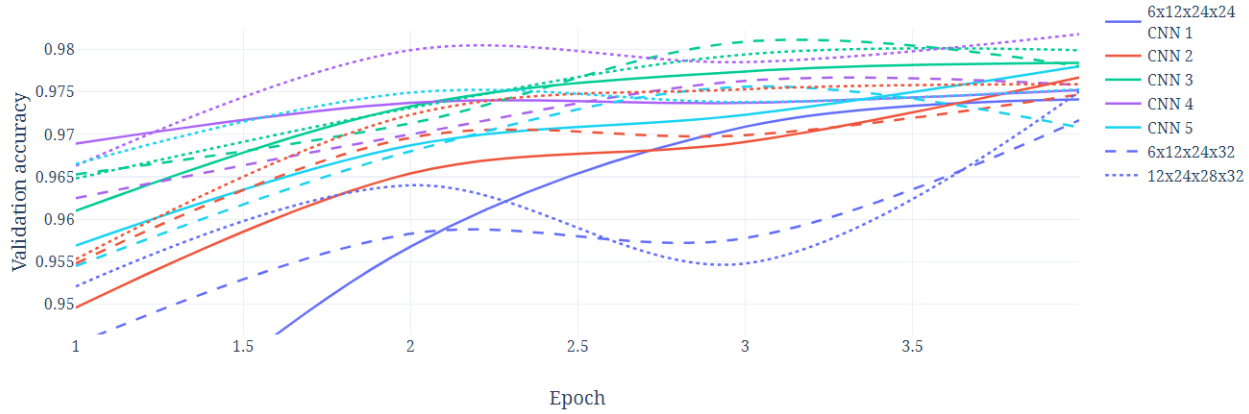

Figure 4.3: Accuracy of 4-layer CNNs.

different colors. Solid, split, and dotted lines show the same networks only with different values in the feature map. In this case, the network number 4 with a feature map value of 12x24x28x32 has the best indicators. The worst result compared to others in the network is number 5 with a feature map value of 6x12x24x32. In this case, network number 4, unlike network number 5, has an average value in kernel size and a stride. Padding has a smaller value by 1 unit in one layer, dilation in two layers is also applied in network number 4, which indicates a decrease in the size of the feature map in one layer and the use of dilation in two initial layers. Feature map here also has a difference in both cases.

Neural Networks	Layers	k/s	s	p	d	i/s	o/s	Feature map		
								6x12x24	8x16x32	12x24x32
CNN1	Layer 1 Layer 2 Layer 3	3 3 3	2 2 2	1 1 1	1 1 1	28 9 3	9 3 1	0.957	0.952	0.966
CNN2	Layer 1 Layer 2 Layer 3	5 3 3	2 3 3	1 1 1	1 1 1	28 12 4	12 4 1	0.957	0.959	0.962
CNN3	Layer 1 Layer 2 Layer 3	5 5 5	3 3 3	2 2 2	1 1 1	28 10 4	10 4 2	0.962	0.968	**0.980**
CNN4	Layer 1 Layer 2 Layer 3	5 5 5	2 2 2	2 2 2	1 1 1	28 14 7	14 7 4	**0.971**	**0.973**	0.976
CNN5	Layer 1 Layer 2 Layer 3	3 3 3	1 1 1	1 1 1	1 1 1	28 14 6	14 6 2	0.963	0.960	0.968

Table 4.1: Accuracy of 3-layer CNNs.

Neural Networks	Layers	k/s	s	p	d	i/s	o/s	Feature map		
								6x12x24 x24	6x12x24 x32	12x24x28 x32
CNN1	Layer 1 Layer 2 Layer 3 Layer 4	3 3 3 3	2 2 2 2	2 2 1 1	1 1 1 1	28 14 7 4	14 7 4 2	0.974	0.971	0.975
CNN2	Layer 1 Layer 2 Layer 3 Layer 4	5 3 3 3	2 2 2 2	2 2 1 1	1 1 1 1	28 14 7 4	14 7 4 2	0.976	0.974	0.975
CNN3	Layer 1 Layer 2 Layer 3 Layer 4	3 3 3 3	1 1 2 2	2 2 1 1	2 2 1 1	28 14 7 4	14 7 4 2	**0.978**	**0.977**	0.979
CNN4	Layer 1 Layer 2 Layer 3 Layer 4	5 5 3 3	1 1 2 2	2 2 1 1	2 2 1 1	28 14 7 4	14 7 4 2	0.975	0.975	**0.981**
CNN5	Layer 1 Layer 2 Layer 3 Layer 4	5 5 5 3	2 2 2 2	2 2 2 1	1 1 1 1	28 14 7 4	14 7 4 2	0.977	0.970	0.975

Table 4.2: Accuracy of 4-layer CNNs.

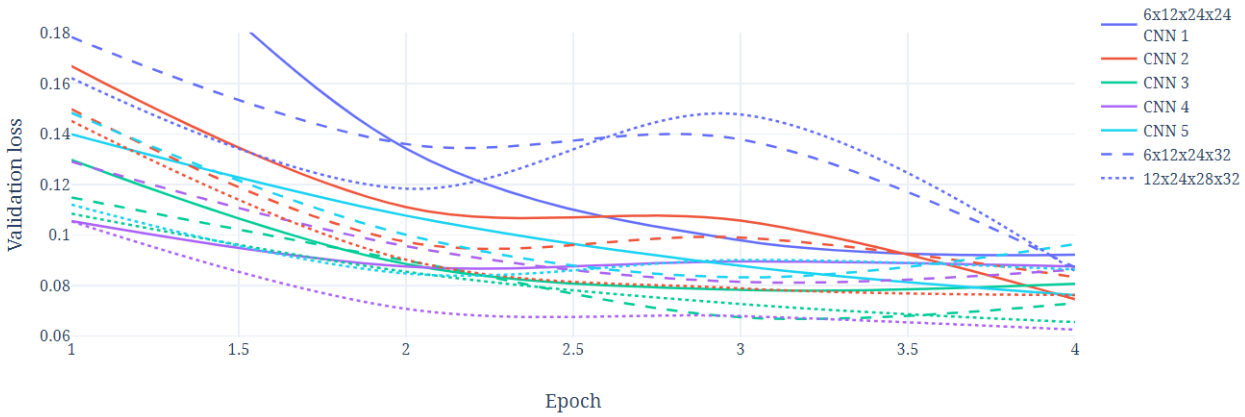

Figure 4.4: Loss of 4-layer CNNs.

For further experiments, the networks with the best indicators in different feature map values were selected. And more specifically, from the three-layer network number 4 in feature maps 6x12x24, 8x16x32 with an accuracy value of 97.1% and 97.3%, respectively, and network number 3 in feature map 12x24x32 with an accuracy value of 98.0%. In the four-layer network number 3 in feature maps 6x12x24x24, 6x12x24x32 with an accuracy value of 97.8% and 97.7%, respectively, and network number 4 in feature map 12x24x28x32 with an accuracy value of 98.1%.

4.2 Batch size and number of epochs

To conduct an experiment with these parameters, values for the size of the batch in the range of 5-200 were selected for each individual value of epochs from 10-30. Below are the results of averaged accuracy from three attempts in two graphs for 3- and 4-layer networks.

In both graphs, the networks are separated by color indicators and the values of the epochs are shown in solid, separated and dotted lines. For 3-layer networks, we can see in the graph that the best indicators for the size of the batch are in the range from 20 to 80 for all the values of the epochs. For four-layer networks, it is noticeable in the graph that the best indicators for the size of the batch are in the range from 20 to 60 for all the values of the epochs.

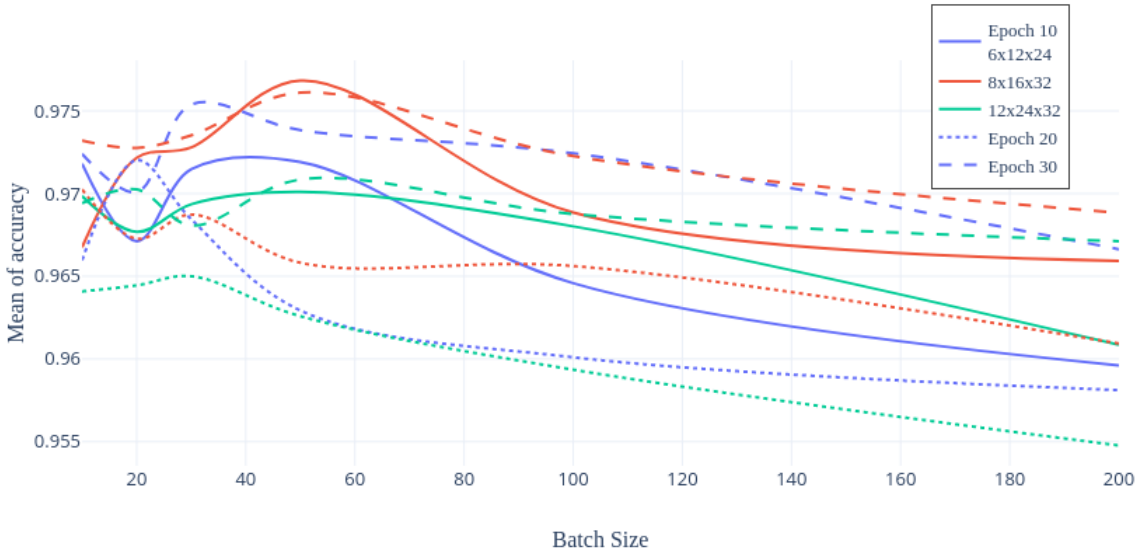

Figure 4.5: Batch size and epoch for 3-layer CNNs.

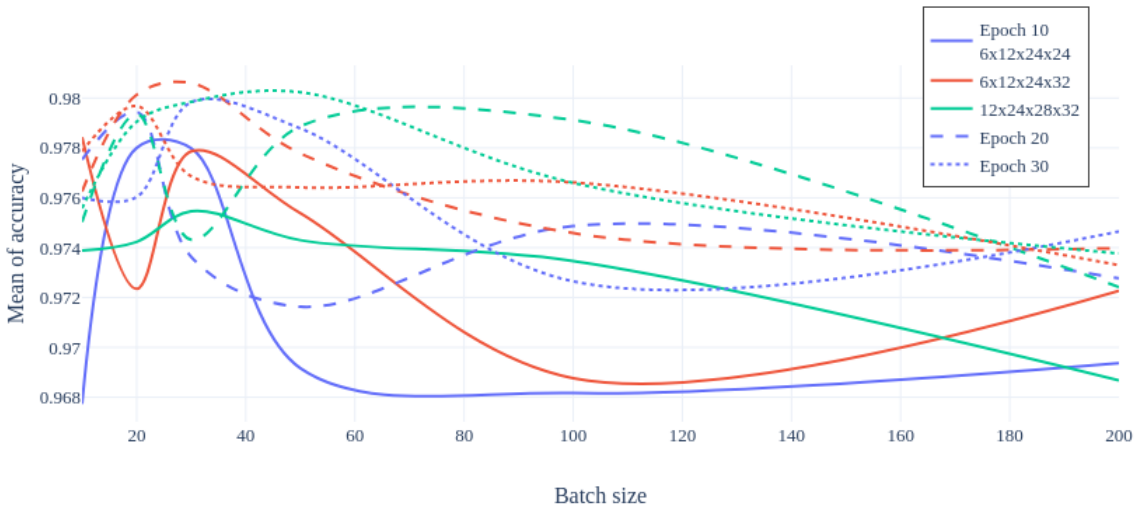

Figure 4.6: Batch size and epoch for 4-layer CNNs.

4.3 Optimization algorithm

For neural networks, the previously mentioned optimization algorithms were selected and the results shown in the graph were obtained. It shows both three-layer and four-layer networks separated by colored indicator lights. The best we can see in the optimizers SGD, RMSprop and Adam.

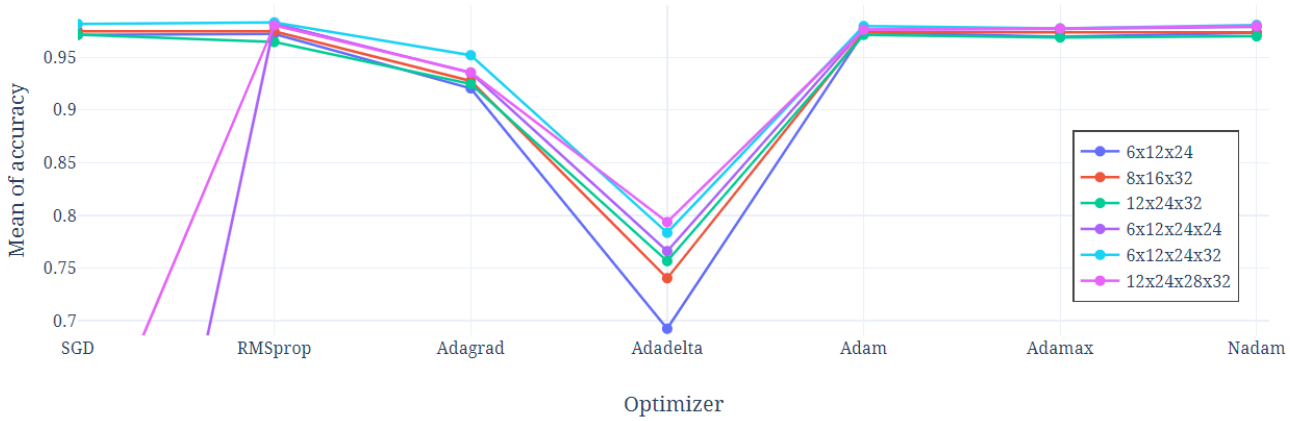

Figure 4.7: Results of performance of optimizers.

4.4 Learning rate

The values for the learning rate are selected in the range from 0.001 to 0.1, but since the network performance has dropped significantly in the value of 0.01, only this segment is shown. In the above graph, the lines are classified as well as in the previous one, and in it we can see that the best value is in the range of 0.002 - 0.004 for all networks.

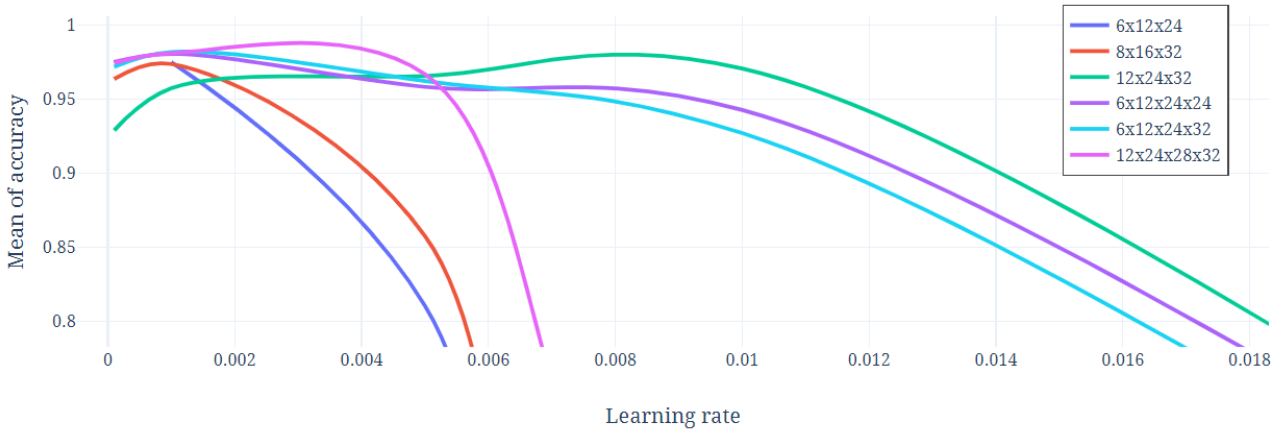

Figure 4.8: Results of learning rate performance.

4.5 Weight initialization

In this case, of the selected values, the worst indicator was zero, which did not fit into the accuracy range from 96-99%. The rest varies slightly in the same values. The best indicators are uniform, normal and he_normal.

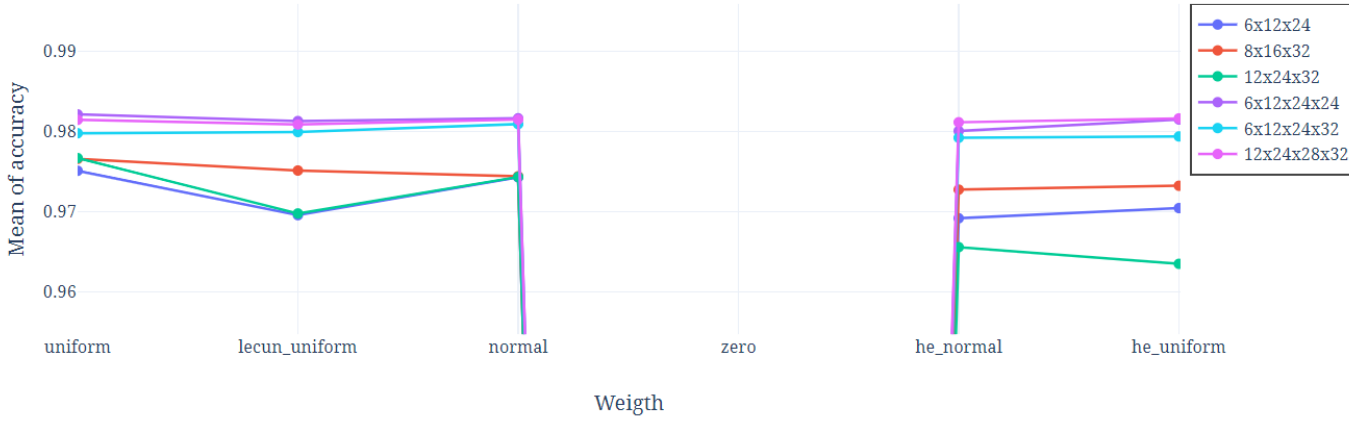

Figure 4.9: Results of weight initialization performance.

4.6 Activation function

Here, too, in the range of selected parameters, the best indicators were awarded the values of soft plus, relu, tanh and sigmoid. But for the softmax, relu and sigmoid values, many networks did not show a very good degree of accuracy and fell out of the range of 92-98% accuracy.

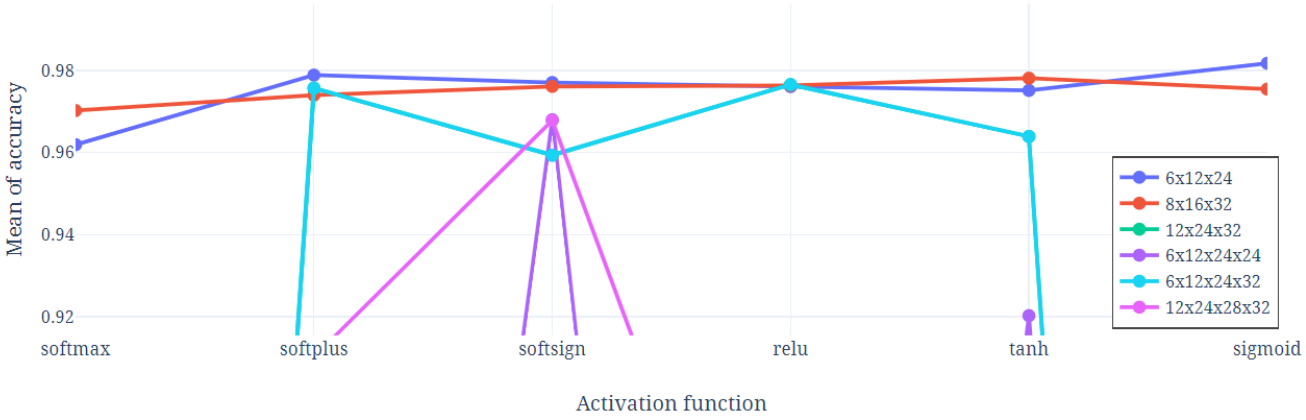

Figure 4.10: Results of activation function performance.

4.7 Dropout

The range of values was chosen from 0 to 1, according to the results of the graph, you can see the best value in the range from 0.2 to 0.4 for all networks. Values above 0.4 lead to large losses, which reduces the overall performance of each network.

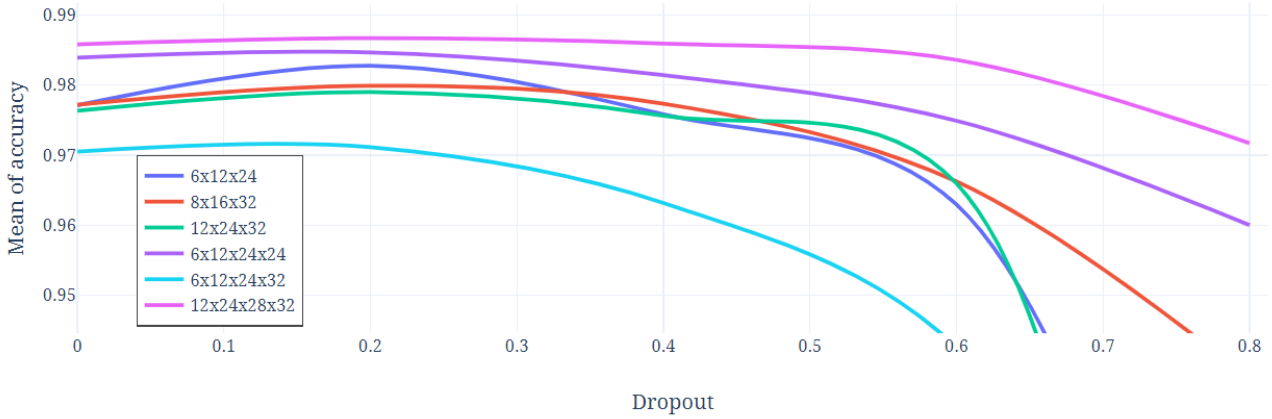

Figure 4.11: Results of dropout performance.

4.8 Discussion

As a result of all the tests, experiments were conducted to assess the influence of hyperparameters both inside the network layers and network parameters with the selection of certain values in a certain range. The result of all tests and improvement by selecting the best parameter for each network is shown in the table below.

Neural Net-works	Feature map	Batch	Epoch	Optimizer	Learning rate	Weight	Activation	Drop out	Before		After	
									Val loss	Val acc	Val loss	Val acc
CNN1	6x12x24	30	30	Adam	0.001	Uniform	Sigmoid	0.2	0.101	0.971	0.040	0.987
CNN2	8x16x32	50	20	RMSprop	0.001	Uniform	Tanh	0.2	0.089	0.973	0.047	0.985
CNN3	12x24x32	50	30	SGD	0.01	Uniform	ReLu	0.2	0.097	0.973	0.046	0.985
CNN4	6x12x24x24	30	30	RMSprop	0.001	Uniform	Softsign	0.2	0.080	0.980	**0.031**	**0.991**
CNN5	6x12x24x32	30	20	RMSprop	0.001	Normal	Softmax	0.2	0.073	0.977	0.035	0.989
CNN6	12x24x28x32	50	30	RMSprop	0.001	Uniform	Softsign	0.2	0.062	0.981	**0.028**	**0.992**

Table 4.3: Overall result.

We can notice that different variations of parameters were selected for each network, which led to different results. This proves that it is necessary to configure the network each time by choosing the parameters individually depending on the task at hand, without relying on similar networks. Since similar networks, as you can see, can show greater performance with different parameters. In our case, this is handwriting recognition of numbers using the MNIST database.

5. Conclusion and future work

By selecting through multiple random values, we were able to determine the effect of various hyperparameters on the convolutional neural network. As a result, a pure convolutional neural network without additional components was developed, which outperforms even densely coupled networks in terms of performance. The answers to the questions were also received.

• RQ1: What is the impact of different hyperparameters on CNN performance?
The influence of hyperparameters depends initially on the selected layers, on their internal hyperparameters, as well as on their number. For the selected task, all the selected hyperparameters had a very substantial impact. This can be observed in the final proper selection of parameters, which led to an increase in the accuracy of each model by roughly 3-4%. This is a very large figure with a range above 95%. Since it is in this circumstance that in many models it is very difficult to raise the value by at least 1%.

• RQ2: In which cases does the CNN more efficiently extract the necessary distinctive features from the image of handwritten characters?
The precise selection of the learning rate, activation function, and dropout can explain these instances. These hyperparameters have the biggest effect on the ability to extract the image's necessary distinguishing features. Their increased value results in enormous losses, and their decreased value results in substantial expenses.

• RQ3: Is it possible to achieve high results using only pure CNN?
As a result of experiments on the selection of the best hyperparameters, it was revealed that it is possible to achieve very good results without using deep neural networks. The highest result is visible in the network number 6 with an accuracy value of 99.2%. This is an indicator of competitiveness with many deep and complex networks.

As future work, it is intended to enhance the size of neural networks for deep learning by adding additional layers and studying their behavior in relation to an excess of layers or a deficiency of layers. In addition, undertake training and validation using many databases to determine the influence of parameters and their task-dependence. There is also the potential of conducting a more in-depth examination and investigation of existing networks based on the highly popular and in-demand convolution networks.

References

[1] Saleh Albelwi and Ausif Mahmood. "A Framework for Designing the Architectures of Deep Convolutional Neural Networks". en. In: *Entropy* 19.6 (May 2017), p. 242. ISSN: 1099-4300. DOI: 10.3390/e19060242. URL: https://www.mdpi.com/1099-4300/19/6/242 (visited on 05/18/2022).

[2] Y. Bengio. *Learning Deep Architectures for AI*. now Publishers Inc, 2009. ISBN: 9781601982957. DOI: 10.1561/9781601982957. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=8187120 (visited on 05/18/2022).

[3] Richard G. Brereton and Gavin R. Lloyd. "Support Vector Machines for classification and regression". en. In: *The Analyst* 135.2 (2010), pp. 230–267. ISSN: 0003-2654, 1364-5528. DOI: 10.1039/B918972F. URL: http://xlink.rsc.org/?DOI=B918972F (visited on 05/18/2022).

[4] Savitri Laxmanrao Chandure and Vandana Inamdar. "Performance analysis of handwritten Devnagari and MODI Character Recognition system". In: *2016 International Conference on Computing, Analytics and Security Trends (CAST)*. Pune, India: IEEE, Dec. 2016, pp. 513–516. ISBN: 9781509013388 DOI: 10.1109/CAST.2016.7915022. URL: http://ieeexplore.ieee.org/document/7915022/ (visited on 05/18/2022).

[5] Robert H. Chen and Chelsea Chen. *Artificial Intelligence: An Introduction for the Inquisitive Reader*. en. 1st ed. Boca Raton: Chapman and Hall/CRC, Apr. 2022. ISBN: 9781003214892. DOI: 10.1201/9781003214892. URL: https://www.taylorfrancis.com/books/9781003214892 (visited on 05/18/2022).

[6] Robert H. Chen and Chelsea Chen. *Artificial Intelligence: An Introduction for the Inquisitive Reader*. en. 1st ed. Boca Raton: Chapman and Hall/CRC,

Apr. 2022. ISBN: 9781003214892. DOI: 10.1201/9781003214892. URL: https://www.taylorfrancis.com/books/9781003214892 (visited on 05/18/2022).

[7] Mohamed Cheriet, Reza Farrahi Moghaddam, and Rachid Hedjam. "A learning framework for the optimization and automation of document binarization methods". en. In: *Computer Vision and Image Understanding* 117.3 (Mar. 2013), pp. 269–280. ISSN: 10773142. DOI: 10.1016/j.cviu.2012.11.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1077314212001877 (visited on 05/18/2022).

[8] Darmatasia and Mohamad Ivan Fanany. "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)". In: *2017 5th International Conference on Information and Communication Technology (ICoIC7)*. Melaka, Malaysia: IEEE, May 2017, pp. 1–6. ISBN: 9781509049127. DOI: 10.1109/ICoICT.2017.8074699. URL: http://ieeexplore.ieee.org/document/8074699/ (visited on 05/18/2022).

[9] Seyed Amirhossein Farjadi and Ali Assar. "Convolutional Neural Network". en. In: (2022). DOI: 10.13140/RG.2.2.29617.45921. URL: https://rgdoi.net/10.13140/RG.2.2.29617.45921 (visited on 05/18/2022).

[10] Reza Farrahi Moghaddam and Mohamed Cheriet. "AdOtsu: An adaptive and parameterless generalization of Otsu's method for document image binarization". en. In: *Pattern Recognition* 45.6 (June 2012), pp. 2419–2431. ISSN: 00313203. DOI: 10.1016/j.patcog.2011.12.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320311005140 (visited on 05/18/2022).

[11] Ashish Ghosh, Nikhil R Pal, and Sankar K Pal. "Neural Computing: An Introduction and Some Applications". en. In: *IETE Journal of Education* 35.1-2 (Jan. 1994), pp. 105–125. ISSN: 0974-7338. DOI: 10.1080/09747338.1994.11436450. URL: http://www.tandfonline.com/doi/full/10.1080/09747338.1994.11436450 (visited on 05/18/2022).

[12] Hesamoddin Hosseini and Ehsan Fazl. "Histograms of Oriented Gradients for Human Detection". id. In: (2021). DOI: 10.13140/RG.2.2.23788.

85122. URL: https://rgdoi.net/10.13140/RG.2.2.23788.85122 (visited on 05/18/2022).

[13] Farah Jamal et al. "Reviewing conceptualisations of community: reflections on a meta-narrative approach". In: *Qualitative Research* 15.3 (June 2015), pp. 314–333. ISSN: 1468-7941, 1741-3109. DOI: 10.1177/1468794113509262. URL: http://journals.sagepub.com/doi/10.1177/1468794113509262 (visited on 05/18/2022).

[14] Conrad D. James et al. "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications". en. In: *Biologically Inspired Cognitive Architectures* 19 (Jan. 2017), pp. 49–64. ISSN: 2212683X. DOI: 10.1016/j.bica.2016.11.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S2212683X16300561 (visited on 05/18/2022).

[15] R. Jayadevan et al. "Automatic processing of Handwritten Bank Cheque Images: A survey". In: *International Journal on Document Analysis and Recognition (IJDAR)* 15.4 (2011), pp. 267–296. DOI: 10.1007/s10032-011-0170-8.

[16] Teja Kattenborn et al. "Review on Convolutional Neural Networks (CNN) in vegetation remote sensing". en. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 173 (Mar. 2021), pp. 24–49. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2020.12.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0924271620303488 (visited on 05/18/2022).

[17] Taranjit Kaur, Barjinder Singh Saini, and Savita Gupta. "Optimized Multi Threshold Brain Tumor Image Segmentation Using Two Dimensional Minimum Cross Entropy Based on Co-occurrence Matrix". In: *Medical Imaging in Clinical Applications*. Ed. by Nilanjan Dey, Vikrant Bhateja, and Aboul Ella Hassanien. Vol. 651. Cham: Springer International Publishing, 2016, pp. 461–486. ISBN: 9783319337913 9783319337937. DOI: 10.1007/978-3-319-33793-7_20. URL: http://link.springer.com/10.1007/978-3-319-33793-7_20 (visited on 05/18/2022).

[18] Haesik Kim. "Deep Learning". en. In: *Artificial Intelligence for 6G*. Cham: Springer International Publishing, 2022, pp. 247–303. ISBN: 9783030950408 9783030950415. DOI: 10.1007/978-3-030-95041-5_6. URL: https:

//link.springer.com/10.1007/978-3-030-95041-5_6 (visited on 05/18/2022).

[19] Sayaka Kindaichi et al. "Potential for using water reservoirs as heat sources in heat pump systems". en. In: *Applied Thermal Engineering* 76 (Feb. 2015), pp. 47–53. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2014.10.091. URL: https://linkinghub.elsevier.com/retrieve/pii/S1359431114009739 (visited on 05/18/2022).

[20] Woo-Young Lee, Seung-Min Park, and Kwee-Bo Sim. "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm". en. In: *Optik* 172 (Nov. 2018), pp. 359–367. ISSN: 00304026. DOI: 10.1016/j.ijleo.2018.07.044. URL: https://linkinghub.elsevier.com/retrieve/pii/S0030402618310167 (visited on 05/18/2022).

[21] Zhiyuan Li et al. "Deep Matching Network for Handwritten Chinese Character Recognition". en. In: *Pattern Recognition* 107 (Nov. 2020), p. 107471. ISSN: 00313203. DOI: 10.1016/j.patcog.2020.107471. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320320302740 (visited on 05/18/2022).

[22] Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia, and L. Alfonso Ureña-López. "Opinion Classification Techniques Applied to a Spanish Corpus". In: *Natural Language Processing and Information Systems*. Ed. by Rafael Muñoz, Andrés Montoyo, and Elisabeth Métais. Vol. 6716. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 169–176. ISBN: 9783642223266 9783642223273. DOI: 10.1007/978-3-642-22327-3_17. URL: http://link.springer.com/10.1007/978-3-642-22327-3_17 (visited on 05/18/2022).

[23] Siddhartha Sankar Nath et al. "A survey of image classification methods and techniques". In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. Kanyakumari District, India: IEEE, July 2014, pp. 554–557. ISBN: 9781479941902 9781479941919 9781479941933 9781479941926. DOI: 10.1109/ICCICCT.2014.6993023. URL: http://ieeexplore.ieee.org/document/6993023/ (visited on 05/18/2022).

[24] Xiao-Xiao Niu and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits". en. In: *Pattern Recognition* 45.4 (Apr. 2012), pp. 1318–1325. ISSN: 00313203. DOI: 10.1016/j.patcog.2011.09.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320311004006 (visited on 05/18/2022).

[25] J. J. de Oliveira et al. "Handwritten Word Recognition Using Multi-view Analysis". en. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by David Hutchison et al. Vol. 5856. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 371–378. ISBN: 9783642102677 9783642102684. DOI: 10.1007/978-3-642-10268-4_44. URL: http://link.springer.com/10.1007/978-3-642-10268-4_44 (visited on 05/18/2022).

[26] Pooja D Pandit. "Case study on Neural Networks for Handwritten English Alphabet Recognition". en. In: (2021). DOI: 10.13140/RG.2.2.35402.49603. URL: http://rgdoi.net/10.13140/RG.2.2.35402.49603 (visited on 05/18/2022).

[27] Arnaldo Pérez Castaño. "Handwritten Digit Recognition". en. In: *Practical Artificial Intelligence*. Berkeley, CA: Apress, 2018, pp. 461–478. ISBN: 9781484233566 9781484233573. DOI: 10.1007/978-1-4842-3357-3_12. URL: http://link.springer.com/10.1007/978-1-4842-3357-3_12 (visited on 05/18/2022).

[28] Made Edwin Wira Putra and Iping Supriana Suwardi. "Structural Off-line Handwriting Character Recognition Using Approximate Subgraph Matching and Levenshtein Distance". en. In: *Procedia Computer Science* 59 (2015), pp. 340–349. ISSN: 18770509. DOI: 10.1016/j.procs.2015.07.529. URL: https://linkinghub.elsevier.com/retrieve/pii/S187705091502058X (visited on 05/18/2022).

[29] V. Rajinikanth, J. P. Aashiha, and A. Atchaya. "Gray-Level Histogram based Multilevel Threshold Selection with Bat Algorithm". In: *International Journal of Computer Applications* 93.16 (May 2014), pp. 1–8. ISSN: 09758887. DOI: 10.5120/16296-6035. URL: http://research.ijcaonline.org/volume93/number16/pxc3896035.pdf (visited on 05/18/2022).

[30] Amjad Rehman and Tanzila Saba. "Off-line cursive script recognition: current advances, comparisons and remaining problems". en. In: *Artificial Intelligence Review* 37.4 (Apr. 2012), pp. 261–288. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-011-9229-7. URL: http://link.springer.com/10.1007/s10462-011-9229-7 (visited on 05/18/2022).

[31] Jeisetti Saiteja, Podila Srivally Rao, and P. Mani Bharadwaj. "HANDWRITTEN DIGIT RECOGNITION". In: *International Journal of Computer Science and Mobile Computing* 11.1 (Jan. 2022), pp. 45–54. ISSN: 2320088X. DOI: 10.47760/ijcsmc.2022.v11i01.007. URL: https://ijcsmc.com/docs/papers/January2022/V11I1202214.pdf (visited on 05/18/2022).

[32] Boran Şekeroğlu and Adnan Khashman. "Performance Evaluation of Binarization Methods for Document Images". en. In: *Proceedings of the International Conference on Advances in Image Processing*. Bangkok Thailand: ACM, Aug. 2017, pp. 96–102. ISBN: 9781450352956. DOI: 10.1145/3133264.3133272. URL: https://dl.acm.org/doi/10.1145/3133264.3133272 (visited on 05/18/2022).

[33] Joel Serey et al. "Artificial Intelligence Methodologies for Data Management". en. In: *Symmetry* 13.11 (Oct. 2021), p. 2040. ISSN: 2073-8994. DOI: 10.3390/sym13112040. URL: https://www.mdpi.com/2073-8994/13/11/2040 (visited on 05/18/2022).

[34] Taniya Seth and Pranab Muhuri. *Optimizing Hyperparameters in CNNs using Bilevel Programming in Time Series Data*. Jan. 2021.

[35] Manoj Kumar Sharma. "A Survey of Thresholding Techniques over Images". en. In: *INROADS- An International Journal of Jaipur National University* 3.2 (2014), p. 461. ISSN: 2277-4904, 2277-4912. DOI: 10.5958/2277-4912.2014.00010.1. URL: http://www.indianjournals.com/ijor.aspx?target=ijor:inroads&volume=3&issue=2&article=010 (visited on 05/18/2022).

[36] Gustavo Siebra Lopes et al. "Recognition of handwritten digits using the signature features and Optimum-Path Forest Classifier". In: *IEEE Latin America Transactions* 14.5 (May 2016), pp. 2455–2460. ISSN: 1548-0992.

DOI: 10.1109/TLA.2016.7530445. URL: http://ieeexplore.ieee.org/document/7530445/ (visited on 05/18/2022).

[37] Xiong Fu-song. "Survey over image thresholding techniques based on entropy". In: *2014 International Conference on Information Science, Electronics and Electrical Engineering*. Sapporo, Japan: IEEE, Apr. 2014, pp. 1330–1334. ISBN: 9781479931972 9781479931965 9781479931958. DOI: 10.1109/InfoSEEE.2014.6947888. URL: http://ieeexplore.ieee.org/document/6947888/ (visited on 05/18/2022).

[38] Yaregal Tadesse et al. "Amharic Handwritten Document Recognition using Deep Learning". en. In: *Computer Vision and Robotics*. Ed. by Jagdish Chand Bansal, Andries Engelbrecht, and Praveen Kumar Shukla. Singapore: Springer Singapore, 2022, pp. 251–264. ISBN: 9789811682247 9789811682254. DOI: 10.1007/978-981-16-8225-4_20. URL: https://link.springer.com/10.1007/978-981-16-8225-4_20 (visited on 05/18/2022).

[39] Khoa Bao Chau Thai, Huu Tien Nguyen, and Huu Dung Tran. "Determination the limit of postprandial blood glucose increase and the metabolism of acetylated wheat starch on healthy mice". vi. In: *Journal of Medicine and Pharmacy* (Aug. 2019), pp. 15–22. ISSN: 1859-3836. DOI: 10.34071/jmp.2019.5.2. URL: http://jmp.huemed-univ.edu.vn/View.aspx?idbb=Xac_dinh_kha_nang_han_che_tang_duong_huyet_sau_an_va_chuyen_hoa_cua_tinh_bot_mi_acetat_tren_chuot_nhat_khoe_manh-20407 (visited on 05/18/2022).

[40] Kübra Uyar, Sakir Tasdemir, and İlker Ali Özkan. "The Analysis and Optimization of CNN Hyperparameters with Fuzzy Tree Model". In: 2021.

[41] YanWei Wang, XiaoQing Ding, and ChangSong Liu. "Post processing for offline Chinese handwritten character string recognition". In: ed. by Christian Viard-Gaudin and Richard Zanibbi. Burlingame, California, United States, Jan. 2012, p. 829709. DOI: 10.1117/12.905985. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.905985 (visited on 05/18/2022).

[42] Wei-Chang Yeh et al. "Convolution Neural Network Hyperparameter Optimization Using Simplified Swarm Optimization". In: *arXiv:2103.03995 [cs]*

(Aug. 2021). arXiv: 2103.03995. URL: http://arxiv.org/abs/2103.03995.

[43] Naigong Yu, Panna Jiao, and Yuling Zheng. "Handwritten digits recognition base on improved LeNet5". In: *The 27th Chinese Control and Decision Conference (2015 CCDC)*. Qingdao, China: IEEE, May 2015, pp. 4871–4875. ISBN: 9781479970179. DOI: 10.1109/CCDC.2015.7162796. URL: http://ieeexplore.ieee.org/document/7162796/ (visited on 05/18/2022).

[44] M. I. Zabezhailo and Y. Y. Trunin. "On the Reliability of Medical Diagnosis Based on Empirical Data". en. In: *Scientific and Technical Information Processing* 48.5 (Dec. 2021), pp. 415–422. ISSN: 0147-6882, 1934-8118. DOI: 10.3103/S0147688221050142. URL: https://link.springer.com/10.3103/S0147688221050142 (visited on 05/18/2022).

[45] Hao Zeng. "An Off-line Handwriting Recognition Employing Tensorflow". In: *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. Fuzhou, China: IEEE, June 2020, pp. 158–161. ISBN: 9781728164991. DOI: 10.1109/ICBAIE49996.2020.00040. URL: https://ieeexplore.ieee.org/document/9196493/ (visited on 05/18/2022).

[46] Hongjian Zhan et al. "Handwritten Digit String Recognition for Indian Scripts". en. In: *Pattern Recognition*. Ed. by Shivakumara Palaiahnakote et al. Vol. 12047. Cham: Springer International Publishing, 2020, pp. 262–273. ISBN: 9783030412982 9783030412999. DOI: 10.1007/978-3-030-41299-9_21. URL: http://link.springer.com/10.1007/978-3-030-41299-9_21 (visited on 05/18/2022).

[47] Sheng Zheng et al. "Sunspot drawings handwritten character recognition method based on deep learning". en. In: *New Astronomy* 45 (May 2016), pp. 54–59. ISSN: 13841076. DOI: 10.1016/j.newast.2015.11.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1384107615001402 (visited on 05/18/2022).

[48] Zahra Ziran et al. "Correction to: Accurate Graphic Symbol Detection in Ancient Document Digital Reproductions". In: *Document Analysis and Recognition – ICDAR 2021 Workshops*. Ed. by Elisa H. Barney Smith and Umapada Pal. Vol. 12916. Cham: Springer International Publishing, 2022,

pp. C1–C1. ISBN: 9783030861971 9783030861988. DOI: 10.1007/978-3-030-86198-8_34. URL: https://link.springer.com/10.1007/978-3-030-86198-8_34 (visited on 05/18/2022).