**Assignment 2** (20%)

CSE 5120 (Section 01) – Introduction to Artificial Intelligence – Fall 2024

*Submitted to*

Department of Computer Science and Engineering  
California State University, San Bernardino, California

*by*

Cameron Hartling (007735059)

Date: *December 6, 2024*

*Email:*

* *007735059@coyote.csusb.edu*

**Assignment Report**

1. **digitRecognizer.py for MNIST dataset**

The MNIST dataset is a collection of 70,000 images of handwritten digits. 60,000 of the images are used for the training of the model and the remaining 10,000 images are used for the test set. All of the images are 28x28 pixels, grayscale, and centered.

digitRecognizer.py

# Handwritten digit recognition for MNIST dataset using Convolutional Neural Networks

# Step 1: Import all required keras libraries

from keras.datasets import mnist # This is used to load the mnist dataset later

from keras.utils import to\_categorical # This will be used to convert your test image to a categorical class (digit from 0 to 9)

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input

import numpy as np

import os

# Step 2: Load and return training and test datasets

def load\_dataset():

# 2a. Load dataset X\_train, X\_test, y\_train, y\_test via imported keras library

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# 2b. reshape for X train and test vars - Hint: X\_train = X\_train.reshape((X\_train.shape[0], 28, 28, 1)).astype('float32')

X\_train = X\_train.reshape((X\_train.shape[0], 28, 28, 1)).astype('float32')

X\_test = X\_test.reshape((X\_test.shape[0], 28, 28, 1)).astype('float32')

# 2c. normalize inputs from 0-255 to 0-1 - Hint: X\_train = X\_train / 255

X\_train = X\_train / 255

X\_test = X\_test / 255

# 2d. Convert y\_train and y\_test to categorical classes - Hint: y\_train = np\_utils.to\_categorical(y\_train)

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# 2e. return your X\_train, X\_test, y\_train, y\_test

return X\_train, X\_test, y\_train, y\_test

# Step 3: define your CNN model here in this function and then later use this function to create your model

def digit\_recognition\_cnn():

# 3a. create your CNN model here with Conv + ReLU + Flatten + Dense layers

model = Sequential()

model.add(Input(shape=(28, 28, 1)))

model.add(Conv2D(30, (5, 5), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(15, (3, 3), activation='relu'))

model.add(MaxPooling2D((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(10, activation='softmax'))

# 3b. Compile your model with categorical\_crossentropy (loss), adam optimizer and accuracy as a metric

model.compile(loss = 'categorical\_crossentropy', metrics = ['accuracy'], optimizer = 'adam')

# 3c. return your model

return model

# Step 4: Call digit\_recognition\_cnn() to build your model

model = digit\_recognition\_cnn()

# Step 5: Train your model and see the result in the Command window. Set epochs to a number between 10 - 20 and batch\_size between 150 - 200

X\_train, X\_test, Y\_train, Y\_test = load\_dataset()

history = model.fit(X\_train, Y\_train, batch\_size = 170, epochs = 15, verbose = 2, validation\_data = (X\_test, Y\_test))

# Step 6: Evaluate your model via your\_model\_name.evaluate() function and copy the result in your report

loss, accuracy = model.evaluate(X\_test, Y\_test, verbose=0)

print('Accuracy: %.2f' % (accuracy \* 100))

# Step 7: Save your model via your\_model\_name.save('digitRecognizer.h5')

model.save('digitRecognizer.h5')

# Code below to make a prediction for a new image.

# Step 8: load required keras libraries

from keras.preprocessing.image import load\_img

from keras.preprocessing.image import img\_to\_array

from keras.models import load\_model

# Step 9: load and normalize new image

def load\_new\_image(path):

# 9a. load new image

newImage = load\_img(path, color\_mode="grayscale", target\_size = (28, 28))

# 9b. Convert image to array

newImage = img\_to\_array(newImage)

# 9c. reshape into a single sample with 1 channel (similar to how you reshaped in load\_dataset function)

newImage = newImage.reshape(1, 28, 28, 1)

# 9d. normalize image data - Hint: newImage = newImage / 255

newImage = newImage / 255

# 9e. return newImage

return newImage

# Step 10: load a new image and predict its class

def test\_model\_performance():

# 10a. Call the above load image function

img = load\_new\_image(r'C:\Users\cbhar\OneDrive\Desktop\School\CSUSB\Classes\Fall2024\Intro to AI\Assignment2\sample\_images\digit1.png')

# 10b. load your CNN model (digitRecognizer.h5 file)

model = load\_model("digitRecognizer.h5")

# 10c. predict the class - Hint: imageClass = your\_model\_name.predict\_classes(img)

imageClass = np.argmax(model.predict(img), axis=-1)

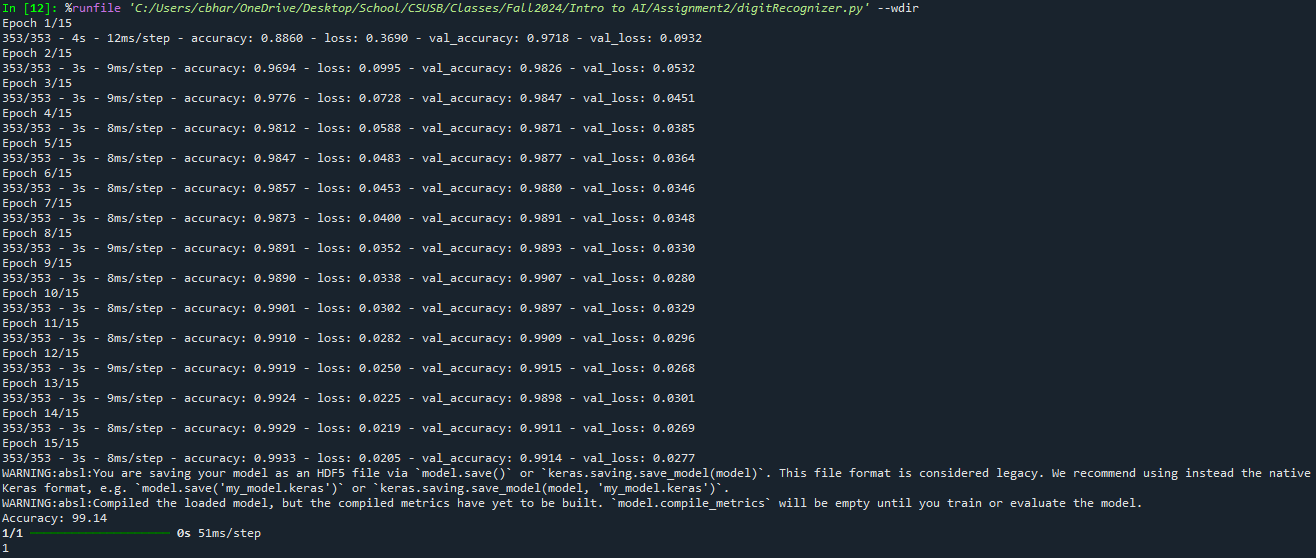
# 10d. Print prediction result

print(imageClass[0])

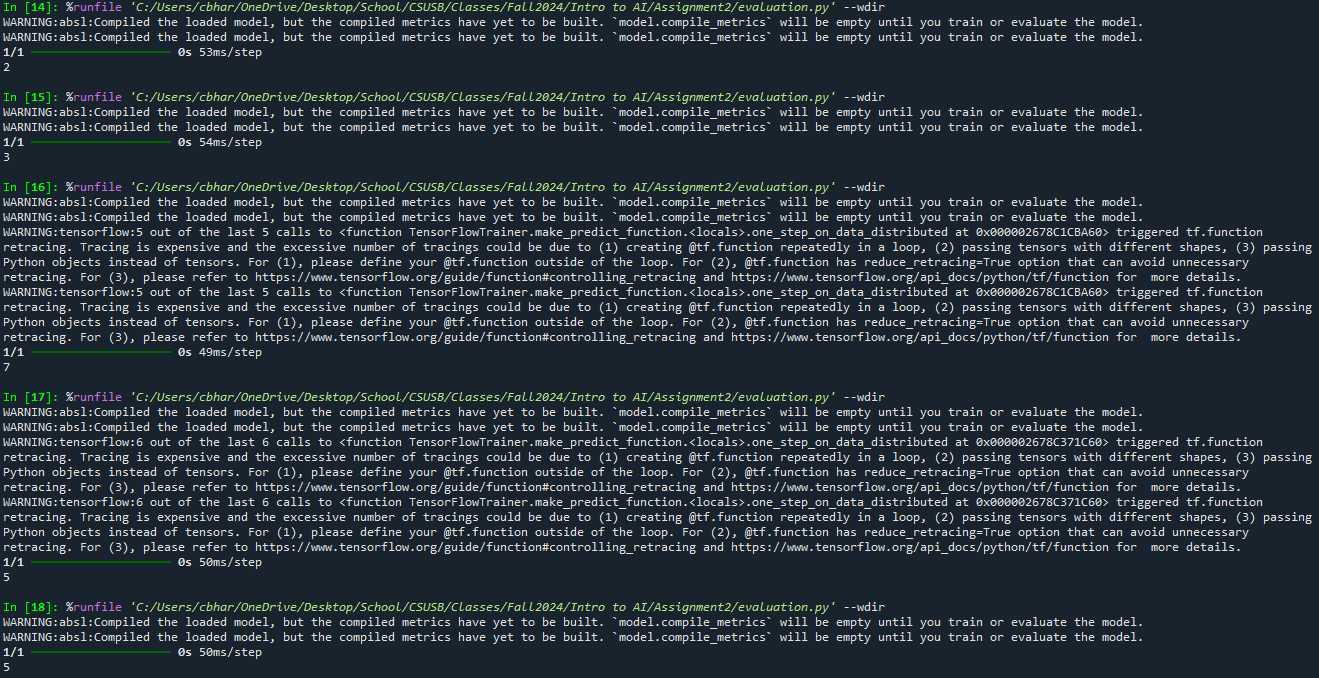
# Step 11: Test model performance here by calling the above test\_model\_performance function

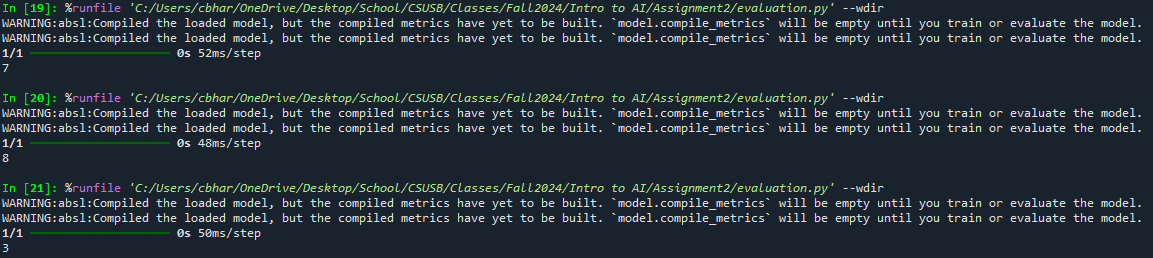
test\_model\_performance()

The following is the result of digitRecognizer.py for digit1.png



The following are digits 2 through 9





1. **Evaluation (evaluation.py) for your model performance evaluation**

The model is loaded into the program by the load\_model() function and saved to a variable.

# Step 3: Load your saved model

model = load\_model("digitRecognizer.h5")