International Journal of Research in Engineering, Science and Management
Volume 5, Issue 1, January 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

Hand Written Digit Recognition Using Machine Learning

Atmakuri Venkata Siva Rama Praneeth[1*], Lokesh Reddy Pappu[2], Gannarapu Pavan Kumar[3], Aparna Sachidanand Bhatta[4], Boddu Rashmi Chowdary[5], Gande Sai Shishwan[6]

***Abstract*: In current instances, with the growth of Artificial Neural Network (ANN), deep getting to know has delivered a dramatic twist in the area of system gaining knowledge of by means of making it more artificially intelligent. Deep learning is remarkably used in big degrees of fields due to its numerous range of packages which include surveillance, health, medication, sports activities, robotics, drones, and so forth. In deep studying, Convolutional Neural Network (CNN) is on the center of remarkable advances that combines Artificial Neural Network (ANN) and up to date deep getting to know techniques. It has been used extensively in sample recognition, sentence category, speech recognition, face recognition, textual content categorization, report evaluation, scene, and handwritten digit reputation. The intention of our project is to examine the variant of accuracies of CNN to categorize handwritten digits the use of numerous numbers of hidden layers and epochs and to make the comparison among the accuracies. In this undertaking, we've got chosen to attention on recognizing handwritten digits to be had within the MNIST database.**

***Keywords*: Machine Learning, Digit recognition.**

1. Introduction

A simple artificial neural community (ANN) has an enter layer, an output layer and some hidden layers among the enter and output layer. CNN has a totally similar architecture as ANN. There are several neurons in every layer in ANN. The weighted sum of all the neurons of a layer will become the center of a neuron of the subsequent layer adding a biased fee. In CNN the layer has 3 dimensions. Here all the neurons aren't absolutely related. Instead, every neuron within the layer is attached to the nearby receptive area. A fee characteristic generates with a purpose to train the community. It compares the output of the community with the preferred output. The sign propagates again to the machine, time and again, to replace the shared weights and biases in all of the receptive fields to decrease the cost of cost characteristic which will increase the network's overall performance. The intention of this newsletter is to examine the have an effect on of hidden layers of a CNN for handwritten digits.

Handwritten digit recognition is the capability of a computer to apprehend the human handwritten digits from distinctive sources like photographs, papers, contact monitors, and many others, and classify them into 10 predefined instructions (0-9). This has been a subject matter of boundless-research within the discipline of deep studying. Digit reputation has many packages like wide variety plate recognition, postal mail sorting, bank check processing, and so on [2]. In Handwritten digit recognition, we are facing many challenges because of distinct types of writing of various peoples because it isn't an Optical individual recognition. This studies gives a complete assessment among different device mastering and deep mastering algorithms for the purpose of handwritten digit recognition. For this, we've used Support Vector Machine, Multilayer Perceptron, and Convolutional Neural Network. The comparison between these algorithms is finished on the premise in their accuracy, errors, and testing-training time corroborated via plots and charts that had been built the usage of matplotlib for visualization. The accuracy of any version is paramount as greater accurate models make higher choices. The models with low accuracy are not suitable for real-international programs. Ex. For an automated financial institution cheque processing machine in which the machine recognizes the amount and date at the take a look at, high accuracy could be very vital. If the gadget incorrectly recognizes a digit, it can cause foremost harm which isn't suited. That's why a set of rules with high accuracy is required in these realworld programs. Hence, we are providing an assessment of various algorithms based on their accuracy in order that the most accurate algorithm with the least probabilities of mistakes can be employed in diverse programs of handwritten digit popularity. This paper offers an affordable information of device studying and deep learning algorithms like SVM, CNN, and MLP for handwritten digit popularity. It furthermore gives you the information approximately which set of rules is efficient in appearing the venture of digit reputation. In in addition sections of this paper, we can be discussing the related paintings that has been finished on this discipline observed by the technique and implementation of all the three algorithms for the fairer expertise of them. Next, it presents the conclusion and result strengthened via the work we have completed in this paper. Moreover, it's going to also provide you with some potential future upgrades that may be performed on this area.

2. Machine Learning Algorithm

A. CNN (Convolutional Neural Network)

Convolution Neural Networks or covnets are neural

*Corresponding author: praneeth.avsr123@gmail.com

networks that percentage their parameters. Imagine you've got an image. It can be represented as a cuboid having its duration, width (dimension of the picture) and height (as image normally have red, inexperienced, and blue channels). Now believe taking a small patch of this picture and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as an end result, we will get some other photo with exclusive width, peak, and depth. Instead of simply R, G and B channels now we have more channels but lesser width and peak. His operation is known as Convolution. If patch size is equal as that of the photo it'll be an ordinary neural community. Because of this small patch, we've fewer weights.

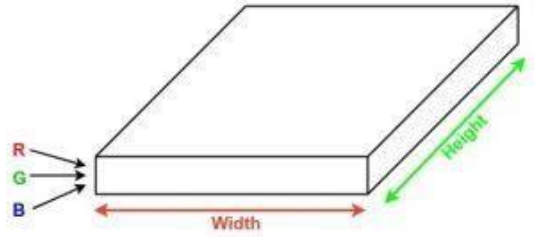

Fig. 1.

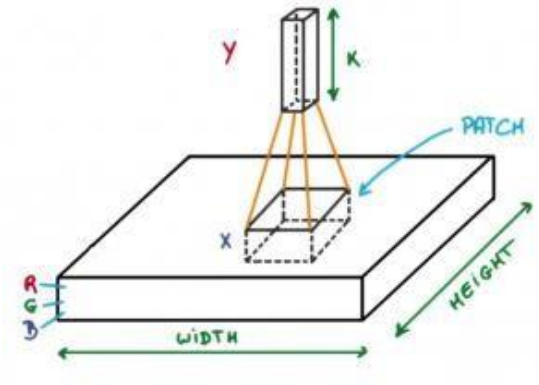

Fig. 2.

A covnets is a chain of layers, and each layer transforms one extent to any other via differentiable function.

Types of layers:

Let's take an example via going for walks a covnets on of picture of measurement 32 x 32 x 3.

1. Input Layer
2. Convolution Layer
3. Activation Function Layer
4. Pool Layer

3. Dataset Used

The dataset was constituted of a number of scanned record dataset available from the National Institute of Standards and Technology (NIST). This is in which the call for the dataset comes from, because the Modified NIST or MNIST dataset.

Images of digits had been taken from an expansion of scanned files, normalized in length and focused. This makes it an wonderful dataset for evaluating fashions, allowing the developer to recognition on the machine learning with little or no information cleaning or instruction required.

Each photograph is a 28 via 28 pixels rectangular (784 pixels total). A general split of the dataset is used to evaluate and examine models, wherein 60,000 snap shots are used to train a version and a separate set of 10,000 pix are used to check it.

4. Methodology

- Load the MNIST dataset in Keras and generate plots of the dataset.

The Keras deep gaining knowledge of library gives a comfort technique for loading the MNIST dataset. The dataset is downloaded robotically the first time this feature is known as and is saved in your house directory in ~/.Keras/datasets/mnist.Pkl.Gz as a 15MB report.

- Reshape the MNIST dataset and develop an easy but nicely performing multi-layer perceptron version on the problem. Train the version with a TensorFlow optimizer, set if for check and train and reshape it to size.
- Use Keras to create convolutional neural community fashions for MNIST. Define a larger version to create a model using tensor flow backend, Save it as cnn.

Larger Convolutional Neural Network:

Import training and function then load and put together the records for CNN. We outline a huge CNN structure with extra convolutional, max pooling layers and fully related layers.

The network topology may be summarized as follows.

1. Convolutional layer with 30 function maps of size 5×5.
2. Pooling layer taking the max over 2*2 patches.
3. Convolutional layer with 15 characteristic maps of length 3×3
4. Pooling layer taking the max over 2*2 patches.
5. Dropout layer with an opportunity of 20%.
6. Flatten layer.
7. Fully connected layer with 128 neurons and rectifier activation.
8. Fully connected layer with 50 neurons and rectifier activation.
9. Output layer.

The model is suit over 10 epochs with a batch size of 2 hundred.

Running the instance prints accuracy on the schooling and validation datasets each epoch and a very last type mistakes rate.

The model takes about one hundred seconds to run according to epoch. This slightly larger version achieves the respectable type blunders charge of 0. Eighty-three%.

Convolutional Neural Network:

CNN is a deep studying set of rules this is widely used for photograph recognition and category. It is a class of deep neural networks that require minimum pre-processing. It inputs the picture inside the shape of small chunks as a substitute than inputting an unmarried pixel at a time, so the community can discover unsure styles (edges) inside the photo greater

correctly. CNN contains three layers specifically, an input layer, an output layer, and a couple of hidden layers which encompass Convolutional layers, Pooling layers (Max and Average pooling), Fully related layers (FC), and normalization layers [12]. CNN uses a filter out (kernel) that is an array of weights to extract functions from the enter photograph. CNN employs distinct activation features at every layer to feature a few non-linearity [13]. As we circulate into the CNN, we have a look at the height and width decrease at the same time as the variety of channels will increase. Finally, the generated column matrix is used to predict the output [14]

Flask:

- First we imported the Flask magnificence. An instance of this elegance will be our WSGI software. Next we created an instance of this class.
- We then used the path() decorator to inform Flask what URL ought to trigger our function.
- We then routed predict to get canvas statistics form the person and made a graph of it to evaluate and reshape the photo statistics to be used in neural network.
- After the picture changed into read lower back in eight-bit black and white model we it writes the output again to the person

CSS & JavaScript:

- We used html and css to make the basic styled page for the user to add to the url.
- To create canvas we used on paint function in js and to get user input.
- It was ased on mouse click and mouse last option with stroke and close path.
- And then it was linked to our page.

5. Conclusion

We were able to achieve an accuracy above 95% and predicted the output using CNN on our WebApp as shown in the figures. In the future, we plan to look at the variant inside the usual category accuracy via varying the quantity of hidden layers and batch size.

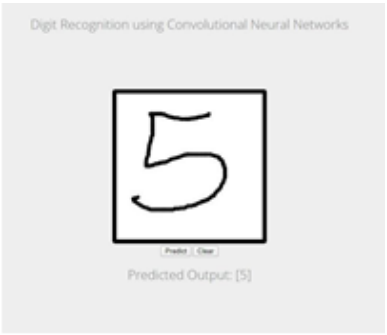

Fig. 3.

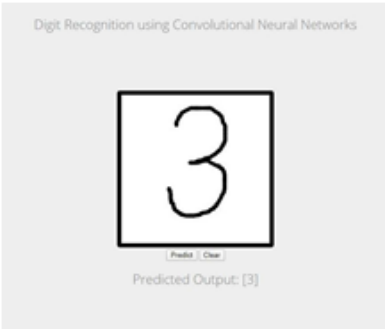

Fig. 4.

```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
K.set_image_dim_ordering('th')
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

# define the larger model
def larger_model():
        # create model
        model = Sequential()
        model.add(Conv2D(30, (5, 5), input_shape=(1, 28, 28), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Conv2D(15, (3, 3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.2))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dense(50, activation='relu'))
        model.add(Dense(num_classes, activation='softmax'))
        # Compile model
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
```

```
from flask import Flask, render_template, request
from scipy.misc import imsave, imread, imresize
import numpy as np
import keras.models
import re
import base64

import sys
import os
sys.path.append(os.path.abspath("./model"))
from load import *

app = Flask(__name__)
global model, graph
model, graph = init()

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict/', methods=['GET','POST'])
def predict():
    # get data from drawing canvas and save as image
    parseImage(request.get_data())

    # read parsed image back in 8-bit, black and white mode (L)
    x = imread('output.png', mode='L')
    x = np.invert(x)
    x = imresize(x,(28,28))
     # reshape image data for use in neural network
    x = x.reshape(1,1,28,28)

    with graph.as_default():
        out = model.predict(x)
        print(out)
        print(np.argmax(out, axis=1))
        response = np.array_str(np.argmax(out, axis=1))
        return response

def parseImage(imgData):
    # parse canvas bytes and save as output.png
    imgstr = re.search(b'base64,(.*)', imgData).group(1)
    with open('output.png','wb') as output:
        output.write(base64.decodebytes(imgstr))

if __name__ == '__main__':
    app.debug = True
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)
```

References

[1] R. Ting, S. Chun-lin and D. Jian, "Handwritten character recognition using principal component analysis", MINI-MICRO Systems, no. 2, pp. 289-292, 2005.

[2] R. Walid and A. Lasfar, "Handwritten digit recognition using sparse deep architectures. in Intelligent Systems: Theories and Applications (SITA-14)", 2014 9th International Conference on, 2014.

[3] Z. Li et al., "Handwritten digit recognition via active belief decision trees", Control Conference (CCC) 2016 35th Chinese, 2016.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview", Neural Networks, vol. 61, pp. 85-117, 2015.
[5] Y. LeCun, Y. Bengio and G. Hinton, Deep learning. Nature, no. 7553, pp. 436-444, 2015.
[6] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in Competition and cooperation in neural nets: Springer, 1982, pp. 267-285.
[7] Y. LeCun et al., "Handwritten digit recognition with a backpropagation network," in Advances in neural information processing systems, 1990, pp. 396-404.
[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradientbased learning applied to document recognition," Proceedings ofthe IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
[9] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in Neural networks for perception: Elsevier, 1992, pp. 65-93.
[10] Y. LeCun, "LeNet-5, convolutional neural networks," URL: http://yann. lecun. com/exdb/lenet, vol. 20, 2015.
[11] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016.
[12] S. Haykin, "Neural networks: A comprehensive foundation: MacMillan College," New York, 1994.
[13] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," IEEE Transactions on Semiconductor Manufacturing, vol. 30, no. 2, pp. 135-142, 2017.
[14] K. G. Pasi and S. R. Naik, "Effect of parameter variations on accuracy of Convolutional Neural Network," in 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016, pp. 398-403: IEEE.
[15] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in Twenty-Second International Joint Conference on Artificial Intelligence, 2011.
[16] K. Isogawa, T. Ida, T. Shiodera, and T. Takeguchi, "Deep shrinkage convolutional neural network for adaptive noise reduction," IEEE Signal Processing Letters, vol. 25, no. 2, pp. 224-228, 2018.
[17] C. C. Park, Y. Kim, and G. Kim, "Retrieval of sentence sequences for an image stream via coherence recurrent convolutional networks," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 945-957, 2018.
[18] Y. Yin, J. Wu, and H. Zheng, "Ncfm: Accurate handwritten digits recognition using convolutional neural networks," in 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 525-531: IEEE.
[19] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing cnn on the loss layer," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4753-4762.
[20] A. Tavanaei and A. S. Maida, "Multi-layer unsupervised learning in a spiking convolutional neural network," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2023-2030: IEEE.
[21] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 1991- 2000, 2014.