sensors

MDPI

Article

Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)

Savita Ahlawat [1], Amit Choudhary [2], Anand Nayyar [3], Saurabh Singh [4] and Byungun Yoon [4,*]

[1] Department of Computer Science and Engineering, Maharaja Surajmal Institute of Technology, New Delhi 110058, India; savita.ahlawat@gmail.com
[2] Department of Computer Science, Maharaja Surajmal Institute, New Delhi 110058, India; amit.choudhary69@gmail.com
[3] Graduate School, Duy Tan University, Da Nang 550000, Vietnam; anandnayyar@duytan.edu.vn
[4] Department of Industrial & Systems Engineering, Dongguk University, Seoul 04620, Korea; saurabh89@dongguk.edu
* Correspondence: postman3@dongguk.edu

Received: 25 May 2020; Accepted: 9 June 2020; Published: 12 June 2020

Abstract: Traditional systems of handwriting recognition have relied on handcrafted features and a large amount of prior knowledge. Training an Optical character recognition (OCR) system based on these prerequisites is a challenging task. Research in the handwriting recognition field is focused around deep learning techniques and has achieved breakthrough performance in the last few years. Still, the rapid growth in the amount of handwritten data and the availability of massive processing power demands improvement in recognition accuracy and deserves further investigation. Convolutional neural networks (CNNs) are very effective in perceiving the structure of handwritten characters/words in ways that help in automatic extraction of distinct features and make CNN the most suitable approach for solving handwriting recognition problems. Our aim in the proposed work is to explore the various design options like number of layers, stride size, receptive field, kernel size, padding and dilution for CNN-based handwritten digit recognition. In addition, we aim to evaluate various SGD optimization algorithms in improving the performance of handwritten digit recognition. A network's recognition accuracy increases by incorporating ensemble architecture. Here, our objective is to achieve comparable accuracy by using a pure CNN architecture without ensemble architecture, as ensemble architectures introduce increased computational cost and high testing complexity. Thus, a CNN architecture is proposed in order to achieve accuracy even better than that of ensemble architectures, along with reduced operational complexity and cost. Moreover, we also present an appropriate combination of learning parameters in designing a CNN that leads us to reach a new absolute record in classifying MNIST handwritten digits. We carried out extensive experiments and achieved a recognition accuracy of 99.87% for a MNIST dataset.

Keywords: convolutional neural networks; handwritten digit recognition; pre-processing; OCR

1. Introduction

In the current age of digitization, handwriting recognition plays an important role in information processing. A lot of information is available on paper, and processing of digital files is cheaper than processing traditional paper files. The aim of a handwriting recognition system is to convert handwritten characters into machine readable formats. The main applications are vehicle license-plate recognition, postal letter-sorting services, Cheque truncation system (CTS) scanning and historical document preservation in archaeology departments, old documents automation in libraries and banks, etc. All these areas deal with large databases and hence demand high recognition accuracy, lesser

computational complexity and consistent performance of the recognition system. It has been suggested that deep neural architectures are more advantageous than shallow neural architectures [1–6]. The key differences are described in Table 1. The deep learning field is ever evolving, and some of its variants are autoencoders, CNNs, recurrent neural networks (RNNs), recursive neural networks, deep belief networks and deep Boltzmann machines. Here, we introduce a convolutional neural network, which is a specific type of deep neural network having wide applications in image classification, object recognition, recommendation systems, signal processing, natural language processing, computer vision, and face recognition. The ability to automatically detect the important features of an object (here an object can be an image, a handwritten character, a face, etc.) without any human supervision or intervention makes them (CNNs) more efficient than their predecessors (Multi layer perceptron (MLP), etc.). The high capability of hierarchical feature learning results in a highly efficient CNN.

Table 1. Shallow neural network vs deep neural network.

Factors	Shallow Neural Network (SNN)	Deep Neural Network (DNN)
Number of hidden layers	- single hidden layer (need to be fully connected).	- multiple hidden layers (not necessarily fully connected).
Feature Engineering	- requires a separate feature extraction process. - some of the famous features used in the literature include local binary patterns (LBPs), histogram of oriented gradients (HOGs), speeded up robust features (SURFs), and scale-invariant feature transform (SIFT).	- supersedes the handcrafted features and works directly on the whole image. - useful in computing complex pattern recognition problems. - can capture complexities inherent in the data.
Requirements	- emphasizes the quality of features and their extraction process. - networks are more dependent on the expert skills of researchers.	- able to automatically detect the important features of an object (here an object can be an image, a handwritten character, a face, etc.) without any human supervision or intervention.
Dependency on data volume	- requires small amount of data.	- requires large amount of data.

A convolutional neural network (CNN) is basically a variation of a multi-layer perceptron (MLP) network and was used for the first time in 1980 [7]. The computing in CNN is inspired by the human brain. Humans perceive or identify objects visually. We (humans) train our children to recognize objects by showing him/her hundreds of pictures of that object. This helps a child identify or make a prediction about objects he/she has never seen before. A CNN works in the same fashion and is popular for analyzing visual imagery. Some of the well-known CNN architectures are GoogLeNet (22 layers), AlexNet (8 layers), VGG (16–19 Ali), and ResNet (152 layers). A CNN integrates the feature extraction and classification steps and requires minimal pre-processing and feature extraction efforts. A CNN can extract affluent and interrelated features automatically from images. Moreover, a CNN can provide considerable recognition accuracy even if there is only a little training data available.

Design particulars and previous knowledge of features are no longer required to be collected. Exploitation of topological information available in the input is the key benefit of using a CNN model towards delivering excellent recognition results. The recognition results of a CNN model are also independent of the rotation and translation of input images. Contrary to this, thorough topological knowledge of inputs is not exploited in MLP models. Furthermore, for a complex problem, MLP is not found to be appropriate, and they do not scale well for higher resolution images because of the full interconnection between nodes, also called the famous phenomenon of the "curse of dimensionality".

In the past few years, the CNN model has been extensively employed for handwritten digit recognition from the MNIST benchmark database. Some researchers have reported accuracy as good as 98% or 99% for handwritten digit recognition [8]. An ensemble model has been designed using a combination of multiple CNN models. The recognition experiment was carried out for MNIST

digits, and an accuracy of 99.73% was reported [9]. Later, this "7-net committee" was extended to the "35-net committee" experiment, and the improved recognition accuracy was reported as 99.77% for the same MNIST dataset [10]. An extraordinary recognition accuracy of 99.81% was reported by Niu and Suen by integrating the SVM (support vector machine) capability of minimizing the structural risk and the capability of a CNN model for extracting the deep features for the MNIST digit recognition experiment [11]. The bend directional feature maps were investigated using CNN for in-air handwritten Chinese character recognition [12]. Recently, the work of Alvear-Sandoval et al. achieved a 0.19% error rate for MNIST by building diverse ensembles of deep neural networks (DNN) [13]. However, on careful investigation, it has been observed that the high recognition accuracy of MNIST dataset images is achieved through ensemble methods only. Ensemble methods help in improving the classification accuracy but at the cost of high testing complexity and increased computational cost for real-world application [14].

The purpose of the proposed work is to achieve comparable accuracy using a pure CNN architecture through extensive investigation of the learning parameters in CNN architecture for MNIST digit recognition. Another purpose is to investigate the role of various hyper-parameters and to perform fine-tuning of hyper-parameters which are essential in improving the performance of CNN architecture.

Therefore, the major contribution of this work is in two respects. First, a comprehensive evaluation of various parameters, such as numbers of layers, stride size, kernel size, padding and dilution, of CNN architecture in handwritten digit recognition is done to improve the performance. Second, optimization of the learning parameters achieved excellent recognition performance on the MNIST dataset. The MNIST database has been used in this work because of the availability of its published results with different classifiers. The database is also popular and mostly used as a benchmark database in comparative studies of various handwritten digit recognition experiments for various regional and international languages.

The novelty of the proposed work lies in the thorough investigation of all the parameters of CNN architecture to deliver the best recognition accuracy among peer researchers for MNIST digit recognition. The recognition accuracy delivered in this work employing a fine-tuned pure CNN model is superior to the recognition accuracies reported by peer researchers using an ensemble architecture. The use of ensemble architecture by peer researchers involves increased computational cost and high testing complexity. Hence, the proposed pure CNN model outperforms the ensemble architecture offered by peer researchers both in terms of recognition accuracy as well as computational complexity.

The rest of the paper is organized as follows: Section 2 describes the related work in the field of handwriting recognition; Sections 3 and 4 describe CNN architecture and the experimental setup, respectively; Section 5 discusses the findings and presents a comparative analysis; and Section 6 presents the conclusion and suggestions for future directions.

2. Related Work

Handwriting recognition has already achieved impressive results using shallow networks [15–24]. Many papers have been published with research detailing new techniques for the classification of handwritten numerals, characters and words. The deep belief networks (DBN) with three layers along with a greedy algorithm were investigated for the MNIST dataset and reported an accuracy of 98.75% [25]. Pham et al. applied a regularization method of dropout to improve the performance of recurrent neural networks (RNNs) in recognizing unconstrained handwriting [26]. The author reported improvement in RNN performance with significant reduction in the character error rate (CER) and word error rate (WER).

The convolutional neural network brings a revolution in the handwriting recognition field and delivered the state-of-the-art performance in this domain [27–32]. In 2003, Simard et al. introduced a general convolutional neural network architecture for visual document analysis and weeded out the complex method of neural network training [33]. Wang et al. proposed a novel approach for

end-to-end text recognition using multi-layer CNNs and achieved excellent performance on benchmark databases, namely, ICDAR 2003 and Street View Text [34]. Recently, Shi et al. integrated the advantages of both the deep CNN (DCNN) and recurrent neural network (RNN) and named it conventional recurrent neural network (CRNN). They applied CRNN for scene text recognition and found it to be superior to traditional methods of recognition [35]. Badrinarayanan et al. proposed a deep convolution network architecture for semantic segmentation. The segmentation architecture is known as SegNet and consists of an encoder network, a decoder network and a pixel-wise classification layer. The proposed method used max-pooling indices of a feature map while decoding and observed good performance. The method is also analyzed and compared with existing techniques for road scene and indoor understanding [36–38]. CNN has shown remarkable abilities in offline handwritten character recognition of Arabic language [39]; handwritten Tamil character recognition [40]; Telugu character recognition [41], handwritten Urdu text recognition [42,43], handwritten character recognition in Indic scripts [44] and Chinese handwritten text recognition [45–47].

Recently, Gupta et al. in [48] proposed a novel multi-objective optimization framework for identifying the most informative local regions from a character image. The work was also evaluated on isolated handwritten English numerals, namely, MNIST images, along with three other popular Indic scripts, namely, handwritten Bangala numerals and handwritten Devanagari characters. The authors used features extracted from a convolutional neural network in their model and achieved 95.96% recognition accuracy. The work of Nguyen et al. in [49] used a multi-scale CNN for extracting spatial classification features for handwritten mathematical expression (HME). The local features and spatial information of HME images were used for clustering HME images. The work observed high performance for the CROHME dataset. They (authors) also concluded that classification can be improved by training the CNN with a combination of global max pooling and global attentive pooling. Ziran et al. [50] developed a faster R-CNN-based framework for text/word location and recognition in historical books. The authors evaluated these deep learning methods on Gutenberg's Bible pages. The handwritten character recognition problem is intelligently addressed in the work of Ptucha et al. [51] by the introduction of an intelligent character recognition (ICR) system using a conventional neural network. The work was evaluated on French-based RIMES lexicon datasets and English-based IAM datasets, showing substantial improvement.

The performance of CNNs depends mainly on the choice of hyper-parameters [52], which are usually decided on a trial-and-error basis. Some of the hyper-parameters are, namely, activation function, number of epochs, kernel size, learning rate, hidden units, hidden layers, etc. These parameters are very important as they control the way an algorithm learns from data [53]. Hyper-parameters differ from model parameters and must be decided before the training begins.

ResNet-52 [54], GoogleNet [55], VGG-16 [56] and AlexNet [57] are some popular CNN models that have a total of 150, 78, 57 and 27 hyper-parameters, respectively. A bad choice for hyper-parameters can incur a high computation cost and lead to poor CNN performance. The researcher's expertise plays an important role in deciding on the configuration of hyper-parameters and requires an intelligent strategic plan. This creates several questions about CNN design for handwriting recognition tasks. How is CNN better in extracting distinct features from handwritten characters? What effect do different hyper-parameters have on CNN performance? What is the role of design parameters in improving CNN performance? In order to guide future research in the handwriting recognition field, it is important to address these questions.

3. Convolutional Neural Network Architecture

A basic convolutional neural network comprises three components, namely, the convolutional layer, the pooling layer and the output layer. The pooling layer is optional sometimes. The typical convolutional neural network architecture with three convolutional layers is well adapted for the classification of handwritten images as shown in Figure 1. It consists of the input layer, multiple hidden layers (repetitions of convolutional, normalization, pooling) and a fully connected and an output layer.

Neurons in one layer connect with some of the neurons present in the next layer, making the scaling easier for the higher resolution images. The operation of pooling or sub-sampling can be used to reduce the dimensions of the input. In a CNN model, the input image is considered as a collection of small sub-regions called the "receptive fields". A mathematical operation of the convolution is applied on the input layer, which emulates the response to the next layer. The response is basically a visual stimulus. The detailed description is as follows:

3.1. Input Layer

The input data is loaded and stored in the input layer. This layer describes the height, width and number of channels (RGB information) of the input image.

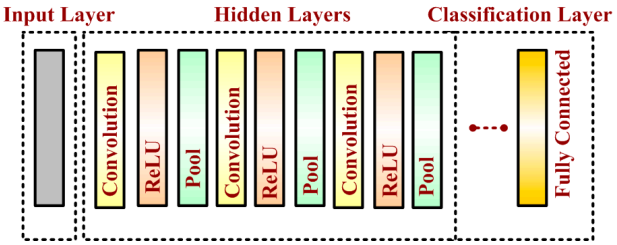


Figure 1. Typical convolutional neural network architecture.

3.2. Hidden Layer

The hidden layers are the backbone of CNN architecture. They perform a feature extraction process where a series of convolution, pooling and activation functions are used. The distinguishable features of handwritten digits are detected at this stage.

3.3. Convolutional Layer

The convolutional layer is the first layer placed above the input image. It is used for extracting the features of an image. The $n \times n$ input neurons of the input layer are convoluted with an $m \times m$ filter and in return deliver $(n - m + 1) \times (n - m + 1)$ as output. It introduces non-linearity through a neural activation function. The main contributors of the convolutional layer are receptive field, stride, dilation and padding, as described in the following paragraph.

CNN computation is inspired by the visual cortex in animals [58]. The visual cortex is a part of the brain that processes the information forwarded from the retina. It processes visual information and is subtle to small sub-regions of the input. Similarly, a receptive field is calculated in a CNN, which is a small region of an input image that can affect a specific region of the network. It is also one of the important design parameters of the CNN architecture and helps in setting other CNN parameters [59]. It has the same size as the kernel and works in a similar fashion as the foveal vision of the human eye works for producing sharp central vision. The receptive field is influenced by striding, pooling, kernel size and depth of the CNN [60]. Receptive field (r), effective receptive field (ERF) and projective field (PF) are terminology used in calculating effective sub-regions in a network. The area of the original image influencing the activation of a neuron is described using the ERF, whereas the PF is a count of neurons to which neurons project their outputs, as described in Figure 2. The visualization of the 5×5-size filter and its activation map are described in Figure 3. Stride is another parameter used in CNN architecture. It is defined as the step size by which the filter moves every time. A stride value of 1 indicates the filter sliding movement pixel by pixel. A larger stride size shows less overlapping between the cells. The working of kernel and stride in the convolution layer is presented in Figure 4.

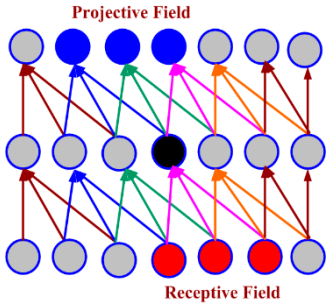


Figure 2. Receptive field and projective field.

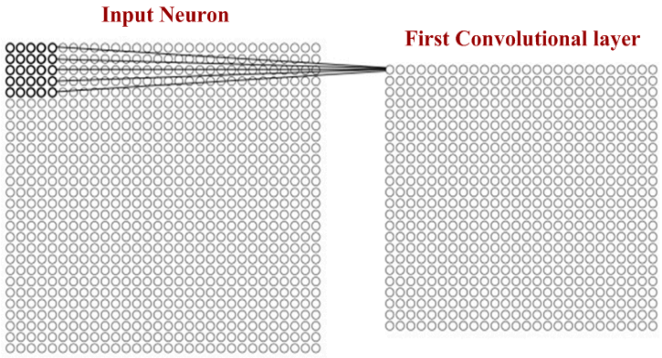


Figure 3. Visualization of filter of size 5×5 with activation map. (Input neuron 28×28 and convolutional layer 24×24).

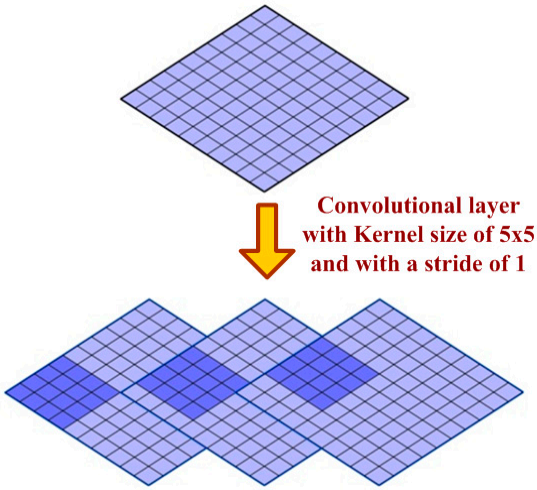


Figure 4. Representation of kernel and stride in a convolutional layer.

The concept of padding is introduced in CNN architecture to get more accuracy. Padding is introduced to control the shrinking of the output of the convolutional layer.

The output from the convolutional layer is a feature map, which is smaller than the input image. The output feature map contains more information on middle pixels and hence loses lots of information present on corners. The rows and the columns of zeros are added to the border of an image to prevent shrinking of the feature map.

Equations (1) and (2) describe the relationship between the size of the feature map, the kernel size and stride while calculating the size of the output feature map.

$$W_{nx} = W_{n-1x} - F_{nx}S_{nx} + 1 \tag{1}$$

$$W_{ny} = W_{n-1y} - F_{ny}S_{ny} + 1 \tag{2}$$

where (W_{nx}, W_{ny}) represent the size of the output feature map, (S_{nx}, S_{ny}) is stride size, and (F_x, F_y) is kernel size. Here 'n' is used to describe the index of layers.

The dilation is another important parameter of CNN architecture that has a direct influence on the receptive field. The dilation can increase the field-of-view (FOV) of a CNN without modifying the feature map [61]. Figure 5 clearly shows that dilation values can exponentially raise the receptive field of a CNN. Too large a dilation can increase the number of computations and hence can slow down the system by increasing the processing time. Therefore, it must be chosen wisely. The relationship between dilation, weight and input is shown in Equations (3) and (4) below.

$$0 - dialation = w[0] * x[0] + w[1] * x[1] + w[2] * x[2]; \quad (3)$$

$$1 - dialation = w[0] * x[0] + w[1] * x[2] + w[2] * x[4]; \quad (4)$$

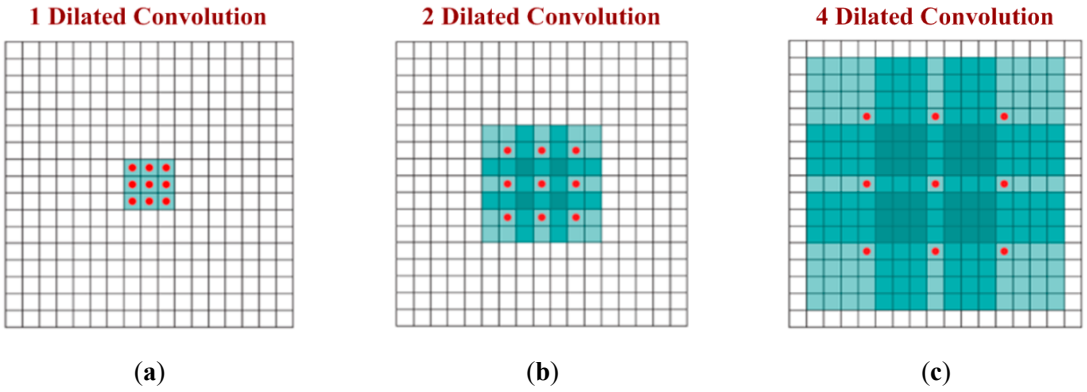


Figure 5. Dilated convolution: (**a**) receptive field of 3×3 using 1-dilated convolution; (**b**) receptive field of 7×7 using 2-dilated convolution; (**c**) receptive field of 15×15 using 4-dilated convolution.

3.4. Pooling Layer

A pooling layer is added between two convolutional layers to reduce the input dimensionality and hence to reduce the computational complexity. Pooling allows the selected values to be passed to the next layer while leaving the unnecessary values behind. The pooling layer also helps in feature selection and in controlling overfitting. The pooling operation is done independently. It works by extracting only one output value from the tiled non-overlapping sub-regions of the input images. The common types of pooling operations are max-pooling and avg-pooling (where max and avg represent maxima and average, respectively). The max-pooling operation is generally favorable in modern applications, because it takes the maximum values from each sub-region, keeping maximum information. This leads to faster convergence and better generalization [62]. The max-pooling operation for converting a 4×4 convolved output into a 2×2 output with stride size 2 is described in Figure 6. The maximum number is taken from each convolved output (of size 2×2) resulting in reducing the overall size to 2×2.

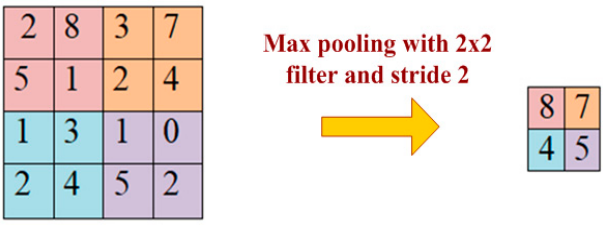


Figure 6. Max pooling with filter 2×2 and stride size.

3.5. Activation Layer

Just like regular neural network architecture, CNN architecture also contains the activation function to introduce the non-linearity in the system. The sigmoid function, rectified linear unit (ReLu) and Softmax are some famous choices among various activation functions exploited extensively in deep learning models. It has been observed that the sigmoid activation function might weaken the CNN model because of the loss of information present in the input data. The activation function used in the present work is the non-linear rectified linear unit (ReLu) function, which has output 0 for input less than 0 and raw output otherwise. Some advantages of the ReLu activation function are its similarity with the human nerve system, simplicity in use and ability to perform faster training for larger networks.

3.6. Classification Layer

The classification layer is the last layer in CNN architecture. It is a fully connected feed forward network, mainly adopted as a classifier. The neurons in the fully connected layers are connected to all the neurons of the previous layer. This layer calculates predicted classes by identifying the input image, which is done by combining all the features learned by previous layers. The number of output classes depends on the number of classes present in the target dataset. In the present work, the classification layer uses the 'softmax' activation function for classifying the generated features of the input image received from the previous layer into various classes based on the training data.

3.7. Gradient Descent Optimization Algorithm

Optimization algorithms are used to optimize neural networks and to generate better performance and faster results. The algorithm helps in minimizing or maximizing a cost function by updating the weight/bias values, which are known as learning parameters of a network, and the algorithm updating these values is termed as the adaptive learning algorithm. These learning parameters directly influence the learning process of a network and have an important role in producing an efficient network model. The aim of all the optimization algorithms is to find the optimum values of these learning parameters. The gradient descent algorithm is one such optimization algorithm. Recent classification experiments based on deep learning reported excellent performance with the progress in learning parameter identification [63–69].

Gradient descent, as the name implies, uses an error gradient to descend along with the error surface. It also allows a minimum of a function to be found when a derivative of it exists and there is only one optimum solution (if we expect local minima). The gradient is the slope of the error surface and gives an indication of how sensitive the error is towards the change in the weights. This sensitivity can be exploited for incremental change in the weights towards the optimum. Gradient descent is classified into three types, namely, batch gradient descent, stochastic gradient descent (SGD) and mini-batch gradient descent.

The SGD algorithm works faster and has been extensively used in deep learning experiments [70–74]. The SGD algorithm avoids redundant computation better than the batch gradient descent algorithm. The algorithm steps for updating the learning parameter for each training data bit are as follows (Algorithm 1):

Algorithm 1 Learning Parameters Update Mechanism

1: **Input:** $(x(i),\ y(i))$ as training data; η as learning rate; $\nabla E(\theta)$ is a gradient of loss (error) function $E(\theta)$ with respect to the θ parameter; momentum factor (m)
2: **Output:** For each training pair, update the learning parameters using the equation $\theta=\theta$-η.∇E(θ;x(i);y(i))
3: **if** stopping condition is met
4: **return** parameter θ.

The SGD algorithm can be further optimized using various optimizers, such as momentum, adaptive moment estimation (Adam), Adagrad and Adadelta. The momentum parameter is selected to speed up SGD optimization. The momentum optimizer can reduce the unnecessary parameter update, which leads to faster convergence of the network. The modified update equations with the momentum factor (m) are given by (5).

$$\left.\begin{array}{c} z(t) = mz(t-1) + \eta \nabla E(\theta) \\ \theta = \theta - z(t) \end{array}\right\} \tag{5}$$

Most implementation usually considers $m = 0.9$ for updating the learning parameters.

Adagrad is one of the SGD algorithms widely preferred in optimizing sparse data. The Adagrad optimizer works by considering the past gradients and modifying the learning rate for every parameter at each time step. The update equation for the Adagrad optimizer is given by (6).

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,\,ii} + \varepsilon}} \cdot g_{t,i} \tag{6}$$

where $g(t,i)$ is a gradient of the loss function for parameter $\theta(i)$ at a time step t; ε is a smoothing term taken to avoid the division by 0; and $G_{t,\,ii}$ is a diagonal matrix representing the sum of gradient squares for parameter $\theta(i)$ for a time step t;

The learning rate in the Adagrad optimizer keeps on decreasing, which causes a slow convergence rate and longer training time. Adadelta is another adaptive learning algorithm and is an extension of the Adagrad optimizer. It can beat the decaying learning rate problem of the previous optimizer, i.e., Adagrad [61]. Adadelta limits the storage of past squared gradients by fixing the storage window size. The moving averages of the square of gradients and square of weights updates are calculated. The learning rate is basically the square root of the ratio between these moving averages. A running average $E\left[g^2\right]_t$ at time step t of the past gradient is calculated using (7).

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma) g_t^2 \tag{7}$$

where $I\_$ is similar to the momentum term.

The parameter update for Adadelta optimization is done using Equations (8) and (9).

$$\Delta\theta_t = -\eta . g_{t,i} \tag{8}$$

$$\Delta\theta_{t+1} = \theta_t + \Delta\theta_t \tag{9}$$

or (10) can be modified as

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \varepsilon}} \odot g_t \tag{10}$$

Replacing the diagonal matrix with the running average $E\left[g^2\right]_t$,

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \odot g_t \tag{11}$$

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} \odot g_t \tag{12}$$

where $RMS[g]_t$ is the parameter update root mean square error and is calculated using (13):

$$RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon} \tag{13}$$

Adam is another famous SGD optimizer having learning weight updating similar to the storing average in Adadelta and decaying average of past squared gradients as present in the momentum

optimizer. Adam outperforms other techniques by performing fast convergence with a fast learning speed. Equations (14) and (15) are used to calculate the first-moment value ($M(t)$) and the variance of gradients ($v(t)$):

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{14}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{15}$$

and the parameter update is given in Equation (16):

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}}\hat{m}_t \tag{16}$$

4. Experimental Setup

To accomplish the task of handwritten digit recognition, a model of the convolutional neural network is developed and analyzed for suitable different learning parameters to optimize recognition accuracy and processing time. We propose to investigate variants of CNN architecture with three layers (CNN_3L) and variants of CNN architecture with four layers (CNN_4L). A total of six cases (case 1 to case 6) have been considered for CNN with three-layer architecture and five cases (case 1 to case 5) for CNN architecture with four layers. All the cases differ in the number of feature maps, stride sizes, padding, dilation and received receptive fields.

The recognition process of the handwritten digits consists of the following steps:

1. To acquire or collect the MNIST handwritten digit images.
2. To divide the input images into training and test images.
3. To apply the pre-processing technique to both the training dataset and the test dataset.
4. To normalize the data so that it ranges from 0 to 1.
5. To divide the training dataset into batches of a suitable size.
6. To train the CNN model and its variants using the labelled data.
7. To use a trained model for the classification.
8. To analyze the recognition accuracy and processing time for all the variants.

In summary, the present work for handwritten digit recognition investigates the role of training parameters, gradient descent optimizers and CNN architecture.

All the experiments were done in MATLAB 2018b on a personal computer with Windows 10, Intel (R) Core (TM) i7-6500 CPU (2.50 GHz), 16.00 GB memory and NVIDIA 1060 GTX GPU. The MNIST database was involved in the training and testing models. The standard MNIST handwritten digit database has 60,000 and 10,000 normalized digit images in its training and testing datasets, respectively. Some of the sample images from the MNIST database are shown in Figure 7.

Data pre-processing plays an important role in any recognition process. To shape the input images in a form suitable for segmentation, the pre-processing methods, such as scaling, noise reduction, centering, slanting and skew estimation, were used. In general, many algorithms work better after the data has been normalized and whitened. One needs to work with different algorithms in order to find out the exact parameters for data pre-processing. In the present work, the MNIST dataset images were size-normalized into a fixed image of size 28×28.

A convolution layer with 28 kernel/filter/patches and kernel sizes of 5×5 and 3×3 is used here to extract the features. Every patch contains the structural information of an image. For example, a convolution operation is computed by sliding a filter of size 5×5 over input image. This layer is responsible for transforming the input data by using a patch/filter of locally connecting neurons from the previous layer.

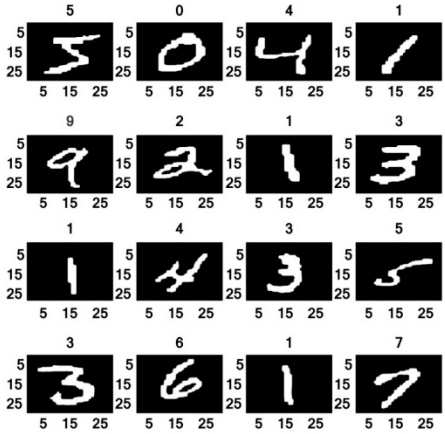


Figure 7. Sample MNIST handwritten digit images.

5. Results and Discussion

The experimental results of the MNIST handwritten digit dataset using different parameters of CNN_3L and CNN_4L architectures are recorded and analyzed in Tables 2 and 3, respectively. An MNIST sample image is represented as a 1-D array of 784 (28 × 28) float values between 0 and 1 (0 stands for black, 1 for white). The receptive field (r) is calculated based on kernel size (k), stride (s), dilation (d), padding (p), input size (i/p) and output size (o/p) of the feature map, and the recognition accuracy and total time elapsed is shown in Tables 2 and 3, employing CNN architecture with three and four layers respectively. The findings of Table 2 confirm the role of different architecture parameters on the performance of our recognition system. The training parameter used here has a learning rate of 0.01 and maximum epoch counts of 4. The highest recognition accuracy achieved in case 3, with CNN_3L architecture having three layers, is 99.76% for the feature map 12-24-32. In case 5, the CNN_4L architecture with four layers achieved the highest recognition accuracy of 99.76 % for the feature map 12-24-28-32, as shown in Table 3.

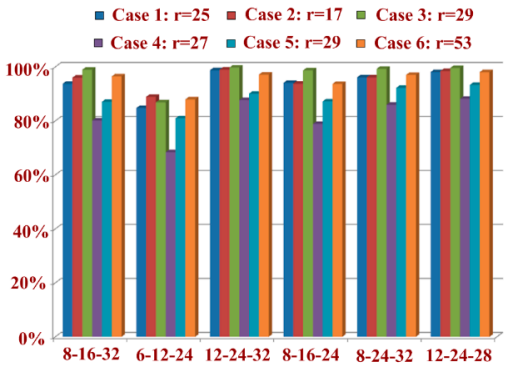
Table 2. Configuration details and accuracy achieved for convolutional neural network with three layers.

Model	Layer	k	s	d	p	i/p	o/p	r	Recognition Accuracy (%) and Total Time Elapsed					
									8-16-32	6-12-24	12-24-32	8-16-24	8-24-32	12-24-28
Case 1	Layer 1	5	2	2	2	28	14	5	93.76% (20 s)	84.76% (42 s)	98.76% (45 s)	94.08% (46 s)	96.12% (42 s)	98.08% (44 s)
	Layer 2	5	2	1	2	14	7	9						
	Layer 3	5	2	1	2	7	4	25						
Case 2	Layer 1	5	2	1	2	28	14	5	96.04% (37 s)	88.91% (27 s)	99% (37 s)	93.80% (37 s)	96.12% (37 s)	98.48% (17 s)
	Layer 2	3	2	1	2	14	7	9						
	Layer 3	3	2	1	2	7	4	17						
Case 3	Layer 1	5	2	1	2	28	14	5	98.96% (27 s)	86.88% (27 s)	**99.7% (29 s)**	98.72% (39 s)	99.28% (31 s)	99.60% (53 s)
	Layer 2	5	2	1	2	14	7	13						
	Layer 3	5	2	1	2	7	4	29						
Case 4	Layer 1	3	3	1	1	28	10	3	80.16% (48 s)	68.40% (29 s)	87.72% (29 s)	78.84% (29 s)	85.96% (51 s)	88.16% (29 s)
	Layer 2	3	3	1	1	10	4	9						
	Layer 3	3	3	1	1	4	2	27						
Case 5	Layer 1	5	3	1	2	28	10	5	87.08% (52 s)	80.96% (30 s)	90.08% (24 s)	87.22% (24 s)	92.24% (24 s)	93.32% (24 s)
	Layer 2	3	3	1	1	10	4	11						
	Layer 3	3	3	1	1	4	2	29						
Case 6	Layer 1	5	3	1	2	28	10	5	96.48% (23 s)	87.96% (23 s)	97.16% (23 s)	93.68% (24 s)	97.04% (24 s)	98.06% (24 s)
	Layer 2	5	3	1	2	10	4	17						
	Layer 3	5	3	1	2	4	2	53						

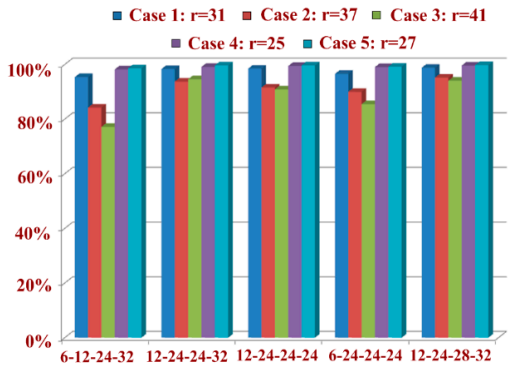
Table 3. Configuration details and accuracy achieved for convolutional neural network with four layers.

Model	Layer	k	s	d	p	i/p	o/p	r	Recognition Accuracy (%) and Total Time Elapsed				
									6-12-24-32	12-24-24-32	12-24-24-24	6-24-24-24	12-24-28-32
Case 1	Layer 1	3	2	1	1	28	14	3	95.36% (56 s)	98.34% (31 s)	98.48% (53 s)	96.56% (44 s)	98.80% (31 s)
	Layer 2	3	2	1	1	14	7	7					
	Layer 3	3	2	1	1	7	4	15					
	Layer 4	3	2	1	1	4	2	31					
Case 2	Layer 1	3	2	2	2	28	14	5	84.20% (26 s)	93.72% (30 s)	91.56% (24 s)	89.96% (26 s)	95.16% (25 s)
	Layer 2	3	2	2	2	14	7	13					
	Layer 3	3	2	1	1	7	4	21					
	Layer 4	3	2	1	1	4	2	37					
Case 3	Layer 1	5	2	2	2	28	14	9	77.16% (32 s)	94.60% (25 s)	90.88% (26 s)	85.48% (25 s)	94.04% (26 s)
	Layer 2	3	2	2	2	14	7	17					
	Layer 3	3	2	1	1	7	4	25					
	Layer 4	3	2	1	1	4	2	41					
Case 4	Layer 1	5	1	2	2	28	17	9	98.20% (30 s)	99.12% (29 s)	99.44% (25 s)	99.04% (22 s)	99.60% (29 s)
	Layer 2	3	2	2	2	17	7	13					
	Layer 3	3	2	1	1	7	4	17					
	Layer 4	3	2	1	1	4	2	25					
Case 5	Layer 1	5	1	2	2	28	28	9	98.60% (27 s)	99.64% (27 s)	99.64% (27 s)	99.20% (27 s)	**99.76% (43 s)**
	Layer 2	5	2	1	2	28	14	13					
	Layer 3	3	2	1	1	14	7	17					
	Layer 4	3	2	1	1	7	4	27					

The values of different architectural parameters have been chosen in such a way as to observe the role of all the parameters. For all convolutional layers, the hyper-parameters include kernel size (1–5), stride (1–3), dilation (1–2) and padding (1–2). The first observation from both of the tables shows the role of the receptive field. The value of the receptive field, when close to the input size, observed good recognition accuracy for all the feature maps (case 3 in Table 2 and case 5 in Table 3). On the other hand, a large gap between receptive field and input size observed poor recognition accuracy (case 3 and case 6 in Table 2 and case 3 in Table 3). The plots of Figure 8a,b clearly describe the relationship between recognition accuracy and receptive field. This highlights that receptive field can easily capture the elementary information like edges and corners from the input images in the lower layers of the CNN, which is passed to the subsequent layers for further processing. From Tables 2 and 3, it can also be observed that an increased number of filters (or increased width of the CNN) helps in improving the performance of CNN architecture. Case 5 of Table 3 also demonstrates the capability of multiple filters in extracting the full features of the handwritten images.



(**a**) Three-layer CNN model architecture vs. percentage accuracy



(**b**) Four-layer CNN model architecture vs. percentage accuracy

Figure 8. Receptive fields and recognition accuracies for CNN: (**a**) architecture having three layers; (**b**) architecture having four layers.

The recognition accuracy of MNIST handwritten digits with different optimizers is shown in Table 4. The optimizers like stochastic gradient descent with momentum (SGDm), Adam, Adagrad and Adadelta are used in the present work to obtain optimized performance. The highest accuracy is achieved using CNN_3L architecture with an Adam optimizer. The Adam optimizer computes adaptive learning rates for each parameter and performs fast convergence. It can be observed that training with the optimizer increases the accuracy of the classifier in both cases involving CNN_3L and CNN_4L. Furthermore, the optimized CNN variant having four layers has less accuracy than the similar variants with three layers. The increased number of layers might cause overfitting and consequently can influence the recognition accuracy. The problem of the overfitting can be avoided by finding out optimal values using trial and error or under some guidance. The concept of dropout may be used to solve the problem of overfitting, in which we can stop some randomly selected neurons (both hidden and visible) from participating in the training process. Basically, dropout is a weight regularization technique and is most preferred in larger networks to achieve better outcomes. Generally, a small value of dropout is preferred; otherwise, the network may be under learning.

The objective of the present work is to thoroughly investigate all the parameters of CNN architecture that deliver best recognition accuracy for a MNIST dataset. Overall, it has been observed that the proposed model of CNN architecture with three layers delivered better recognition accuracy of 99.89% with the Adam optimizer.

Table 4. Recognition accuracy with different optimizers.

Model	Recognition Accuracy (%)			
	Momentum (Sgdm)	Adam	Adagrad	Adadelta
CNN_3L	99.76%	99.89	98.67	99.77
CNN_4L	99.76%	99.35	98	99.73

The comparison of the proposed CNN-based approach with other approaches for handwritten numeral recognition is provided in Table 5. It can be observed that our CNN model outperforms the various similar CNN models proposed by various researchers using the same MNIST benchmark dataset. Some researchers used ensemble CNN architectures for the same dataset to improve their recognition accuracy but at the cost of increased computational cost and high testing complexity. The proposed CNN model achieved recognition accuracy of 99.89% for the MNIST dataset even without employing ensemble architecture.

Table 5. Comparison of proposed CNN architecture for numeral recognition with other techniques.

Handwritten Numeral Recognition				
Reference	**Approach**	**Database**	**Features**	**Accuracy (%)/Error Rate**
[75]	CNN	MNIST	Pixel based	0.23%
[76]	CNN	MNIST	Pixel based	0.19%
[8]	CNN	MNIST	Pixel based	0.53%
[77]	CNN	MNIST	Pixel based	0.21%
[78]	CNN	MNIST	Pixel based	0.17%
[79]	Deep Learning	The Chars74K	Pixel based	88.89% (GoogleNet) 77.77% (Alexnet)
[43]	CNN	Urdu Nasta'liq handwritten dataset (UNHD)	Pixel and geometrical based	98.3%
Proposed approach	CNN	MNIST	Pixel and geometrical based	99.89%

6. Conclusions

In this work, with the aim of improving the performance of handwritten digit recognition, we evaluated variants of a convolutional neural network to avoid complex pre-processing, costly feature extraction and a complex ensemble (classifier combination) approach of a traditional recognition system. Through extensive evaluation using a MNIST dataset, the present work suggests the role of various hyper-parameters. We also verified that fine tuning of hyper-parameters is essential in improving the performance of CNN architecture. We achieved a recognition rate of 99.89% with the Adam optimizer for the MNIST database, which is better than all previously reported results. The effect of increasing the number of convolutional layers in CNN architecture on the performance of handwritten digit recognition is clearly presented through the experiments.

The novelty of the present work is that it thoroughly investigates all the parameters of CNN architecture that deliver best recognition accuracy for a MNIST dataset. Peer researchers could not match this accuracy using a pure CNN model. Some researchers used ensemble CNN network architectures for the same dataset to improve their recognition accuracy at the cost of increased computational cost and high testing complexity but with comparable accuracy as achieved in the present work.

In future, different architectures of CNN, namely, hybrid CNN, viz., CNN-RNN and CNN-HMM models, and domain-specific recognition systems, can be investigated. Evolutionary algorithms can be explored for optimizing CNN learning parameters, namely, the number of layers, learning rate and kernel sizes of convolutional filters.

Author Contributions: data curation, S.A. and A.C.; formal analysis, S.A., A.C. and A.N.; funding acquisition, B.Y.; investigation, S.S.; methodology, A.N.; software, A.C.; validation, S.S.; writing—original draft, S.A., A.N. and S.S.; writing—review and editing, B.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Dongguk University Research Fund (S-2019-G0001-00043).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893.
2. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 2. [CrossRef]
3. Xiao, J.; Xuehong, Z.; Chuangxia, H.; Xiaoguang, Y.; Fenghua, W.; Zhong, M. A new approach for stock price analysis and prediction based on SSA and SVM. *Int. J. Inf. Technol. Decis. Making* **2019**, *18*, 287–310. [CrossRef]
4. Wang, D.; Lihong, H.; Longkun, T. Dissipativity and synchronization of generalized BAM neural networks with multivariate discontinuous activations. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 3815–3827. [PubMed]
5. Kuang, F.; Siyang, Z.; Zhong, J.; Weihong, X. A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. *Soft Comput.* **2015**, *19*, 1187–1199. [CrossRef]
6. Choudhary, A.; Ahlawat, S.; Rishi, R. A binarization feature extraction approach to OCR: MLP vs. RBF. In Proceedings of the International Conference on Distributed Computing and Technology ICDCIT, Bhubaneswar, India, 6–9 February 2014; Springer: Cham, Switzerland, 2014; pp. 341–346.
7. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **1980**, *36*, 193–202. [CrossRef]
8. Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; LeCun, Y. What is the best multi-stage architecture for object recognition. In Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV), Kyoto, Japam, 29 September–2 October 2009.

9. Ciresan, D.C.; Meier, U.; Masci, J.; Gambardella, L.M.; Schmidhuber, J. High-performance neural networks for visual object classification. *arXiv* **2011**, arXiv:1102.0183v1.
10. Ciresan, D.C.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. *arXiv* **2012**, arXiv:1202.2745.
11. Niu, X.X.; Suen, C.Y. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recognit.* **2012**, *45*, 1318–1325. [CrossRef]
12. Qu, X.; Wang, W.; Lu, K.; Zhou, J. Data augmentation and directional feature maps extraction for in-air handwritten Chinese character recognition based on convolutional neural network. *Pattern Recognit. Lett.* **2018**, *111*, 9–15. [CrossRef]
13. Alvear-Sandoval, R.; Figueiras-Vidal, A. On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. *Inf. Fusion* **2018**, *39*, 41–52. [CrossRef]
14. Demir, C.; Alpaydin, E. Cost-conscious classifier ensembles. *Pattern Recognit. Lett.* **2005**, *26*, 2206–2214. [CrossRef]
15. Choudhary, A.; Ahlawat, S.; Rishi, R. A neural approach to cursive handwritten character recognition using features extracted from binarization technique. *Complex Syst. Model. Control Intell. Soft Comput.* **2015**, *319*, 745–771.
16. Choudhary, A.; Rishi, R.; Ahlawat, S. Handwritten numeral recognition using modified BP ANN structure. In Proceedings of the Communication in Computer and Information Sciences (CCIS-133), Advanced Computing, CCSIT 2011, Royal Orchid Central, Bangalore, India, 2–4 January 2011; Springer: Berling/Heildelberg, Germany, 2011; pp. 56–65.
17. Cai, Z.W.; Li-Hong, H. Finite-time synchronization by switching state-feedback control for discontinuous Cohen–Grossberg neural networks with mixed delays. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1683–1695. [CrossRef]
18. Zeng, D.; Dai, Y.; Li, F.; Sherratt, R.S.; Wang, J. Adversarial learning for distant supervised relation extraction. *Comput. Mater. Contin.* **2018**, *55*, 121–136.
19. Long, M.; Yan, Z. Detecting iris liveness with batch normalized convolutional neural network. *Comput. Mater. Contin.* **2019**, *58*, 493–504. [CrossRef]
20. Chuangxia, H.; Liu, B. New studies on dynamic analysis of inertial neural networks involving non-reduced order method. *Neurocomputing* **2019**, *325*, 283–287.
21. Xiang, L.; Li, Y.; Hao, W.; Yang, P.; Shen, X. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput. Mater. Contin.* **2018**, *55*, 541–559.
22. Huang, Y.S.; Wang, Z.Y. Decentralized adaptive fuzzy control for a class of large-scale MIMO nonlinear systems with strong interconnection and its application to automated highway systems. *Inf. Sci.* **2014**, *274*, 210–224. [CrossRef]
23. Choudhary, A.; Rishi, R. Improving the character recognition efficiency of feed forward bp neural network. *Int. J. Comput. Sci. Inf. Technol.* **2011**, *3*, 85–96. [CrossRef]
24. Ahlawat, S.; Rishi, R. A genetic algorithm based feature selection for handwritten digit recognition. *Recent Pat. Comput. Sci.* **2019**, *12*, 304–316. [CrossRef]
25. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
26. Pham, V.; Bluche, T.; Kermorvant, C.; Louradour, J. Dropout improves recurrent neural networks for handwriting recognition. In Proceedings of the 14th Int. Conf. on Frontiers in Handwriting Recognition, Heraklion, Greece, 1–4 September 2014.
27. Tabik, S.; Alvear-Sandoval, R.F.; Ruiz, M.M.; Sancho-Gómez, J.L.; Figueiras-Vidal, A.R.; Herrera, F. MNIST-NET10: A heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. *Ensembles Overv. Proposal Inf. Fusion* **2020**, *62*, 73–80. [CrossRef]
28. Lang, G.; Qingguo, L.; Mingjie, C.; Tian, Y.; Qimei, X. Incremental approaches to knowledge reduction based on characteristic matrices. *Int. J. Mach. Learn. Cybern.* **2017**, *8*, 203–222. [CrossRef]
29. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
30. He, S.; Zeng, W.; Xie, K.; Yang, H.; Lai, M.; Su, X. PPNC: Privacy preserving scheme for random linear network coding in smart grid. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 1510–1532.

31. Sueiras, J.; Ruiz, V.; Sanchez, A.; Velez, J.F. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing.* **2018**, *289*, 119–128. [CrossRef]
32. Liang, T.; Xu, X.; Xiao, P. A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. *Pattern Recognit. Lett.* **2017**, *94*, 105–111. [CrossRef]
33. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practice for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, UK, 3–6 August 2003.
34. Wang, T.; Wu, D.J.; Coates, A.; Ng, A.Y. End-to-end text recognition with convolutional neural networks. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012.
35. Shi, B.; Bai, X.; Yao, C. An End-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2298–2304. [CrossRef]
36. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
37. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef]
38. Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, Araucano Park, Las Condes, Chille, 11–18 December 2015.
39. Boufenar, C.; Kerboua, A.; Batouche, M. Investigation on deep learning for off-line handwritten Arabic character recognition. *Cogn. Syst. Res.* **2018**, *50*, 180–195. [CrossRef]
40. Kavitha, B.; Srimathi, C. Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks. *J. King Saud Univ. Comput. Inf. Sci.* **2019**. [CrossRef]
41. Dewan, S.; Chakravarthy, S. A system for offline character recognition using auto-encoder networks. In Proceedings of the International Conference on Neural Information Processing, Doha, Qatar, 12–15 November 2012.
42. Ahmed, S.; Naz, S.; Swati, S.; Razzak, M.I. Handwritten Urdu character recognition using one-dimensional BLSTM classifier. *Neural Comput. Appl.* **2019**, *31*, 1143–1151. [CrossRef]
43. Husnain, M.; Saad Missen, M.; Mumtaz, S.; Jhanidr, M.Z.; Coustaty, M.; Luqman, M.M.; Ogier, J.-M.; Choi, G.S. Recognition of urdu handwritten characters using convolutional neural network. *Appl. Sci.* **2019**, *9*, 2758. [CrossRef]
44. Sarkhel, R.; Das, N.; Das, A.; Kundu, M.; Nasipuri, M. A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts. *Pattern Recognit.* **2017**, *71*, 78–93. [CrossRef]
45. Xie, Z.; Sun, Z.; Jin, L.; Feng, Z.; Zhang, S. Fully convolutional recurrent network for handwritten chinese text recognition. In Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016), Cancun, Mexico, 4–8 December 2016.
46. Liu, C.; Yin, F.; Wang, D.; Wang, Q.-F. Online and offline handwritten Chinese character recognition: Benchmarking on new databases. *Pattern Recognit.* **2013**, *46*, 155–162. [CrossRef]
47. Wu, Y.-C.; Yin, F.; Liu, C.-L. Improving handwritten chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognit.* **2017**, *65*, 251–264. [CrossRef]
48. Gupta, A.; Sarkhel, R.; Das, N.; Kundu, M. Multiobjective optimization for recognition of isolated handwritten Indic scripts. *Pattern Recognit. Lett.* **2019**, *128*, 318–325. [CrossRef]
49. Nguyen, C.T.; Khuong, V.T.M.; Nguyen, H.T.; Nakagawa, M. CNN based spatial classification features for clustering offline handwritten mathematical expressions. *Pattern Recognit. Lett.* **2019**. [CrossRef]
50. Ziran, Z.; Pic, X.; Innocenti, S.U.; Mugnai, D.; Marinai, S. Text alignment in early printed books combining deep learning and dynamic programming. *Pattern Recognit. Lett.* **2020**, *133*, 109–115. [CrossRef]
51. Ptucha, R.; Such, F.; Pillai, S.; Brokler, F.; Singh, V.; Hutkowski, P. Intelligent character recognition using fully convolutional neural networks. *Pattern Recognit.* **2019**, *88*, 604–613. [CrossRef]
52. Cui, H.; Bai, J. A new hyperparameters optimization method for convolutional neural networks. *Pattern Recognit. Lett.* **2019**, *125*, 828–834. [CrossRef]

53. Tso, W.W.; Burnak, B.; Pistikopoulos, E.N. HY-POP: Hyperparameter optimization of machine learning models through parametric programming. *Comput. Chem. Eng.* **2020**, *139*, 106902. [CrossRef]
54. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016.
55. Christian, S.; Wei, L.; Yangqing, J.; Pierre, S.; Scott, R.; Dragomir, A.; Andrew, R. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
56. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
57. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
58. Eickenberg, M.; Gramfort, A.; Varoquaux, G.; Thirion, B. Seeing it all: Convolutional network layers map the function of the human visual system. *NeuroImage* **2017**, *152*, 184–194. [CrossRef] [PubMed]
59. Le, H.; Borji, A. What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks? *arXiv* **2018**, arXiv:1705.07049.
60. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. *arXiv* **2017**, arXiv:1701.04128.
61. Lin, G.; Wu, Q.; Qiu, L.; Huang, X. Image super-resolution using a dilated convolutional neural network. *Neurocomputing* **2018**, *275*, 1219–1230. [CrossRef]
62. Scherer, D.; Muller, A.; Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In Proceedings of the International Conference on Artificial Neural Networks, Thessaloniki, Greece, 15–18 September 2010.
63. Shi, Z.; Ye, Y.; Wu, Y. Rank-based pooling for deep convolutional neural networks. *Neural Netw.* **2016**, *83*, 21–31. [CrossRef]
64. Wu, H.; Gu, X. Towards dropout training for convolutional neural networks. *Neural Netw.* **2015**, *71*, 1–10. [CrossRef] [PubMed]
65. Saeed, F.; Paul, A.; Karthigaikumar, P.; Nayyar, A. Convolutional neural network based early fire detection. *Multimed. Tools Appl.* **2019**, 1–17. [CrossRef]
66. Alzubi, J.; Nayyar, A.; Kumar, A. Machine learning from theory to algorithms: An overview. *J. Phys. Conf. Series* **2018**, *1142*, 012012. [CrossRef]
67. Duchi, J.; Hazen, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
68. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. *arXiv* **2012**, arXiv:abs/1212.5701.
69. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
70. Bartlett, P.; Hazan, E.; Rakhlin, A. Adaptive online gradient descent. In Proceedings of the NIPS, Vancuver, BC, Canada, 8–11 December 2008.
71. Do, C.B.; Le, Q.V.; Foo, C.S. Proximal regularization for online and batch learning. In Proceedings of the ICML, Montreal, QC, Canada, 14–18 June 2009.
72. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**. [CrossRef] [PubMed]
73. Shalev-Shwartz, S.; Singer, Y.; Srebro, N. Pegasos: Primal estimated sub-gradient solver for svm. In Proceedings of the ICML, Corvallis, OR, USA, 20–24 June 2007.
74. Zinkevich, M.; Weimer, M.; Smola, A.; Li, L. Parallelized stochastic gradient descent. *NIPS* **2010**, *2*, 2595–2603.
75. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
76. Dietterich, T.; Bakiri, G. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Intell. Res.* **1995**, *1*, 263–286. [CrossRef]
77. Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; Fergus, R. Regularization of neural networks using DropConnect. In Proceedings of the 30th International Conference on Machine Learning (PMLR), Atlanta, GA, USA, 16–21 June 2013.

78. Loquercio, A.; Della Torre, A.; Buscema, M. Computational Eco-Systems for handwritten digits recognition. *arXiv* **2017**, arXiv:1703.01872v1.
79. Soomro, M.; Farooq, M.A.; Raza, M.A. Performance evaluation of advanced deep learning architectures for offline handwritten character recognition. In Proceedings of the International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 18–20 December 2017; pp. 362–367.