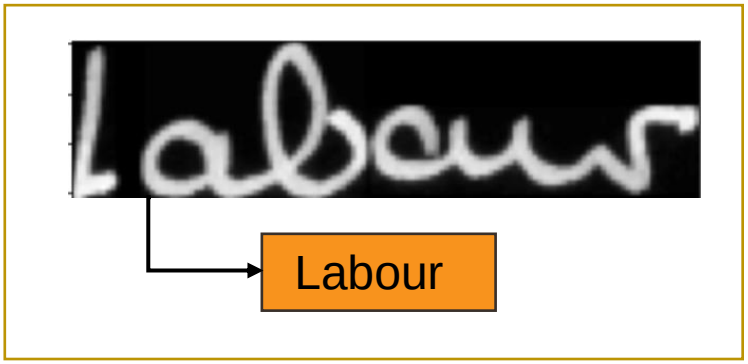
Handwritten Character Recognition

Mid-Program Project 2

Table of Contents

Aim of the Project

The aim of this project is to automatically convert handwritten text into machine encoded text.



Learning Outcome

1. Implementation of Convolutional Neural Network to extract features from images
2. Implementation of Recurrent Neural Network (GRU, LSTM, Bi-LSTM, etc.) to process sequential data
3. Preparation and preprocessing of image data
4. Preparation and preprocessing of image data
5. Development, error analysis, and deep learning model improvement

Problem Statement

A medical company needs to take handwritten prescription and retrieve the text from it. To do this manually, it will require a lot of time and lots of cost to the company. So, the company wants to automate this task. You need to create a neural network model that will take images as input, read the text images, and convert it into digital text.

To solve this problem, you need to create a CNN LSTM model.

Strategy

The project has 4 major components:

1. Preprocess image and text data
2. Implementation of CNN layers to extract features
3. Implementation of RNN (Bi-LSTM) layers to the sequence model
4. CTC_loss and CTC_decode

1. Preprocess image and text data:

Unpack and load the pre-processed dataset with images. You can download the images using the following link:

https://drive.google.com/file/d/1idCx6pr1ptmHrEHjbbiQxF4xoCZ9dNtv/view?usp=sharing

To download the text use the following link:

https://drive.google.com/file/d/1lZA6WjzssJQYBEN7XLq4rhtxuRpqqmhT/view?usp=sharing

Resize and normalize all the images. For text data convert each word to encoded text, give an ID to each character and use that ID to generate encoded text.

2. CNN layers for feature extraction model

Add several CNN and Pooling layers to the model to extract features from the image. You can also add Normalization layers and Dropout Layers to prevent overfitting.

3. Bi-LSTM layers to learn the sequential data in the image

Add several Bi-directional LSTM layers, whose input will be the extracted features from the CNN layers, to learn the sequential pattern from the images. You can add dropout layers for regularization. Finally, add the output layer to the model, where the number of classes will be the number of characters.

4. Use CTC_loss and CTC_decode (Connectionist Temporal Categorical)

The output sequence from the output layer will be fed to the CTC layer. A normal loss function optimizes just one objective, but CTC optimizes both the length of the word and the classes of the predicted sequence of characters.

Also, ctc_decode can be used to get the final prediction.

To learn more about CTC, refer to the following:

https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

ctc_batch_cost: https://www.tensorflow.org/api_docs/python/tf/keras/backend/ctc_batch_cost

ctc_decode: https://www.tensorflow.org/api_docs/python/tf/keras/backend/ctc_decode

Dataset Description

Dataset: IAM Dataset

You will be using the IAM dataset for training and testing your model. We will use the words subset of the dataset, in which each image contains a word. The dataset contains over 8000 images and their digital text.

This is a sample from text file,

a01-000u-01-03 ok 156 1400 937 294 59 NN Labour

where,

a01-000u-01-03: image location+ image id

Labour: actual word

Image file contains one word in each image

Task to be done:

1. Read the parser.txt file containing the image id and the respective word for that image and take the first 10000 instances for training and testing of the model **Marks: 10**
2. Images can be of different shape thus resize all your images to have the same shape (for example = (128,32)) Marks: 10
3. Currently, the pixel values are between 0 to 255, normalize the images so that the pixel values are in range 0 to 1 Marks: 10
4. Create a list of all characters and use the character's index to encode the actual words into digits
5. Pad all the words to have a similar length Marks: 10
6. Split your dataset for training and testing Marks: 5
7. Create a model for training: Marks: 30
   a) Add several CNN layers to extract the sequence of features
   b) Add Bi-LSTM layers to propagate through the sequence
   c) Add a dense layer (output layer) with total number of neurons as (total number of characters + 1) and the activation as softmax.
8. The output sequence from the output layer will be fed to the CTC layer. **Marks: 5**

Hint:

```
def ctc_lambda_func(args):
    y_pred, labels, input_length, label_length = args

    return K.ctc_batch_cost(labels, y_pred, input_length, label_length)

loss  = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([outputs, the_labels,
input_length, label_length])

#model to be used at training time
model = Model(inputs=[inputs, the_labels, input_length, label_length], outputs=loss)
```

```
model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer = optimizer_name
, metrics=['accuracy'])
```

9. Predict output using your model (do not use the last loss layer) on validation images, use ctc_decode to decode your output and then print the actual words using the indexes from your character's list. Marks: 20

How to Start with the Project?

1. Log-in to the Google Co-lab, load the notebook to the environment. Go to Runtime to choose the "Change runtime type." For faster training, choose GPU as the hardware accelerator and SAVE it.

2. Import all the necessary Python packages. Numpy and Pandas for numerical processing, data importing, preprocessing etc. Matplotlib for plotting pose joints and showing images, cv2 package for image processing functions, sklearn for splitting datasets, keras for deep learning model creation, training, testing, inference, etc.

3. Dataset for Character Recognition: The IAM dataset used in this project is made available by ICDAR. These datasets can be downloaded using the paths mentioned earlier and uploaded to the google colab current working directory or it can be kept in Google Drive, which can be mounted to the Colab working directory.

4. Since the dataset does not provide the validation set (Validation set helps us to monitor the training after each epoch) part of the training set can be considered as a validation set using function train_test_split. The parameter test_size is the ratio between the number of training and validation set samples, you set the test_size = 10% of the total dataset.

How to submit your project?

Following are the tasks, which need to be developed while executing the project:

- Share your project via Google Colab to support@edureka.co
- The .ipynb file with details of each step in the markdown
- Save the model and mail the .h5 created and the downloaded .ipynb to support@edureka.co
- You can even upload your code into your github repository and share your repository with us