Journal of Machine Learning Research 1 (2021) 1-48 Submitted 4/00; Published 10/00

Handwritten Bangla Letter Recognition Using Deep Convolutional Neural Network

Pritom Kumar Das bsse0919@iit.du.ac.bd
Department of Software Engineering
University of Dhaka

Ishrat Jahan Emu bsse0927@iit.du.ac.bd
Department of Software Engineering
University of Dhaka

Nazifa Tasnim Hia bsse0930@iit.du.ac.bd
Department of Software Engineering
University of Dhaka

Dr. B M Mainul Hossain mainul@du.ac.bd
Associate Professor
Department of Software Engineering
University of Dhaka

Abstract

Handwritten letter recognition in any language is always a problematic and time-consuming task. Depending on the number of letters we may have more than one thousand classes. For the Bengali language, it is even more difficult because it is highly stylized, morphologically complex, and comprises potentially juxtapositional letters. Even though it has more than 230 million users worldwide and is the fifth most spoken language in the world, the improvements over the years in Bengali letter recognition are significantly less as compared to the other languages. In this paper, we investigate the performance of recognization of Bengali handwritten letters using some deep convolutional neural network (DCNN) models. We use the state-of-the-art pre-trained neural network models to learn about some special characteristics of the letters and use densely connected layers for the discrimination task. We trained our system on the CMATERdb dataset and test on the Ekush dataset. By using two different datasets for the training and testing purpose we can be more confident that our network is more robust and handles more variation. It achieves 97.74% accuracy on the validation dataset and 87.85% accuracy on the test dataset.

Keywords: Bangla Handwritten Letter Recognition, Deep Convolutional Neural Network Model

1. Introduction

More than 230 million people speak in Bangla all over the world. People from Bangladesh, West Bengal, Assam, Tripura, and some immigrant communities of the United Kingdom, the United States, and the Middle East speak in the Bengali language. It is the 5th most widely spoken language in the world. It has 11 vowels and 39 consonants. In this era, all of the documents or files are converting into e-files. As we know, most of the previous documents, such as Birth certificates, school records were handwritten documents. So for the digitalization of such huge documents, the authority needs automation. So research on Bengali Handwritten letters will be helpful for such automation.

In the Bangla language, a letter differs from its similar one with a single dot or mark. Some letters are identical and very close. Besides, handwritten letters also differ from person to person. The Bengali language has 50 individual letters and some of them are very similar as well as identical. It indicates that simple classification techniques cannot provide better results in handwritten Bangla letter recognition systems. In this work, we investigate the handwritten Bangla letter recognition on Bangla letters using the deep convolutional neural network (DCNN) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998) models. The contributions of this paper can be summarized as follows: We have implemented comprehensive evaluation of the state-of-the-art DCNN models including inception V3 C. Szegedy, V. Vanhoucke ,S. Ioffe , J. Shlens , Z.Wojna (2016), VGG Net vgg (2014), Residual Network (ResNet) res (2016), and Densely connected convolutional Network (DenseNet) G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten (2017)on the application of handwritten Bangla letter recognition.

2. Related Work

There has been some research on Bangla letter recognition in recent times. From all those works, very few are notable. Pal and Chaudhuri U. Pal and B. Chaudhuri (2000) presented a new feature extraction-based method for handwritten Bangla letter recognition. Liu and Suen C.-L. Liu and C. Y. Suen (2019) suggested directional gradient features for handwritten Bangla digit classification using a famous dataset named ISI Bangla numeral dataset B. Chaudhuri (2006), that involves 19,392 training samples, 4000 test samples, and 10 classes (i.e., 0 to 9). It is to be noted that they worked on digit recognition which has fewer classes, Das et al. N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu (2012) used a genetic algorithm-based region sampling method for local feature selection and achieved better accuracy on Handwritten Bangla Character Recognition. Xu et al. J.-W. Xu, J. Xu, and Y. Lu (2008) used a hierarchical Bayesian network that directly takes raw images as the network inputs and classifies them using a bottom-up approach. Sparse representation classifiers have also been applied for Bangla digit recognition, where accuracy was reported for handwritten digit recognition. Handwritten Bangla basic and compound letter recognition using multilayer perceptron (MLP) and SVM classifier was suggested, while handwritten Bangla numeral recognition using MLP was presented in where the average recognition rate reached.

In recent years, deep learning-based methods have been increasingly used in handwritten letter recognition systems. Ciregan and Meier D. Ciregan and U. Meier (2015) applied multicolumn CNNs to Chinese letter classification. Kim and Xie G. E. Hinton and Teh (2015) applied DCNN to Hangul handwritten letter recognition and superior performance has been achieved against classical methods. A CNN-based Handwritten Bangla Letter Recognition scheme was proposed in M. M. Rahman (2015) where the best recognition accuracy reached 85.36%on their own dataset.

3. Methodology

Deep neural network (DNN) is a highly popular and widely used field of machine learning and it uses mainly three popular network architectures: Deep Belief Net (DBN) Kim and Xie (2006), Stacked Autoencoder (SAE) P. Vincent and Manzagol (2008), and CNN. As DNN uses many interconnected layers with deep connections, DNN methods are generally more capable of representing the highly varying nonlinear functions in machine learning problems than shallow neural networks with weak layers et al (2009). Shallow architectures mainly try to compensate for their depth with less computation time and complexity. However, It generally fails to make the necessary number of connections to know all the necessary features in a large dataset. In the case of DNN, the low and middle level extract the feature from the input image, whereas the high level uses the extracted features for classification operations. By doing this, DNNs can create a complete and sophisticated network that is formed by integrating with all necessary layers within a single network. As a result, in the case of a large dataset with a large number of parameters DNN models often lead to better overall accuracy compared to other machine learning approaches. Nowadays, the successful use of DNN covers a variety of topics such as radar signal examination E. Protopapadakis and Bimpas (2017), medical image analysis, electricity consumption monitoring J. Kim and Kim (2017), food security, and remote sensing. Among all deep learning approaches, Convolutional Neural Network (CNN)Aharon Azulay (2019) models are one of the most popular and widely used in the field of image classification. It provides state-of-the-art performance on human action recognition, image super-resolution, segmentation, scene labeling, and visual tracking K. Zhang and Yang (2016).

3.1 Convolutional Neural Network (CNN)

LeCun et al. Y. LeCun and Haffner (1998). initially applied CNN to digit recognition tasks. Various variants of CNN are applied to various image processing applications. The architecture of CNN closely resembles human visual data processing methods, and it derives high accuracy for tasks that involve 2D images. Moreover, CNN can effectively learn from the dataset and can extract various features from the 2D images. Features like max-pooling are very effective in absorbing shape variations and reducing the calculation space. Furthermore, CNN can make sparse connections with tied weights to make a fully connected network of fewer parameters than other networks of similar size. Also, CNN is trained with a gradient-based learning algorithm and suffers less from the diminishing gradient problem. It reduces the chance of making a useless network where later layers have no impact on the classification. By using the gradient-based algorithm the network minimizes error criterion and produces good generalization performance M. Matsugu and Kaneda (2003) and highly optimized weights. In Figure 1, the overall architecture of a CNN is shown. It consists of two main parts: feature extractor and classifier. In the feature extraction unit, each layer of the network receives the output from its immediate previous layer as inputs. And then it passes the current output as inputs to the immediate next layer. On the other hand, the classification part is used to generate the predicted outputs associated with the input data and create the necessary features needed to correctly classify the images. The two basic layers in CNN architecture are the convolution and pooling layers. In the convolution layer, each node extracts the features from the input images by doing the convolution operation on the input nodes. The max-pooling layer reduces the dimension size by abstracting the features through average or maximum operation on input nodes. The outputs of the layer are used as input for the next layer, where the inputs go through a set of kernels followed by the nonlinear activation function ReLU. For example, if xil-1 inputs from the l-1th layer, ki,jl are kernels of lth layer. The biases of the lth layer are represented with bjl . Then, the convolution operation can be expressed as 1

$$x_j^l = f(x_i^{l-1} * k_i^j) + b_j^l \tag{1}$$

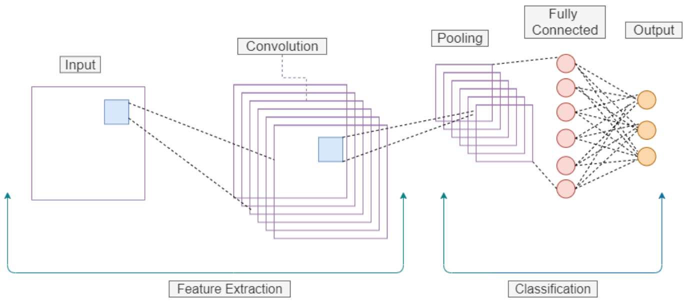


Figure 1: Basic CNN architecture

The subsampling or pooling layer abstracts the feature through average or maximum operation on input nodes and gets the low and high-level features for the output result. For example, if a downsampling kernel is applied, then each output dimension will be half of the corresponding input dimension for all the inputs. The pooling operation can be stated as follows:

$$x_j^l = down(x_i^{l-1}) \tag{2}$$

In contrast to other traditional neural networks, CNN extracts low- to high-level features rather than the other way around. The higher-level features can be derived from the propagated feature of the lower-level layers as the lower-level features often act as the building block for higher-level features. As the features propagate to the highest layer, the dimension of the feature can be reduced depending on the size of the pooling masks and convolution layers. However, the number of features for mapping is usually increased for selecting or mapping the extremely suitable features of the input images for better classification accuracy. For the final output, the last layer of CNN is used as inputs to the fully connected network and it typically uses an activation function like Softmax to produce the classification outputs. For an input sample x, weight vector w, and K distinct linear functions, the Softmax operation can be defined for the ith class as follows:

$$P(y = i|x) = \frac{exp(x^T w_i)}{\sum_{i=1}^{n} exp(x^T w_k)} \tag{3}$$

Over the years there are different variants of D-CNN architecture have been proposed. The following section discusses some of the most popular D-CNN architectures and the use of these architectures in our experiment.

3.2 CNN Variants

With the widespread use of CNN networks, by combining various important and fundamental components many efficient D-CNN networks have been created. These components are the convolution layer, pooling layer, fully connected layer, and Softmax layer. The advanced architecture of this network consists of a stack of convolutional layers and max-pooling layers followed by fully connected and Softmax layers at the end. Many researchers in various companies and universities have proposed various models including LeNet, AlexNet, VGG Net, All-Conv, NiN, ResNet, FractalNet, DenseNet, and many more. However, some topological differences can be observed in different modern D-CNN architectures. Out of many D-CNN architectures, AlexNet, VGG Net, IncetionNet, ResNet, DenseNet, and FractalNet can be viewed as the most popular architectures with respect to their enormous performance on different benchmarks for object classification. All of these networks use different methods to solve common CNN problems like vanishing gradients and use different combinations of layers to get their desired effects. Among these models, some of the models are designed especially for large-scale implementation such as ResNet and InceptionNet, whereas the VGG Net consists of general architecture. In contrast, DenseNet's architecture is unique in terms of unit connectivity where every layer to all subsequent layers is directly connected. All of these models have a slight variation that makes them more robust in different types of datasets. For example, DenseNet has many variants and some of the popular ones among them are DenseNet121, DenseNet161, DenseNet169, DenseNet201. All of these variants have different numbers of layers and produce slightly different results even in the same dataset. In our paper we have used VGGNet 16, VGG Net 19, ResNet50, InceptionV3, DenseNet121, DenseNet169, and DenseNet201 for Bangla Letter Recognition.

There are five steps in our work.

1. Collect the dataset for training and testing.
2. Preprocess the dataset and perform data augmentation.
3. Train the dataset on the D-CNN models and their variants.
4. Test the models on a completely different dataset. and analyze the results.

By doing this experiment we want to see which of these deep architectures are best suited for this type of task and create a future guideline for further research.

4. Dataset

The data used to support the findings of this study are available at CMATERdb Center for Microprocessor Application for Training Education and Research (2012). It consists of a Train folder and a Test folder, containing 12,000 and 3,000 images respectively. All the images were rescaled to standard dimensions of 32x32. For our experiment, we have used VGG, Resnet, Densenet, and Inception models. Of these four models, the minimum image size limit for the Inception V3 model is 75x75. So, for this, we have resized the CMATERdb dataset to 100x100 images. We have also inverted the colors for this network because it reduces complexity. For the other 3 models VGG, Resnet, and Densenet the minimum size limit is 32x32. So, rescaling to 32x32 is useful here. We used the lowest resolution possible because it reduces training cost and time. We validated our result with the testing dataset of 3000 images.

For the testing dataset, the accuracy for the validation set may indicate overfitting in the dataset. For this reason, we have used a completely different dataset for the testing set. We have used the Ekush Dataset for testing purposes. This dataset contains two different datasets of both male and female participants. For better testability, we have created a custom test set from these datasets. We have taken 100 images from both datasets randomly to create a 200 image per class dataset. As we have 50 Bangla Characters to identify, we have 50 classes. So the total number of images in the test set is 10000 images. We have also resized the datasets to different model preferences. Just like the training dataset we have resized the images 100x100 for the inception model and 32x32 for the other models.

5. Experimental Results and Analysis

For the experiment, we have decided to run every model for 100 epochs. We have selected the batch size as 16 and have given an early stopping function. We also reduce the learning rate if the loss function does not improve for 5 epochs. Each model took on an average of 5 hours to train. There were some failed attempts due to connectivity issues in google collaboratory. These are the results for all the successful models.

Types	**Method Name**	**Accuracy of Testset (CMATERdb 3.1.2)**	**Accuracy of Testset (Ekush Male and Female)**
Existing approaches	Bhowmick et al.	84.33 %	—
	Basu et al.	80.58%	—
	Bhattacharya et al.	95.84%	—
	BHCR-CNN	85.96%	—
D-CNN	VGG 16	97.56%	85.61%
	VGG 19	95.88%	83.30%
	Residual Network (Resnet) 50	93.87%	80.67%
	Inception V3	97.70%	12.85%
	DenseNet121	97.73%	86.43%
	DenseNet169	97.74%	87.85%
	DenseNet201	97.23%	85.86 %

Table 1: Accuracy of different models

Most of the errors incurred by our model in recognition tasks are due to extreme proximity in shapes among characters. There were some characters that were overwritten on top of one another to make it more confusing. For this reason, a significant number of errors were caused by mislabeled, irrecoverably distorted, and illegal data examples.

Here we see that for the CMATERdb 3.1.2 dataset the results of the test set have very high accuracy when compared to the previous results that were conducted on the same dataset. By using various pre-trained models such as VGG, ResNet, and DenseNet we have a comparative result of which network performs best depending on this completely new testing dataset. The total parameters, trainable parameters, and non-trainable parameters for these models are described in the following table.

Deep CNN Models	**Total Params**	**Trainable Params**	**Non-trainable Params**
VGG 16	15,794,546	15,792,498	2,048
VGG 19	21,104,242	21,102,194	2,048
ResNet 50	26,246,578	26,188,338	58,240
Inception V3	21,905,234	21,870,802	34,432
DenseNet121	8,643,698	8,556,978	86,720
DenseNet169	14,906,994	14,744,242	162,752
DenseNet201	20,849,266	20,615,346	233,920

Table 2: Parameters of different D-CNN models

Here the various pre-trained D-CNN Models have different numbers of trained and non-trained parameters. After going through all of the results we can see that Dense net 169 has the highest result at 87.85%. This can gain better results if we consider only one kind of dataset, where its accuracy is 89.95%. But with merging two datasets we have more versatility for this model to work in all kinds of situations.

Now we will provide the training and testing results of various D-CNN models.

5.0.1 VGG 16

The following two figures are the loss and accuracy graph for VGG 16 Model.

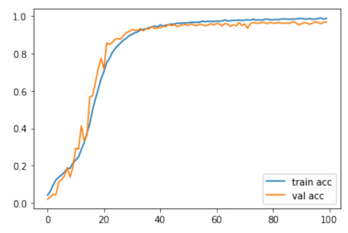


Figure 2: Loss Of VGG 16

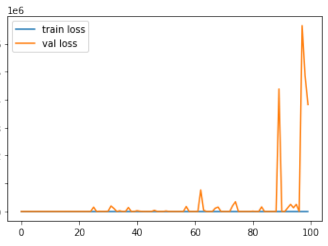


Figure 3: Accuracy Of VGG 16

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is high loss for this model and its accuracy has gone up sharply for the first 40 epochs. After that the result was a gradual climb and after 60 epochs it has stopped improving.

5.0.2 VGG 19

The following two figures are the loss and accuracy graph for VGG 19 Model.

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is high loss for this model and its accuracy has gone up sharply for the first 50 epochs. After that the result was a gradual

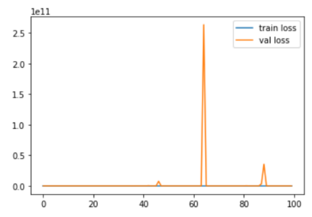


Figure 4: Loss Of VGG 19

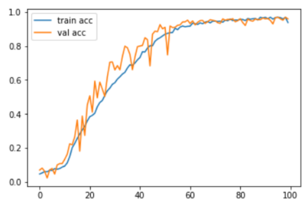


Figure 5: Accuracy Of VGG 19

climb and after 70 epochs it has stopped improving. Even though. it has more parameters than the VGG 16 model; its performance is worse in both of the testing dataset.

5.0.3 Resnet 50

The following two figures are the loss and accuracy graph for the Resnet 50 Model.

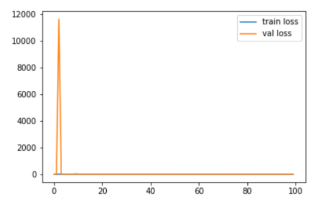


Figure 6: Loss Of Resnet 50

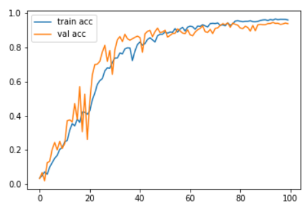


Figure 7: Accuracy Of Resnet 50

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is significantly low loss for this model and its accuracy has gone up sharply for the first 50 epochs. After that the result was a gradual climb and after 80 epochs it has stopped improving. The resnet model is deeper than other D-CNN models. However, its result is average compared to the other models. It has a 93.87% accuracy for the CMATERdb testset and only 80.67% accuracy in our custom Ekush testset. So, we did not bother to work with the other versions of the resnet model.

5.0.4 Inception V3

The following two figures are the loss and accuracy graph for the Inception V3 Model.

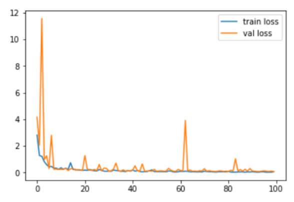


Figure 8: Loss OfInception V3

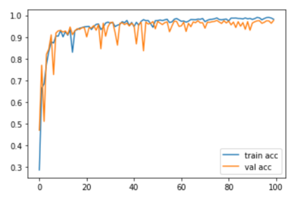


Figure 9: Accuracy Of Inception V3

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is significantly low loss for this model and its accuracy has gone up sharply for the first 25 epochs. After that the result was a gradual climb and after 60 epochs it has stopped improving. The result of this model is shocking. It has a 97.70% accuracy for the CMATERdb testset and an abysmal 12.83% accuracy in our custom Ekush testset. This indicates massive overfitting. Point to note that this is the only dataset where we had 100x100 pictures. We also used color inversion on this dataset. So, changing image size to bigger dimensions may lead to overfitting for the HBCR process. Therefore, our decision to use lower resolution images is more accurate.

5.0.5 Densenet 121

The following two figures are the loss and accuracy graph for the Densenet 121 Model.

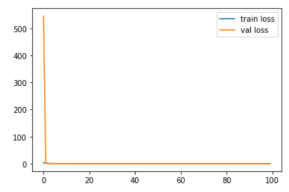
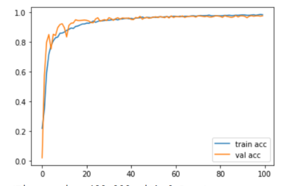


Figure 10: Loss Of Densenet 121



```
<Figure size 432x288 with 0 Axes>
```

Figure 11: Accuracy Of Densenet 121

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is significantly low loss for this model and its accuracy has gone up sharply for the first 25 epochs. After that the result was a gradual climb and after 40 epochs it has stopped improving. Even after that it has better accuracy than the previous models and testing. Meaning densenet architecture is a great match for the Bangla Letter Classification problem.

5.0.6 Densenet 169

The following two figures are the loss and accuracy graph for the Densenet 169 Model. Here

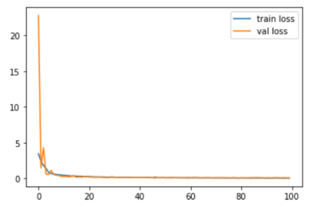


Figure 12: Loss Of Densenet 169

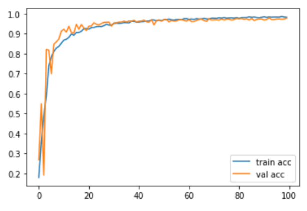


Figure 13: Accuracy Of Densenet 169

on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is significantly low loss for this model and its accuracy has gone up sharply for the first 30 epochs. After that the result was a gradual climb and after 50 epochs it has stopped improving. Even though the improving rate of this model is slower compared to the Densenet 121 model. It has better overall results.

5.0.7 Densenet 201

The following two figures are the loss and accuracy graph for the Densenet 201 Model.

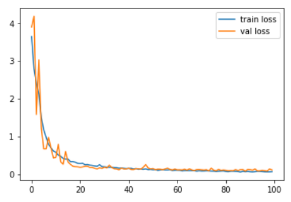
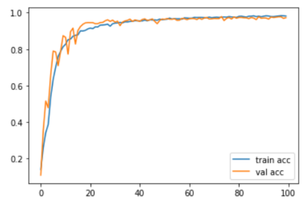


Figure 14: Loss Of Densenet 201



```
<Figure size 432x288 with 0 Axes>
```

Figure 15: Accuracy Of Densenet 201

Here on the pictures the x axis is representing the number of epochs and the y axis refers to the loss/accuracy. From this we can see that there is significantly low loss for this model and its accuracy has gone up sharply for the first 25 epochs. After that the result was a gradual climb and after 55 epochs it has stopped improving. This is the most heavy model of the densenet models that we have tested. Despite this, its result is the lowest when compared to the other two densenet models. It can mean that more params doesn't necessarily mean better results.

6. Conclusion

In this research, we investigated the performance of several popular deep convolutional neural networks (D-CNNs) for handwritten Bangla letter recognition in various settings. From the experimental results we can derive that DenseNet169 is the best performer in classifying Bangla letters out of all the other D-CNN models. To the best of our knowledge, these are one of the best recognition results on the CMATERdb dataset as well as the EKush dataset. In future, we will like to try other fusion-based D-CNN models, such as Inception Recurrent Convolutional Neural Network (IRCNN) Henning Lange (2021), to eplore and develop for handwritten Bangla handwritten letter recognition.

References

Very deep convolutional networks for large-scale image recognition. `http://arxiv.org/abs/1409.1556`, 2014.

Deep residual learning for image recognition. `https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html`, 2016.

Yair Weiss. Aharon Azulay. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 2019.

B. Chaudhuri. A complete handwritten numeral database of bangla–a major indic script, 2006.

C.-L. Liu and C. Y. Suen. A new benchmark on the recognition of handwritten bangla and farsi numeral characters, 2019.

C. Szegedy, V. Vanhoucke ,S. Ioffe , J. Shlens , Z.Wojna. Rethinking the inception architecture for computer vision, 2016.

Center for Microprocessor Application for Training Education and Research. CMATERdb Dataset. `https://code.google.com/archive/p/cmaterdb/`, 2012.

D. Ciregan and U. Meier. Multi-column deep neural networks for offline handwritten chinese character classification, 2015.

A. Doulamis N. Doulamis D. Dres E. Protopapadakis, A. Voulodimos and M. Bimpas. Stacked autoencoders for outlier detection in over-the-horizon radar signals. *Computational Intelligence and Neuroscience,*, page 11, 2017.

Y. Bengio et al. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2:1–127, 2009.

S. Osindero G. E. Hinton and Y.-W. Teh. Handwritten hangul recognition using deep convolutional neural networks. *Neural Computation*, 18:1–13, 2015.

G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks, 2017.

J. Nathan Kutz Henning Lange, Steven L. Brunton. From fourier to koopman: Spectral methods for long-term time series prediction. *Journal of Machine Learning Research 22*, page 1–38, 2021.

T.-T.-H. Le J. Kim and H. Kim. Nonintrusive load monitoring based on advanced deep learning and novel signature. *International Journal on Document Analysis and Recognition*, 2017:22, 2017.

J.-W. Xu, J. Xu, and Y. Lu. Handwritten bangla digit recognition using hierarchical bayesian network, 2008.

Y. Wu K. Zhang, Q. Liu and M.-H. Yang. Robust visual tracking via convolutional networks without training. *IEEE Transactions on Image Processing*, 25:1779–1792, 2016.

I.-J. Kim and X. Xie. A fast learning algorithm for deep belief nets. *International Journal on Document Analysis and Recognition*, 18:1527–1554, 2006.

S. Islam P. C. Shill M. H. Rahman M. M. Rahman, M. Akhand. Bangla handwritten character recognition using convolutional neural network. *International Journal of Image, Graphics and Signal Processing*, 2015.

Y. Mitari M. Matsugu, K. Mori and Y. Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16:555–559, 2003.

N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application, 2012.

Y. Bengio P. Vincent, H. Larochelle and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. *25th International Conference on Machine Learning*, page 1096–1103, 2008.

U. Pal and B. Chaudhuri. Automatic recognition of unconstrained off-line bangla handwritten numerals, 2000.

Y. Bengio Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition, 1998.