Ministry of Higher Education and Research
University of Sfax
National Engineering School of Sfax

Deep Learning Project

English Handwriting Recognition

Academic Year : 2023-2024

Realized by

Oussama Slimani

GI2-S4

General Introduction

Introduction

✓ Handwriting recognition is a fascinating and challenging problem in the field of artificial intelligence and deep learning.

✓ It involves interpreting handwritten text, converting it into digital text, and understanding its meaning.

✓ This project focuses on developing a deep learning model to recognize and interpret English handwriting.

Importance of Handwriting Recognition

✓ **Digitizing historical documents:** Converting handwritten historical records into digital format for easy access and preservation.

✓ **Automating data entry:** Reducing manual data entry efforts in forms, cheques, and other handwritten documents.

✓ **Enhancing accessibility**: Assisting individuals with disabilities to interact with handwritten content.

Challenges in Handwriting Recognition

✓ **Variability in handwriting styles**: Different individuals have unique ways of writing, which can vary widely in terms of size, slant, and shape of characters.

✓ **Noise and distortion:** Handwritten text may be affected by noise, smudges, and distortions, making it harder to recognize accurately.

✓ **Context understanding:** Recognizing individual characters is not enough; understanding the context and word-level recognition is essential for accurate interpretation.

Deep Learning for Handwriting Recognition

1. Dataset (source)

kaggle

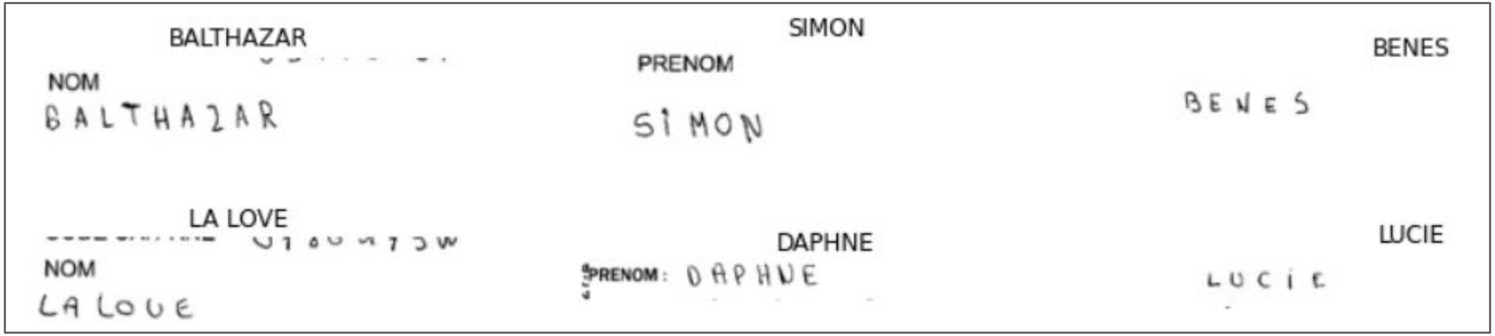
✓ Dataset from Kaggle contains over **400,000** handwritten names gathered from charity projects.

✓ These images are organized into training, testing, and validation sets, enabling researchers and practitioners to train and evaluate machine learning models effectively.

✓ Each image label follows a specific naming format, facilitating the extension of the dataset with additional data.

2. Data Preprocessing (Loading and Viewing Data)

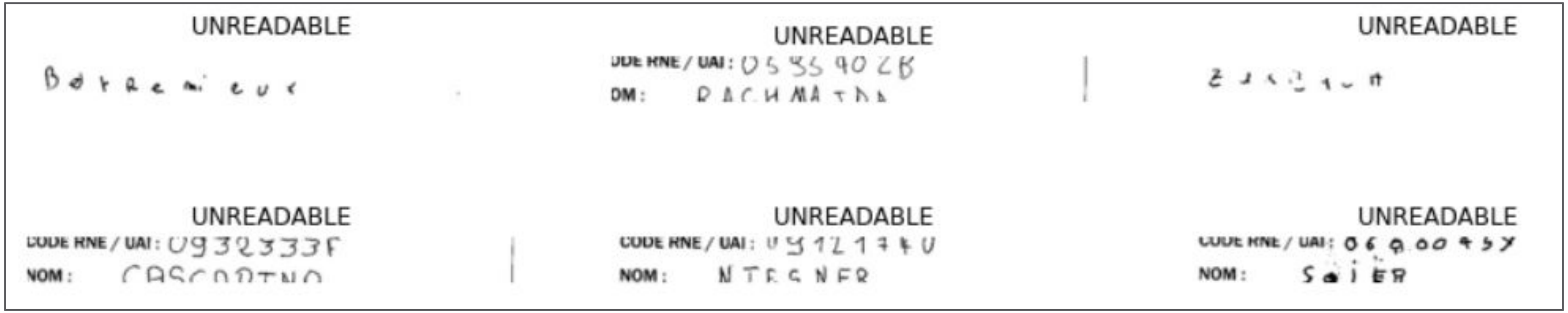
✓ The dataset is inspected to understand its structure and characteristics, gaining insights into the distribution of handwritten names and associated images.



2. Data Preprocessing (Cleaning Data)

✓ Missing values are addressed, and unreadable samples are filtered out to ensure data quality.

```
Number of NaNs in train set      :  565
Number of NaNs in validation set :  78
```



2. Data Preprocessing (Preprocessing Data)

✓ **Resize and Crop**: Ensure images fit within a `64x256` size by cropping excess dimensions.

✓ **Background and Rotation**: Place cropped images on a `64x256` white background and rotate 90 degrees clockwise.

✓ **Normalization**: Scale pixel values to [0, 1] by dividing by 255.

✓ **Convert labels to numerical format and back:** Prepares the data for training a model with CTC loss, which requires numerical labels and their lengths for alignment during training.

2. Data Preprocessing (Split Data)

✓ The dataset used in my work consists of **30,000** training samples and **3,000** validation samples, along with some samples for testing, all selected from the 400,000 images in the dataset.

✓ It has been cropped, normalized, processed, and cleaned, with numerical labels assigned to the data.

3. Model Architecture (1/6)

✓ This model combines **Convolutional Neural Networks** (CNNs) and **Recurrent Neural Networks** (RNNs) to perform handwriting recognition.

1. Input Layer

⇒ The model expects input images of shape (256, 64, 1), where 256 is the height, 64 is the width, and 1 represents the grayscale channel.

3. Model Architecture (2/6)

2. Convolutional Neural Network (CNN) Layers

⇒ The CNN layers are used for feature extraction from the input images.

✓ **First Convolutional Block**

- Conv2D: 32 filters, (3, 3), 'same' padding, He normal initialization.

- BatchNormalization

- ReLU Activation

- MaxPooling2D: (2, 2)

3. Model Architecture (3/6)

✓ **Second Convolutional Block**

- Conv2D: 64 filters, (3, 3), 'same' padding, He normal initialization.
- BatchNormalization
- ReLU Activation
- MaxPooling2D: (2, 2)
- Dropout: 0.3

3. Model Architecture (4/6)

✓ **Third Convolutional Block**

- Conv2D: 128 filters, (3, 3), 'same' padding, He normal initialization.
- BatchNormalization
- ReLU Activation
- MaxPooling2D: (1, 2)
- Dropout: 0.3

3. Model Architecture (5/6)

3. Reshape Layer

⇒ Reshape:

Converts the output from the CNN layers into a 2D tensor suitable for the RNN layers, with shape (64, 1024).

⇒ Dense Layer:

64 units with ReLU activation to process the reshaped data.

3. Model Architecture (6/6)

4. Recurrent Neural Network (RNN) Layers

⇒ The CRNN model architecture is designed to effectively capture spatial and temporal features inherent in handwritten text.

✓ **Recurrent Layers**: Utilizing recurrent neural network (RNN) layers, such as Long Short-Term Memory (LSTM) units, to capture temporal dependencies and contextual information within sequences of characters.

✓ **Connectionist Temporal Classification (CTC) Loss:** The CTC loss function is utilized to train the model, enabling it to handle variable-length input sequences and predict sequences of characters accurately.

4. Compile & Train the Model

1. Model Compilation

✓ **Loss**: Custom CTC loss function

✓ **Optimizer**: Adam optimizer with learning rate of 0.0001.

2. Model Training

✓ **Epochs:** 60

✓ **Batch Size**: 128

```
Epoch 54/60
235/235 [==============================] - 32s 138ms/step - loss: 1.2794 - val_loss: 2.0305
Epoch 55/60
235/235 [==============================] - 32s 138ms/step - loss: 1.2518 - val_loss: 2.0469
Epoch 56/60
235/235 [==============================] - 32s 138ms/step - loss: 1.2313 - val_loss: 2.0653
Epoch 57/60
235/235 [==============================] - 33s 139ms/step - loss: 1.2018 - val_loss: 2.0503
Epoch 58/60
235/235 [==============================] - 32s 138ms/step - loss: 1.1733 - val_loss: 2.0275
Epoch 59/60
235/235 [==============================] - 32s 137ms/step - loss: 1.1540 - val_loss: 2.0384
Epoch 60/60
235/235 [==============================] - 32s 138ms/step - loss: 1.1233 - val_loss: 2.0683
```

5. Model Evaluation

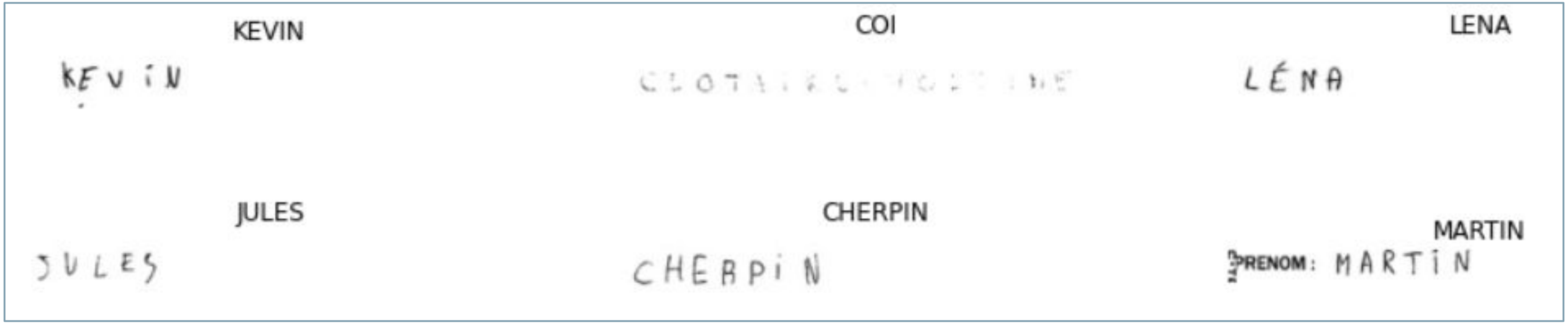
✓ Following training, the performance of the CRNN model is evaluated to assess its effectiveness in handwriting recognition.

✓ **Character-level Accuracy:** Assessing the model's accuracy in predicting individual characters.

✓ **Word-level Accuracy:** Evaluating the model's accuracy in predicting complete words.

```
Correct characters predicted : 89.35%
Correct words predicted      : 75.87%
```

5. Some predictions



CONCLUSION

Conclusion

This presentation provides a comprehensive overview of the challenging process involved in training a CRNN model for handwriting recognition, emphasizing the importance of each step in achieving accurate and acceptable results.

Thank You

For your attention