Bahria University

Karachi Campus

PROJECT REPORT

PROJECT TITLE: "HANDWRITTEN DIGIT RECOGNIZER"

COURSE: CSL-411 ARTIFICIAL INTELLIGENCE LAB
TERM: FALL 2020, CLASS: BSE- 5(B)

Submitted By:

Shizrah Khalid	**57229**
(Name)	**(Reg. No.)**

Submitted To:
Engr. Muhammad Rehan Baig

Signed **Remarks:** **Score:**

Table of Contents

ABSTRACT

Machine Learning is one of the most popular approaches in Artificial Intelligence. Over the past decade, Machine Learning has become one of the integral parts of our life. It is implemented in a task as simple as recognizing human handwriting or as complex as self-driving cars. It is also expected that in a couple of decades, the more mechanical repetitive task will be over. With the increasing amounts of data becoming available there is a good reason to believe that Machine Learning will become even more prevalent as a necessary element for technological progress.

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. Classification problems include:

- Given an example, classify if it is spam or not.
- Given a handwritten character, classify it as one of the known characters.
- Given recent user behavior, classify as churn or not.

From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn. A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label.

INTRODUCTION

To make machines more intelligent, the developers are diving into machine learning and deep learning techniques. A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks. Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. For example – object recognition, image and sound classification, object detection, image segmentation, etc.

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

I have implemented a handwritten digit recognition system using the MNIST dataset and a special type of deep neural network that is Convolutional Neural Networks. I have built GUI in which we can draw the digit and recognize it straight away. For deep learning I used Keras, tensorflow, numpy and pillow libraries and the Tkinter library for building GUI.

ABOUT DATASET

For this project I have used The MNIST dataset. Abbreviation for MNIST is Modified National Institute of Standards and Technology database. It is a huge dataset of handwritten digits that is commonly used for training various image classification systems. This is a dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. It is considered to be a good database for people who try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. The MNIST database of handwritten digits is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

Link: http://yann.lecun.com/exdb/mnist/

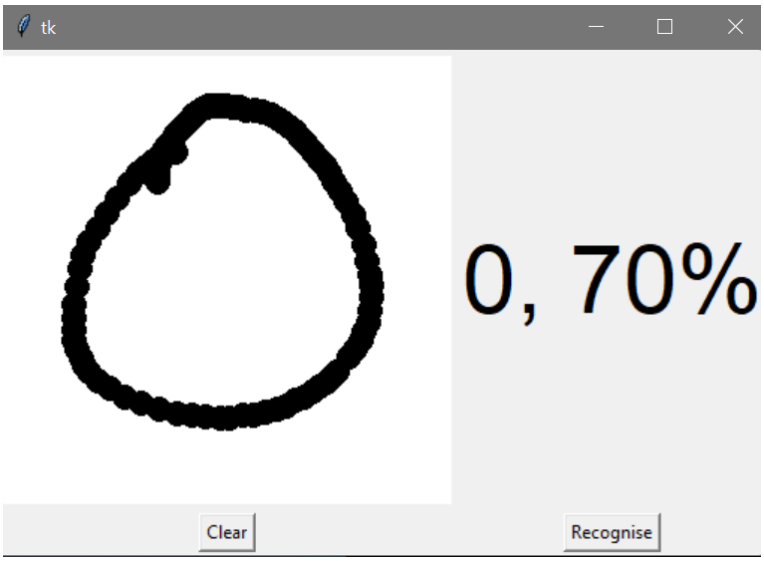
CODE

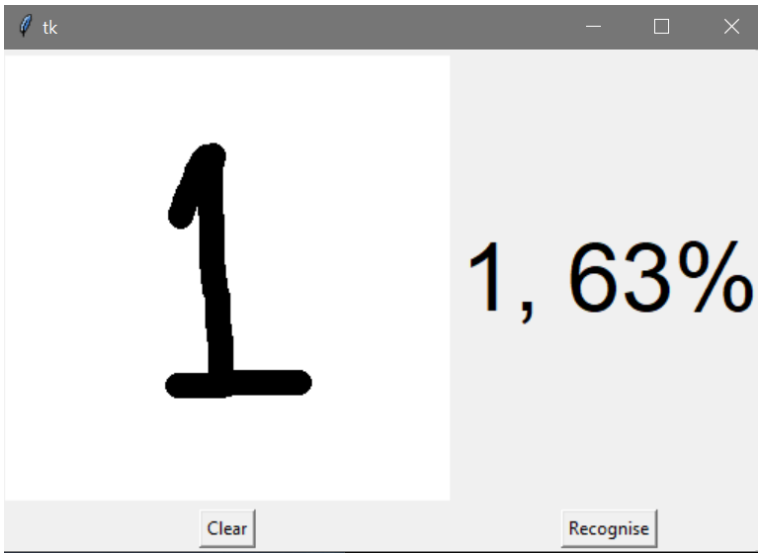
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape, y_train.shape)
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
batch_size = 128
num_classes = 10
epochs = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

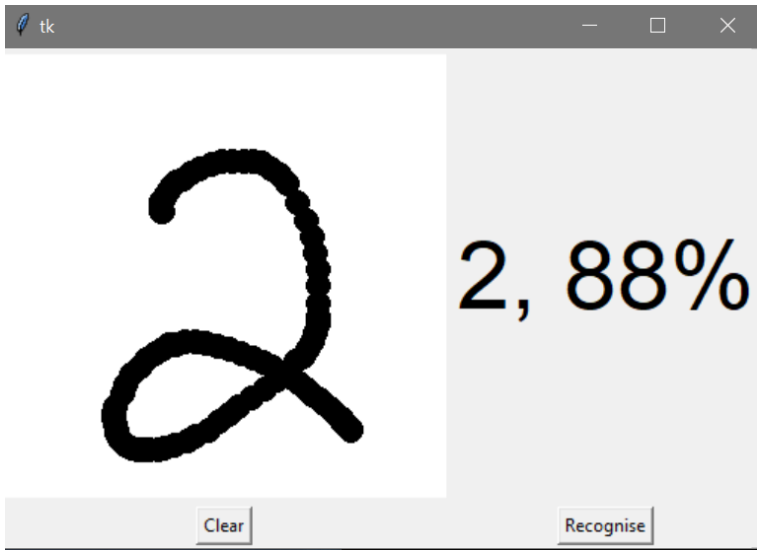
```
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])
hist = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
print("The model has successfully trained")
model.save('mnist.h5')
print("Saving the model as mnist.h5")
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])


from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np
model = load_model('mnist.h5')
def predict_digit(img):
    img = img.resize((28,28))
    img = img.convert('L')
    img = np.array(img)
    img = img.reshape(1,28,28,1)
    img = img/255.0
    res = model.predict([img])[0]
    return np.argmax(res), max(res)
class App(tk.Tk):
    def __init__(self):
```

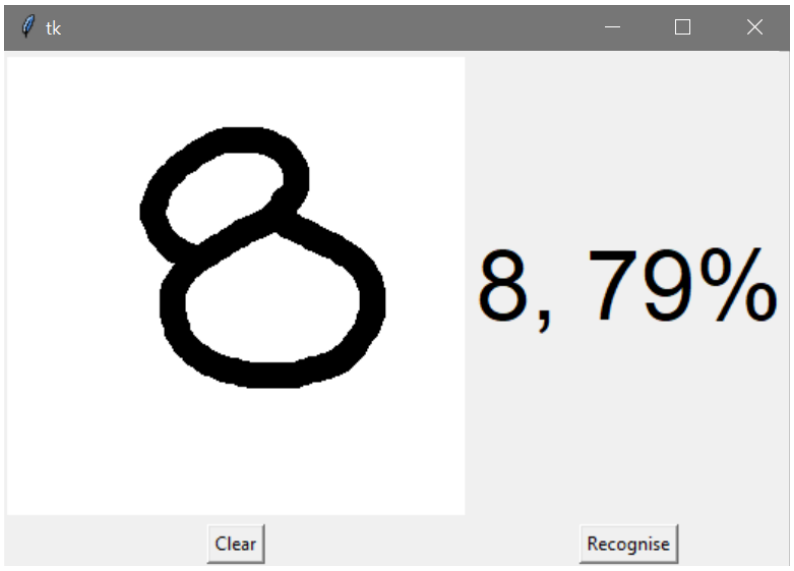
```
        tk.Tk.__init__(self)
        self.x = self.y = 0
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")
        self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise", command =        self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command = self.clear_all)
        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)
        self.canvas.bind("<B1-Motion>", self.draw_lines)
    def clear_all(self):
        self.canvas.delete("all")
    def classify_handwriting(self):
        HWND = self.canvas.winfo_id()
        rect = win32gui.GetWindowRect(HWND)
        im = ImageGrab.grab(rect)
        digit, acc = predict_digit(im)
        self.label.configure(text= str(digit)+', '+ str(int(acc*100))+'%')
    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')
app = App()
mainloop()
```

OUTPUT


tk
0, 70%
Clear
Recognise


tk
1, 63%
Clear
Recognise


tk
2, 88%
Clear
Recognise



Submitted by: Shizrah Khalid

Training;

```
C:\Windows\System32\cmd.exe
MPZQ2ME2ZZHJJRJ3JIKNDB7.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
(60000, 28, 28) (60000,)
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
2021-01-21 15:37:23.958649: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-01-21 15:37:23.967819: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'nvcuda.dll'; dlerror: nv
2021-01-21 15:37:23.972324: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuInit: UNKNOWN ERROR (303)
2021-01-21 15:37:23.982151: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: SK-Laptop
2021-01-21 15:37:23.991519: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: SK-Laptop
2021-01-21 15:37:23.996433: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network
 the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-01-21 15:37:24.008039: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-01-21 15:37:24.297675: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (regist
Epoch 1/10
469/469 [==============================] - 48s 100ms/step - loss: 2.3041 - accuracy: 0.1176 - val_loss: 2.2882 - val_accuracy: 0.1941
Epoch 2/10
469/469 [==============================] - 66s 140ms/step - loss: 2.2913 - accuracy: 0.1386 - val_loss: 2.2742 - val_accuracy: 0.2474
Epoch 3/10
469/469 [==============================] - 60s 129ms/step - loss: 2.2780 - accuracy: 0.1563 - val_loss: 2.2597 - val_accuracy: 0.3014
Epoch 4/10
469/469 [==============================] - 63s 134ms/step - loss: 2.2649 - accuracy: 0.1740 - val_loss: 2.2438 - val_accuracy: 0.3464
Epoch 5/10
469/469 [==============================] - 62s 132ms/step - loss: 2.2522 - accuracy: 0.1922 - val_loss: 2.2260 - val_accuracy: 0.3852
Epoch 6/10
469/469 [==============================] - 56s 119ms/step - loss: 2.2372 - accuracy: 0.2116 - val_loss: 2.2064 - val_accuracy: 0.4199
Epoch 7/10
469/469 [==============================] - 58s 123ms/step - loss: 2.2215 - accuracy: 0.2269 - val_loss: 2.1846 - val_accuracy: 0.4520
Epoch 8/10
469/469 [==============================] - 62s 133ms/step - loss: 2.2033 - accuracy: 0.2443 - val_loss: 2.1606 - val_accuracy: 0.4798
Epoch 9/10
469/469 [==============================] - 61s 130ms/step - loss: 2.1823 - accuracy: 0.2646 - val_loss: 2.1349 - val_accuracy: 0.4983
Epoch 10/10
469/469 [==============================] - 62s 132ms/step - loss: 2.1641 - accuracy: 0.2785 - val_loss: 2.1061 - val_accuracy: 0.5123
The model has successfully trained
Test loss: 2.1061065196990967
Test accuracy: 0.5123000144958496
Saving the model as mnist.h5
```

EXPLANATION

Handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a tough job as hand written digits are not perfect. In this project I have tried to combine the approach of KNN classification algorithm with CNNs to make my machine learning project more efficient with the help of keras which makes it easier to understand too. The idea is to build a machine learning project that can classify the handwritten digits.

KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras is a powerful and easy-to-use free open source Python library for developing and evaluating machine learning models. It wraps the efficient numerical computation libraries like TensorFlow and allows us to define and train neural network models in just a few lines of code.

Built on top of TensorFlow 2.0, Keras is an industry-strength framework that can scale to large clusters of GPUs or an entire TPU pod. It's not only possible but now it's easy too. We can export Keras models to JavaScript to run directly in the browser, to TF Lite to run on iOS, Android, and embedded devices. It's also easy to serve Keras models as via a web API.

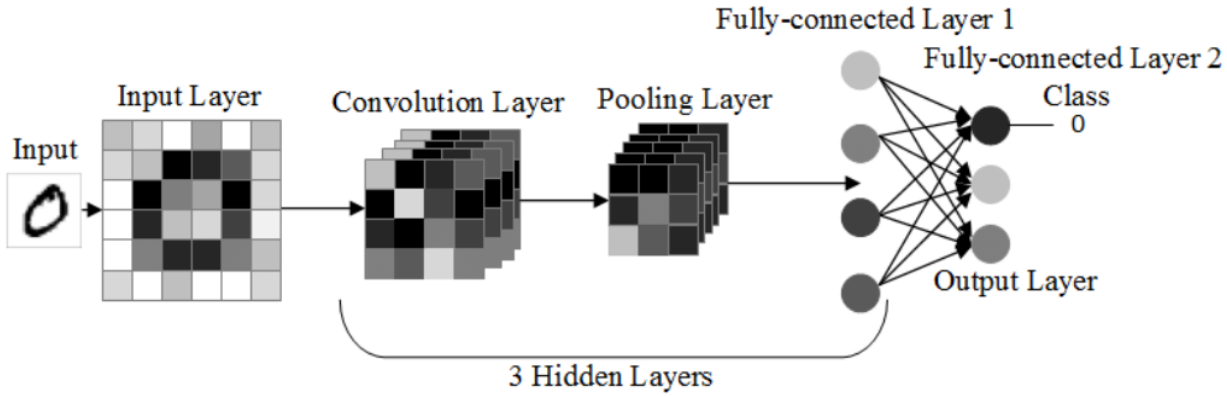
KNN AND CNN

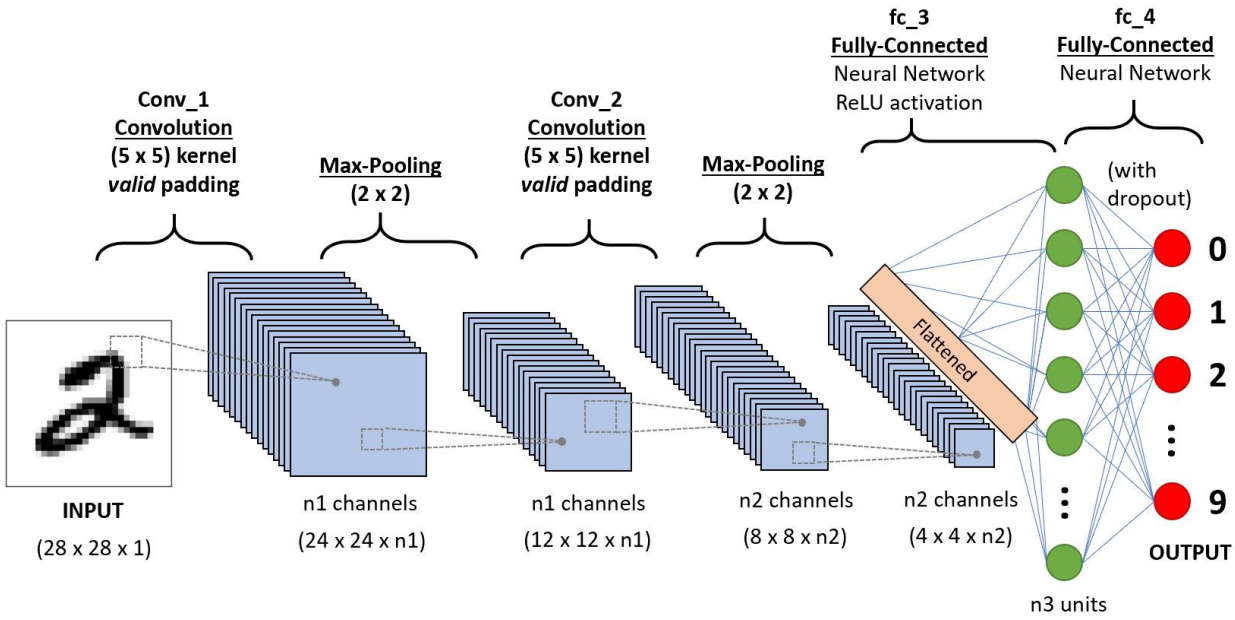
K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures. Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase. In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

Convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They have applications in image and video recognition, image classification, medical image analysis, natural language processing, brain-computer interfaces, etc. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. CNNs are known for their high connectedness and low complexity. CNNs use relatively little pre-processing

compared to other classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

BLOCK DIAGRAM





Convolutional layers convolve the input and pass its result to the next layer. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.

GITHUB LINK:

https://github.com/ShizrahKhalid/handwritten_digit_recognizer.git