East West University

Department of CSE

Project Title

Handwritten Capital Letter Recognition

Submitted By

Shodorson Nath

ID: 2020-1-60-224

Course Code: CSE366

Section: 05

Submitted To

Sadika Islam Sneha

Lecturer

Department of CSE

East West University, Dhaka.

Problem Description: Handwritten Capital Letter Recognition. In this problem, we tackle the task of recognizing handwritten capital letter from the famous A_Z Handwritten Dataset.

Problem Code Summary:

1. Objective:

The provided code aims to train a convolutional neural network (CNN) to recognize handwritten characters from the A-Z Handwritten Data dataset. It also allows users to input an image, preprocess it, and predict the character it contains.

2. Libraries Used:

- pandas: For data manipulation and loading CSV files.
- numpy: For numerical computations.
- tensorflow: For building and training neural networks.
- matplotlib.pyplot: For data visualization.
- sklearn.model_selection.train_test_split: For splitting the dataset into training and testing sets.
- PIL.Image: For image processing.

3. Data Handling:

- The dataset is loaded from a CSV file named "A_Z Handwritten Data.csv".
- Features and labels are extracted from the dataset.

4. Data Preprocessing:

The features (images) are reshaped to a 4D array of shape (num_samples, 28, 28, 1) where num_samples is the number of data points. The pixel values are normalized to be between 0 and 1.

5. Model Architecture:

The model is a sequential neural network with the following layers:

- Convolutional layer with 32 filters, a (3, 3) kernel, and ReLU activation.
- Max pooling layer with a (2, 2) pool size.
- Flatten layer to convert the 2D feature maps to a 1D array.
- Dense hidden layer with 128 units and ReLU activation.
- Dense output layer with 26 units (one for each letter) and softmax activation.

6. Model Training:

The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss. It is trained for 10 epochs with a validation split of 20%.

7. Model Evaluation:

The model's performance is evaluated on the test set using accuracy and loss metrics.

8. Training History Visualization:

Training and validation accuracy and loss are plotted over epochs.

9. Random Predictions:

30 random images from the test set are selected and the model's predictions are displayed.

10. Saving and Loading the Model:

The trained model is saved as "handwritten_character_model.h5" for later use.

11. User Input Processing:

- The code prompts the user to enter the path of an image file.
- The input image is preprocessed to match the format used in training.

12. Image Prediction:

- The model predicts the character in the input image.
- The input image and the predicted character are displayed.

Problem Code:

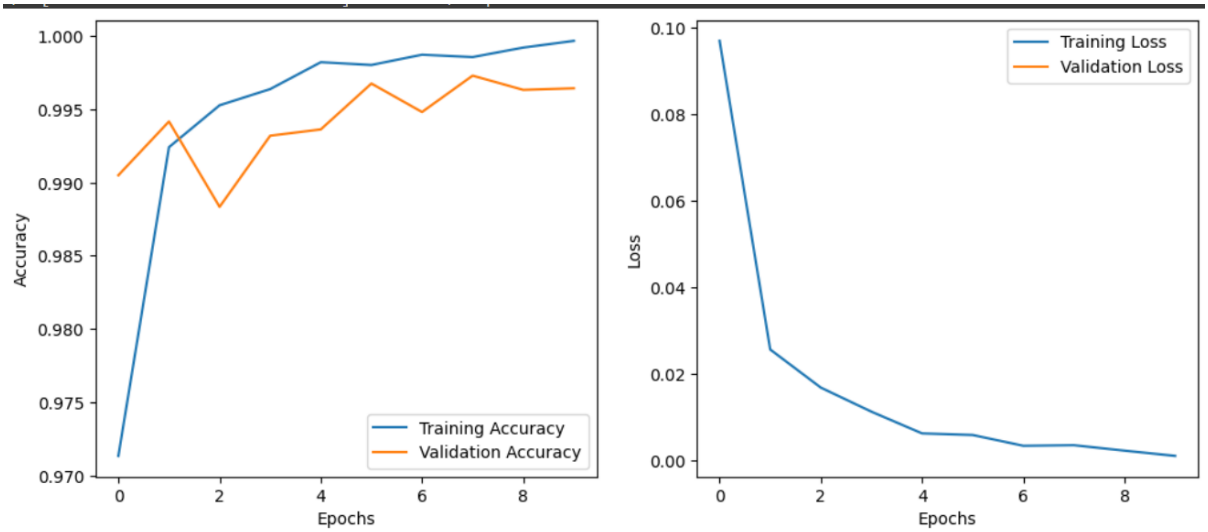
```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from PIL import Image

# Load the dataset
data = pd.read_csv("/content/A_Z Handwritten Data.csv")

# Extract features and labels
X = data.values[:, 1:]
y = data.values[:, 0]

# Reshape (28x28x1) features and normalize
X = X.reshape(-1, 28, 28, 1)
X = X.astype('float32') / 255.0

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(26, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=10,
validation_split=0.2)
model.save('handwritten_character_model.h5')

# Visualize the training history
print("\nAccuracy and Losses Graphs:")
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_acc:.4f}')

# Visualize some predictions
plt.figure(figsize=(15, 8))
for i in range(30):
    plt.subplot(5, 6, i+1)
    index = np.random.randint(0, len(X_test))
    predicted_label =
np.argmax(model.predict(X_test[index:index+1]))
    plt.imshow(X_test[index].reshape(28, 28), cmap='gray')
    plt.title(f'Predicted: {chr(65 + predicted_label)}')
    plt.axis('off')

plt.tight_layout()
plt.show()

# Load the model after training
model = tf.keras.models.load_model('handwritten_character_model.h5')

# Function to preprocess user-input image
def preprocess_image(image_path):
    img = Image.open(image_path).convert('L')
    img = img.resize((28, 28))
    img = np.array(img)
    img = img.reshape(1, 28, 28, 1)
    img = img.astype('float32') / 255.0
    return img

# Get user input for image file
image_path = input("\nEnter the path of the image: ")

# Preprocess the image
input_image = preprocess_image(image_path)

# Predict the character
predicted_label = np.argmax(model.predict(input_image))

# Show the input image and prediction
plt.figure(figsize=(5, 5))
plt.imshow(input_image.reshape(28, 28), cmap='gray')
plt.title(f'Predicted: {chr(65 + predicted_label)}')
plt.axis('off')
```

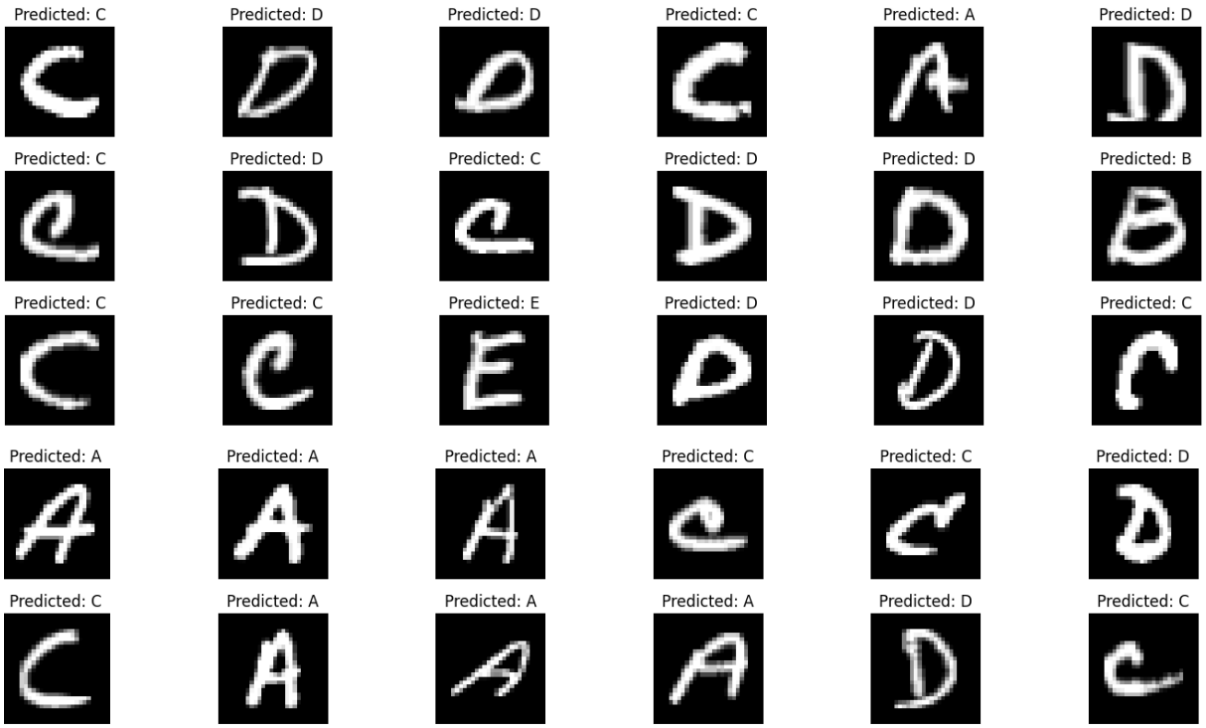
```
plt.show()

print(f"The predicted character is: {chr(65 + predicted_label)}")
```

Output:

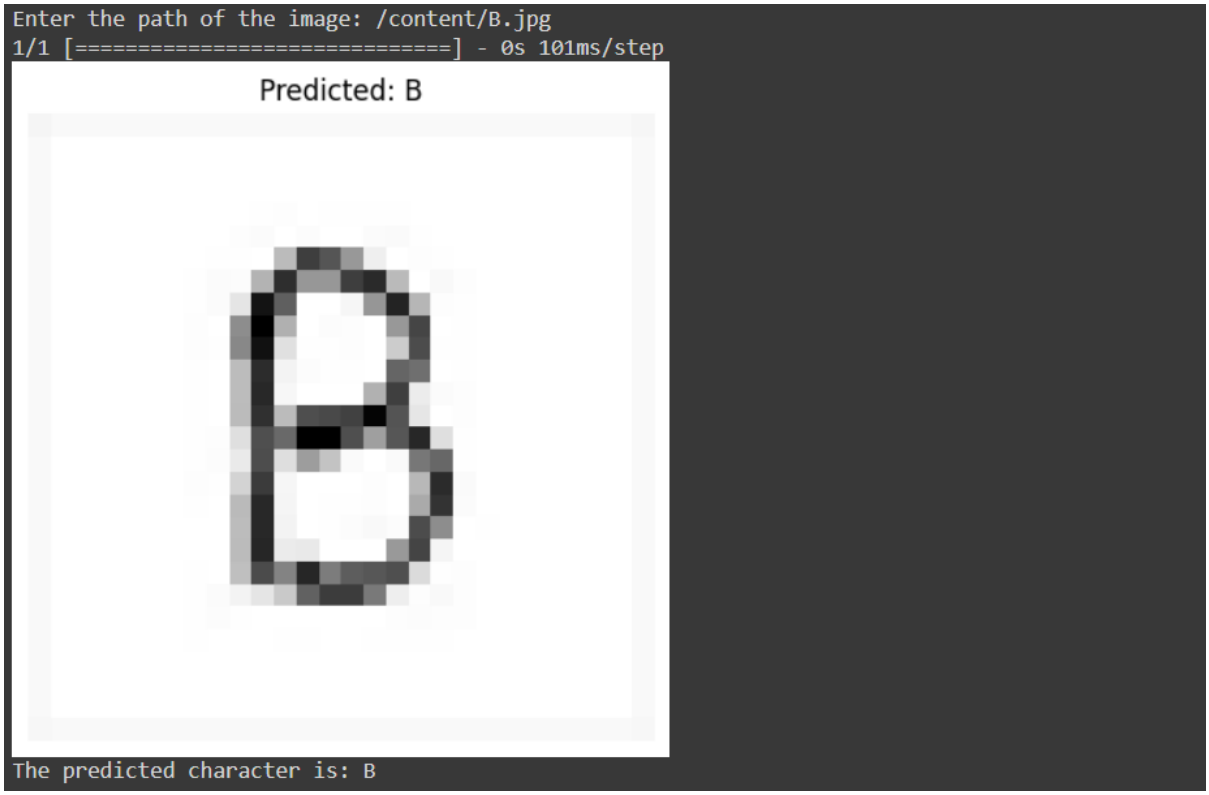
Accuracy and Losses Graph:



Some Prediction from Dataset:



Prediction from Input an Image:



Learnings: The "Handwritten Capital Letter Recognition" problem used the famous A_Z Handwritten dataset, which helps to learn machine learning Concepts. This code facilitates the learning of a Convolutional Neural Network (CNN) for handwritten character recognition. It begins by loading an A_Z Handwritten dataset containing images of handwritten characters and their corresponding labels. These images are preprocessed to ensure uniformity in format and to facilitate the learning process. The CNN architecture, defined using TensorFlow's Keras API, employs layers specialized in feature extraction and classification. Through multiple epochs of training, the model learns to recognize patterns and features in the images, gradually improving its ability to correctly classify characters. The training progress is monitored and visualized through accuracy and loss curves. Following training, the model's performance is evaluated on a separate test set to gauge its generalization capabilities. The trained model is then saved for future use, allowing it to be readily applied to new, unseen data. Additionally, the code enables user interaction, where individuals can input images for character recognition. These user-provided images undergo preprocessing to align them with the format used during training. The loaded model is then employed to predict the character in the provided image, and the outcome is displayed for user confirmation. This code thus encompasses the complete pipeline of training, evaluating, and using a CNN for handwritten character recognition, showcasing the iterative nature of machine learning.