# A simple CNN architecture for MNIST dataset

Darun Arumugham, Varshitha Seralathan

***Abstract*—Machine Learning methods are allowing vision recognition to become a common practice and are under constant development worldwide. Recognition of handwritten digits has long been an active research topic in computer vision and pattern recognition. Here we present a simple model for image recognition called Convolutional Neural Network that describes the theoretical and implementation steps for addressing this challenge. Using the MNIST handwritten digital database, a convolutional neural network was trained to identify pictures and predict exactly what the numbers in the pictures are.**

***Keywords—CNN, accuracy, pixels***

## I. Introduction

### A. Problem Definition

In the current era of digitization, handwriting recognition is crucial to the processing of information. The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning. The handwritten digits are generally different in strokes, size, thickness, orientation, and distance from the margins thereby increasing the complexity for recognition.[1] Convolutional neural networks (CNNs) are the most successful method for resolving handwriting recognition issues because they are very good at understanding the structure of handwritten characters and words in ways that facilitate the automatic extraction of distinctive features. This paper tackles how we approached the problem and created a Convolutional Neural Network architecture for identifying handwritten digits.

### B. Our Solution

Our goal is to use a CNN that can recognize handwritten digits to classify a given image of a handwritten digit into one of ten categories that represent integer values from 0 to 9, inclusive. A common deep neural network for computer vision applications is the convolutional neural networks. So, using the MNIST dataset, we will construct an image classifier using TensorFlow's Keras API. Firstly, we obtain a suitable MNIST dataset. Then, data exploration is performed on the obtained dataset to understand its features, identify missing values and outliers, and validate the distribution of various labels or classes. It also provides some insights into the data that we're working with. Once the data is cleaned and prepared, the CNN model is built by using multiple layers and trained using a portion of the dataset. The CNN model to be built has to consider the number of layers and the number of nodes in each of those layers. Such attributes determine the accuracy of the resulting model. After the model is built, it is validated against a small subset of data. Lastly, when satisfactory accuracy is obtained, the model is used for testing against real data and compared with other existing models to distinguish the features, training duration, and output accuracy.

## II. Data Exploration

MNIST database (Modified National Institute of Standards and Technology database) is sizable database of handwritten numbers which is frequently used to train different image processing systems.

Fig. 1. Sample MNIST dataset

MNIST dataset images are grayscale, with pixels ranging between 0 and 255. Each image has a height of 28 pixels and a width of 28 pixels for a total of 784 pixels. Each pixel has a single pixel value that describes its lightness or darkness, with larger numbers representing darker pixels.

- Training data

Training data is the data our model learns from. The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

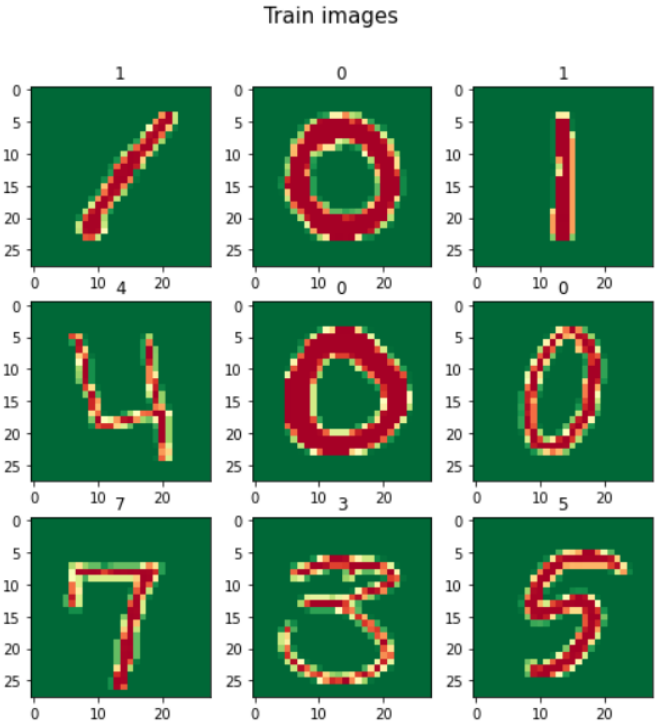


Fig. 2. Training images

- Test data

The test data is kept secret from the model until after it has been trained. Test data is used to evaluate our model. The test data set, (test.csv), is the same as the training set, except that it does not contain the "label" column.

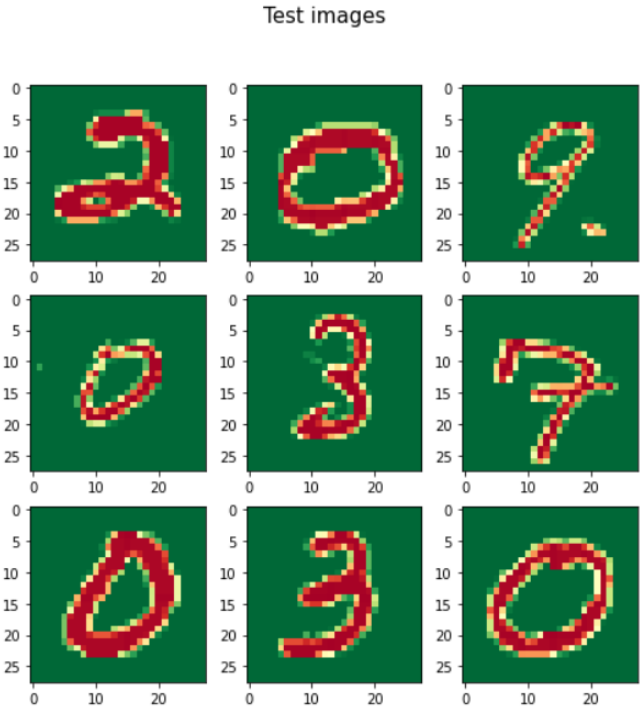


Fig. 3. Test images

In this file, the digit images are divided into two groups: x_train and x_test, and y_train and y_test. In both the x_train and x_test parts, there are greyscale RGB codes (0-255). A label is present on each of the y_train and y_test parts between 0 and 9.

## A. *Explaratory Data Analysis*

An exploratory data analysis involves performing initial investigations on data with the help of summary statistics and graphical representations so that patterns can be discovered, anomalies can be spotted, hypotheses can be tested, and assumptions can be tested.

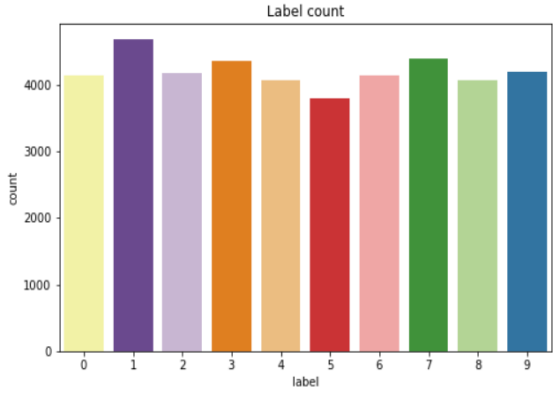


Fig. 4. Distribution of labels in dataset

From fig 5, we can see that Each image is *28* pixels in height and *28* pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

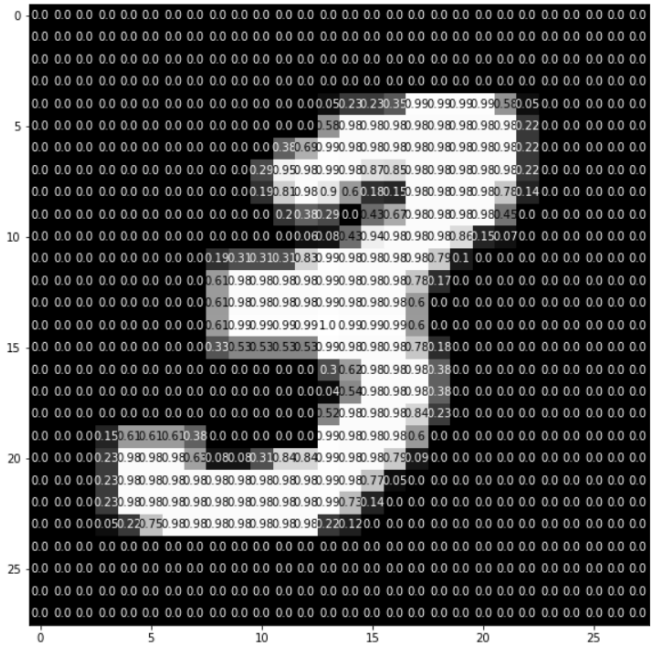

Fig. 5. Pixel representation of a sample MNIST dataset

## B. *Data Preprocessing*

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

### 1. Normalization

Normalization alters the range of pixel intensity values to make an image appear more familiar or normal to the senses. It is always necessary to normalize our data when modelling neural networks. To reduce the effects of illumination differences, we perform a grayscale normalization. Normalization increases the efficiency of the model.

### 2. Reshaping the data

Keras requires an additional dimension at the end, which corresponds to channels. As grayscale images, MNIST images only have one channel. RGB images have 3 channels, so we would need to reshape vectors (784 px) into 3D matrices 28x28x3.

### 3. Encoding

This dataset's labels are numerical values between 0 and 9, but it's critical that our algorithm treats these as elements in a collection rather than ordinal values. In our dataset, the values "0" and "9" are just two alternative classifications from our possible values; neither is smaller than the other.

It is incorrect to claim that the model was "off by 8" if our algorithm predicts "8" when it should predict "0"—it basically predicted the incorrect category. In a similar manner, predicting "7" when we should have said "8" is no better than predicting "0" when we should have predicted "8" – both are just incorrect.

A “one-hot encoded” vector is the best method to address this problem when making predictions about categorical data (as opposed to continuous data values). In other words, we create a vector as long as the number of categories we have, and force the model to set exactly one value to 1 and the rest to 0 (the 1 is the "hot" value).

## III. Methodology

Identifying the label of the image provided as a class between 0 to 9 using a CNN model is the objective. Described in the following sub-headings are the steps followed to create the model and validate its performance.

### A. Deep Learning

Deep Learning is a part of machine learning models.[2] Deep learning includes Supervised, Semi-supervised or unsupervised learning. Deep learning architectures which are applied to computer vision, speech recognition, natural language processing, audio processing, social network filtering, machine translation, bioinformatics, and medical image analysis fields include Deep neural networks, Deep belief networks, and Recurrent neural networks.[3] The results they have produced are comparable to and sometimes even better than humans. [4]

### B. Convolutional Neural Network

Convolutional neural network (CNN) is one of the most popular deep neural networks.[5] Convolutional neural networks are built from neurons that have weights and biases that can be learned, and just like regular neural networks, data fed to them are processed by the convolution layer. CNN is made up of several layers, and each layer of a CNN uses a differentiable technique to change one activation into another. The three primary layers utilized to create the CNN architecture are the Convolutional Layer, Pooling Layer, and Fully-Connected Layer. To create a complete CNN design, we shall execute these layers in succession. The typical convolutional neural network architecture with three convolutional layers is well adapted for the classification of handwritten images as shown in Figure 6. It consists of the input layer, multiple hidden layers (repetitions of convolutional, normalization, pooling), and a fully connected and output layer.[6]

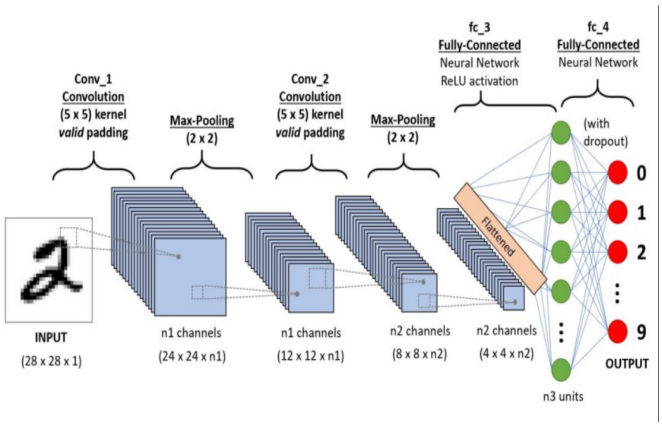


Fig. 6. Typical convolutional neural network architecture.

#### a. Input Layer

The input layer is loaded with the input data and stored there. The RGB information (height, width, and the number of channels) of the input image are described in this layer.

#### b. Hidden Layer

The foundation of CNN's architecture is its hidden layers. A sequence of convolution, pooling, and activation functions are used during the feature extraction process. At this step, the distinguishing characteristics of handwritten digits are identified.

#### c. Convolutional Layer

The input image is absorbed by the convolutional layer, which is the topmost layer. It is used to draw out an image's features. An m × m filter is convolved with the input layer's n × n input neurons, producing an output of (n - m + 1) × (n - m + 1). The convolutional layer's primary components are the receptive field, stride, dilation, and padding. During the pass, move each filter across the input volume and compute dot products between the filter and input.[7] The values of that particular filter are given in the form of a two-dimensional activation map. When CNN sees a specific type of feature of the image, the filter learns to activate. A single two dimension Activation map is produced for each filter in every convolutional layer. On piling the activation maps against the depth of the neurons, one can achieve the resulting magnitude.[4]

#### d. Pooling Layer

In order to decrease the input dimensionality and hence the computational complexity, a pooling layer is inserted between two convolutional layers. Pooling enables the selected data to be sent to the following layer while the unnecessary values are left behind. Additionally, the pooling layer aids in feature selection and overfitting management. The pooling process is carried out independently. It operates by taking a single output value from the input images' tiled, non-overlapping subregions. Max-pooling and avg-pooling are two popular forms of pooling operations (where max and avg represent maxima and average, respectively). Modern applications often prefer the max-pooling process since it retains the most information while taking the maximum values from each sub-region.

#### e. Activation Layer

The activation function is a component of CNN architecture, just like it is in standard neural network architecture, and it is used to inject non-linearity into the system. Among the several activation functions that are widely used in deep learning models are the sigmoid function, rectified linear unit (ReLu), and Softmax. The loss of information in the input data caused by the sigmoid activation function has been seen to impair the CNN model. The non-linear rectified linear unit (ReLu) function, which has output 0 for input less than 0 and raw output otherwise, is the activation function employed in the current work. ReLu activation function benefits include its resemblance to the human nervous system, ease of usage, and capacity for quicker training for bigger networks.

#### f. Classification or Fully Connected Layer

The final layer in the CNN architecture is the classification layer. It is a fully connected feed-forward network that is mostly used as a classifier. All of the neurons in the previous layer are connected to the neurons in the fully connected layers. Combining all the information previously learnt by layers, this layer determines predicted classes by locating the

input image. The number of classes in the target dataset determines how many output classes there will be.

### C. Building Convolutional Neural Network Model

The Keras sequential API is used to define the model by adding one layer at a time, starting from the input. Two elements make up the model:

1. Frontend of feature extraction with convolutional and pooling layers
2. Backend classifier that will make predictions

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 232,650
Trainable params: 232,650
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. CNN model

We begin with the 2D convolutional layer with 32 filters and a kernel size of (3,3) for the feature extraction front-end. This layer, which is essentially a mask, can be used to blur, sharpen, emboss, and identify edges in an image. It makes use of the ReLU's activation function, or Rectifier Linear Unit. Without the activation function, a deep neural network can only be used to fit a linear model to data. A max-pooling layer comes after the convolutional layer. The classifier can then use the flattened filter maps as features.

This is followed by another convolutional layer with 64 filters for feature extraction. This is also followed by a max-pooling layer. There is a dropout of 0.2 after these four layers.

The resulting Matrix was flattened after the features were extracted and provided as the input to the fully connected network. Relu and Softmax activation techniques are utilized in the fully connected layer, with a dropout of 0.2. Relu is the greatest option for the activation function because all of the images are in grayscale. Figures 8 and 9 represent the training and validation accuracy and loss respectively.

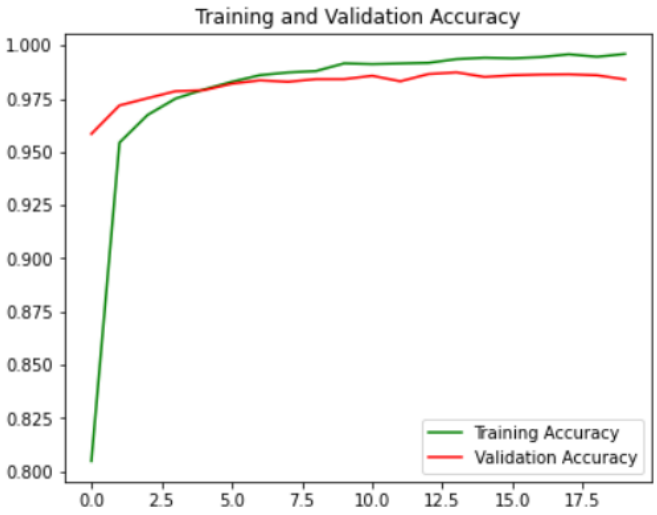


Fig. 8. Training and Validation Accuracy

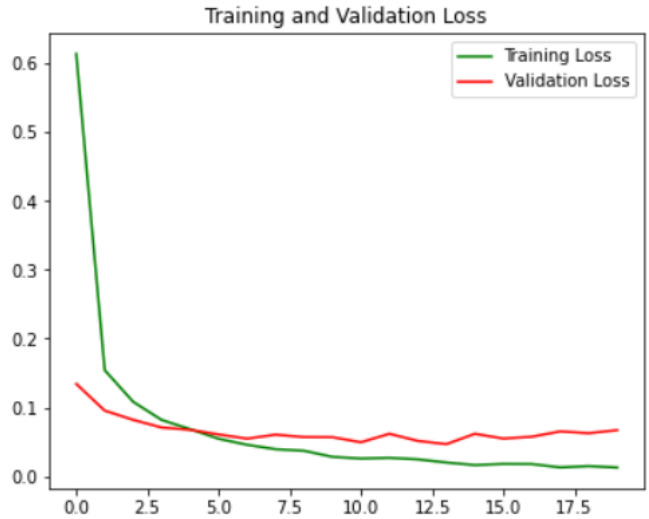


Fig. 9. Training and Validation Loss

We have only so far produced an unoptimized, empty CNN. So, using the optimizer, the loss function, and the metrics as parameters, we will now compile our network. Adam is the optimization algorithm we have chosen because it is effective and best suited for problems with many parameters, like our own. We selected Categorical Crossentropy as the loss function because we are working with a multiclass classification problem. This function efficiently calculates the discrepancy between the output that was actually produced and the output that the model had anticipated, which represents the prediction error. This error is sent as input to the optimizer (Adam), which creates a better model with each iteration. We have selected accuracy to be tracked for our metric.

## IV. Testing and Results

We trained and validated our model using the datasets mentioned. Figure 10 shows the metrics report of the model built.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Class 0 | 0.99 | 1.00 | 0.99 | 1174 |
| Class 1 | 0.99 | 0.99 | 0.99 | 1481 |
| Class 2 | 0.99 | 0.98 | 0.99 | 1223 |
| Class 3 | 0.99 | 0.99 | 0.99 | 1274 |
| Class 4 | 0.98 | 0.99 | 0.99 | 1244 |
| Class 5 | 0.99 | 0.98 | 0.99 | 1117 |
| Class 6 | 0.98 | 0.99 | 0.99 | 1273 |
| Class 7 | 0.99 | 0.99 | 0.99 | 1335 |
| Class 8 | 0.99 | 0.99 | 0.99 | 1210 |
| Class 9 | 0.99 | 0.98 | 0.99 | 1269 |
| accuracy | | | 0.99 | 12600 |
| macro avg | 0.99 | 0.99 | 0.99 | 12600 |
| weighted avg | 0.99 | 0.99 | 0.99 | 12600 |

Fig. 10. Classification Report metrics

To better understand the performance of the model, we have also created a confusion matrix as shown in figure 11.

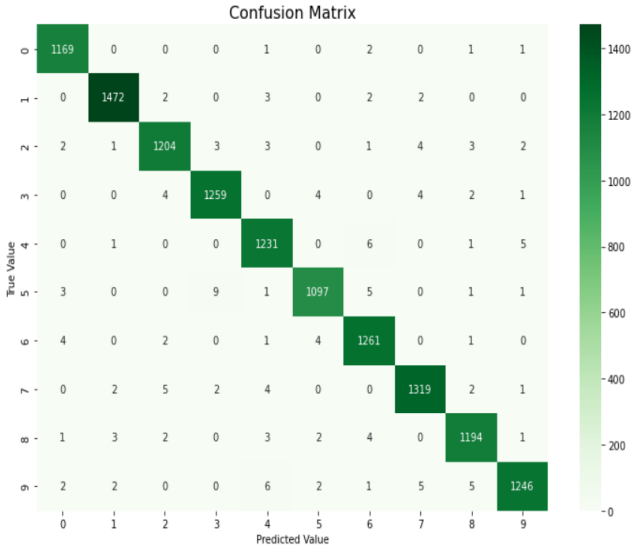


Fig. 11. Confusion Matrix

The model is then used to predict the class of random images used as tests and the output is verified for accuracy. The following shows a sample of the output predicted by the model built.

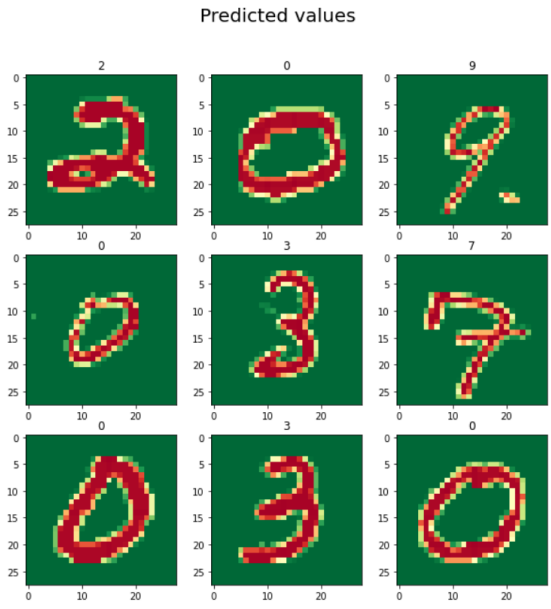


Fig. 12. Predition 1

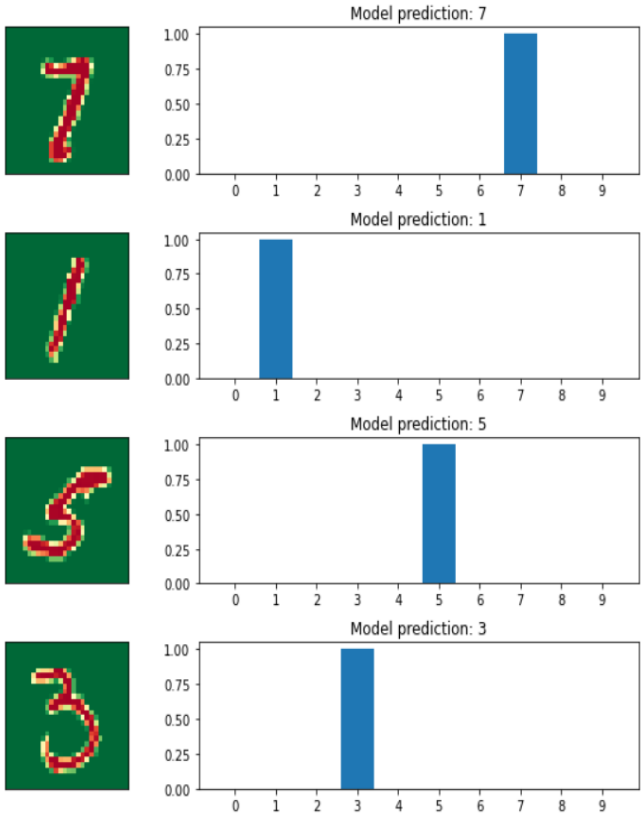


Fig. 13. Prediction 2

## V. Comparison with Existing Models

As a comparison, we looked at some existing CNN models found on the internet, which omitted one or more critical layers or features. Various options and methods exist for building our customized CNN architecture. It is more important to consider the model's complexity than its accuracy.[8]

In Model-1, there are three pairs of convolutional subsampling, but there are no advanced features. In the second case, 32 maps are used in the first layer and 64 maps are used in the second layer. It is not worth the additional computation cost to build architectures with more maps since they only perform slightly better. Only the dense layer and learnable subsampling layers are included in the third model. Additionally, batch normalization is used in the third model.[8]

| CNN Model | Accuracy |
|---|---|
| Model 1 | 99.14 % |
| Model 2 | 99.27 % |
| Model 3 | 99.14 % |
| Proposed Model | 99.6 % |

Fig. 14. Model Accuracy Comparison

## VI. Future Work

Various parameters and layers of the CNN model can be compared and developed to further increase accuracy and efficiency. It is possible to examine many CNN architectures, including hybrid CNN models like CNN-RNN and CNN-HMM, as well as domain-specific recognition systems. The number of layers, learning rate, and kernel sizes of

convolutional filters are three CNN learning parameters that can be optimized using evolutionary algorithms.

## VII. Conclusion

In this paper, a CNN model has been implemented. This deep neural network type has two convolutional layers. Convolutional layers provide several types of visual features of the image, as is well known. In this work, we downloaded a batch of random image data, tested the model, and discovered the important findings that were covered in the previous section.

## VIII. References


[1] Y Bengio, A Courville and P Vincent, "Representation Learning: A Review and New Perspectives", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2018.

[2] Yann LeCun and Yoshua Bengio, "Convolutional networks for images speech and time series" in The handbook of brain theory and neural networks, pp. 276-278, 1995.

[3] D Yu, L Ma and H Lu, "Lottery Digit Recognition Based on Multi-features", IEEE Systems and Information Engineering Design Symposium, pp. 1-4, 2007.

[4] A. Garg, D. Gupta, S. Saxena and P. P. Sahadev, "Validation of Random Dataset Using an Efficient CNN Model Trained on MNIST Handwritten Dataset," 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), 2019, pp. 602-606, doi: 10.1109/SPIN.2019.8711703.

[5] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). IEEE.

[6] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, “Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN),” Sensors, vol. 20, no. 12, p. 3344, Jun. 2020, doi: 10.3390/s20123344.

[7] Dan Ciregan, Ueli Meier and Jürgen Schmidhuber, "Multi-column deep neural networks for image classification", 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642-3649, 2012.

[8] Cdeotte, “How to choose CNN architecture mnist,” *Kaggle*, 24-Sep-2018. [Online]. Available: https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist/notebook. [Accessed: 17-Oct-2022].