Article

Recognition of Handwritten Arabic and Hindi Numerals Using Convolutional Neural Networks

Amin Alqudah [1,*], Ali Mohammad Alqudah [2], Hiam Alquran [2], Hussein R. Al-Zoubi [1], Mohammed Al-Qodah [3] and Mahmood A. Al-Khassaweneh [1,4]

1 Department of Computer Engineering, Yarmouk University, Irbid 21163, Jordan; halzoubi@yu.edu.jo (H.R.A.-Z.); khassaweneh@ieee.org (M.A.A.-K.)
2 Department of Biomedical Systems and Informatics Engineering, Yarmouk University, Irbid 21163, Jordan; ali_qudah@yu.edu.jo (A.M.A.); heyam.q@yu.edu.jo (H.A.)
3 Department of Electrical Engineering, Prince Sattam Bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia; m.alqodah@psau.edu.sa
4 Department of Engineering, Computing and Mathematical Sciences, Lewis University, Romeoville, IL 60446, USA
* Correspondence: amin.alqudah@yu.edu.jo

Abstract: Arabic and Hindi handwritten numeral detection and classification is one of the most popular fields in the automation research. It has many applications in different fields. Automatic detection and automatic classification of handwritten numerals have persistently received attention from researchers around the world due to the robotic revolution in the past decades. Therefore, many great efforts and contributions have been made to provide highly accurate detection and classification methodologies with high performance. In this paper, we propose a two-stage methodology for the detection and classification of Arabic and Hindi handwritten numerals. The classification was based on convolutional neural networks (CNNs). The first stage of the methodology is the detection of the input numeral to be either Arabic or Hindi. The second stage is to detect the input numeral according to the language it came from. The simulation results show very high performance; the recognition rate was close to 100%.

Keywords: Arabic numerals; CNN; deep learning; handwritten; numerals Hindi numerals

Citation: Alqudah, A.; Alqudah, A.M.; Alquran, H.; Al-Zoubi, H.R.; Al-Qodah, M.; Al-Khassaweneh, M.A. Recognition of Handwritten Arabic and Hindi Numerals Using Convolutional Neural Networks. *Appl. Sci.* **2021**, *11*, 1573. https://doi.org/10.3390/app11041573

Received: 9 December 2020
Accepted: 22 January 2021
Published: 9 February 2021

1. Introduction

Automatic recognition is widely applied in many aspects; such as face recognition, fingerprint recognition, and numerals recognition. Our traditional life is transforming from paper full to paperless. Nowadays, we communicate using different methods, the main methods are Internet and mobile telephones. Digital services are used in almost every aspect of our lives. Nevertheless, there are still some daily activities and tasks that depend on traditional methods of communication, such as using paper and pen. Therefore, automatic handwriting recognition can undoubtedly save time and effort. Handwritten digit recognition is one of the most successful applications of automatic pattern recognition either on-line or off-line. Most of such these applications were performed on Arabic digits, because it is the most known numbering system in the world. However, Hindi digits are widely used in some countries in Africa [1–4].

Handwritten recognition is one of the most difficult tasks due stylistic variations of the writers. Due to the importance of automated handwritten recognition in various applications such as reading postal codes, checking verifications, automatic recognition of handwritten numerals is one of the fields that gains interest by many researchers in order to develop algorithms and/or methodologies to increase recognition rates [5–11]. In this study, we employed convolutional neural networks (CNNs) for the offline recognition of Arabic and Hindi handwritten decimal numbers and through different number of scenarios and experiments, we demonstrated that very high recognition rates can be achieved.

2. Background and State-of-the-Art Research

In this section, we review the latest published methodologies in the field of numeral recognition focusing on the recognition of handwritten numerals. In our survey on related work about handwritten numeral recognition, we encountered few studies employing various classification techniques. Here, we briefly review the most recent research of this active research area, which plays an important role in the field of artificial intelligence.

In [12], handwritten Arabic (Indian) numerals were classified using a two-stage classification technique with Nearest Mean (NMC), K-Nearest Neighbor (K-NNC), and Hidden Markov Models (HMMC) being used in the first stage and a Structural Classifier (SC) being applied in the second stage. The achieved recognition ratios were around 98%, on average. In their work [13], Choudhary et al. used a supervised learning technique based on the artificial neural network (ANN) for offline handwritten numeral recognition, where they employed a multilayered perceptron (MLP) with one hidden layer. The drawback of this work is the small-sized data set with very few samples.

The authors of [14] applied Adaboost on an ensemble of MLP classifiers for the recognition of three handwritten Indian numerals: Devanagari, Bangla, and Oriya. The work in [15] focused on recognizing the seven segments display of LCDs with an obtained recognition accuracy of 79%. Shape context descriptor methodology was used in [16] to recognize Marathi and English characters and numerals. The achieved recognition accuracy was between 83% and 95%. The study in [17] employed deep learning in the recognition of handwritten Arabic numerals and achieved an accuracy of 97.4%. In [18], the authors used Spiking neural networks (SNNs) in the recognition of handwritten numerals, where they were able to improve the recognition accuracy compared to the use of BP neural networks.

The authors of [19] employed deep learning techniques for the recognition of numerals from multiple languages including Eastern Arabic, Persian, Devanagari, Urdu, Western Arabic, where they were able to achieve high recognition rates. The author of [20] adopted the restricted Boltzmann machine (RBM) in his work as the representative deep learning methodology for the recognition of English and Devanagari numerals with an SVM-based classifier.

In [21], the authors surveyed the most-recent techniques for the recognition of printed and handwritten numerals and characters of both Indic and non-Indic scripts, where the non-Indic scripts include Bangla, Devanagari, Gurmukhi, Kannada, Oriya, and Tamil. The focus of the study in [22] was on the segmentation of a string of digits into individual numerals as the authors applied a combination of PCA and SVM classifiers and called it PCA-SVMNet in the recognition of strings of unknown lengths. The technique applied in [23] combined low-level stroke (LLS) and polar histogram for the recognition of handwritten Gujarati, Devanagari, and English numerals.

Nevertheless, we noticed certain shortcomings in each of previous studies including the lack of a comprehensive data set with large enough samples written by a large number of people for both Indian and Arabic numerals, in addition to unsatisfactory obtained recognition rates. Deep learning is a promising path toward a more capable artificial intelligence that has proved a success in various pattern recognition disciplines. In this study, we address these issues and present a state-of-the-art recognition approach based on deep learning methods.

Researchers and technologists are witnessing the widespread popularity of deep learning theoretically and practically. Deep learning is a branch of artificial neural networks (ANNs), where deep layers are adopted instead of shallow networks used previously in ANNs. We refer interested readers to several survey papers in the field [24–28], where the authors of [24] survey technologies, learning mechanisms, platforms, and applications of deep learning. A review of the recent algorithms and architectures that enhance both training and testing in terms of accuracy and time can be found in [25]. The survey in [26] focuses on joint representation, coordinated representation, and encoder-decoder, which are the frameworks used in Deep Multimodal Representation Learning. The authors of [27] reviewed the application of deep learning in the visual content analysis in multimedia data consisting of images and videos. Deep learning has been extensively used in image processing applications; meanwhile, this field has also promoted new deep learning algorithms and architectures [28].

In our previous work [10,29] we have tackled the problem of automatic numeral recognition, wherein [10] we addressed the problem of printed Arabic numerals. In that study, the correlation coefficient was utilized to find the best match between the numeral under test and a set of reference images representing the baseline for comparison. The task of handwritten numeral recognition is more challenging than that of printed counterparts. The reason for this is due to the imperfection in the handwriting of most individuals. This includes so many writing styles with very large variances in terms of size, orientation, line thickness, and overlapping, among many others. Therefore, the purpose of our next work [29] was to handle the recognition of handwritten Arabic numerals, where we employed self-organizing maps in this task.

Although in our previous work [10–29] we were able to achieve high recognition ratios, our mission in this research is to extend our findings and to build upon our previous results. Specifically, the objective of this work is to build a universal numeral recognizer; one that can recognize both printed and handwritten multilingual numerals. Moreover, we aim also at using the results and findings of this work in the recognition of printed and handwritten multilingual text, in general.

Herein, we present a fast and high accurate methodology using a deep learning technique that exhibits very high recognition rates for the offline detection of numerals (Arabic or Hindi) then classifying of the handwritten of both languages decimal numbers for all the numerals (0–9). The same CNN architecture has been used for all stages of recognition but trained in a different dataset. The training is done using three different types of solvers (Adaptive Moment Learning Rate (*Adam*), Root Mean Square Propagation (*RMSProp*), and Stochastic Gradient Descent with Momentum (*SGDM*)).

The novelty of this paper is obtaining a highly accurate, fast, and reliable system for recognition off-line numerals handwritten in either Arabic or Hindi format, that could be used as a universal system in different fields. All previous studies did not reach 100% accuracy in their work, besides, they applied their algorithms on limited datasets, while this work was applied on three types of huge datasets.

This paper is organized as follows: a description of the data processing, methodology, and CNN is presented in Section 2. Section 3 explains our scenarios and provides detailed results. Section 4 provides the time requirements of the proposed methodology/scenarios. Finally, Section 5 concludes the paper.

3. Materials and Methods

The proposed recognition approach consists of two main stages: the first stage is to detect whether the input numeral is in Arabic or Hindi format. The second stage is to recognize the value of each numeral. Figure 1 shows a flow diagram for the proposed methodology. In both stages, CNN was used to do the classification and the recognition. Three training algorithms were employed: Adaptive Moment Learning Rate (*Adam*), Root Mean Square Propagation (*RMSProp*), and Stochastic Gradient Descent with Momentum (*SGDM*). In the case of *Adam* training algorithms, different learning rates were used.

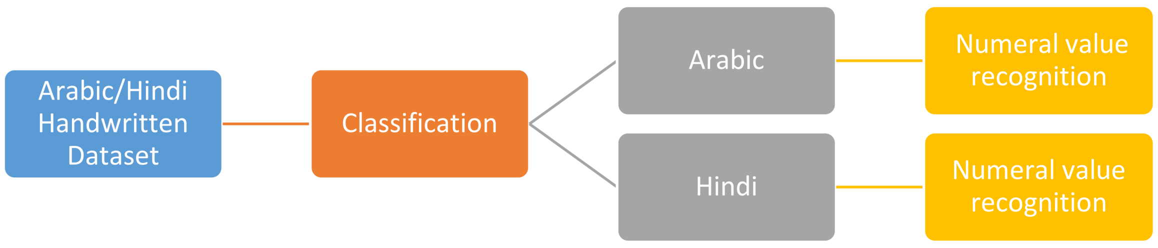

Arabic/Hindi Handwritten Dataset
Classification
Arabic
Hindi
Numeral value recognition
Numeral value recognition

Figure 1. Flow diagram of the proposed methodology.

3.1. Dataset

In this research, we used our dataset proposed in [29]. The images in the dataset were obtained by scanning huge number of handwritten numerals. The scanned image (original image) was transformed to a gray-level image and then into binary images afterward. The binary image was cleaned by opening and closing transforms and by erosion and dilation in order to smooth the numeral boundaries, maintain its size, and remove any noise. The original image dimensions used in this work are (32 $\times$ 32) pixels (including the number and background) [29]. The size of the data set was 49,900 images for all (0–9) Arabic numerals distributed equally, and 48,600 images for all (٠–٩) Hindi numerals distributed equally as well. Table 1 shows the 10 Arabic numerals and their corresponding Hindi numerals.

Table 1. Arabic numerals and their corresponding Hindi numerals.

Arabic Number	Hindi Number
0	٠
1	١
2	٢
3	٣
4	٤
5	٥
6	٦
7	٧
8	٨
9	٩

3.2. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are a deep learning algorithm, which is commonly used in automatic image features extraction, segmentation, and classification systems. It is a specific kind of artificial neural network (ANN) that is based on supervised machine learning techniques [30]. The preprocessing stage here is not necessarily as other classification algorithms. It is composed of an input layer, an output layer, and many hidden layers [31]. The distinguishing features of CNN are the composition of the hidden layer, which consists of many layers, such as the convolutional layer, which is responsible for detecting the high-level features using weight matrix or filters. These weights are adapted so the cost function is minimized. The various weights combination may extract edges, colors, or lines. The output here is called the feature map. This layer reduces the computation complexity significantly through several hyperparameters, such as depth, stride, and zero paddings [32]. The feature map's dimension can be reduced by passing it to the Pooling layer, which leads to decrease the computation time and complexity as well [33]. Besides, it controls overfitting [34]. There are two types of operations in the pooling layer: max pooling or average pooling. These operations act as an important role in the downstream process, which is the key to reducing the variations in data and preserving the most significant details [35]. The resultant reduced features map is modified by activation function such as rectified linear unit (ReLU), which replaces the negative value by 0 and leaves the positive value as it is. ReLU is defined as $(x) = max(0, x)$. ReLU is the most commonly used activation function. This layer is called the ReLU layer [33,36]. After that, a fully connected layer takes the output of convolution or Pooling layers to predict the best label to describe the image. A fully connected layer is often ended by a *softmax* layer which applies a *softmax* function to its input, which is expressed by the corresponding equation [37]:

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j\ )} \tag{1}$$

where x is the input vector of size K, j = 1:K, and x_i is ith individual input. The *Softmax* function limits the output between 0 and 1, which leads the output to be interpreted as a probability. It is commonly used in multivariant classifications purposes. The fully connected layer has nodes with the same number of output classes [38]. In the paper, we utilized the properties in MATLAB® 2019 [39] to build our own CNN which consists of 18 layers and employs it in discriminating the numerals. The following graph shows the flowchart of the proposed CNN.

Figure 2 illustrates the internal architecture of the proposed CNN. The input is a grayscale image with size 32 × 32, and the first convolutional layer uses 32 filters with size 3 × 3 and padding zeros is one, whereas the rest convolutional layers use 16 filters with the same size, except the third one uses eight filters only. To accelerate the training stage in CNN, and to decrease the sensitivity of the network initialization, the batch normalization layer is added between the convolutional layer and the nonlinearity layer (ReLU) [40]. Max pooling layer is employed in the suggested CNN with window size 2 × 2 and increments 2 pixels as well. The output layer consists of a fully connected layer with an output size 10, a softmax layer, and a classification layer. In the used CNN the loss function was cross entropy loss which is used during the training to modify model layer weights during preparation. The main aim of cross entropy loss is to minimize the loss, i.e., the smaller the loss, the better the model. The optimal model has a cross-entropic loss of 0. Table 2 shows the details of each layer in the used CNN architecture.

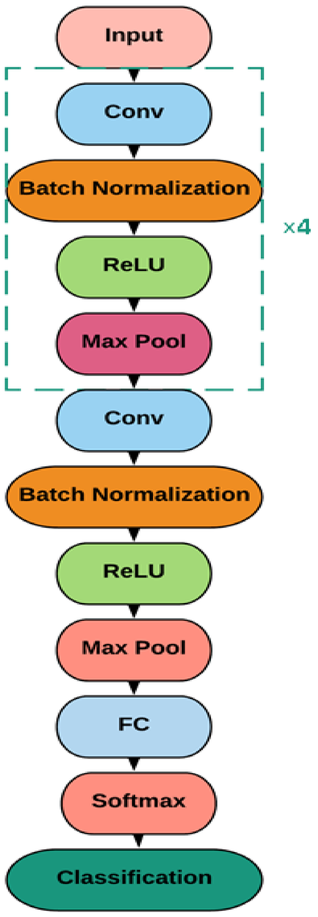


Figure 2. Proposed convolutional neural network (CNN) architecture.

Table 2. Layers information for the proposed CNN architecture.

#	Layer	Information	
1	Input Layer	Size	32×32
2	Conv_1	Number of filters	48
		Kernel size	3×3
		Activation	RELU
3	Batch_Norm_1	Number of channels	48
5	Maxpol_1	Kernel size	2×2
		Stride	2×2
6	Conv_2	Number of filters	32
		Kernel size	3×3
		Activation	RELU
7	Batch_Norm_2	Number of channels	32
9	Maxpol_2	Kernel size	2×2
		Stride	2×2
10	Conv_3	Number of filters	16
		Kernel size	3×3
		Activation	RELU
11	Batch_Norm_3	Number of channels	16
13	Maxpol_3	Kernel size	2×2
		Stride	2×2
14	Conv_4	Number of filters	32
		Kernel size	3×3
		Activation	RELU
15	Batch_Norm_4	Number of channels	32
16	Conv_5	Number of filters	32
		Kernel size	3×3
		Activation	RELU
17	Batch_Norm_3	Number of channels	16
18	Maxpol_2	Kernel size	2×2
		Stride	2×2

3.3. *Adaptive Moment Learning Rate* (Adam)

The adaptive moment learning rate (*Adam*) is a form of stochastic gradient descent with an adaptive learning rate. Stochastic is one of the most common optimization algorithms designed to tackle the very complex problem of optimization and is commonly used in deep learning to update weights based on a subset of training samples as shown in the equation below.

$$L_t(W) = \frac{1}{b}\sum_{j=1}^{b} l\left(W; x_{ij}; y_{ij}\right) + \gamma r(W) \tag{2}$$

where $\left\{\left(x_{ij}; y_{ij}\right)_{j=1}^{b}\right\}$ is the random mini-batch size chosen at iteration t, γ is the forgetting factor, l is the loss function, b the number of training samples, W are the weights, and r is the convex regularize [41].

The illustration for updating the weights in the *Adam* algorithm is shown in Table 3 [42,43],

Table 3. Steps of the *Adam* algorithm.

Step #	Equation	Explanation
1	$M_o = 0,\ R_o = 0$	Initialization for t = 1, . . . , T
2	$M_t = \beta_1 M_{t-1} + (1-\beta_1)\nabla l_t(W_{t-1})$	1st momentum estimate
3	$R_t = \beta_2 R_{t-1} + (1-\beta_2)\nabla l_t(W_{t-1})^2$	2nd momentum estimate
4	$\hat{M_t} = \frac{M_t}{(1-\beta_1)^t}$	1st momentum bias correction
5	$\hat{R_t} = \frac{MR_t}{(1-\beta_2)^t}$	2nd momentum bias correction
6	$W_t = W_{t-1} - \alpha \frac{\hat{M_t}}{\sqrt{\hat{R_t}+\varepsilon}}$	Update
7	*Return W*	Returning value

where M is the 1st momentum estimate, R is the 2nd momentum estimate, $\hat{M}$ is the 1st momentum correction, $\hat{R}$ is the 2nd momentum correction, W is the weights, α is the learning rate, β_1 and β_2 are the hyper-parameters, and ∇l_t is the gradient evaluated at timestep t.

3.4. Root Mean Square Propagation (RMSProp)

Neural networks that are based on distance measurements and Gaussian activation functions cannot be trained using only simple mini-batch gradient descent or momentum. This means no or only extremely slow convergence can be achieved. Even for shallow networks, this is the case. The solution is to apply *RMSProp* algorithm (root mean square propagation), which renders the training possible in the first place. *RMSprop* keeps a moving average of the element-wise squares of the parameter gradients, that is described by Equation (3)

$$v_l = \beta_2 v_{l-1} + (1-\beta_2)[\nabla E(\theta_l)]^2 \tag{3}$$

v is the moving avearge β_2 is the decay rate of the moving average, the value that was used is 0.9. θ is the parameter vector and *E(θ)* is the loss function. *RMSProp* updates its parameter individually as shown in Equation (4)

$$\theta_{l+1} = \theta_l - \frac{\alpha \nabla E(\theta_l)}{\sqrt{v_l} + \varepsilon} \tag{4}$$

α represents the learning rate, ε is a small value to avoid division on zero. *RMSProp* has shown excellent adaptation of indifferent software to the learning levels [44].

3.5. Stochastic Gradient Descent with Momentum

Stochastic Gradient Descent with Momentum (*SGDM*) has a difficulty negotiating ravines, i.e., places where the terrain bends in one dimension much more steeply than in another, which is normal around local optima [45]. Across these cases, *SGDM* oscillates around the slopes of the ravine while only advancing hesitantly along the bottom toward the optimal locale. A momentum is a tool that helps accelerate SGD and dampen oscillations in the related direction. It does so by adding to the current update vector a fraction of the update vector of the past time step. It is summarized by Equation (5) [42]:

$$\theta_{l+1} = \theta_l - \alpha\nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{5}$$

Typically, the momentum term γ is set to 0.9 or a similar value which is always less than 1. The process of updating parameters can be explained as a ball is going down a hill, accumulating energy as it is going down and getting faster and faster.

For dimensions whose gradients point in the same directions, the momentum term increases and decreases updates for dimensions whose gradients change direction. As a result, we are gaining faster convergence and reducing oscillation.

4. Results

In this paper, different scenarios were performed. First, we applied CNNs to classify the input images to be either Hindi numerals or Arabic numerals. Three training algorithms were used: *Adam*, *RMSProp*, and *SGDM*. The second scenario is to classify the Arabic images into their 10 subclasses. The third scenario is to classify the Hindi images into their 10 subclasses, as well. The same training algorithms that were used in the first scenario were used in the second and third scenarios. In both scenarios, *Adam* was used with different learning rates. In this research, the Adam optimization method was used because it provided a combined heuristics optimization of both Stochastic Gradient Descent with Momentum and Root Mean Square Propagation (*RMSProp*), and it can build a gradient descent which combats the pathological curvature of the problem with a faster search at the same time. The following subsections show detailed results.

4.1. Arabic–Hindi Classification

In this scenario, the main goal is to classify the images under consideration into two classes: Arabic numerals or Hindi numerals. The CNN was trained using the training data that were explained previously. In this regard, we had to exclude some numbers that could be written in the same manner in both languages (Hindi: ١, ٥, ٦, ٩, Arabic: **1**, **0**, **7**, **9**). Therefore, the total number of data used to train and test is 59,100 images, instead of 98,500 images.

Three algorithms were trained and tested in this regard: *Adam*, *RMSProp*, and *SGDM*. Table 4 shows part of the results of this scenario. In this table, the recognition rate was high for all classifiers, they all show promising results. The recognition rate was close to 100%. *Adam* is providing a combined heuristics optimization of both Stochastic Gradient Descent with Momentum and Root Mean Square Propagation, and it can build a gradient descent which combats the pathological curvature of the problem with a faster search at the same time.

Table 4. Summary of results for Arabic–Hindi classification using three different training algorithms.

Training Algorithm	Training Data				Testing Data		
	Number of Arabic Images	Number of Hindi Images	Recognition Rate (%)	Training Time Hours (Approx.)	Number of Arabic Images	Number of Hindi Images	Recognition Rate (%)
Adam	20,957	20,412	100	72	4454	4330	99.80
RMSProp	20,957	20,412	99.96	37	4447	4339	99.73
SGDM	20,957	20,412	100	40	4443	4347	99.73

Table 5 shows detailed results of *Adam* classifier, only 18 images out of 8784 images were misrecognized. The recognition rate was 99.8% (99.7% for Hindi, 99.8% for Arabic). Table 6 shows all the misrecognized images for Arabic–Hindi classification using *Adam* training algorithm. It is obvious from this table that some of the images were misrecognized due to unclear handwriting. From both tables, it is obvious that the recognition rate of Arabic images was slightly better than the recognition rate of Hindi images.

Table 5. Recognition rates for Arabic–Hindi classification using *Adam* training algorithm.

Numeral Value	**Number of Images Misrecognized in Hindi**	**Number of Images Misrecognized in Arabic**	**Total Number of Misrecognized Images**
0	0	0	0
1	0	0	0
2	3	2	5
3	2	0	2
4	3	1	4
5	0	1	1
6	0	0	0
7	3	0	3
8	2	1	3
9	0	0	0
Total Number of Misrecognized Images	13	5	18
Total Number of Correctly Recognized Images	4317	4449	8766
Total Number of Testing Images	4330	4454	8784
Recognition Rate (%)	99.7	99.89	99.8

Table 7 shows detailed results of *RMSProp* classifier, only 23 images out of 8784 images were misrecognized. The recognition rate was 99.73% (99.65% for Hindi, 99.82% for Arabic). Table 8 shows all the misrecognized images for Arabic–Hindi classification using *RMSProp* training algorithm. It is obvious from this table that some of the images were misrecognized due to unclear handwriting. From both tables, it is obvious that the recognition rate of Arabic images was slightly better than the recognition rate of Hindi images.

Table 9 shows detailed results of *SGDM* classifier, only 24 images out of 8784 images were misrecognized. The recognition rate was 99.73% (99.68% for Hindi, 99.77% for Arabic). Table 10 shows all the misrecognized images for Arabic–Hindi classification using *SGDM* training algorithm. It is obvious from this table that some of the images were misrecognized due to unclear handwriting. From both tables, it is obvious that the recognition rate of Arabic images was slightly better than the recognition rate of Hindi images.

4.2. *Arabic Classification*

In this scenario, the main goal is classify Arabic images into 10 subclasses (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Three training algorithms were investigated: *Adam*, *RMSProp*, and *SGDM. Adam* was trained with three different learning rates. As shown in Table 11, the size of the training data was 34,929 images, the recognition rate of all cases was high. *Adam* classifier has better performance when the learning rate was 0.0001. *Adam* is providing a combined heuristics optimization of both Stochastic Gradient Descent with Momentum and Root Mean Square Propagation, and it can build a gradient descent which combats the pathological curvature of the problem with a faster search at the same time.

Table 6. Misrecognized images for Arabic–Hindi classification using *Adam* training algorithm.

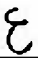
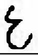
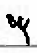
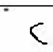
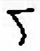
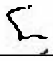
Numerals Format	Image	Standard Format	Misrecognized As	Recognition Rate (%)
Hindi		٨	Arabic	99.7
		٨	Arabic	
		٧	Arabic	
		٧	Arabic	
		٧	Arabic	
		٤	Arabic	
		٤	Arabic	
		٤	Arabic	
		٣	Arabic	
		٣	Arabic	
		٢	Arabic	
		٢	Arabic	
		٢	Arabic	
Arabic		4	Hindi	99.89
		2	Hindi	
		8	Hindi	
		5	Hindi	
		2	Hindi	

Table 12 shows detailed results of *Adam* classifier when the learning rate was 0.0001. Only 34 images out of 7384 images were misrecognized. The recognition rate was 99.54%. All numerals achieved recognition rate higher than 99%. The least recognition rate was for numeral "4". It was misrecognized seven times; six of them as "9". This happened because of the way that number was written, which is sometimes can be misleading. Table 13 shows all the misrecognized images for Arabic classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 14 shows detailed results of *Adam* classifier when the learning rate was 0.001. Only 38 images out of 7350 images were misrecognized. The recognition rate was 99.48%. Most numerals achieved recognition rate higher than 99%. The least recognition rate was for numeral "4". It was misrecognized nine times; five of them as "9". This happened because of the way that number was written, which is sometimes can be misleading. Table 15 shows all the misrecognized images for Arabic classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 7. Recognition rates for Arabic–Hindi classification using *RMSProp* training algorithm.

Numeral Value	**Number of Images Misrecognized in Hindi**	**Number of Images Misrecognized in Arabic**	**Total Number of Misrecognized Images**
0	0	0	0
1	0	0	0
2	4	2	6
3	2	0	2
4	6	2	8
5	0	2	2
6	0	2	2
7	2	0	2
8	1	0	1
9	0	0	0
Total Number of Misrecognized Images	15	8	23
Total Number of Correctly Recognized Images	4324	4439	8763
Total Number of Testing Images	4339	4447	8786
Recognition Rate (%)	99.65	99.82	99.73

Table 16 shows detailed results of *Adam* classifier when the learning rate was 0.01. Only 75 images out of 7331 images were misrecognized. The recognition rate was 98.98%. Most numerals achieved recognition rate higher than 99%. The least recognition rate was for numeral "4" and numeral "7". Numeral "4" was misrecognized 16 times; 15 of them as "9" and numeral "7" was misrecognized 16 times; seven of them as "1". This happened because of the way those numbers were written, which is sometimes can be misleading. Table 17 shows all the misrecognized images for Arabic classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 18 shows detailed results of *RMSProp* classifier. Only 56 images out of 7332 images were misrecognized. The recognition rate was 99.24%. Most numerals achieved recognition rate higher than 99%. The least recognition rate was for numeral "1". Numeral "1" was misrecognized 22 times; seven of them as "2" and seven of them as "4". This happened because of the way those numbers were written, which is sometimes can be misleading. Table 19 shows all the misrecognized images for Arabic classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 8. Misrecognized images for Arabic–Hindi classification using *RMSProp* training algorithm.

Numerals Format	Image	Standard Format	Misrecognized As	Recognition Rate (%)
Hindi		٤	Arabic	99.65
		٢	Arabic	
		٢	Arabic	
		٤	Arabic	
		٧	Arabic	
		٧	Arabic	
		٣	Arabic	
		٢	Arabic	
		٢	Arabic	
		٤	Arabic	
		٤	Arabic	
		٤	Arabic	
		٣	Arabi	
		٨	Arabi	
		٤	Arabi	
Arabic		6	Hindi	99.82
		6	Hindi	
		4	Hindi	
		5	Hindi	
		2	Hindi	
		4	Hindi	
		2	Hindi	
		5	Hindi	

Table 9. Recognition rates for Arabic–Hindi classification using *SGDM* training algorithm.

Numeral Value	**Number of Images Misrecognized in Hindi**	**Number of Images Misrecognized in Arabic**	**Total Number of Misrecognized Images**
0	0	0	0
1	0	0	0
2	1	4	5
3	3	0	3
4	8	2	10
5	0	2	2
6	0	0	0
7	2	0	2
8	0	2	2
9	0	0	0
Total Number of Misrecognized Images	14	10	24
Total Number of Correctly Recognized Images	4333	4433	8766
Total Number of Testing Images	4347	4443	8790
Recognition Rate (%)	99.68	99.77	99.73

Table 20 shows detailed results of *SGDM* classifier. Only 52 images out of 7329 images were misrecognized. The recognition rate was 99.29%. Most numerals achieved recognition rate higher than 99%. The least recognition rate was for numeral "4". Numeral "4" was misrecognized 10 times; eight of them as "9". This happened because of the way those numbers were written, which is sometimes can be misleading. Table 21 shows all the misrecognized images for Arabic classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 10. Misrecognized images for Arabic–Hindi classification using *SGDM* training algorithm.

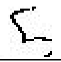
Numerals Format	Image	Standard Format	Misrecognized As	Recognition Rate (%)
Hindi		٢	Arabic	99.68
		٤	Arabic	
		٣	Arabic	
		٣	Arabic	
		٧	Arabic	
		٤	Arabic	
		٧	Arabic	
		٤	Arabic	
		٤	Arabic	
		٤	Arabic	
		٤	Arabic	
		٤	Arabic	
		٣	Arabi	
		٤	Arabi	
Arabic		2	Hindi	99.77
		2	Hindi	
		4	Hindi	
		2	Hindi	
		2	Hindi	
		4	Hindi	
		8	Hindi	
		8	Hindi	
		5	Hindi	
		5	Hindi	

Table 11. Summary of results for Arabic numerals classification using three different training algorithms.

Training Algorithm	Training Data			Testing Data	
	Number of Arabic Images	Recognition Rate (%)	Training Time Hours (Approx.)	Number of Arabic Images	Recognition Rate (%)
Adam (learning rate = 0.0001)	34,929	100	65	7348	99.54
Adam (learning rate = 0.001)	34,929	100	63	7350	99.48
Adam (learning rate = 0.01)	34,929	99.01	62	7331	98.98
RMSProp	34,929	99.93	50	7332	99.24
SGDM	34,929	100	52	7329	99.29

Table 12. Recognition rates for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.0001).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9		
0	732	0	0	0	0	0	1	0	0	0	1	99.86
1	1	738	3	1	0	0	0	0	0	0	5	99.33
2	0	2	732	0	0	0	0	0	1	0	3	99.59
3	0	1	0	730	0	2	0	0	1	0	4	99.46
4	0	0	0	0	726	0	0	1	0	6	7	99.05
5	0	0	0	1	0	736	0	0	0	0	1	99.86
6	2	1	0	0	0	2	736	0	1	0	6	99.19
7	0	0	0	2	0	0	0	733	0	0	2	99.73
8	2	0	0	0	0	0	0	0	729	0	2	99.73
9	0	0	0	0	2	0	0	0	1	722	3	99.59
Total Number of Misrecognized											34	99.54
Total Number of Correctly Recognized											7314	

4.3. Hindi Classification

In this scenario, the main goal is classify Hindi images into 10 subclasses (٠، ١، ٢، ٣، ٤، ٥، ٦، ٧، ٨، ٩). Three training algorithms were investigated: *Adam*, *RMSProp*, and *SGDM*. *Adam* was trained with three different learning rates. As shown in Table 22, the size of the training data was 34,020 images, the recognition rate of all cases was high. *Adam* classifier has better performance when the learning rate was 0.001. *Adam* is providing a combined heuristics optimization of both Stochastic Gradient Descent with Momentum and Root Mean Square Propagation, and it can build a gradient descent which combats the pathological curvature of the problem with a faster search at the same time.

Table 13. Misrecognized images for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.0001).

Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
0		6									99.86
1		3		2		2		2		0	99.33
2		1		8		1					99.59
3		5		5		1		8			99.46
4		9		9		9		9		7	99.05
		9		9							
5		3									99.86
6		5		5		1		0		8	99.19
		0									
7		3		3							99.73
8		0		0							99.73
9		4		8		4					99.59

Table 14. Recognition rates for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.001).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9		
0	733	0	0	0	0	0	1	0	0	0	1	99.86
1	1	736	1	0	2	2	0	0	0	0	6	99.19
2	0	2	732	0	0	0	0	0	2	0	4	99.46
3	0	0	0	731	0	0	0	0	2	0	2	99.73
4	1	0	0	1	722	0	0	1	1	5	9	98.77
5	0	0	0	3	0	733	0	0	0	0	3	99.59
6	1	0	0	0	0	1	742	0	1	0	3	99.6
7	0	0	0	1	0	0	0	729	1	0	2	99.73
8	1	0	0	1	0	0	1	0	732	0	3	99.59
9	0	0	0	0	3	0	0	1	1	722	5	99.31
Total Number of Misrecognized											38	99.48
Total Number of Correctly Recognized											7312	

Table 15. Misrecognized images for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.001).

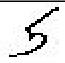
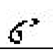
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
0		6									99.86
1		5		5		4		4		2	99.19
		0									
2		1		8		8		1			99.46
3		8		8							99.73
4		9		9		3		9		8	98.77
		9		9		7		0			
5		3		3		3					99.59
6		5		0		8					99.6
7		3		8							99.73
8		6		3		0					99.59
9		8		4		4		7		4	99.31

Table 16. Recognition rates for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.01).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9		
0	726	0	0	0	0	0	0	0	2	0	2	99.73
1	0	730	2	1	1	0	1	1	0	0	6	99.18
2	0	3	717	2	1	0	0	0	1	1	8	98.9
3	0	0	0	739	0	2	0	0	2	0	4	99.46
4	0	0	0	0	715	0	0	1	0	15	16	97.81
5	0	0	0	4	1	728	0	0	0	3	8	98.91
6	0	0	0	0	0	0	734	0	4	0	4	99.46
7	0	7	1	4	0	0	2	713	0	2	16	97.81
8	0	0	1	2	0	2	0	0	730	2	7	99.05
9	0	0	0	0	3	0	0	1	0	724	4	99.45
Total Number of Misrecognized											75	98.98
Total Number of Correctly Recognized											7256	

Table 17. Misrecognized images for Arabic numerals classification using *Adam* training algorithm (learning rate = 0.01).

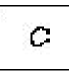
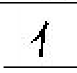
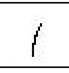
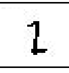
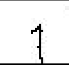
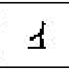
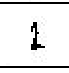
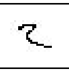
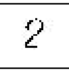
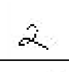
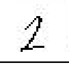
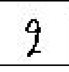
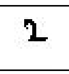
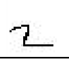
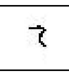
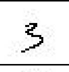
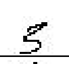
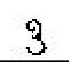
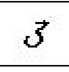
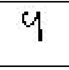
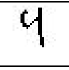
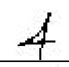
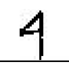
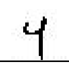
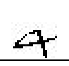
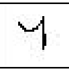
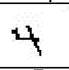
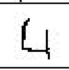
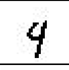
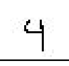
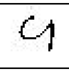
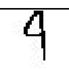
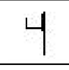
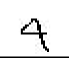
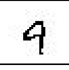
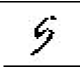
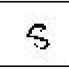
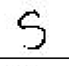
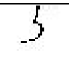
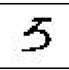
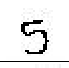
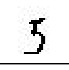
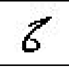
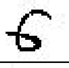
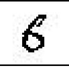
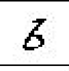
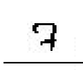
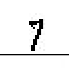
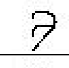
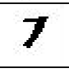
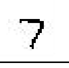
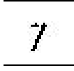
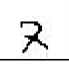
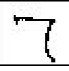
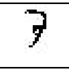
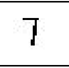
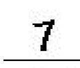
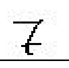
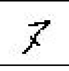
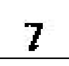
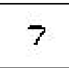
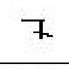
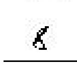
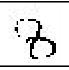
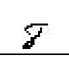
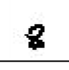
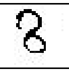
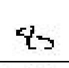
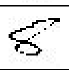
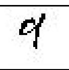
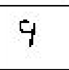
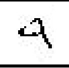
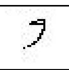
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
0		8		8							99.73
1		4		6		2		7		3	99.18
		2									
2		3		8		4		1		9	98.9
		1		1		3					
3		5		5		8		8			99.46
4		9		9		7		7		9	97.81
		9		9		9		9		9	
		9		9		9		9		9	
		9									
5		9		3		9		3		3	98.91
		9		4		3					
6		8		8		8		8			99.46
7		9		1		9		1		2	97.81
		1		3		6		3		1	
		1		6		3		1		3	
		1									
8		5		3		9		2		3	99.05
		9		5							
9		4		4		4		7			99.45

Table 18. Recognition rates for Arabic numerals classification using *RMSProp* training algorithm.

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9		
0	731	0	0	1	0	0	0	0	1	0	2	99.73
1	1	708	7	0	7	1	1	2	0	3	22	96.99
2	0	3	729	2	0	0	0	0	0	0	5	99.32
3	0	0	0	733	0	2	0	0	1	0	3	99.59
4	1	0	0	0	731	0	0	0	0	4	5	99.32
5	0	0	0	1	0	737	0	0	1	1	3	99.59
6	1	0	0	0	0	1	734	0	1	0	3	99.59
7	0	0	1	1	2	0	0	733	0	1	5	99.32
8	1	0	0	0	0	2	0	0	723	1	4	99.45
9	1	0	0	0	3	0	0	0	0	717	4	99.45
Total Number of Misrecognized											56	99.24
Total Number of Correctly Recognized											7276	

Table 19. Misrecognized images for Arabic numerals classification using *RMSProp* training algorithm.

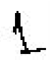
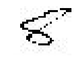
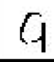
Numeral Value	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Recognition Rate (%)**
0		3		8							99.73
		4		2		4		4		9	
		5		2		0		2		2	
1		2		4		6		4		4	96.99
		7		7		4		2		9	
		9		2							
2		3		1		1		1		3	99.32
3		5		8		5					99.59
4		9		0		9		9		9	99.32
5		8		3		9					99.59
6		8		5		0					99.59
7		9		4		3		4		2	99.32
8		9		0		5		5			99.45
9		4		4		4		0			99.45

Table 20. Recognition rates for Arabic numerals classification using *SGDM* training algorithm.

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9		
0	731	0	0	0	0	0	0	0	1	0	1	99.86
1	1	726	5	0	0	1	1	0	0	1	9	98.78
2	0	3	724	1	0	0	0	3	0	1	8	98.91
3	0	0	0	731	0	1	0	0	1	0	2	99.73
4	0	1	0	0	722	0	0	0	1	8	10	98.63
5	0	0	0	0	0	735	1	0	0	0	1	99.86
6	0	0	0	0	0	2	730	0	1	0	3	99.59
7	0	2	0	2	0	0	0	732	0	2	6	99.19
8	1	0	1	1	0	2	1	0	725	1	7	99.04
9	0	0	0	0	3	1	0	0	1	721	5	99.31
Total Number of Misrecognized											52	99.29
Total Number of Correctly Recognized											7277	

Table 21. Misrecognized images for Arabic numerals classification using *SGDM* training algorithm.

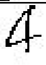
Numeral Value	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Missed Image**	**Recognized As**	**Recognition Rate (%)**
0		8									99.86
1		2		6		5		2		2	98.78
		9		2		2		0			
2		3		7		1		9		1	98.91
		7		1		7					
3		5		8							99.73
4		1		9		9		9		8	98.63
		9		9		9		9		9	
5		6									99.86
6		5		5		8					99.59
7		3		1		3		9		9	99.19
		1									
8		3		2		0		6		5	99.04
		9		5							
9		4		8		5		4		4	99.31

Table 22. Summary of results for Hindi numerals classification using three different training algorithms.

Training Algorithm	Training Data			Testing Data	
	Number of Hindi Images	Recognition Rate (%)	Training Time Hours (Approx.)	Number of Hindi Images	Recognition Rate (%)
Adam (learning rate = 0.0001)	34,020	99.80	62	7059	99.6
Adam (learning rate = 0.001)	34,020	99.97	61	7089	99.73
Adam (learning rate = 0.01)	34,020	98.73	58	7056	99.56
RMSProp	34,020	99.79	49	7089	99.62
SGDM	34,020	99.80	50	7082	99.73

Table 23 shows detailed results of *Adam* classifier when the learning rate was 0.0001. Only 28 images out of 7059 images were misrecognized. The recognition rate was 99.60%. Most numerals achieved recognition rate higher than 99%. Numeral "٠" and numeral "٧" achieved 100% recognition rate. The least recognition rate was for numeral "٢". It was misrecognized 10 times; five of them as "٣". This happened because of the way that number was written, which is sometimes can be misleading. Table 24 shows all the misrecognized images for Hindi classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 25 shows detailed results of *Adam* classifier when the learning rate was 0.001. Only 19 images out of 7089 images were misrecognized. The recognition rate was 99.73%. All numerals achieved recognition rate higher than 99%. Numeral "٠" and numeral "٧" achieved 100% recognition rate. The least recognition rate was for numeral "٢". It was misrecognized five times; two of them as "٣". This happened because of the way that number was written, which is sometimes can be misleading. Table 26 shows all the misrecognized images for Hindi classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 27 shows detailed results of *Adam* classifier when the learning rate was 0.01. Only 31 images out of 7056 images were misrecognized. The recognition rate was 99.56%. Most numerals achieved recognition rate higher than 99%. Numeral "٠", numeral "٤", and numeral "٥" achieved 100% recognition rate. The least recognition rate was for numeral "٢". It was misrecognized 11 times; four of them as "٣". This happened because of the way that number was written, which is sometimes can be misleading. Table 28 shows all the misrecognized images for Hindi classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 29 shows detailed results of *RMSProp* classifier. Only 27 images out of 7089 images were misrecognized. The recognition rate was 99.62%. Most numerals achieved recognition rate higher than 99%. Numeral "٠", numeral "٦", and numeral "٩" achieved 100% recognition rate. The least recognition rate was for numeral "٢". It was misrecognized 10 times; four of them as "٣". This happened because of the way that number was written, which is sometimes can be misleading. Table 30 shows all the misrecognized images for Hindi classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 23. Recognition rates for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.0001).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	٠	١	٢	٣	٤	٥	٦	٧	٨	٩		
٠	700	0	0	0	0	0	0	0	0	0	0	100
١	0	668	2	0	0	0	3	0	1	1	7	98.96
٢	0	1	688	5	2	1	1	0	0	0	10	98.57
٣	0	0	0	703	0	0	0	1	0	0	1	99.86
٤	0	0	0	0	712	0	0	0	0	2	2	99.72
٥	0	0	0	0	0	701	0	0	0	1	1	99.86
٦	0	0	0	0	0	0	715	0	0	1	1	99.86
٧	0	0	0	0	0	0	0	717	0	0	0	100
٨	0	0	0	0	0	1	0	0	722	1	2	99.72
٩	0	1	0	0	1	0	2	0	0	705	4	99.44
Total Number of Misrecognized											28	99.6
Total Number of Correctly Recognized											7031	

Table 24. Misrecognized images for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.0001).

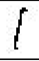
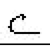
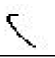
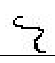
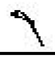
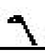
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
٠											100
١		٦		٨		٢		٦		٦	98.96
		٩		٢							
٢		٣		٣		٣		٤		٥	98.57
		٤		١		٣		٣		٦	
٣		٧									99.86
٤		٩		٩							99.72
٥		٩									99.86
٦		٩									99.86
٧											100
٨		٩		٥							99.72
٩		٤		١		٦		٦			99.44

Table 25. Recognition rates for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.001).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	٠	١	٢	٣	٤	٥	٦	٧	٨	٩		
٠	694	0	0	0	0	0	0	0	0	0	0	100
١	0	677	0	0	0	1	1	1	1	0	4	99.41
٢	0	1	695	2	1	1	0	0	0	0	5	99.29
٣	0	0	0	705	0	0	0	1	0	0	1	99.86
٤	0	0	0	0	715	0	0	0	0	1	1	99.86
٥	0	0	0	0	0	710	0	1	0	1	2	99.72
٦	0	0	0	0	0	0	720	0	0	1	1	99.86
٧	0	0	0	0	0	0	0	719	0	0	0	100
٨	0	0	0	0	0	0	0	0	726	1	1	99.86
٩	0	1	0	1	0	0	2	0	0	709	4	99.44
Total Number of Misrecognized											19	99.73
Total Number of Correctly Recognized											7070	

Table 26. Misrecognized images for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.001).

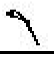
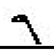
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
٠											100
١		٥		٧		٦		٨			99.41
٢		١		٤		٥		٣		٣	99.29
٣		٧									99.86
٤		٩									99.86
٥		٩		٧							99.72
٦		٩									99.86
٧											100
٨		٩									99.86
٩		١		٦		٦		٣			99.44

Table 27. Recognition rates for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.01).

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	٠	١	٢	٣	٤	٥	٦	٧	٨	٩		
٠	701	0	0	0	0	0	0	0	0	0	0	100
١	0	658	2	0	0	1	0	1	0	3	7	98.95
٢	0	0	690	4	3	2	0	1	0	1	11	98.43
٣	0	0	0	696	0	0	0	2	0	0	2	99.71
٤	0	0	0	0	718	0	0	0	0	0	0	100
٥	0	0	0	0	0	698	0	0	0	0	0	100
٦	0	0	0	0	0	0	716	0	0	2	2	99.72
٧	0	1	0	0	0	0	0	716	0	2	3	99.58
٨	0	0	0	0	0	1	0	1	719	2	4	99.45
٩	0	0	0	0	1	0	1	0	0	713	2	99.72
Total Number of Misrecognized											31	99.56
Total Number of Correctly Recognized											7025	

Table 28. Misrecognized images for Hindi numerals classification using *Adam* training algorithm (learning rate = 0.01).

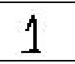
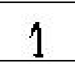
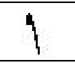
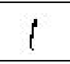
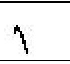
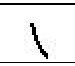
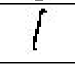
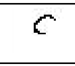
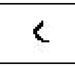
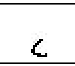
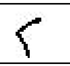
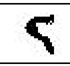
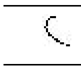
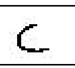
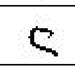
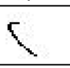
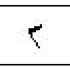
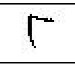
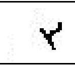
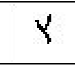
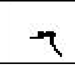
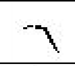
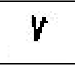
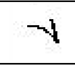
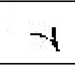
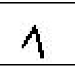
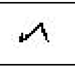
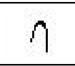
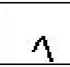
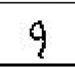
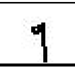
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
٠											100
١		٥		٩		٩		٢		٩	98.95
		٧		٢							
٢		٥		٣		٧		٤		٣	
		٤		٥		٩		٤		٣	98.43
		٣									
٣		٧		٧							99.71
٤											100
٥											100
٦		٩		٩							99.72
٧		١		٩		٩					99.58
٨		٩		٧		٥		٩			99.45
٩		٤		٦							99.72

Table 29. Recognition rates for Hindi numerals classification using *RMSProp* training algorithm.

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	٠	١	٢	٣	٤	٥	٦	٧	٨	٩		
٠	685	0	0	0	0	0	0	0	0	0	0	100
١	0	686	2	0	1	0	0	0	0	1	4	99.42
٢	0	2	686	7	0	1	0	0	0	0	10	98.56
٣	0	0	0	715	0	0	0	1	0	0	1	99.86
٤	0	1	1	0	714	0	0	0	0	0	2	99.72
٥	0	0	0	0	0	707	0	0	1	1	2	99.72
٦	0	0	0	0	0	0	720	0	0	0	0	100
٧	0	0	0	0	0	4	0	718	1	0	5	99.31
٨	0	0	0	0	0	2	1	0	720	0	3	99.59
٩	0	0	0	0	0	0	0	0	0	711	0	100
Total Number of Misrecognized											27	99.62
Total Number of Correctly Recognized											7062	

Table 30. Misrecognized images for Hindi numerals classification using *RMSProp* training algorithm.

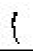
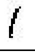
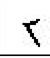
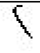
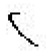
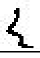
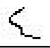
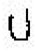
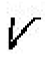
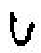
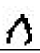
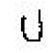
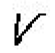
Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
٠											100
١		٤		٢		٩		٢			99.42
٢		٣		٣		٣		١		٣	98.56
		٣		٥		٣		١		٣	
٣		٧									99.86
٤		١		٢							99.72
٥		٩		٨							99.72
٦											100
٧		٥		٥		٥		٨		٥	99.31
٨		٦		٥		٥					99.59
٩											100

Table 31 shows detailed results of *SGDM* classifier. Only 19 images out of 7082 images were than 99%. Numeral “٠” and numeral “٤” achieved 100% recognition rate. The least recognition rate was for numeral "٢". It was misrecognized four times; two of them as "٣". This happened because of the way that number was written, which is sometimes can be misleading. Table 32 shows all the misrecognized images for Hindi classification using this setup. It is obvious from this table that some of the images were misrecognized due to unclear and misleading handwriting.

Table 31. Recognition rates for Hindi numerals classification using *SGDM* training algorithm.

Numeral	Number of Times Recognized As										Number of Misrecognized	Recognition Rate (%)
	٠	١	٢	٣	٤	٥	٦	٧	٨	٩		
٠	688	0	0	0	0	0	0	0	0	0	0	100
١	0	703	1	0	0	0	0	1	0	1	3	99.58
٢	0	1	694	2	0	1	0	0	0	0	4	99.43
٣	0	1	0	711	0	0	0	1	0	0	2	99.72
٤	0	0	0	0	709	0	0	0	0	0	0	100
٥	0	0	0	0	0	706	0	1	0	0	1	99.86
٦	0	0	0	0	0	0	715	0	0	1	1	99.86
٧	0	0	1	0	0	3	0	719	0	0	4	99.45
٨	0	0	0	0	0	2	0	0	716	1	3	99.58
٩	0	1	0	0	0	0	0	0	0	702	1	99.86
Total Number of Misrecognized											19	99.73
Total Number of Correctly Recognized											7063	

Table 32. Misrecognized images for Hindi numerals classification using *SGDM* training algorithm.

Numeral Value	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Missed Image	Recognized As	Recognition Rate (%)
٠											100
١		٩		٢		٧					99.58
٢		٣		٥		٣		١			99.43
٣		٧		١							99.72
٤											100
٥		٧									99.86
٦		٩									99.86
٧		٥		٥		٥		٢			99.45
٨		٩		٥		٥					99.58
٩		١									99.86

5. Time Requirements

To evaluate the proposed methodology performance in terms of time, the training time as well as the testing time were recorded. The machine that was used is an Intel Core-I7 computer with 16 Gb of RAM. It is obvious that *Adam* consumes more time than other optimization methods in terms of training. This was because *Adam* is a hybrid method of two other optimizers. In terms of testing times. Al classifiers show a very fast testing time per image. The recognition time was in the order of milliseconds; which is considered fast. Table 33 shows detailed time records.

Table 33. Average training time and average testing time.

Scenario	Training Time (Hour)	Average Testing Time Per Image (ms)
	Arabic/Hindi Classification	
Adam (Learning Rate = 0.0001)	72	185
Adam (Learning Rate = 0.001)	67	150
Adam (Learning Rate = 0.01)	64	105
RMSProp	37	111
SGDM	40	103
	Arabic Numerals Recognition	
Adam (Learning Rate = 0.0001)	65	110
Adam (Learning Rate = 0.001)	63	166
Adam (Learning Rate = 0.01)	62	187
RMSProp	50	174
SGDM	52	152
	Hindi Numerals Recognition	
Adam (Learning Rate = 0.0001)	62	203
Adam (Learning Rate = 0.001)	61	186
Adam (Learning Rate = 0.01)	58	109
SGDM	49	134
RMSProp	50	117

6. Discussion and Conclusion

Handwritten numerals recognition is an active research area which always needs an improvement in accuracy. The proposed model is a modified version from AOCT-Net [42] where the input size is 32 $\times$ 32, and the number of classes of the fully connected (FC) layer is 10, in order to make the proposed model more efficient and suitable to the data. AOCTNet [42] previously showed that it is very simple and efficient. A simpler model means requiring less resources for both training and testing stages, and it makes the model suitable to be designed in embedded systems. Comparing the used model with other models shows that our proposed model has higher accuracy and requires less execution time. The architecture that was used in our model is totally different from other types of architectures that have been used in literature. In [46], the authors built their own CNN with only six layers, three of them are convolutional layers, one fully connected layer, one softmax layer, and an output or classification layer. In [47], the authors utilized VGG19 [48] which is a pretrained CNN structure. It is composed from 19 levels (25 layers; 16 convolution layers, three fully connected layers, five MaxPool layers, and one SoftMax layer). They utilized it for deep features extraction. On the other hand, in [49] the authors used Deep Recurrent Convolutional Neural Network (DRCNN) [50] as a predictor model for bankruptcy, while in [51] the authors built their own CNN structure model based on Efficient Net architecture with nine levels (nine layers) to classify X-ray images into three classes (Normal, Pneumonia, and COVID-19). Table 34 shows the differences and the similarities between the proposed structure and the existing structures in literature.

Table 34. Comparison between the proposed method and previous work.

REF	Architecture	Goal	# of Layers	Accuracy
[42]	AOCT-Net	Classification of optical coherence tomography into five classes	5 levels (18 layers)	95.3%
[46]	New Structure of CNN	Classification of aerial images	6 levels (18 layers)	92.2%
[47]	VGG19 [48] is a pretrained CNN on ImageNet	Extracting deep features	19 levels (25 layers)	90%
[49]	Deep Recurrent Convolutional Neural Network existed in 2017 [50]	Prediction bankruptcy models	24 input nodes, 14 hidden nodes, and 2 output nodes	93.50%
[51]	Efficient Net	Classification of X-ray images (Normal, Pneumonia, or COVID-19)	9 levels (9 layers)	96.7%
Proposed	New CNN Model	Numerals classification (Hindi and Arabic formats) 10 classes	5 levels (18 layers)	~100%

This paper utilized the benefits of deep learning algorithms with various learning rate optimizations methods in discriminating of handwritten images between Hindi and Arabic numerals alongside recognizing the numerals to their 10 classes either in Hindi or in Arabic format. The purpose of employing deep learning was to take advantages of the power of CNNs that can manage input data with large dimensions and share their weights. The three scenarios (Arabic–Hindi classification, Hindi numerals recognition, and Arabic numerals recognition) have exploited various known optimization methods; *Adam*, *RMSprop*, and *SGDM*. *Adam* shows highly accurate recognition rate in all scenarios alongside requiring more time for processing than others. It was because *Adam* is providing a combined heuristics optimization of both Stochastic Gradient Descent with Momentum and Root Mean Square Propagation.

The proposed recognition approach has two advantages. First, the method is highly accurate, the results are close to 100% (recognition rate of Arabic numerals was slightly better than the recognition rate of Hindi numerals). Second, the proposed method exhibits high performance: the proposed approach performs recognition in only a few milliseconds. It was obvious from the previous results that some of the images were misrecognized due to unclear and misleading handwriting. This issue can be solved in the future by adding and augmenting more unclear and misleading data and training the system again. This system can be used in many applications such as in banks and exam grading.

Author Contributions: Conceptualization, A.A., A.M.A., H.A., H.R.A.-Z., M.A.-Q. and M.A.A.-K.; data curation, A.A., H.A., H.R.A.-Z. and M.A.A.-K.; methodology, A.A., A.M.A., H.A., H.R.A.-Z.; validation, A.A., A.M.A., H.A., H.R.A.-Z., M.A.-Q. and M.A.A.-K.; writing, A.A., A.M.A., H.A., H.R.A.-Z., M.A.-Q. and M.A.A.-K.; project administration, A.A. and M.A.-Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University, grant number 2017/01/7551.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This project was supported by the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University under the research project # 2017/01/7551.

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

References

1. Li, X.; Sun, Y.; Tang, M.; Yan, X.; Kang, Y. A Neural Network-Based Intelligent Image Target Identification Method and Its Performance Analysis. *Intell. Autom. Soft Comput.* **2011**, *17*, 885–896. [CrossRef]
2. El Hindi, K.; Khayyat, M.; Abu Kar, A. Comparing the Machine Ability to Recognize Hand-Written Hindu and Arabic Digits. *Intell. Autom. Soft Comput.* **2017**, *23*, 295–301. [CrossRef]
3. Stefano, C.D.; Iuliano, A.; Marcelli, A. Shape-Based Algorithm for Detecting Ligatures in On-Line Handwriting. *Intell. Autom. Soft Comput.* **2011**, *7*, 187–194. [CrossRef]
4. Sánchez, E.G.; Dimitriadis, Y.A.; Mas, M.S.R.; García, P.S.; Izquierdo, J.C.; Coronado, J.L. On-Line Character Analysis and Recognition With Fuzzy Neural Networks. *Intell. Autom. Soft Comput.* **2001**, *7*, 163–175. [CrossRef]
5. Abirami, S.; Murugappan, S. Scripts and Numerals Identification from Printed Multilingual Document Images. *Comput. Sci. Inform. Technol.* **2011**, *1*, 129–146.
6. Ahmed, F.; Moskowitz, S. Correlation based watermarking method for image authentication applications. *Opt. Eng.* **2004**, *43*, 1833–1838.
7. Alhoniemi, E. Unsupervised Pattern Recognition Methods for Exploratory Analysis of Industrial Process Data. Ph.D. Thesis, Helsinki University of Technology, Espoo, Finland, 2002.
8. Al-Omari, F.; Al-Jarrah, O. Handwritten Indian Numerals Recognition System Using Probabilis-tic Neural Networks. *Adv. Eng. Inform.* **2004**, *18*, 9–16. [CrossRef]
9. Alqudah, A.; Al-Zoubi, H. Efficient k-Class Approach for Face Recognition. *Comput. Electr. Eng.* **2015**, *45*, 260–273. [CrossRef]
10. Alqudah, A.; Al-Zoubi, H.; Al-Khassaweneh, M. Shift and Scale Invariant Recognition of Printed Numerals. *J. Abhath Al-Yarmouk Basic Sci. Eng.* **2012**, *21*, 41–49.
11. Al-Zoubi, H.; Al-Khassaweneh, M.; Alqudah, A. Precise and Accurate Decimal Number Recognition Using Global Motion Estimation. *Int. J. Artif. Intell. Soft Comput.* **2011**, *2*, 287–301. [CrossRef]
12. Sabri, A.; Marwan, M.; Abu-Amara, H. Recognition of handwritten Arabic (Indian) Numerals Using Radon-Fourier-based Features. In Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA'10), Cambridge, UK, 20–22 February 2010; pp. 158–163.
13. Choudhary, A.; Rishi, R.; Ahlawat, S. Handwritten Numeral Recognition Using Modified BP ANN Structure. In *Advanced Computing. CCSIT 2011. Communications in Computer and Information Science*; Meghanathan, N., Kaushik, B.K., Nagamalai, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 133.
14. Jindal, T.; Bhattacharya, U. Recognition of offline handwritten numerals using an ensemble of MLPs combined by Adaboost. In Proceedings of the 4th International Workshop on Multilingual OCR, Washington, DC, USA, 24 August 2013.
15. Kulkarni, P.H.; Kute, P.D. Optical numeral recognition algorithm for seven segment display. In Proceedings of the 2016 Conference on Advances in Signal Processing (CASP), Pune, India, 9–11 June 2016; pp. 397–401.
16. Zinjore, R.S.; Ramteke, R.J. Recognition of handwritten bilingual Characters-Numerals using shape context. In Proceedings of the 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Pune, India, 19–21 December 2016; pp. 265–268.
17. Ashiquzzaman, A.; Tushar, A.K. Handwritten Arabic numeral recognition using deep learning neural networks. In Proceedings of the 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, Bangladesh, 13–14 February 2017; pp. 1–4.

18. Xie, Y.; Liu, Y. A handwritten numeral recognition method based on STDP based with unsupervised learning. In Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 22–24 March 2017; pp. 839–842.
19. Latif, G.; Alghazo, J.; Alzubaidi, L.; Naseer, M.M.; Alghazo, Y.M. Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals. In Proceedings of the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR), London, UK, 12–14 March 2018; pp. 90–95.
20. Thomas, S. A Study of Representation Learning for Handwritten Numeral Recognition of Multilin-gual Data Set. In *Information and Communication Technology for Sustainable Development. Lecture Notes in Networks and Systems*; Mishra, D., Nayak, M., Joshi, A., Eds.; Springer: Singapore, 2018; Volume 10.
21. Kumar, M.; Jindal, M.K.; Sharma, R.K.; Jindal, S.R. Character and numeral recognition for non-Indic and Indic scripts: A survey. *Artif. Intell. Rev.* **2019**, *52*, 2235–2261. [CrossRef]
22. Aly, S.; Mohamed, A. Unknown-Length Handwritten Numeral String Recognition Using Cascade of PCA-SVMNet Classifiers. *IEEE Access* **2019**, *7*, 52024–52034. [CrossRef]
23. Parekh, K.A.; Goswami, M.M.; Mitra, S.K. Handwritten Numeral Recognition Using Polar Histogram of Low-Level Stroke Features. In *Proceedings of 3rd International Conference on Computer Vision and Image Processing*; Advances in Intelligent Systems and Computing; Chaudhuri, B., Nakagawa, M., Khanna, P., Kumar, S., Eds.; Springer: Singapore, 2020; Volume 1022.
24. Hatcher, W.G.; Yu, W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* **2018**, *6*, 24411–24432. [CrossRef]
25. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, *7*, 53040–53065.
26. Guo, W.; Wang, J.; Wang, S. Deep Multimodal Representation Learning: A Survey. *IEEE Access* **2019**, *7*, 63373–63394. [CrossRef]
27. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A Survey of Deep Learning-Based Object Detection. *IEEE Access* **2019**, *7*, 128837–128868. [CrossRef]
28. Jiao, L.; Zhao, J. A Survey on the New Generation of Deep Learning in Image Processing. *IEEE Access* **2019**, *7*, 172231–172263. [CrossRef]
29. Alqudah, A.; Al-Zoubi, H.R.; Al-Khassaweneh, M.A.; Al-Qodah, M. Highly Accurate Recognition of Handwritten Arabic Decimal Numbers Based on a Self-Organizing Maps Ap-proach. *Intell. Autom. Soft Comput.* **2018**, *24*, 493–505. [CrossRef]
30. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Van Esesn, B.C.; Awwal, A.A.S.; Asari, V.K. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv* **2018**, arXiv:1803.01164.
31. Aman, D.; Pahwa, P. Image retrieval techniques: A survey. *Int. J. Eng. Technol.* **2018**, *7*, 215–219.
32. Keiron, O.; Nash, R. An introduction to convolutional neural networks. *arXiv preprint* **2015**, arXiv:1511.08458.
33. John, M. *An Overview of Convolutional Neural Network Architectures for Deep Learning*; Microway, Inc.: Plymouth, MA, USA, 2016.
34. Chartrand, G.; Cheng, P.M.; Vorontsov, E.; Drozdzal, M.; Turcotte, S.; Pal, C.J.; Kadoury, S.; Tang, A. Deep learning: A primer for radiologists. *Radiographics* **2017**, *37*, 2113–2131. [CrossRef]
35. Lee, C.Y.; Gallagher, P.W.; Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *Artif. Intel. Stat.* **2016**, *51*, 464–472.
36. Christopher, M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
37. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint* **2018**, arXiv:1811.03378.
38. Alqudah, A. Brain Tumor Classification Using Deep Learning Technique—A Comparison between Cropped, Uncropped, and Segmented Lesion Images with Different Sizes. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 3684–3691. [CrossRef]
39. Andrea, V.; Lenc, K. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM International Conference on Multimedia*; ACM: New York, NY, USA, 2015.
40. Fabian, S. *The Effect of Batch Normalization on Deep Convolutional Neural Networks*; KTH Royal Institute of Technology, School of Computer Science and Communication: Stockholm, Sweden, 2016.
41. Bushaev, V. Adam—Latest Trends in Deep Learning Optimization. *Towards Data Sci. Listopad*. Available online: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (accessed on 28 June 2018).
42. Alqudah, A.M. AOCT-NET: A convolutional network automated classification of multiclass retinal diseases using spectral-domain optical coherence tomography images. *Med. Biol. Eng. Comput.* **2020**, *58*, 41–53. [CrossRef]
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Christian, R. Machine Learning, A Probabilistic Perspective. *Chance* **2014**, *27*, 62–63. [CrossRef]
45. Jiawei, Z. Gradient descent-based optimization algorithms for deep learning models training. *arXiv* **2019**, arXiv:1903.03614.
46. Mahajan, P.; Abrol, P.; Lehana, P.K. Scene based Classification of Aerial Images using Convolution Neural Networks. *J. Sci. Ind. Res.* **2020**, *79*, 1087–1094.
47. Ziegelmayer, S.; Kaissis, G.; Harder, F.; Jungmann, F.; Müller, T.; Makowski, M.; Braren, R. Deep Convo-lutional Neural Network-Assisted Feature Extraction for Diagnostic Discrimination and Feature Visualization in Pancreatic Ductal Adenocarcinoma (PDAC) versus Autoimmune Pancreatitis (AIP). *J. Clin. Med.* **2020**, *9*, 4013. [CrossRef]
48. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
49. Becerra-Vicario, R.; Alaminos, D.; Aranda, E.; Fernández-Gámez, M.A. Deep Recurrent Convolutional Neural Network for Bankruptcy Prediction: A Case of the Restaurant Industry. *Sustainability* **2020**, *12*, 5180.

50. Huang, C.-W.; Narayanan, S.S. Deep convolutional recurrent neural network with attention mechanism for robust speech emotion recognition. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo, Hong Kong, China, 10–14 July 2017; pp. 583–588.
51. Marques, G.; Agarwal, D.; de la Torre Díez, I. Automated medical diagnosis of COVID-19 through Efficient Net convolutional neural network. *Appl. Soft Comput.* **2020**, *96*, 106691. [CrossRef]