J.Natn.Sci.Foundation Sri Lanka 2021 49 (4): 503 - 511
DOI: http://dx.doi.org/10.4038/jnsfsr.v49i4.9825

RESEARCH ARTICLE

Recognition of handwritten characters using deep convolution neural network

S Arivazhagan[1], M Arun[1*] and D Rathina[2]
[1] *Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, Sivakasi, India.*
[2] *Department of Electronics and Communication Engineering, Renganayagi Varatharaj College of Engineering, Sivakasi, India.*

Submitted: 24 March 2020; Revised: 18 January 2021; Accepted: 23 April 2021

Abstract: Character recognition is a very interesting technique in the field of pattern recognition. Specifically, handwritten character recognition is gaining the attention of researchers as it is necessary for historical documents, archives, or mass digitization of hand-filled forms. The correct classification of handwritten characters is really a challenging task due to its variability in the writing styles of an individual at different times and among different individuals such as size, shape, speed of writing and thickness of characters, etc. To solve this challenging task, the features extracted from the characters should be suitable for the variability of the characters. In this research work, Deep Convolution Neural Network (DCNN) has been used instead of hand-crafted features from the handwritten characters, to automatically learn the best features for this task. The proposed DCNN framework is trained and tested on the Chars74K handwritten dataset in all the aspects of handwritten numbers and handwritten English alphabets with various training and testing proportions and various subproblems. The recognition rate for the proposed DCNN provides better results when compared with the other schemes. The recognition rates for 62 classes in the Chars74K dataset with 50:50, 70:30 and 80:20 train test ratio is 88.05 %, 89.21 % and 90.32 %, respectively.

Keywords: Character recognition, Chars74K, CNN, deep learning, handwritten characters.

INTRODUCTION

The human brain can store trillions of bytes of information. With time, the memory in human brain gets faded, so, written information is very important for the memory retrieval process. The handwritten information is very useful in both private (letters, notes, addresses, reminders, lists, diaries, etc.) and official correspondence (bank cheques, tax forms, admission forms, historical documents, etc.). Hence, there is a dire need to digitize all these paper documents, enabling people to easily search for necessary information and get access to the important sources of knowledge written in papers. To digitize these documents, the image of handwritten text is captured using a digital camera, pre-processed, segmented into individual characters and finally are recognized by a classifier. This entire process is known as Handwritten Character Recognition (HCR), which interprets the handwritten characters in digitized form which can be used for applications such as document verification, digital library, reading bank deposit slips, cheque interpretation, data entry of application, loan, and tax forms. HCR contributes to the advancement of automation process by reducing the human work (Attigeri, 2018). In the field of Pattern Recognition, HCR is a challenging part due to its un-deterministic characteristics. The handwritten characters differ in size and style from person to person and they differ in the same person due to ageing factor. HCR is of two types depending on the handwritten input source, namely, online handwritten character recognition and offline handwritten character recognition. Online handwritten character recognition is the process of interpreting the

* Corresponding author (m_arun@outlook.com; https://orcid.org/0000-0002-7860-8230)

handwritten characters as it is written on the touch screen by a digital pen. Here, recognizing the characters involves finding the position of the pen tip movements and the up/down pen strokes. The information is extracted from the strokes and the structures of the handwritten character, and the character is recognized instantly. Offline character recognition is a process of interpreting the handwritten characters from the scanned image of the handwritten document. The handwritten character in the image is subjected to feature extraction based on shape, orientation, and other transformable features. The offline HCR is quite complex than the online HCR because the information of pen tip movement is not available. The only way to recognize offline handwritten images is to extract the features which are more suitable for character recognition. In this research work, we have studied the effect of Deep Convolution Neural Network (DCNN) in the field of HCR.

Many researchers have worked on character recognition for the past few decades. The performance for character recognition using the multi-layer perceptron classifier is compared with the convolutional neural network (Driss *et al*., 2017). Handwritten digits recognition by convolutional neural networks and with traditional features extraction (Hu invariant moments, Fourier descriptors, projections histograms, horizontal cell projections, local line fitting and zoning) and classification techniques (k-NN, Mahalanobis distance and support vector machines) is done and compared (Enriquez *et al*., 2019). A novel feature extraction method known as Deep Contextual Stroke Pooling (DCSP) has been introduced for Scene Character Recognition (Zhang *et al*., 2018). An adaptive deep Q-learning strategy for handwritten digit recognition was proposed by Qiao *et al*. (2018). This Q-learning strategy merges the feature extracting capability of deep learning and the decision-making of reinforcement learning to form a deep belief network. A new data augmentation method and directional feature maps using CNNs have been introduced for handwritten Chinese character recognition (Qu *et al*., 2018).

CNN architecture for outputting the arbitrary length symbol streams from the handwritten text was presented by Ptucha *et al*. (2019). A new method for combining position embeddings with residual networks (ResNets) and bidirectional long short-term memory (BiLSTM) networks for unconstrained offline handwritten word recognition was proposed by Wu *et al*. (2019). A feature extraction technique using non-redundant Stockwell transform for handwritten digit recognition of Odia language was introduced (Dash *et al*., 2015). Another transformed domain feature extraction using Slantlet coefficients is also proposed by the same researchers. A DIGI-Net CNN for recognizing digits, which has the ability to learn common features from three different formats (handwritten, natural images, printed font) was designed (Madakannu & Selvaraj, 2019). A shared-hidden-layer deep convolutional neural network (SHL-CNN) for image character recognition for different languages was designed (Bai *et al*., 2014). A convolutional network for handwritten character recognition based on subspace method was designed by Gatto *et al*. (2017). A machine learning model for recognizing handwritten characters on form document based on CNN for feature extraction and support vector machines (SVM) for classification was proposed (Darmatasia, 2017). A method called DropSample, a new training method for enhancing the deep convolutional neural networks for large-scale unconstrained handwritten Chinese character recognition was introduced by Yang *et al*. (2016). Recently, Sinhala handwritten character recognition was done by convolutional neural networks (Mariyathas *et al*., 2020). Considering the popularity of the Convolutional Neural Network, in this research work, a new DCNN architecture is proposed for recognizing the handwritten characters rather than using famous architectures like ResNet, VGGNet and GoogleNet, etc. because of their computational complexity and memory requirement. Also, the proposed architecture is a full sparse convolutional, as it does not have any fully connected layers except the output layer.

METHODOLOGY

The handwritten dataset should have diverse writing styles by numerous authors in order to design a good character recognition model. This research work used the Chars74K dataset (*http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/*) for recognizing the individual handwritten characters. This dataset has handwritten images which are drawn using a tablet PC. It consists of 3410 handwritten images, which includes 62 classes (0-9, A-Z, a-z); each class consists of 55 images. Apart from the handwritten characters, the dataset also has 7705 natural images and 62992 images of synthetic fonts used in the computer, making the total number of images in the dataset as 74000 (74K). The sample handwritten images from the Chars74K dataset are shown in the Figure 1.

Figure 1: Sample handwritten images of the chars 74K image dataset

The block diagram of the proposed method is shown in the Figure 2. The first block in the proposed methodology is the pre-processing stage. The pre-processing stage in this research work is a two-step process. The first step in the pre-processing stage is the binarization, which converts the RGB image to a binary image. As it is simple to process the low-level colour, the input image is binarized by means of the Otsu algorithm (Haseena & Clara, 2017). The second process in the pre-processing stage is to get the interested portion from the dataset image. The handwritten character images in the Chars74k dataset have a wide background area more than the Region of Interest (ROI). The bounding box approach is used to extract the ROI from the handwritten character images. The excess background is removed based on the connectivity of the binary image. The image is cropped based on the bounding box and the ROI is obtained.

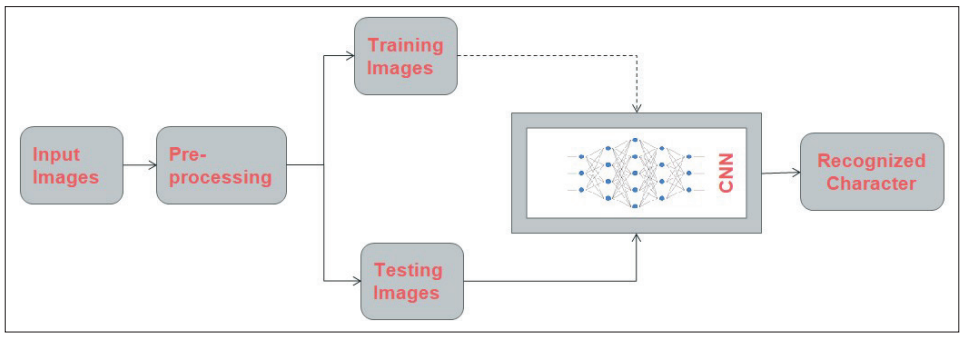

Figure 2: Block diagram of proposed methodology

After the pre-processing stage, the images were split into different train and test ratios such as 80:20, 70:30 and 50:50. Then the images are trained and tested using the proposed DCNN architecture. The proposed DCNN architecture has seven (07) stages of consecutive hidden layers between the input layer and the fully connected layer. The architecture of the proposed DCNN is shown in Figure 3.

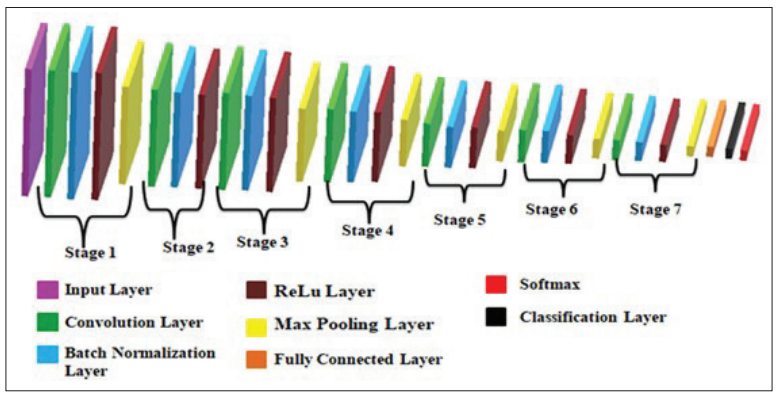

Figure 3: Architecture of proposed deep convolution neural network

The proposed DCNN architecture has 7 hidden layers, which are the convolution layers, along with batch normalization and activation functions (ReLU). Between every hidden layer, there is a pooling operation, which takes place except in the 2nd hidden layer.

Convolution layer is the heart of the CNNs. They are responsible for the automatic learning of the features from the given input image. This layer applies the filter to the input to create a feature map that encapsulates the occurrence of the detected features in the input. The filters in the neural networks are handcrafted, but the idea of CNN is to learn the filters during the training stage in the context of a specific application. The convolution operation is shown in Figure 4.

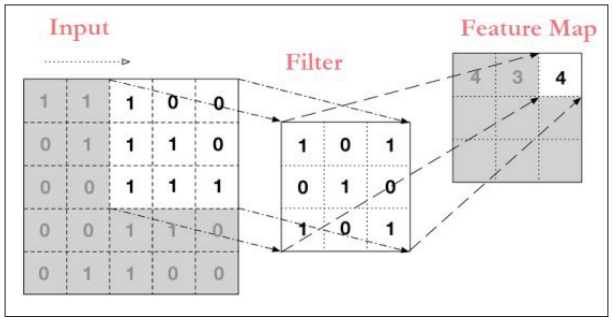

Figure 4: Convolution operation in convolution neural network

The size of the feature map after the convolution operation is given by equation 1.

$$O_f = \left[\frac{I+2P-F_c}{S}\right] + 1 \qquad \ldots (1)$$

where O_f is the size of the feature map, I is the size of the input to the convolution layer, P and S stands for length of the padding and stride and F_c is the size of the convolution filter.

Training deep neural networks is a challenging task. The CNN is updated layer-by-layer backward during the back propagation from the output to the input using the estimate of error that assumes the weights in the layers prior to the current layer are fixed. As all layers are getting updated simultaneously during an update, the update procedure will be trying to chase a moving target. Batch normalization is the technique to coordinate the update of multiple layers in the CNN. The batch normalization can be applied after or before the activation function. For ReLU activation function, it is identified that batch normalization should be used before the activation function. Batch normalization provides a well-designed way of re-parameterizing any deep network. The re-parameterization drastically reduces the problem of coordinating updates across many layers. Standardizing the activations of the prior layer makes spread and distribution of inputs of the subsequent layer does not change during the weight update, at least not dramatically. This effects stabilizing and speeding-up the training process of deep neural networks.

Activation functions are really important in a Convolution Neural Network (CNN) to identify the non-linear complex functional mappings between the input and output. If the activation function has not been added, the CNN would simply be a linear regression model of polynomial function of one degree. Therefore, the hidden layers in the CNNs are added with an activation function such as tanh, ReLU or sigmoid, etc. In this research work, we have used ReLU as the activation function as it avoids and rectifies vanishing gradient problem. The only limitation of this ReLU activation function is that it can be used only in the hidden layers. Therefore, we have used Softmax activation function in the output layer as we are dealing with a multiclass classification problem.

A limitation of the feature map output of convolution layer is that the feature maps trace the accurate location of features in the input making the feature map translation variant. Therefore, small movements in the location of the feature in the input will result in a different feature map. To avoid this situation, pooling layer is introduced for down-sampling. A pooling layer is added mostly after the convolution layer. Even though more pooling operations are available, in this research work we have used Max Pooling in the DCNN. We have added 6 pooling layers between the 7 hidden layers and randomly eliminated a pooling layer between the 2nd and 3rd hidden layer.

After the sequence of 7 stages, the fully connected output layer is stacked to the DCNN. Neurons present in this layer are connected to all the activations of hidden layers in the architecture. This layer flattens the two-dimensional vectors into one-dimensional features. The final output from this layer consists of N × 1 dimensions, where N denotes the number of classes. Then the decision-making action is taken by the Softmax layer. Table 1 specifies the parameters of the proposed DCNN in every layer.

Table 1: Parameters of the proposed DCNN architecture in every layer

Layer	Filter size	No. of feature maps	Input size	Output size
1-Conv	3 × 3	128	64 × 64	64 × 64
MaxPooling	2 × 2	128	64 × 64	32 × 32
2-Conv	3 × 3	128	32 × 32	32 × 32
3- Conv	3 × 3	128	32 × 32	32 × 32
MaxPooling	2 × 2	128	32 × 32	16 ×16
4-Conv	3 × 3	128	16 × 16	16 ×16
MaxPooling	2 × 2	128	16 × 16	8 × 8
5-Conv	3 × 3	128	8 × 8	8 × 8
MaxPooling	2 × 2	128	8 × 8	4 × 4
6-Conv	3 × 3	128	4 × 4	4 × 4
MaxPooling	2 × 2	128	4 × 4	2 × 2
7-Conv	3 × 3	128	2 × 2	2 × 2
MaxPooling	2 × 2	128	2 × 2	1 × 1

The training process involves the back propagation technique, which adjusts the weight value depending on the previous trained validation. The back propagation does the optimization process, which involves forward pass, loss function, background pass and weight update. The forward pass is used to interchange the input and the output values. The loss function, which is used for the optimization is mean square error (MSE) between the predicted and the target one. The backward pass is defined by the weight update value that gives the least loss measure.

RESULTS AND DISCUSSION

The entire research work was done on the Chars74K handwritten dataset using the proposed DCNN architecture, and was carried out by MATLAB R2018a software in an Intel core i5 Processor @3.2 GHz with 64 bit OS.

First the experiment was carried out for finding the optimum number of epochs. The experiment was done on the handwritten characters of the Chars74K dataset with different train and test ratios. Also, the experiment was carried out as 4 different subproblems. The first subproblem considers only the numerals of the dataset (10 class problem). The numerals 0–9 is considered and only 550 images out of 3410 images were used. The second subproblem considers only the uppercase alphabets (A-Z), making it a 26-class problem. The third subproblem considers only the lowercase alphabets (a-z), making it as another 26-class problem. Finally, all the alphanumeric handwritten characters (0-9, A-Z, a-z) were considered, and made into a 62-class recognition problem. The experiment was carried out for 80:20, 70:30 and 50:50 train test ratios for different epochs and the recognition rates are tabulated in Table 2.

It is evident from Table 2 that 20 epochs give the best recognition rate. When the number of epochs increased, we observed the problem of overfit; also, the computational time is increased with the increase of the epochs. Therefore, further experimentation is carried out with 20 epochs. Now the experiment was done to find the optimum filter size for the convolution layer. Filter size is also an important factor for extracting the features. Most of the best features are usually local in nature, so a convolution filter capturing the local features will make sense. To get the useful feature of the image, a single filter slides all over the height and width of the image. In this research work, it is decided to keep the filter sizes common across all the seven hidden convolution layers. To find the optimum filter size, the experiment was carried out with several filter sizes such as 3×3, 5×5, 7×7, 9×9 and its recognition rate is tabulated in Table 2.

Table 2: Recognition rates for different epochs using the proposed DCNN

No. of epochs	Train test ratio (%)	Recognition rate (%)			
		Numerals (0-9)	Uppercase alphabets (A-Z)	Lowercase alphabets (A-Z)	Alphanumeric (0-9, A-Z, a-z)
20	50:50	98.52	98.72	94.02	88.05
	70:30	100	99.04	96.15	89.21
	80:20	100	97.90	93.36	90.32
50	50:50	98.52	98.01	92.88	87.69
	70:30	99.38	98.56	94.95	88.91
	80:20	70	98.95	93.71	89.15
100	50:50	97.41	98.43	93.30	87.22
	70:30	98.75	97.60	93.75	87.30
	80:20	76.36	96.85	94.41	88.56

Table 3: Recognition rates for different filter sizes using the proposed DCNN

Filter size	Train test ratio (%)	Recognition rate (%)			
		Numerals (0-9)	Uppercase alphabets (A-Z)	Lowercase alphabets (A-Z)	Alphanumeric (0-9, A-Z, a-z)
3 × 3	50–50%	98.52	98.72	94.02	88.05
	70–30%	100	99.04	96.15	89.21
	80–20%	100	97.90	93.36	90.32
5 × 5	50–50%	99.63	96.44	92.17	85.90
	70–30%	96.88	95.91	93.75	87.90
	80–20%	99.09	98.60	96.15	88.56
7 × 7	50–50%	94.81	96.44	90.17	84.11
	70–30%	100	95.67	92.55	85.08
	80–20%	100	95.45	94.06	88.71
9 × 9	50–50%	94.81	96.44	88.46	82.20
	70–30%	95	95.91	93.51	85.99
	80–20%	98.18	96.85	92.31	85.45

While using different filter sizes, the padding was adjusted so that it does not affect the dimensions of the architecture described in Table 1. For the filter sizes 3×3, 5×5, 7×7 and 9×9, the padding sizes used are 1, 2, 3, and 4, respectively. Again, the same four subproblems were carried out using different filter sizes with adjusted padding. The recognition rate for the different filter sizes for the subproblems is listed in Table 3.

It is not an easy task to pick the optimum filter size from Table 3. For an example, 5×5 filter size is better than 3×3 filter size for 50:50 train test ratio in numeral subproblem. But for other test ratios, 3×3 filter size performs better. The same was observed with lowercase alphabets also. Filter sizes 7×7 and 9×9 provided more or less comparable results among each other. But for the subproblem of recognizing the alphanumeric (62 class)

characters, without any doubt, the filter size 3× 3 performs better. Therefore, it was observed that lower filter size extracts more local features than the higher filter size. In larger filter size, it was observed that weight sharing is poor compared to the smaller filter size because of the padding size. Further experiments were carried out using the filter size 3×3.

Learning rate, learning rate schedule, L2 regularization parameter, learning rate drop factor, drop period and batch size are the parameters, which are fixed as per the values given in Table 4.

Table 4: Fixed values of the options for the proposed DCNN

S. no.	Options	Value
	Learning rate	0.001
	Learning rate schedule	piecewise
	L2 regularization parameter	0.0001
	Learn rate drop factor	0.1
	Drop period	8 epochs
	Batch size	4

Learning rate (Brownlee, 2019) is defined as the amount that the weights are updated during training. The learning rate is a tuneable parameter that has a small positive value, often in the range between 0.0 and 1.0. It is cited as a positive scalar determining the size of the step. During training phase of the DCNN, the back propagation of error estimates the amount of error for which the weights of a node in the network are responsible. Instead of updating the weight with the full amount, it is scaled by the learning rate. For example, if the learning rate is 0.001, it means that weights in the network are updated by 0.1% of the estimated weighted error whenever the weights are updated. Similarly, large weights in a DCNN lead to a more complex network that has overfitting. Penalizing the DCNN based on the size of the network weights during training can reduce overfitting, so that an L1 or L2 vector norm penalty is added to the optimization of the network to encourage smaller weights (Brownlee, 2018)

The pictorial representation of the confusion matrix for the recognition of the alpha numeric (62 classes) of the proposed DCNN with 20 epochs, 3×3 filter size, 0.001 as learning rate and 0.0001 as L2 regularization parameter is shown in Figure 5.

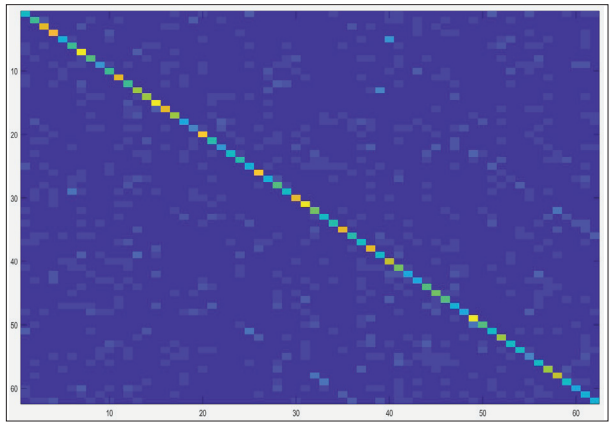

Figure 5: The pictorial plot of the confusion matrix

From the pictorial plot, it is noted that 37 % of the testing images of numeric character 0 is wrongly predicted as the uppercase alphabet 'O', and 14 % of the numeric character 0 is wrongly predicted as the lowercase alphabet 'o'. Only 48 % test images of numeric character 0 is predicted as numeric zero. Another 1 % is predicted as other character than 'O' or 'o'. In similar manner, only 62 % lower case alphabet 'o' is correctly predicted. Therefore, to resolve this problem, a different subproblem (36 class problem considering case insensitiveness in alphabets and numerals) was formulated and the training was done with the above DCNN architecture. Now, the experimentation is done in a case insensitive method and the recognition rate is tabulated in the Table 5.

Table 5: Recognition rate for the case insensitive problem using the proposed DCNN

Train test ratio (%)	Recognition rate (%)	
	Alphabets (case insensitive) + numerals (A-Z & a-z)	Alphabets (case insensitive) + numerals (A-Z & a-z, 0-9)
50:50	94.55	90.53
70:30	94.99	94.20
80:20	95.97	94.43

The recognition of the model improves when we consider case insensitiveness, as 'O' and 'o' have the same features. The system could now train wisely with

the features automatically learnt by the proposed DCNN architecture. The recognition rate of the handwritten character recognition (62 class problem) of the proposed DCNN is compared with other available methods in Table 6. The aggregation table shows that the proposed DCNN architecture performs well compared with the other schemes. From the table, we also inferred that the machine-tuned features have better recognition results than man-tuned features.

Table 6: Recognition rate comparison of state-of-art methods on the Chars74K dataset with 62 classes

State-of-art methods	Recognition rate (%)
GoogleNet (Soomro *et al*., 2017)	88.89
Alex Net (Soomro *et al*., 2017)	77.77
Multiscale HoG Features (Newell & Griffin, 2011)	80.00
ConvNet (Sundaresan & Lin, 2015)	71.69
Proposed DCNN architecture	90.32

CONCLUSION

A deep convolution neural network (DCNN) was designed for classifying handwritten characters and the proposed DCNN architecture achieved better recognition rate compared with the state-of-art methods. Detailed experimentation was conducted for tuning the parameter of the filter size and found that smaller filter size would give a better recognition rate. It was observed that the smaller filter size provides useful features, and it is computationally effective. The recognition of handwritten characters using the proposed DCNN architecture is well suitable for the Chars74K dataset. Therefore, this model can be used for digitizing bank deposit slips, cheque interpretations, and data entry of applications, loan and tax forms when combined with a good segmentation method.

Conflict of interest

The authors declare that there is no conflict of interest.

REFERENCES

Attigeri S. (2018). Neural network based handwritten character recognition system. *International Journal of Engineering and Computer Science* **7**(3): 23761–23768.
DOI: https://doi.org/10.18535/ijecs/v7i3.18

Bai J., Chen Z., Feng B. & Xu B. (2014). Image character recognition using deep convolutional neural network learned from different languages, *Proceedings of 2014 IEEE International Conference on Image Processing (ICIP)*, October 27–30, Paris, France, pp. 2560–2564.

Brownlee J. (2018). Use weight regularization to reduce overfitting of deep learning models. Available at *https://machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/*

Brownlee J. (2019). How to configure the learning rate when training deep learning neural networks. Available at *https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/*

Darmatasia & Fanany M.I. (2017). Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM), *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, Malacca City, pp. 1–6.
DOI: https://doi.org/10.1109/ICoICT.2017.8074699

Dash K.S., Puhan N.B. & Panda G. (2015). Handwritten numeral recognition using non-redundant Stockwell transform and bio-inspired optimal zoning. *IET Image Processing* **9**(10): 874–882.
DOI : https://doi.org/10.1049/iet-ipr.2015.0146

Driss S.B., Soua M., Kachouri R. & Akil M. (2017). A comparison study between MLP and convolutional neural network models for character recognition, *Proceedings of SPIE 10223, Real-Time Image and Video Processing* **2017**: 1022306.
DOI: https://doi.org/10.1117/12.2262589

Enriquez E.A., Gordillo N., Bergasa L.M., Romera E. & Hu´elamo C.G. (2019). Convolutional neural network vs traditional methods for offline recognition of handwritten digits. In: *Advances in Physical Agents*. *WAF 2018. Advances in Intelligent Systems and Computing* (eds. R.F. Pizán, Á.G. Olaya, M.S. Lorente, J.I. Martínez, A.L. Espino), volume 855. Springer, Switzerland.
DOI: https://doi.org/10.1007/978-3-319-99885-5_7

Gatto B.B., Santos E.M. dos & Fukui K. (2017). Subspace-based convolutional network for handwritten character recognition, *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto, Japan, pp. 1044–1049.
DOI: https://doi.org/10.1109/ICDAR.2017.173

Haseena M. & Clara A.R. (2017). A review on an efficient iterative thinning algorithm. *International Journal of Innovative Research in Science, Engineering and Technology* **6**(11): 541–548.

Madakannu A. & Selvaraj A. (2020). DIGI-Net: a deep convolutional neural network for multi-format digit recognition. *Neural Computing and Applications* **32**: 11373–11383.
DOI: https://doi.org/10.1007/s00521-019-04632-9

Mariyathas J., Shanmuganathan V. & Kuhaneswaran B. (2020). Sinhala handwritten character recognition using convolutional neural network, *5th International Conference*

on Information Technology Research (ICITR), Moratuwa, Sri Lanka, pp. 1–6.
DOI: https://doi.org/10.1109/ICITR51448.2020.9310914

Newell A.J. & Griffin L.D. (2011). Multiscale histogram of oriented gradient descriptors for robust character recognition, *2011 International Conference on Document Analysis and Recognition*, Beijing, China, pp. 1085–1089.
DOI: https://doi.org/10.1109/ICDAR.2011.219

Ptucha R., Such F.P., Pillai S., Brockler F., Singh V. & Hutkowski P. (2019). Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition* **88**: 604–613.
DOI: https://doi.org/10.1016/j.patcog.2018.12.017

Qiao J., Wang G., Li W. & Chen M. (2018). An adaptive deep Q-learning strategy for handwritten digit recognition. *Neural Networks* **107**: 61–71.
DOI: https://doi.org/10.1016/j.neunet.2018.02.010

Qu X., Wang W., Lu K. & Zhou J. (2018). Data augmentation and directional feature maps extraction for in-air handwritten Chinese character recognition based on convolutional neural network. *Pattern Recognition Letters* **111**: 9–15.
DOI: https://doi.org/10.1016/j.patrec.2018.04.001

Soomro M., Farooq M.A. & Raza R.H. (2017). Performance evaluation of advanced deep learning architectures for offline handwritten character recognition, *2017 International Conference on Frontiers of Information Technology (FIT)*, Islamabad, Pakistan, pp. 362–367.
DOI: https://doi.org/10.1109/FIT.2017.00071

Sundaresan V. & Lin J. (2015). Recognizing handwritten digits and characters. Available at *http://cs231n.stanford.edu/reports/2015/pdfs/vishnu_final.pdf*

Wu X., Chen Q., You J. & Xiao Y. (2019). Unconstrained offline handwritten word recognition by position embedding integrated ResNets Model. *IEEE Signal Processing Letters* **26**(4): 597–601.
DOI: https://doi.org/10.1109/LSP.2019.2895967

Yang W., Jin L., Tao D., Xie Z. & Feng Z. (2016). DropSample: a new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten Chinese character recognition. *Pattern Recognition* **58**: 190–203.
DOI: https://doi.org/10.1016/j.patcog.2016.04.007

Zhang Z., Wang H., Liu S. & Xiao B. (2018). Deep contextual stroke pooling for scene character recognition. *IEEE Access* **6**: 6454–16463.
DOI: https://doi.org/10.1109/ACCESS.2018.2817342