Assignment 2 (20%)

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

Submitted to

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

by

Nicholas Linder (007627900)

Date: *May 4, 2024*

Email:
- *007627900@coyote.csusb.edu*

Assignment Report

Brief description of your work here acknowledging your collaboration with your class fellow (or a friend from other CSE 5120 section), and the capacity at which he/she collaborated with you, followed by the algorithms you implemented.

1. digitRecognizer.py for MNIST dataset

Your brief explanation of the dataset, your code solution, and any documentation with screenshots of your code Evaluation (results from digitRecognizer.py)

My assignment code basically just goes down the steps that were supplied and supplying the steps as asked. Some changes were made such as the keras.utils import to to_categorical rather than what was supplied because it was causing an error not allowing the program to run. The code begins with importing keras libraries as well as numpy which will all be used for the code. It follows with loading the dataset and reshaping the images to match the input shape that is to be expected. It then normalizes the input to a value between 0 and 1 and converts them to categorical format. The next step is one of the most important parts because it defines the cnn model creating the layout as instructed. Followed by building the model that has been defined. It is then trained which will display the results of the model and save them. Last is loading the new image for prediction and then testing the performance of the image.

digitRecognizertest.py 7 × evaluationtest.py 5

digitRecognizertest.py > ...

```python
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('digit1.png')
    # 10b. load your CNN model (digitRecognizer.h5 file)
    model = load_model('digitRecognizer.h5')
    # 10c. predict the class probabilities
    class_probabilities = model.predict(img)
    # 10d. Extract the class with the highest probability
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Epoch 9/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9974 - loss: 0.0092 - val_accuracy: 0.9869 - val_loss: 0.0417
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9984 - loss: 0.0066 - val_accuracy: 0.9850 - val_loss: 0.0520
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9986 - loss: 0.0057 - val_accuracy: 0.9868 - val_loss: 0.0439
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9996 - loss: 0.0032 - val_accuracy: 0.9845 - val_loss: 0.0554
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9984 - loss: 0.0053 - val_accuracy: 0.9878 - val_loss: 0.0471
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9995 - loss: 0.0031 - val_accuracy: 0.9863 - val_loss: 0.0539
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9995 - loss: 0.0025 - val_accuracy: 0.9873 - val_loss: 0.0537
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9835 - loss: 0.0658
Test Loss: 0.05369662493467331
Test Accuracy: 0.9872999787330627
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
ras.saving.save_model(model, 'my_model.keras')`.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty un
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Predicted Class: 1
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
```

⊗ 0 ⚠ 12 0

digitRecognizertest.py 7 × evaluationtest.py 5

digitRecognizertest.py > test_model_performance

```python
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
    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('digit5.png')
    # 10b. load your CNN model (digitRecognizer.h5 file)
    model = load_model('digitRecognizer.h5')
    # 10c. predict the class probabilities
    class_probabilities = model.predict(img)
    # 10d. Extract the class with the highest probability
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Epoch 9/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9971 - loss: 0.0112 - val_accuracy: 0.9870 - val_
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9977 - loss: 0.0099 - val_accuracy: 0.9856 - val_
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - accuracy: 0.9978 - loss: 0.0084 - val_accuracy: 0.9861 - val_
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9991 - loss: 0.0047 - val_accuracy: 0.9862 - val_
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - accuracy: 0.9987 - loss: 0.0057 - val_accuracy: 0.9835 - val_
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - accuracy: 0.9992 - loss: 0.0037 - val_accuracy: 0.9841 - val_
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9978 - loss: 0.0072 - val_accuracy: 0.9871 - val_
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9848 - loss: 0.0653
Test Loss: 0.05278543755412102
Test Accuracy: 0.9871000051498413
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(mod
ras.saving.save_model(model, 'my_model.keras')`.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_met
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted Class: 5
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
```

digitRecognizertest.py 7 × evaluationtest.py 5

digitRecognizertest.py > test_model_performance

```python
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_datas
    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('digit9.png')
    # 10b. load your CNN model (digitRecognizer.h5 file)
    model = load_model('digitRecognizer.h5')
    # 10c. predict the class probabilities
    class_probabilities = model.predict(img)
    # 10d. Extract the class with the highest probability
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Epoch 9/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9978 - loss: 0.0086 - val_accuracy: 0.9867 - val_loss: 0
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9981 - loss: 0.0077 - val_accuracy: 0.9878 - val_loss: 0
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9989 - loss: 0.0051 - val_accuracy: 0.9856 - val_loss: 0
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9989 - loss: 0.0045 - val_accuracy: 0.9873 - val_loss: 0
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9987 - loss: 0.0050 - val_accuracy: 0.9876 - val_loss: 0
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9993 - loss: 0.0033 - val_accuracy: 0.9873 - val_loss: 0
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9997 - loss: 0.0020 - val_accuracy: 0.9879 - val_loss: 0
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9831 - loss: 0.0665
Test Loss: 0.049887944012880325
Test Accuracy: 0.9879000186920166
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
ras.saving.save_model(model, 'my_model.keras')`.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` w
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted Class: 3
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
```

⊗ 0 ⚠ 12 0

2. Evaluation (evaluation.py) for your model performance evaluation

You can also provide brief description of your code written in evaluation.py to load your saved model that can be readily used on test dataset for the staff.

The evaluation is pretty similar to the digitRecognizer file except without the digit recognition definition. The code begins with importing keras libraries as well as numpy which will all be used for the code. It follows with loading the dataset and reshaping the images to match the input shape that is to be expected. It then normalizes the input to a value between 0 and 1 and converts them to categorical format. It then loads a pre-trained model where the data will be saved. The next step is the evaluation model where the loaded faded will be tested for loss and accuracy. The next state is to load the specific image and pass it through the trained model for prediction.

digitRecognizertest.py 7 | evaluationtest.py 5

evaluationtest.py > test_model_performance

```python
# Step 6: load and normalize new image
def load_new_image(path):
    # 6a. load new image
    newImage = load_img(path, color_mode="grayscale", target_size=(28, 28))
    # 6b. Convert image to array
    newImage = img_to_array(newImage)
    # 6c. reshape into a single sample with 1 channel (similar to how you reshaped in load_dataset func
    newImage = newImage.reshape(1, 28, 28, 1)
    # 6d. normalize image data - Hint: newImage = newImage / 255
    newImage = newImage / 255
    # 6e. return newImage
    return newImage

# Step 7: load a new image and predict its class
def test_model_performance(model):
    # 7a. Call the above load image function
    img = load_new_image('digit1.png')
    # 7b. predict the class probabilities
    class_probabilities = model.predict(img)
    # 7c. Extract the class with the highest probability
    predicted_class = np.argmax(class_probabilities, axis=1)
    # 7d. Print prediction result
    print("Predicted Class:", predicted_class[0])

# Step 8: Test model performance here by calling the above test_model_performance function
X_train, X_test, y_train, y_test = load_dataset()
model = load_saved_model()
evaluate_model(model, X_test, y_test)
test_model_performance(model)
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
_and_sample_images-1-1/evaluationtest.py
2024-05-05 23:13:27.543863: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly di
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:13:28.222698: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly di
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:13:30.078643: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler fl
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
Test Loss: 0.049887944012880325
Test Accuracy: 0.9879000186920166
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
Predicted Class: 1
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
```

digitRecognizertest.py 7 | evaluationtest.py 5

evaluationtest.py > test_model_performance

```python
# Step 6: load and normalize new image
def load_new_image(path):
    # 6a. load new image
    newImage = load_img(path, color_mode="grayscale", target_size=(28, 2
    # 6b. Convert image to array
    newImage = img_to_array(newImage)
    # 6c. reshape into a single sample with 1 channel (similar to how yo
    newImage = newImage.reshape(1, 28, 28, 1)
    # 6d. normalize image data - Hint: newImage = newImage / 255
    newImage = newImage / 255
    # 6e. return newImage
    return newImage

# Step 7: load a new image and predict its class
def test_model_performance(model):
    # 7a. Call the above load image function
    img = load_new_image('digit5.png')
    # 7b. predict the class probabilities
    class_probabilities = model.predict(img)
    # 7c. Extract the class with the highest probability
    predicted_class = np.argmax(class_probabilities, axis=1)
    # 7d. Print prediction result
    print("Predicted Class:", predicted_class[0])

# Step 8: Test model performance here by calling the above test_model_pe
X_train, X_test, y_train, y_test = load_dataset()
model = load_saved_model()
evaluate_model(model, X_test, y_test)
test_model_performance(model)
```

PROBLEMS 12 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sam
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sam
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sam
crosoft\WindowsApps\python3.11.exe' 'c:\Users\Nick\.vscode\extensions\ms-python.debugp
_edit_and_sample_images-1-1\evaluationtest.py'
2024-05-05 23:15:36.243915: I tensorflow/core/util/port.cc:113] oneDNN custom operatio
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:15:36.905485: I tensorflow/core/util/port.cc:113] oneDNN custom operatio
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:15:38.773337: I tensorflow/core/platform/cpu_feature_guard.cc:210] This
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlo
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built.
Test Loss: 0.049887944012880325
Test Accuracy: 0.9879000186920166
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
```

digitRecognizertest.py 7 | evaluationtest.py 5

evaluationtest.py > test_model_performance

```python
# Step 6: load and normalize new image
def load_new_image(path):
    # 6a. load new image
    newImage = load_img(path, color_mode="grayscale", target_size=(28, 28))
    # 6b. Convert image to array
    newImage = img_to_array(newImage)
    # 6c. reshape into a single sample with 1 channel (similar to how you reshaped in I
    newImage = newImage.reshape(1, 28, 28, 1)
    # 6d. normalize image data - Hint: newImage = newImage / 255
    newImage = newImage / 255
    # 6e. return newImage
    return newImage

# Step 7: load a new image and predict its class
def test_model_performance(model):
    # 7a. Call the above load image function
    img = load_new_image('digit9.png')
    # 7b. predict the class probabilities
    class_probabilities = model.predict(img)
    # 7c. Extract the class with the highest probability
    predicted_class = np.argmax(class_probabilities, axis=1)
    # 7d. Print prediction result
    print("Predicted Class:", predicted_class[0])

# Step 8: Test model performance here by calling the above test_model_performance funct
X_train, X_test, y_train, y_test = load_dataset()
model = load_saved_model()
evaluate_model(model, X_test, y_test)
test_model_performance(model)
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
crosoft\WindowsApps\python3.11.exe' 'c:\Users\Nick\.vscode\extensions\ms-python.debugpy-2024.6.0-win32
_edit_and_sample_images-1-1\evaluationtest.py'
2024-05-05 23:16:47.659576: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You m
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:16:48.325263: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You m
e environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-05 23:16:50.200762: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binar
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appro
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_
Test Loss: 0.049887944012880325
Test Accuracy: 0.9879000186920166
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
Predicted Class: 3
PS C:\Users\Nick\Desktop\github\5120-AI-Digit-Recognizer\CSE5120_Files_to_edit_and_sample_images-1-1>
```