Machine Learning Engineer Nanodegree

by Mohammed Ayaz S

Proposal

Domain Background

Optical Character Recognition is used to convert handwritten or printed text into Machine Encoded text. The aim of Optical Character Recognition (OCR) is to predict the handwritten text with maximum accuracy and minimal errors. Optical Recognition has a wide range of application.

Early Optical Character Recognition was mainly focused towards building tools to aid people who are blind. Since, then Optical Character Recognition has evolved to recognize sign boards for translation, navigation and for text-to-speech. OCR now has a wide range of applications from day-to-day applications to scientific applications.

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks, but in this project - it is used to recognize hand-written character text.

Bidirectional recurrent neural networks(RNN) are just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation.

The aim is to implement character recognition using Bidirectional Recurrent Neural Network and compare the accuracy with the Conventional Recurrent Neural Network.

Problem Statement

Implement a model that recognizes a character set using Conventional RNN and also build the same model using Bidirectional Recurrent Neural Network then compare the accuracy of both models.

Datasets and Inputs

Dataset used - http://ai.stanford.edu/~btaskar/ocr/

The dataset used in the Optical Character Recognition problem is a (slightly modified) OCR dataset provided by MIT. This dataset contains a total of roughly 6800 words where every character of the words is represented in the form of images. The longest word is 14 characters in length.

Inputs- The input to our recurrent neural network, that is the features, is a sequence of character images where every image is 16x8 pixels. We set up a helper function to get the features and labels

from the lines of data that we just read in. The data that we just read in has a unique sequential integer ID assigned to every line, so we sort on this character ID. Character sequences which make up words are maintained. Set up lists of the data and the target values. The data is the input sequence of images which make up words, and the target is the word as a sequence of actual characters, not images

The feature vectors of our training data are a sequence of images where every image is one character. We won't feed in a single character at a time. Instead, we'll feed in a sequence of characters which make up a word. Each image has processed and rasterized to be of the same shape. Every character image is 16 pixels by 8 pixels, having a total of 128 pixels to represent 1 character. Every pixel is either 0 or 1, so these are not color images. Color images tend to be far more complex and probably more suited for a convolutional neural network, which can detect patterns in images. For a recurrent neural network, these are simple 0s or 1s image.

Output - The accuracy of the RRN and Bi-Directional RNN is compared with help of graphs and visualizations further various factors involved

Solution Statement

The intended solution is a comparison between the conventional RNN and Bi-directional RNN. To check if there is an improvement in the accuracy of using the Bidirectional RNN. It is expected that there is an improvement in accuracy when Bi-directional RNNs are used.

Benchmark Model

There are various present models and parameters to benchmark with. Here, two models (RNNs and Bi-directional RNNs) are benchmarked with each other test.

Evaluation Metrics

The evaluation metric would be straightforward in the OCR. It is the accuracy of the prediction of the image text to the machine-encoded test. This would depend on various factors like size of the image and the quality of the image. There are labeled images and test images where the predicted label is compared with the labeled image for the accuracy.

Project Design

Programming language: Python 2.7+

Libraries: Tensorflow 1.7

Workflow :

- Dataset is downloaded and loaded into the program.
- Features and labels - The input to our recurrent neural network, that is the features, are a sequence of character images where every image is 16x8 pixels. We set up a helper function to get the features and labels from the lines of data that we just read in. The data that we just read

in has a unique sequential integer ID assigned to every line, so we sort on this character ID. Character sequences which make up words are maintained. Set up lists of the data and the target values. The data is the input sequence of images which make up words, and the target is the word as a sequence of actual characters, not images

- Shuffle and Feed in Training Data - Training data should always be fed into our machine learning model in shuffled form so that there are no unwanted patterns that our model picks up in our input. Shuffle indices using np.random.permutation and then set up shuffle features and corresponding labels. The dataset is split into test and train dataset. The training data will be used to generate our machine learning model parameters, and the test data will be used to measure how well our model performs on a dataset that it hasn't seen before. Sixty-six percent of our data will be used for training, and the remaining for the test. Split the data, as well as the labels into training and test portions. There are 4538 words in our training dataset and test is the remaining files. The test and training data is shuffled.
- Training a Recurrent Neural Network - the RNN is trained and evaluated by looking at the tensor
- Setup the Bidirectional RNN using the Tensorflow
- Train the Bidirectional RNN and compare the results.

References :

Recursive Recurrent Nets with Attention Modeling for OCR in the Wild - https://arxiv.org/pdf/1603.03101v1.pdf

Dataset - http://ai.stanford.edu/~btaskar/ocr/