Handwritten digit classification using CNN

Rohan Gandhi, Rohan Shahane, Santosh Kannan and Tirth Doshi

***Abstract* – Processing noisy and high dimensional image data and interpreting it in text format is a key component for numerous applications worldwide. Over the course of time, different machine learning algorithms have been leveraged to achieve this task with a high success rate. In this paper, we present a Convolutional Neural Network (CNN) based model to accurately process noisy, 300 x 300 dimensional handwritten digits. This is achieved by mitigating the noise from the images by transforming the images to binary format and training the CNN on a refined dataset. This paper also describes the implementation of the model backs the functionality of the model with experimental results observed. Furthermore, the paper addresses the challenges faced during the implementation phase and also discusses various experiments conducted to achieve the goal of accurately classifying the data.**

I. INTRODUCTION

Convolutional Neural Network is a class of deep learning neural network which can efficiently analyze image data. Processing hand-written image data and recognizing patterns can be handled precisely using CNN. Researchers over the years have explored and proposed numerous models based on CNN. We briefly analyze few of them which are relevant to our project.

Researchers Y. Hou and H. Zhao explored an approach to combine an artificial neural network with a convolutional neural network to obtain high accuracy for handwritten digit recognition. They designed the neural network with Gabor Filters, which analyze a specific frequency in the localized area of the filter and pass the new images after extraction into the convolutional neural network. The primary objective of the research was to reduce the size of network in order to minimize the processing time. The proposed method obtained higher accuracy for with the combined CNN model as compared to the legacy CNN model. However, since the training and testing for the model was performed on MNIST dataset, we cannot gauge the impact due to noisy images on this model.[1]

Another paper implements a KNN classifier to classify digits written in "*Devanagiri*" script.[2] The approach involved resizing the images to obtain consistency. The paper describes a method to extract features from a CNN and applying KNN and SVM classifier to classify the digits. Support Vector Machines create a hyper plane such that the margin of the classifiers is maximized, and support vectors are the data points that are closest to the hyperplane and influence the choice of hyper plane [2].

A paper published in the Iranian Conference on Machine Vision compares various models for recognizing Persian handwritten digits. The model was trained and evaluated using the popular hoda dataset after resizing these images with padding. Different experiments were conducted with KNN, SVM's, Random Forest and CNN based classifiers. The paper concluded that the best performance was achieved using CNN [3].

In general, all these papers are inclined towards CNN. Since the classroom dataset is noisy as well as high dimensional, we aim to implement preprocessing strategies for images and build a CNN model to classify the data. A detailed guideline on the implementation and relevant experiments is described in following sections.

II. IMPLEMENTATION

The model has been implemented using the Keras package and the TensorFlow framework from the Python library. In order to leverage the inbuilt features of the libraries, importing all necessary packages is mandatory. The classroom dataset contains 1200 data points and training the model on only this dataset leads to overfitting of the model on classroom data with a low degree of generalization. Thus, additional data points were inserted from the MNIST database. The dimensions for these MNIST data points need to be converted to match the training data provided. Additional steps are taken to filter and refine the color standards of each image. The implementation of our model follows the below steps:

- Importing required Packages
- Reading Datasets
- Augmenting the Datasets
- Training the Model
- Testing the Model Performance

Importing required Packages: Since we process image data and implement a Keras CNN model, all dependant packages are required to be imported. Essential packages include Pandas, NumPy, Keras, TensorFlow, PIL, Matplotlib, sklearn

Reading Datasets: Apart from the provided dataset, in order to generalise the model, we added 2000 images from Keras MNIST train dataset to the training set and 2000 images from MNIST test dataset to the model test set. Both datasets are read into NumPy arrays.

Augmenting the Datasets: Data Augmentation is required in multiple phases to refine the different datasets and make them consistent for processing.

- Training Dataset Pre-processing: - The provided dataset contains noisy images and is refined in two steps:
  - Noise Removal – The images are converted into Binary colour scale by converting all bits in the image as 0 or 1, thereby removing any noise. This is achieved by assuming a threshold of 120, meaning all pixels with value below 120 are set to 1, and 0 otherwise.

- Inverting Colours – Assuming a threshold of 150, the bits are reversed to set the background for the characters as 0 (black). This way, the mean for the dataset ~ 0.

- MNIST Data Pre-processing: - In order to process MNIST data together with the provided dataset, it is essential to convert the MNIST images from 28x28 into 300x300. This is achieved by padding the 28x28 images to make their dimensions 100x100 and resizing the padded images to 300x300. Padding is done in order to maintain the aspect ratio of the image and avoid image blurring.

- Splitting Data into Test and Train sets: - For the purpose of training and evaluating the model, we split the data as below:
  - Train set – 970 images from provided dataset + 2000 images from MNIST train dataset
  - Model Evaluation set – 120 images from provided dataset
  - Model prediction Test set – 110 images from classroom dataset + 2000 images from MNIST test set (referred as Unseen data in following sections)

Training the Model: Once the datasets are made consistent, one final data augmentation step is required before processing the model. Channel field is added to the data arrays and the label arrays are converted into one hot encoded arrays.

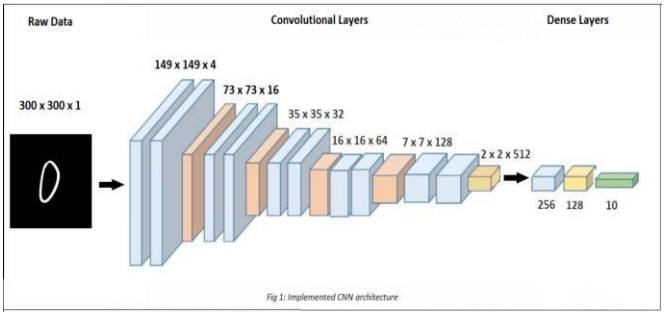


Fig 1 depicts the architecture of the model implemented.

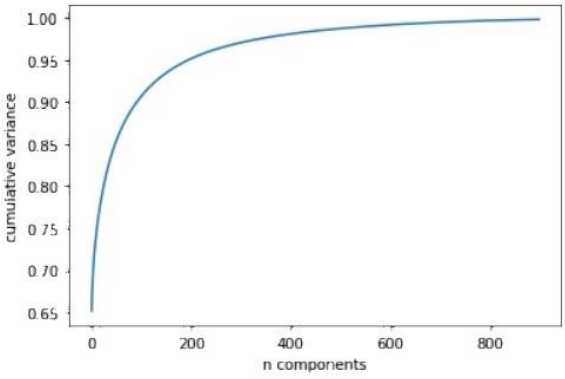
Multiple convolutional layers are used with "*Relu*" activation function to increase the learning ability of the model. Convolutional layers are followed by 2 dense layers with *"Relu*" activation which assist in processing the non-linear nature of data. Finally, a 10-neuron dense layer with the "*Softmax*" activation is used to classify the data into 10 different classes.

Testing the Model: Model is validated in two phases. First, the model is tested on the model evaluation set and the parameters are adjusted to achieve higher accuracy. Next, the model is tested on the Prediction test set (unseen dataset) to gauge the accuracy of the model and its ability to generalise.

III. EXPERIMENTS

To train the model, we selected 970 data points from the classroom data as the training set. Various experiments have been conducted during different phases of designing the model. Details for each experiment with the concluded results are mentioned below:

1. Dimensionality Reduction: Since the images are 300 x 300 shaped, time required for any model to process the data would be high. To reduce the dataset into smaller dimensions, we applied PCA transformation, since PCA is one of the best models for image processing.



Result: As shown in the graph above, nearly 900 components are required to preserve a close to 100% variance. However, due to the noise in the data, the accuracy of the model significantly dropped. Moreover, since CNN works by convolving the data and reducing the dimensions at each layer, we decided to discard the external dimensionality reduction process.

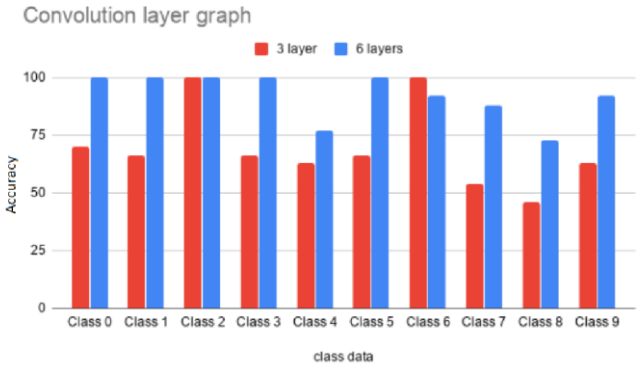
2. Augmenting Classroom Dataset: The dataset provided contains lot of noisy images. In order to maintain consistency across dataset, we removed the noise from the datasets by converting all faint pixels into white. Further, we inverted the colours for all images, resulting in the white coloured digits in black background. Since black pixels are represented as 0, the mean for the image would be close to 0. In order to decide the threshold values for removing noise and inverting the images, we followed a trial and error approach.

Threshold for de-noising the data	Threshold for inverting images	Images inaccurately transformed
120	120	22
120	150	8
110	120	32
110	150	14
150	150	27

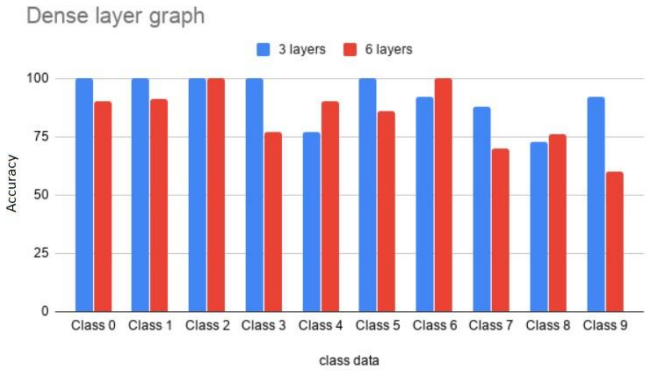
Result: A threshold of 120 for removing the noise and 150 for inverting the images resulted in least loss of data. 5.

3. Choice of Model: For the purpose of classifying the digits accurately, we considered three approaches- CNN, KNN and SVM. CNN stands out due to its capability of learning the features automatically, sharing weights at each layer and computing the results efficiently. Both KNN and SVM have its own pros and cons and are used in image classification for various use cases.
Result: Through our preliminary analysis, we chose CNN model, due to its inherent advantages of adaptive learning and feature extractions.

4. Number of Layers: We conducted experiments to test the accuracy of the model with respect to number of convolutional layers and dense layers. This was done by varying the number of convolutional and dense layers alternatively. We tested the model for either 3 or 6 convolutional layers and 3 or 6 dense layers. First, we fixed values for dense layers and varied the convolutional layers.



After finalizing 6 convolutional layers, we compared accuracy for models with 3 dense layers and 6 dense layers.



Result: From the experiments conducted on different number of layers in the configuration, we decided on using 6 convolutional layers and 3 dense layers.

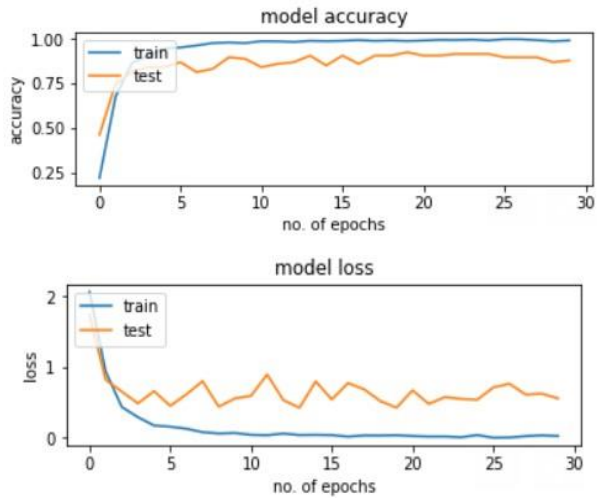
Configuring the Convolutional and Dense Layers: After finalizing 6 convolutional layers and 3 dense layers, we computed the parameters at each layer starting with 300 x 300 x 4 for layer 1. Since a (3,3) kernel window is used, we know that the dataset shrinks by ~half after each layer. Thus, we configured the subsequent layers as 8,16,32,64,128 and compared the model with higher filter models. For the purpose of evaluating the generalization level, we created a news dataset unseen by the model. Top 5 models observed with high accuracies are mentioned below:

Convolutional Layer Configuration	Classroom Accuracy	MNIST accuracy	Unseen Data
4, 8, 16, 32, 64, 128	91	64	65
4, 16, 32, 32, 64, 64	92	71	72
4, 16,32,64,64, 128	94	66	67
4,16,32,64,128,256	93	69	68
4,16,32,64,128, 512	95	77	79

Thus, we decided on the configuration (4, 16, 32, 64, 128, 512) since it has high accuracy with high scope for generalizing to new data.
Since the final convolutional layer has 512 neurons, we experimented dense layers with values in the power of 2. i.e. 512, 256, 128, 64, 32. Final layer is fixed with 10 neurons since the output should classify the data into 10 different classes. Upon iterating the model over different values for dense layers, we arrived at a configuration (256, 128, 10) which resulted in high accuracy.

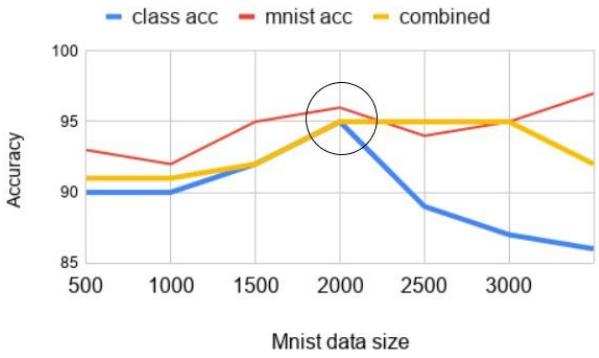
Result: Thus, we implemented the model with a configuration of (4,16,32,64,128,512) for convolutional layers and (256,128,10) for dense layers.

6. Number of Epochs: Once the configuration for convolutional layers and dense layers were computed, we simulated the model for 30 epochs to gauge the accuracy of the model vs epochs.



Result: As shown in the graph above, at 19 epochs the model obtains high accuracy with low data loss. Thus, we finalized the number of epochs as 19.

7. Addition of MNIST data to the training set: After augmenting the MNIST data, we appended MNIST data to classroom dataset. In order to avoid overfitting the model on MNIST data, it was essential to obtain a generalised model for MNIST and classroom data. We thus iterated the model with 970 classroom data points and varying amounts of MNIST data.



Result: The model was tested on three different test data sets as mentioned in the image. As shown in the graph, highest accuracy for the combined dataset, while achieving a high accuracy for classroom dataset is obtained with 2000 MNIST data points

8. Validation of the Model: To completely validate the model and its ability to generalize to new data, we applied K Folds validation technique to evaluate the model.

K-fold	Classroom test set	MNIST test set
5	94.34	91.66
8	94.29	91.66
10	97.33	92.45
15	99.01	93.23

We observed that for the classroom dataset, the model starts overfitting for values of K higher than 8. However, this is not the case with MNIST data. This is because the classroom dataset is not large enough to fit 15 folds cross validation. On the other hand, since MNIST data is more expansive, higher values of K are acceptable.

Result: Thus, for values of K < 10, we were able to obtain high accuracy values

IV. CONCLUSION

Through the experiments carried out, we can conclude that Convolutional Neural Network is an efficient model with high accuracy for handwritten digits recognition. We were able to address the challenges in each phase of design and implementation and build an efficient model. Moreover, we can infer that a model which processes higher dimensional image data is likely to provide higher accuracy. Since the dataset only contained 1200 data points, the model could not generalize to unseen data. This was rectified by adding MNIST dataset, thereby proving that higher data volumes can help in generalization.

In our case, the model loss stagnates after 20 epochs, indicating that the accuracy does not improve anymore. We believe, this model performance could be enhanced with advanced image processing techniques and complex configurations.

REFERENCES

[1] Y. Hou and H. Zhao, "Handwritten digit recognition based on depth neural network," 2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Okinawa, 2017, pp. 35-38.

[2] N. Yu, P. Jiao and Y. Zheng, "Handwritten digits recognition base on improved LeNet5," The 27th Chinese Control and Decision Conference (2015 CCDC), Qingdao, 2015, pp. 4871-4875.

[3] Y. Zamani, Y. Souri, H. Rashidi and S. Kasaei, "Persian handwritten digit recognition by random forest and convolutional neural networks," 2015 9th Iranian Conference on Machine Vision and Image Processing (MVIP), Tehran, 2015, pp. 37-40.

[4] Machinelearningmastery, https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/ (2016)