Text Detection using Convolutional Recurrent Neural Networks

Business Objective

Text detection is the process of detecting the text present in the image. Several applications include solving the captcha, identifying vehicles by reading their license plates, etc.

Convolutional neural networks are deep learning algorithms that are very powerful for the analysis of images. On the other hand, Recurrent Neural Networks (RNNs) are used for sequential data such as text. RNNs are ideal for solving problems where the sequence is more important than the individual items themselves.

In our case, to perform image processing and sequence prediction, we are going to use both CNN and RNN networks. We will apply convolutions on the images, and on these convolutions, we will run the recurrent neural network model to predict the text in the picture. This model is called Convolutional Recurrent Neural Network (CRNN).

This model is best used for images with a single line of text in them, so that we will build this model on images with single-line texts.

This project aims to build a convolutional recurrent neural network that can detect the text from a given image.

Data Description

We will be using the TRSynth100K dataset. This dataset contains around, 100k images along with their labels in a text file.

Aim

To build a CRNN network that can predict the single-line text in a given image.

Tech stack

- Language - Python
- Libraries – cv2, torch, numpy, pandas, albumentations, matplotlib, pickle

Approach

1. Importing the required libraries
2. Download the required dataset
3. Pre-processing
    - Find null values and replace those missing values with string “null”
    - Create a data frame with image path and corresponding labels (save as csv file)
    - Create a mapping from characters to integer and save in pickle format(char2int.pkl)
    - Create a mapping from integer to character and save in pickle format (int2char.pkl)
4. Define configurations paths
5. Training the model
    - Split the data into train and validation
    - Create train and validation datasets
    - Define the loss function
    - Create the model object
    - Define the optimizer
    - Training loop
    - Save the trained model
6. Predictions
    - Select image for prediction
    - Apply augmentations
    - Take the model output
    - Apply softmax and take label predictions
    - Use the ‘ph’ string character and convert integer predictions to string
    - Display the output

Modular code overview

```
input
  |_TRSynth100K image folder
  |_int2char.pkl
  |_char2int.pkl
  |_data_file.csv

src
  |_source
          |_dataset.py
          |_network.py
  |_train.py
  |_config.py
  |_predict.py
  |_preprocessing.py
  |_utlis.py

lib
  |_crnn.ipynb


output
  |_model.pth
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. lib
4. output

1. input **-** It contains all the data that we have for analysis. There are two pickles files and one csv file
    - int2char.pkl
    - char2int.pkl
    - data_file.csv

2. src - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:
    - source folder – this contains a dataset.py and network.py files
    - train, config, predict, pre-processing, and utlis files

3. Output folder **–** The output folder contains the best-fitted model that we trained for this data. This model can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
   Note: This model is built over a chunk of data. One can obtain the model for the entire data by running engine.py by taking the entire data to train the models.

4. Lib folder **-** This is a reference folder. It contains the original ipython notebook that we saw in the videos.

Project Takeaways

1. Understanding the business problem
2. Understanding what is Convolutional Neural Network (CNN)
3. Understanding what is Recurrent Neural Network (RNN)
4. Understanding the CRNN Architecture
5. Understanding the CTC loss function
6. Importing the dataset and required libraries
7. Convert data to pickle file and CSV file
8. How to perform environment setup
9. How to define configurations
10. Perform data pre-processing
11. How to build a CRNN model
12. Split the data into train and validation datasets
13. How to train the model
14. How to make predictions on the model
15. How to interpret the generated results.