SiLiHaSeR

Shivam Pandey
Department of Computer Science
University of California, Davis

Michael Yang
Department of Computer Science
University of California, Davis

Aniket Banginwar
Department of Computer Science
University of California, Davis

March 17, 2022

1 Problem Statement and Motivation

The research and development on handwritten sentence recognition are to a large degree motivated by many application areas, such as data acquisition in banks, text-voice conversion, security, etc. Many applications demand very high recognition accuracy and reliability. There are existing tools such as *Google's Tesseract*. However, its output will have poor quality if the input images are not preprocessed to suit it.

We have come up with a Machine Learning (ML) model utilizing different concepts involved in ML such as supervised and unsupervised learning, along with some dynamic programming. We have named our ML model as "*SiLiHaSeR*", which stands for *Single-Line Handwritten Sentence Recognizer*. The final goal of *SiLiHaSeR* is to recognize a single-line handwritten sentence with minimal error. We have evaluated the model based on some performance metric to assess its efficiency and reliability. Notwithstanding the fact that *SiLiHaSeR* is not "good" enough to be deployed for mission-critical tasks, it can still serve us by recognizing fairly accurate text from a single-line handwritten sentence.

2 Datasets

The IAM handwriting dataset [1] consists of several forms of handwritten English words along with labels. This dataset is used to train and test the Neural Network (NN) implemented in *Phase*-2 of the *SiLiHaSeR*, to recognize handwritten words.

3 Design and Technical Approach

SiLiHaSeR is developed sequentially using an unsupervised machine learning algorithm, named *K-means Clustering* and an iterative dynamic programming technique known as the *Kadane's* algorithm. It also employs deep learning methods such as *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN) using *Long Short Term Memory* (LSTM) units, and *Connectionist Temporal Classification* (CTC).

In a nutshell, an image (JPG or PNG) of a handwritten sentence in *English* language is given as the input to *SiLiHaSeR*, and the model gives the output which is expected to be a sentence in machine-printed format.

The proposed design of *SiLiHaSeR* involves 3 phases: *Phase*-1 is segmenting a sentence image into a sequence of images of individual words, *Phase*-2 is recognition of individual words from images using deep learning, and *Phase*-3 is correction of errors in recognized words. Each of the three phases are discussed in the following sections. We will see in the experimental results section below that *SiLiHaSeR* performs "surprisingly good" by combining *Phase*-1 and *Phase*-2 *only*, without deploying *Phase*-3. However, we could further improve *SiLiHaSeR*'s accuracy by "diligently" implementing *Phase*-3. We thought about building a *Language Model* (LM) using *unigrams* and conditional probability table (CPT) from *bigrams*, along with a tree search approach like *Uniform Cost Search* to correct some errors in recognized sentence. The detailed explanation about our potential ideas and scheme for *Phase*-3 is discussed in section 3.3.

3.1 Phase-1: Segmentation of Sentence into Words

In the first phase of its working, *SiLiHaSeR* segments the input image of a handwritten sentence image into a sequence of individual words' images. This phase involves the deployment of *Kadane's* and *K-means Clustering* algorithms, along with *computer vision* libraries. Upon receiving the raw input image of a sentence, some noise is removed and the image is converted into a binary image. Thereafter, each *black* pixel is linked to a large positive weight, and each *white* pixel is linked to a small negative weight. Subsequently, a vertical and horizontal projection of the image is obtained on the vertical and horizontal axes respectively. It is done in such a way that the projection indices of columns having all *black* pixels will get positive values, and the projection indices of columns having all *white* pixels will get negative values. After this, *Kadane's* algorithm is used to find the maximum sum subarray of the horizontal projection. Using this information, the precise location of the sentence is obtained, and the leading and trailing white spaces are trimmed. This same procedure is performed utilizing

vertical projection for trimming the top and bottom spaces in the sentence. All the spaces in the sentence are discovered using the horizontal projection, as regions with spaces will have small negative values. Without loss of generality, it is assumed that interword spaces are larger than intraword spaces. Then, we employ unsupervised learning using *K-means Clustering* algorithm to "learn" and "cluster" the interword and intraword spaces. The sentence is sliced at the bisector of the learnt interword spaces, eventually leading to sentence segmentation. Using the aforementioned technique, the white space encompassing each word is also trimmed. In this manner, we get a sequence of images of each word.

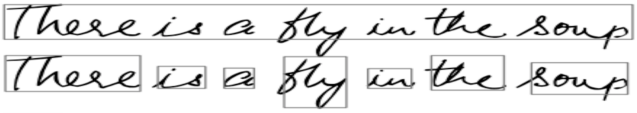


Figure 1: Line Segmentation into Words

3.2 Phase-2: Handwritten Word Recognition

In the second phase of its working, *SiLiHaSeR* utilizes an existing NN architecture [2] to recognise each of the word from the sequence of images obtained in phase-1. In figure 2, the architecture of the NN is explained. This NN consists of CNN layers, RNN layers with LSTM units and a CTC layer.

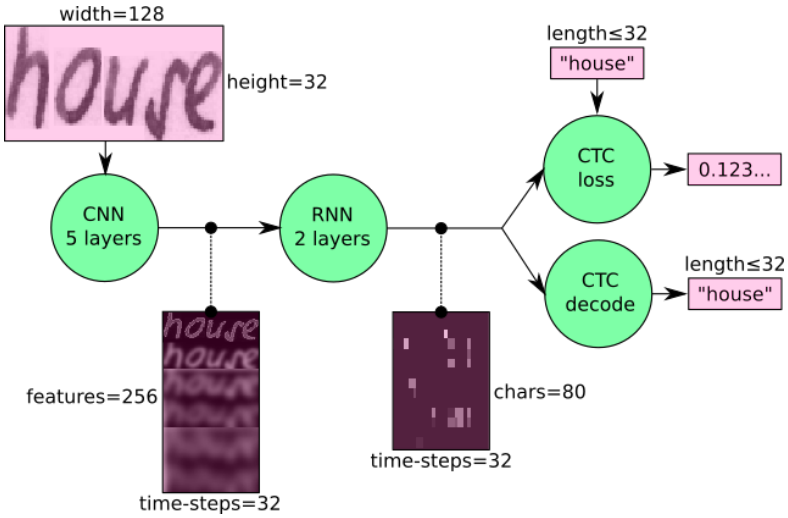


Figure 2: Convolutional-Recurrent Neural Network (CRNN) Architecture and Design

The CRNN is trained on words' images from the *IAM* dataset. Eventually, for each word image given as input to the NN, it is expected to output the *machine-printed* word string. The CRNN architecture details are described below.

- CNN : The input word image is fed into the 5 CNN layers which are trained to extract relevant features from the image. A 5×5 and a 3×3 filter kernel is used in the first 2 layers and last 3 layers respectively, to perform convolutions. This is followed by a RELU function to introduce non-linearity. Eventually, a pooling layer generates a downsized version of the input. The output feature map is 32×256.
- RNN : The sequence model employed here is LSTM-RNN, as it can tackle vanishing gradient problem and learn long-term dependencies. There are 256 features per time-step in the feature sequence, and this model propagates crucial information through each time step. The LSTM-RNN output sequence is mapped to a 32×80 matrix. The IAM dataset consists of 79 different characters, and one additional character is needed for the CTC operation (CTC blank label). Thus, there are 80 entries for each of the 32 time-steps.
- CTC: The CTC layer is given the LSTM-RNN output matrix and the ground truth while training. It computes the loss value. While testing, the CTC is only given the output matrix to decode it into a word.

After utilizing this NN model, the phase-2 is executed and *SiLiHaSeR* outputs the machine-printed words.

3.3 Phase-3: Error Correction in Recognized Words

In this phase, we deal with correction of some errors that might have been introduced in recognizing words by utilizing *Phase*-1 + *Phase*-2. Note that there *can be* errors in the form of spellings, or recognizing words which do not make sense in the context of sentence. We have proposed below to build a *Language Model* using a relevant dataset to tackle this problem.

Firstly, we collect a large text corpus from an existing dataset. Then, data cleaning is performed to get rid of every other symbol except *characters* and *spaces*. After data cleaning, the resultant data is used to create two sets, one for unigrams and the other for bigrams. Let us denote any arbitrary "word" as "w". The probabilities $P(w)$ of unigrams are computed, and a CPT is constructed from bigrams. Then, a tree is created such that at any level, a node represents a word from the set of words. The root node designates the empty word. The number of nodes at each level equals the cardinality of this set. Each node at a particular level is connected to every node in the next consecutive level. The weight of the edge connecting a node w_i at i^{th} level to another node w_{i+1} at $(i+1)^{th}$ level is computed by -

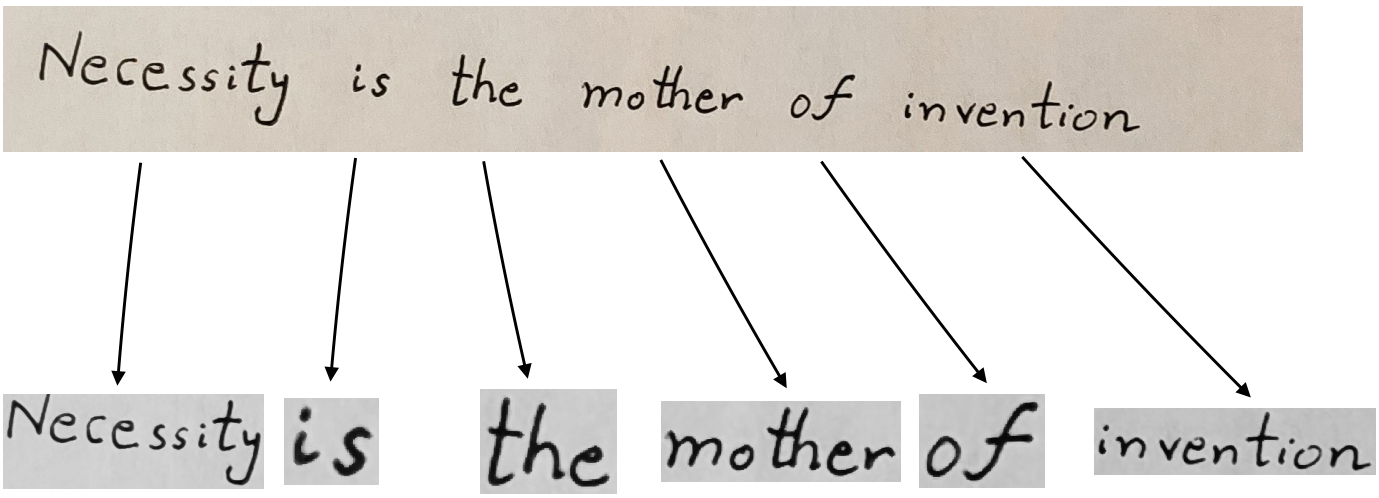
$$\textit{Levenshtein Distance}\ (\text{word at } (i+1)^{th} \text{ level , } (i+1)^{th} \text{ word in the current sentence}) \times (1 - P(w_{i+1}|w_i)),$$

where, the *Levenshtein distance* between two words is computed as the minimum number of single character edits like insertions, substitutions, or removals, to change one word into the other. Here, $P(w_{i+1}|w_i)$ can be calculated using the conditional probability table. Then, a trail/path from the root node to a leaf node represents a sentence. The length of this trail/path would signify the likelihood that the sentence constructed by the nodes visited in this path is the one given as input from *Phase-1 + Phase-2*. A tree search algorithm, named *Uniform cost search* can be run on this tree. Eventually, it is expected that the computed shortest path would give the correctly determined sentence.

***Note**: We tried to implement this phase, however, we did not get promising results. We require some more time and resources to improve this model. There are some potential improvement ideas which are discussed in future directions section.*

4 Experimental Results

4.1 Phase-1: Line Segmentation Result



It is observed that the line segmentation code correctly segments the sentence into individual words, and provides the output in the form of a sequence of words' PNG images.

4.2 Phase-1 + Phase-2: Sentence Recognition Results

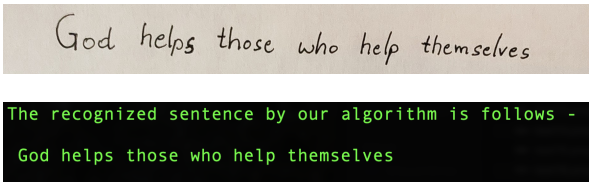


Figure 3: *SiLiHaSeR* correctly recognizes the sentence here.

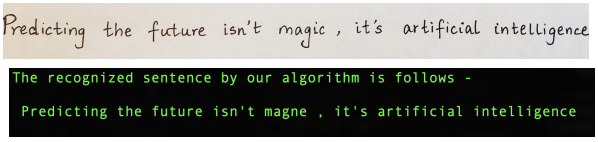


Figure 4: Here, *SiLiHaSeR* recognized most words correctly, including punctuation mark. However,it failed to recognize "magic" correctly.

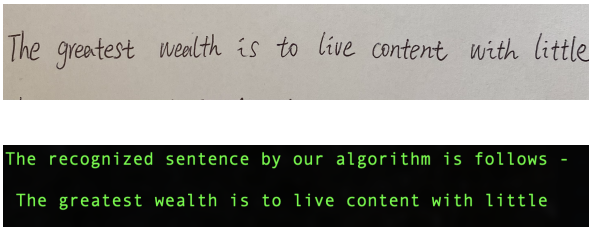


Figure 5: *SiLiHaSeR* correctly recognizes the sentence here.

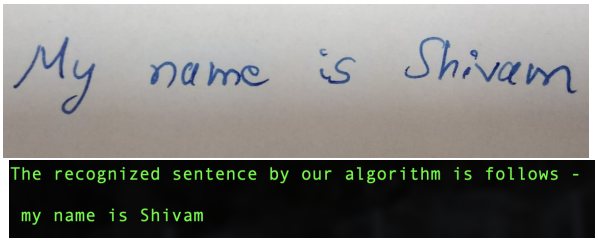


Figure 6: *SiLiHaSeR* correctly recognizes all the words. However, it mis-predicted "M" as "m".

4.3 Testing *SiLiHaSeR*

We tested our model on 34 test images, which were hand-written by all of the team members. Below is the image showing the predictions for each test image, along with the accuracy of the model on test set.

```
['or work on line level', 'my name is Shivam', 'God helps those who help themselves', 'Quick brown fo
r jumps over the lacy dog', 'Adversity and loss make a man wise', 'A foo and his money are soon parte
d', 'All good things come to an end', 'Always put your best foot jorward', "All's fair in love and wa
r", 'Necessity is the mother of invention', 'No news is good news', "She doesn't study geometry on Su
ndany", 'life is a dream for the wise', 'Peligion is regarded by the commons as true', 'learn to spea
k welll and listen better', 'Give more than you take', 'Yesterday is the deadlined for alll ccomplain
ts', 'No one can ruin your play without your permission', 'Doy the plifficute things while they are e
asy', 'The greatest wealth is to live content with little', 'Happiness and Areedomt beging with oone
principle', 'Actions speak louder than words', 'A computer is able to learn from experience', "Predic
ting the future isn't magne , it's artificial intelligence", 'A bad workman always blames his tools',
 'Better Late than neves', 'Ignorance is bliss', 'Cleanliness is next to codliness', 'Familiarity bre
eds Contempt', 'Fortune favors the brave', 'Blood is thicker than water', 'All that qlitters is not G
old', 'A drowning man will clutch at a straw', 'Five boxing wizards jumped quickly']
Accuracy (considering case-sensitivity) =  0.9146919431279621
Accuracy (not considering case-sensitivity) =  0.933649289099526
```

We have considered 2 different model evaluation metrics. Below are their definitions and results for our 34 custom test images.

1. **Test Accuracy (considering case-sensitivity)** (α_c): This is computed by checking how many words have been recognized by *SiLiHaSeR* correctly; if the case of a character is not recognized correctly, then the word will be considered as "incorrectly recognized". Mathematically,

$$\alpha_c = \frac{\text{Correctly recognized words considering case of each character of the word}}{\text{Total number of words}} = 91.469\%$$

2. **Test Accuracy (without considering case-sensitivity)**(α_{nc}): This is computed by checking how many words have been recognized by *SiLiHaSeR* correctly, not considering the case of the characters of each word. Mathematically,

$$\alpha_{nc} = \frac{\text{Correctly recognized words not considering case of each character of the word}}{\text{Total number of words}} = 93.364\%$$

5 Contributions of Team Members

- The line segmentation unsupervised learning algorithm for *Phase-1* is developed by **Shivam Pandey**.
- Understanding the NN architecture and implementing the word recognition deep learning model for *Phase-2* is done by **Michael Yang** and **Aniket Banginwar**.
- The proposed design of language model for *Phase*-3 is collaboratively done by **all** team members.

Overall, all team members gave adequate efforts on this project, and brainstormed together to develop a "good enough" working model.

6 Conclusion and Future Directions

We successfully deployed a "good enough" working deep learning model using the concepts of *supervised* and *unsupervised* learning, along with some help of dynamic programming. To our surprise, the *SiLiHaSeR* gave very nice sentence predictions by utilizing *Phase*-1 + *Phase*-2, without using *Phase*-3. However, there are some assumptions which were taken into consideration while developing *Phase*-1 and *Phase*-2.

For instance, while developing the unsupervised clustering algorithm for *Phase*-1, we assumed that the interword spaces are not slanted (like in the case of extremely cursive handwriting). In slant cursive handwriting, we would not get the desired result by finding the horizontal interword space. One way to tackle this problem is by taking projection of the sentence on a new tilted axis. This tilting of axis should be carefully done by an optimal amount, so that the spaces are properly obtained by the projection of our sentence on the new tilted axis. Further, if the size of intraword space is comparable to the size of interword space, then the problem becomes more challenging, and we might need to incorporate some additional contextual information to detect interword spaces.

Data augmentation can be done to improve the accuracy of our model. This can be done by some random transformation to the training and test images. Deslanting could be used to further improve the NN accuracy. Even adding extra CNN layers, and using a 2D-LSTM can help better train the model. For building the *Language model*, *trigrams* and in general, *ngrams* may be used to improve the results.

References

[1] IAM. URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database.

[2] Harald Scheidl. URL: https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5.