COMPUTER VISION

Project-1

Program: Handwritten digits recognition using Convolutional Neural Networks

Name: **NIKHIL CHEZIAN**

Matriculation Number: **11027706**

Professor: **Dr. MILAN GNJATOVIC**

Date: **1st June 2023**

ABSTRACT

The field of computer vision has witnessed significant advancements in solving complex tasks such as handwritten digit recognition. Handwritten digit recognition plays a crucial role in various applications, from postal mail sorting to bank check processing and digit-based authentication systems. With the advancement of computer vision techniques, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for solving complex image recognition tasks. This project focuses on developing a CNN-based model specifically tailored for accurate classification of handwritten digits. By utilizing the MNIST dataset, which contains a diverse range of labeled digit images, the model learns to identify underlying patterns and features that differentiate different digits. The training process involves optimizing the model's parameters to minimize loss and maximize classification accuracy. The evaluation of the model's performance is conducted using well-established metrics such as accuracy and loss. Additionally, practical validation is performed using a set of PNG image files representing handwritten digits.

INTRODUCTION

A convolutional neural network consists of an input layer, hidden layers and an output layer. In a convolutional neural network, the hidden layers include one or more layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix.

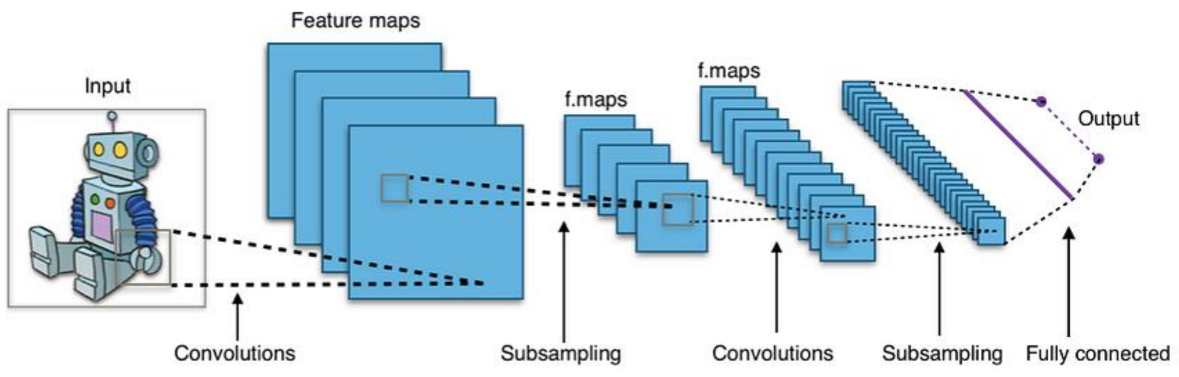


Fig.1: Illustration of principle of Convolutional Neural Network

In this project, a Convolutional Neural Network (CNN) model was developed to accurately classify handwritten digits from the MNIST dataset. The model is trained on a large collection of labeled digit images and learns to extract meaningful features from the input data. Through an iterative training process, the model optimizes its parameters to minimize the loss and improve classification accuracy. The trained model achieves a high accuracy rate on the test set, demonstrating its effectiveness in recognizing handwritten digits. Additionally, a set of PNG image files representing handwritten digits are used to validate the model's performance in a real-world scenario.

DATASET

The MNIST dataset is a widely used benchmark dataset for handwritten digit recognition. It consists of 60,000 training images and 10,000 testing images, each of size 28x28 pixels. The dataset is preprocessed by reshaping the images into a suitable format and normalizing the pixel values between 0 and 1.

MODEL ARCHITECTURE

The CNN architecture for this project is based on the LeNet-5 architecture proposed by Yann LeCun et al. in 1998. The LeNet-5 model was chosen because it was specifically designed for handwritten digit recognition tasks, such as the classification of digits in the MNIST dataset. The LeNet-5 architecture consists of convolutional layers followed by subsampling layers, and then fully connected layers for classification. Similarly, this code defines a model with Conv2D layers, MaxPooling2D layers, and Dense layers for classification. It is designed to extract relevant features from the input images and make accurate predictions.

This model consists of multiple layers, each serving a specific purpose in the overall architecture.

1. The first layer is a Conv2D layer with 32 filters and a kernel size of (3, 3). This layer applies a set of filters to the input image, convolving them across the image to capture local patterns and features. The ReLU activation function is used to introduce non-linearity to the network.

2. Following the Conv2D layer, a MaxPooling2D layer is employed with a pool size of (2, 2). This layer reduces the spatial dimensions of the input feature maps, effectively downsampling the information while retaining the most important features. Max pooling helps to extract robust and invariant features by selecting the maximum value within each pooling region.

3. The next Conv2D layer has 64 filters and a kernel size of (3, 3). Similar to the previous Conv2D layer, it performs convolutions on the feature maps to capture higher-level features and patterns. Again, the ReLU activation function is used to introduce non-linearity.

4. Another MaxPooling2D layer with a pool size of (2, 2) follows the second Conv2D layer, further reducing the spatial dimensions and extracting more abstract features from the feature maps.

5. To prepare the extracted features for classification, a Flatten layer is applied. This layer reshapes the 2D feature maps into a 1D vector, effectively "flattening" the data.

6. Two fully connected Dense layers with 128 units each are added to introduce more non-linearity and learn complex relationships within the data. The ReLU activation function is used in these layers to introduce non-linearity and help the model approximate the target function more effectively.

7. Finally, a Dense layer with 10 units is employed, representing the number of classes in the classification task (digits 0 to 9). The softmax activation function is used in this layer to generate a probability distribution over the classes, enabling the model to assign a probability to each class and make predictions based on the highest probability.

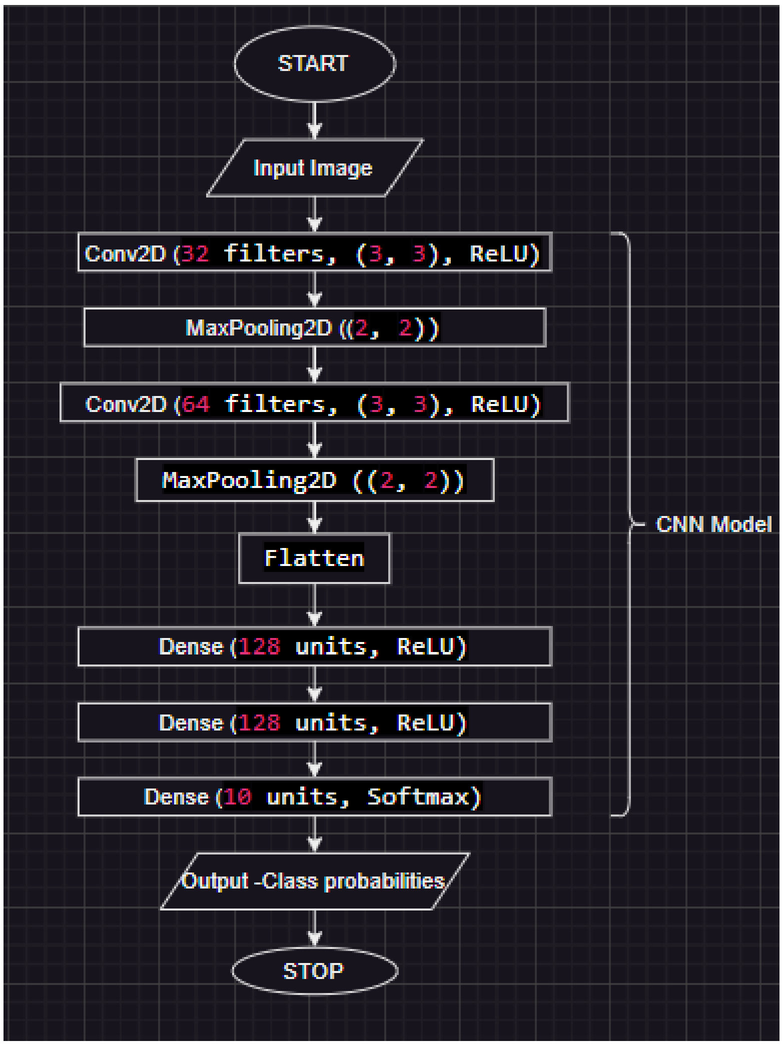


Fig.2: Pictorial representation of model architecture developed in this project.

This code uses two Conv2D layers with ReLU activation, followed by MaxPooling2D layers for downsampling. Then, the feature maps are flattened and connected to Dense layers for classification.

Overall, this architecture allows the model to learn and extract increasingly complex features from the input images, enabling accurate classification of handwritten digits. The combination of convolutional and pooling layers helps the model capture local and global patterns, while the fully connected layers facilitate high-level feature learning and classification.

MODEL TRAINING

The model is compiled with the Adam optimizer, sparse categorical cross-entropy loss function, and accuracy as the evaluation metric.

- Adam optimizer - is an algorithm used to update the weights of a neural network during training by adapting the learning rate for each parameter based on estimates of first and second moments of the gradients.
- Sparse categorical cross-entropy - is a loss function commonly used in classification tasks where the classes are mutually exclusive and represented by integers, providing a measure of dissimilarity between the predicted class probabilities and the true class labels.
- Loss - is a measure of how well the predictions of a model align with the true values during training, indicating the degree of error in the model's output.
- Accuracy - is a metric that quantifies the performance of a classification model by calculating the proportion of correct predictions compared to the total number of predictions made.

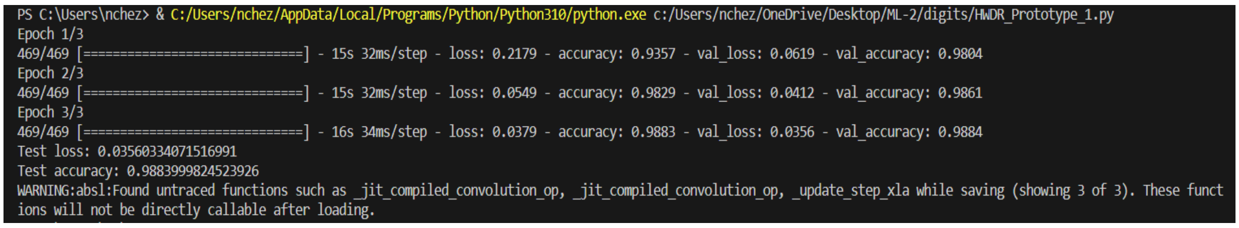
It is then trained on the MNIST dataset with a batch size of 128 and three epochs.

- Epoch - in machine learning refers to a complete iteration or pass through the entire training dataset during the training phase of a model. During each epoch, the model goes through all the training examples and updates its parameters based on the calculated loss and the chosen optimization algorithm.
- Batch Size – 128 - it is the number of training examples used in a single forward and backward pass during each iteration of an epoch. Instead of updating the model's parameters after processing each individual example (batch size of 1).

The training process involves displaying progress updates and evaluating the model's performance on the validation data. After training, the model is evaluated on the test set, providing the loss and accuracy metrics. Finally, the trained model is loaded, and predictions are made on a set of PNG images representing handwritten digits. The predicted class and corresponding image are displayed for each prediction.

RESULTS & ANALYSIS

The trained model achieved an impressive test accuracy of 0.9884, indicating its high performance in accurately recognizing handwritten digits. The high accuracy rate showcases the effectiveness of convolutional neural networks (CNNs) in image recognition tasks and underscores their relevance in solving real-world problems. In this case, it indicates that the chosen model architecture, along with the training configuration and optimization technique (Adam optimizer), effectively learned and extracted meaningful features from the input images. The low test loss value of 0.0356 further confirms the model's ability to minimize errors and make accurate predictions.



```
PS C:\Users\nchez> & C:/Users/nchez/AppData/Local/Programs/Python/Python310/python.exe c:/Users/nchez/OneDrive/Desktop/ML-2/digits/HWDR_Prototype_1.py
Epoch 1/3
469/469 [==============================] - 15s 32ms/step - loss: 0.2179 - accuracy: 0.9357 - val_loss: 0.0619 - val_accuracy: 0.9804
Epoch 2/3
469/469 [==============================] - 15s 32ms/step - loss: 0.0549 - accuracy: 0.9829 - val_loss: 0.0412 - val_accuracy: 0.9861
Epoch 3/3
469/469 [==============================] - 16s 34ms/step - loss: 0.0379 - accuracy: 0.9883 - val_loss: 0.0356 - val_accuracy: 0.9884
Test loss: 0.03560334071516991
Test accuracy: 0.9883999824523926
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _update_step_xla while saving (showing 3 of 3). These funct
ions will not be directly callable after loading.
```

Fig.3: The model was evaluated to obtain an accuracy of 98.8%

IMAGE VERIFICATION

The trained model is used to verify a set of 10 PNG image files, where each image represents a handwritten digit. The images are loaded, preprocessed, and fed into the model for prediction. The predicted class for each image is obtained, and the image along with the predicted class is displayed for visual inspection.

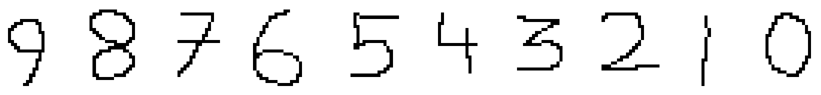
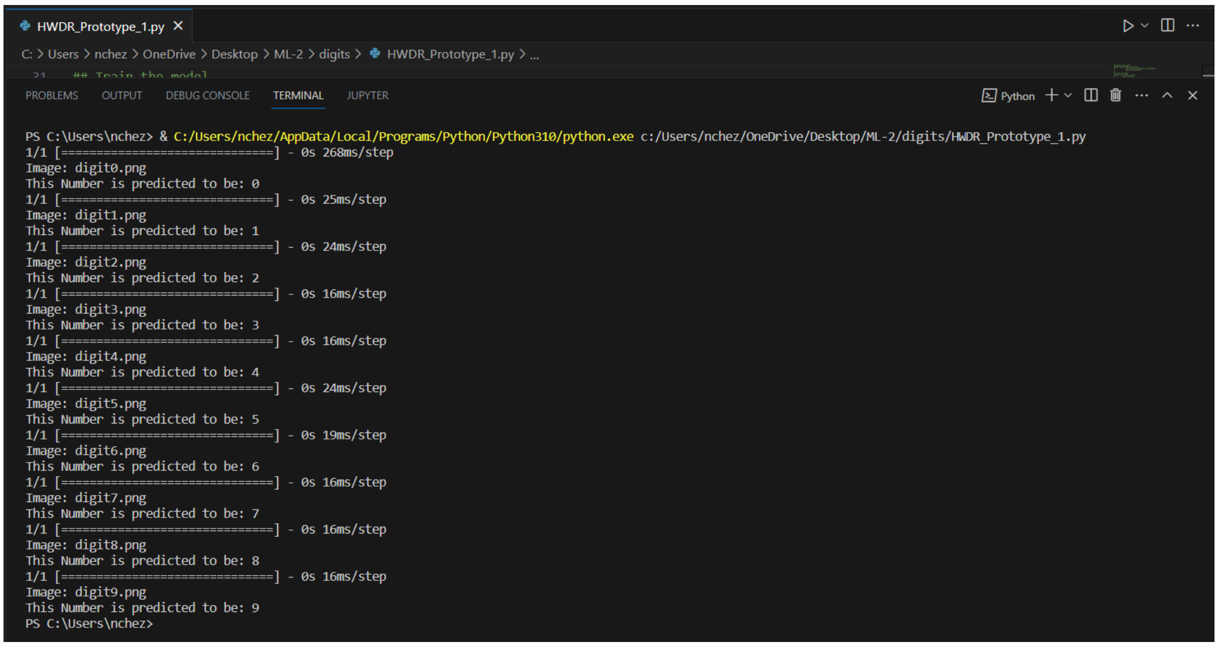


Fig.4.A: Handwritten Input Images of digits 0-9; 28x28 Pixels, PNG-Format.


HWDR_Prototype_1.py
C: > Users > nchez > OneDrive > Desktop > ML-2 > digits > HWDR_Prototype_1.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
Python

```
PS C:\Users\nchez> & C:/Users/nchez/AppData/Local/Programs/Python/Python310/python.exe c:/Users/nchez/OneDrive/Desktop/ML-2/digits/HWDR_Prototype_1.py
1/1 [==============================] - 0s 268ms/step
Image: digit0.png
This Number is predicted to be: 0
1/1 [==============================] - 0s 25ms/step
Image: digit1.png
This Number is predicted to be: 1
1/1 [==============================] - 0s 24ms/step
Image: digit2.png
This Number is predicted to be: 2
1/1 [==============================] - 0s 16ms/step
Image: digit3.png
This Number is predicted to be: 3
1/1 [==============================] - 0s 16ms/step
Image: digit4.png
This Number is predicted to be: 4
1/1 [==============================] - 0s 24ms/step
Image: digit5.png
This Number is predicted to be: 5
1/1 [==============================] - 0s 19ms/step
Image: digit6.png
This Number is predicted to be: 6
1/1 [==============================] - 0s 16ms/step
Image: digit7.png
This Number is predicted to be: 7
1/1 [==============================] - 0s 16ms/step
Image: digit8.png
This Number is predicted to be: 8
1/1 [==============================] - 0s 16ms/step
Image: digit9.png
This Number is predicted to be: 9
PS C:\Users\nchez>
```

Fig.4.B: The 9 input images of digits from 0-9 were correctly predicted.

CONCLUSION

In conclusion, this project successfully develops a CNN model for handwritten digit recognition using the MNIST dataset. The model demonstrates high accuracy in classifying handwritten digits, showcasing the effectiveness of CNNs in image recognition tasks. The project lays the foundation for further improvements and applications in digit recognition.

References

1. http://gnjatovic.info/imageprocessing/
2. Keras documentation: https://keras.io/
3. MNIST dataset: http://yann.lecun.com/exdb/mnist/
4. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
5. Convolutional Neural Networks for Visual Recognition by Andrej Karpathy: https://cs231n.github.io/convolutional-networks/
6. This video was taken as reference to construct the model architecture's skeletal code. Procedure to create a CNN model, saving, and calling it to further validate the model. Concept to create input image set for digits(0-9) in paint and feed it to predict class labels. https://www.youtube.com/watch?v=bte8Er0QhDg&t=766s
   Practice model created from the video: **'handwritten.model'**

```
#mnist= tf.keras.datasets.mnist
#(x_train, y_train), (x_test, y_test)= mnist.load_data()
#
#x_train= tf.keras.utils.normalize(x_train, axis=1)
#x_test= tf.keras.utils.normalize(x_test, axis=1)
#
#model = tf.keras.models.Sequential()
#model.add( tf.keras.layers.Flatten(input_shape=(28, 28)))
#model.add(tf.keras.layers.Dense(128, activation='relu'))
#model.add(tf.keras.layers.Dense(128, activation='relu'))
#model.add(tf.keras.layers.Dense(10, activation='softmax'))
#
#model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
#model.fit(x_train, y_train, epochs=3)
#print('loss=','sparse_categorical_crossentropy')
#print ('accuracy')
#model.save('handwritten.model')
#model = tf.keras.models.load_model('handwritten.model')
model = tf.keras.models.load_model('handwritten.model')
```

   Independent model created after due diligence, as present in the source code: **'handwrittenpt1.model'**
7. https://chat.openai.com/
   - Concept of functions like dense, dropout, etc.- Line: 6.
   - Deriving the syntax for LeNet5 model Architecture- Lines:22-30.
   - Deriving the syntax for updated model fitting- Lines: 40-44.
   - Deriving the syntax for image reshaping with appropriate parameters- Line: 74.
   - Apart from the above-mentioned lines, ChatGPT was mainly used to debug errors like installing missing libraries and understanding importing functions.

**** The above lines of code were extracted from ChatGPT to adhere to syntax, however the methods used were determined after due diligence.**

¶¶¶ Thank You for your Patience ¶¶¶