Project Report

Matching handwritten document images using CNN

Submitted by:

Anjana K	Shivani Naik	S Maneesha	Vaishali Jain
MT2016011	MT2016088	MT2016119	MT2016149

Problem Statement

Our aim is to calculate similarity between documents written by different individuals. We compare two documents at an image level by segmenting the words and comparing these segmented words to obtain a similarity score of the two documents.

This method avoids the conversion of text to OCR, which is not reliable for most type of documents. The assigned similarity score would be decided irrespective of factors like word form variation, order in which words appear in different documents, format of the document and paraphrasing. We will be using word spotting which is a known technique to retrieve text from images.

Motivation for solving this problem:

Our main motivation for taking up this problem was to contribute to the field of academia for automating the scoring of answer scripts and plagiarism detection in handwritten documents.

This project also involves the application of major machine learning techniques like CNN for training the data. This would give us an insight into some of the advanced concepts in machine learning and their applications.

Word segmentation makes use of algorithms like connected components analysis and watershed, which will help us understand some techniques of image processing.

Summary of related papers:

Base Paper

Title: Matching Handwritten Document Images
Authors: Praveen Krishnan and C.V Jawahar

Content in brief:

The paper discusses a method for estimating similarity between handwritten documents. Similarity score is determined by estimating common key words across the two documents.

Dataset Construction: In this problem there was lack of data for training handwritten word images. To address this a synthetic handwritten dataset of 1 million word images very similar to natural handwriting was built. It was formed out of 750 different handwritten fonts. For each sample 100 samples were randomly sampled and image rendered. Parameters such as kerning level , stroke width, and mean foreground and background pixel distributions were varied during the process to ensure maximum variance amongst samples.

The method is implemented in two distinct modules as follows:

Feature extraction and Classification of Words: A document is basically a distribution of words. Initially a classifier is constructed for about 10000 most common words from English Language vocabulary. The paper proposes use of Convolutional Neural Network for constructing the classifier. The CNN architecture defined consists of convolutional layers of 4, 128, 256, 512 and 512 square

filters with dimensions: 5, 5, 3, 3 and 3 respectively, followed by two fully connected layers. In the work, the classes were determined from the last FC layer to represent
each handwritten word image.

Document Segmentation and Similarity Estimation: The documents need to be segmented to component words for estimation of similarity. The paper proposes a connected component based method to achieve segmentation. The words from both documents are classified using classifier constructed earlier. The similarity is then computed as the symmetric distance between the best word matches across the documents.

Reference Paper 1:

Title: ICDAR2013 Handwriting Segmentation Contest
Authors: Nikoloas Stamatopoulos, Basilis Gatos et-al

Content in brief:

The paper discusses various segmentation approaches resulting from Handwriting Segmentation Contest organized in the context of ICDAR2013. A brief description of each of the eleven submitted segmentation algorithms is given in the paper. The algorithms were evaluated against a standard benchmarking dataset created solely for the purpose.

GOLESTAN Algorithm: Handwritten text image is first filtered using a 2D Gaussian filter. Filtered image is divided into a number of overlapped blocks, the skew angle for each of which is determined. Adaptive thresholding is then applied and binarized. Binarized blocks are concatenated to get text lines. Text lines are then filtered using a 2DGaussian filter. Ascenders and descenders are then eliminated and thresholding applied again to finally extract the words.

Connected Component Algorithm: One of the algorithm is based on connected component analysis. The average width and height of connected components (CCs) are first estimated using statistical metrics methods. The CCs of normal size that are close to each other and almost at the same latitude are grouped into short text lines. At a next step, the previously detected text lines are merged into long text lines according to their direction, latitude and the intersections between them. Finally, the CCs with abnormal size are merged with the existing text lines by checking the neighborhood. Once the text lines are detected, the horizontal density of each text line is estimated and a closing operation is applied according to it. Finally, the average distance between adjacent words is calculated and is used to merge adjacent words whose distances are smaller than this value

Other methods included energy minimization framework considering the fitting errors of text lines and the distances between detected text lines,adaptive thresholding followed by double smoothing and merging, K-means clustering approach, construction of graph using skeleton of image and path finding.

Reference Paper 2:

Title: Handwritten Word Spotting in Old Manuscript Images using a Pseudo-Structural Descriptor Organized in a Hash Structure
Authors: David Fern´andez, Josep Llad´os, and Alicia Forn´es

Content in brief:
One of the implementation of handwriting recognition is digitizing the historical documents. There is an increased emphasis on them due to the vast source of information they provide and the need of preservation due to their fragile state. This task faces a large number of difficulties like degraded quality of document and increased noise due to their age. The paper proposes word spotting technique for the same which is a pattern classification problem in which keywords are detected and words are represented as shape features. The paper uses query-by-example approach of the technique and considers word as shapes and spotting is achieved through shape dissimilarity functions. It has three steps:

- Pre-Processing Step:
  This is used to improve the quality of the documents due to the falling quality of the documents.
- Feature Extraction:
  In this ,feature is composed by the number of intersections in the four directions.
- Matching Words:
  This is the retrieval approach and consists of organizing the encoded the features in lookup table. Classification is then done by taking the best matching query of the word from the table.

Dataset:

IIIT-HWS Data Set is a synthetic handwritten dataset of 90K word images which are very similar to natural handwriting. It is formed from 750 handwritten fonts. It consists of 100 images for each word written in different fonts. We are using a subset of this data set.

A list of 500 most commonly used words starting with all alphabets was made to select the dataset words. We have used 50,000 images from this dataset, corresponding to the 500 word classes.

Approach:

Our project has three modules:

- The CNN will be trained using the word images dataset.
- Given two input document images, we will be getting the constituent words from the document images using segmentation of the image.
- Next, we will be using our trained CNN to predict the class of each segmented word and calculating similarity score.

We have worked on the first 2 modules so far.

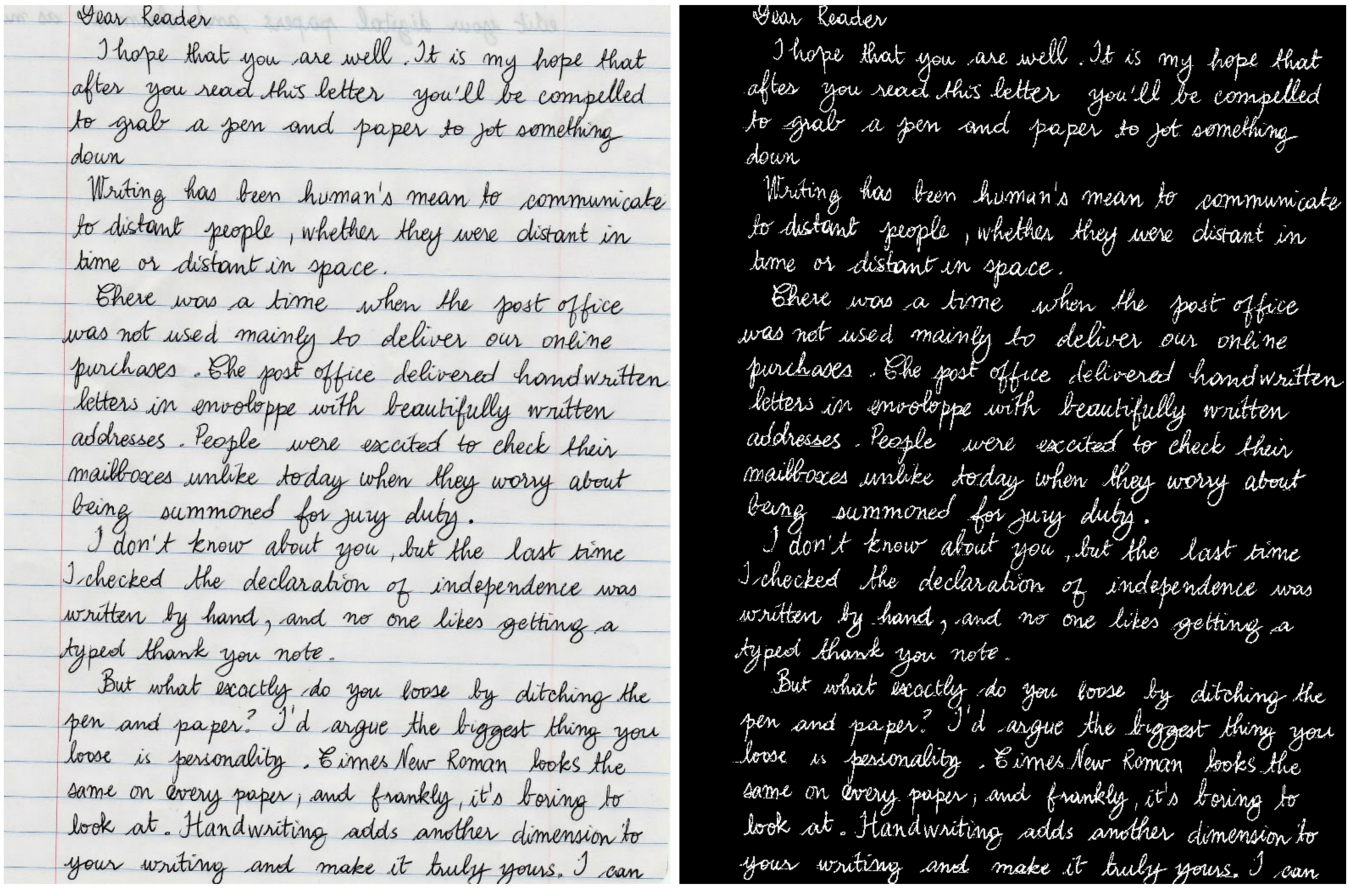
Module 1] Segmentation of document images:

Each document will be represented as a word distribution. For this, we need to detect the constituent words present in the documents, which is done using segmentation.

This is the preprocessing step for every document image.

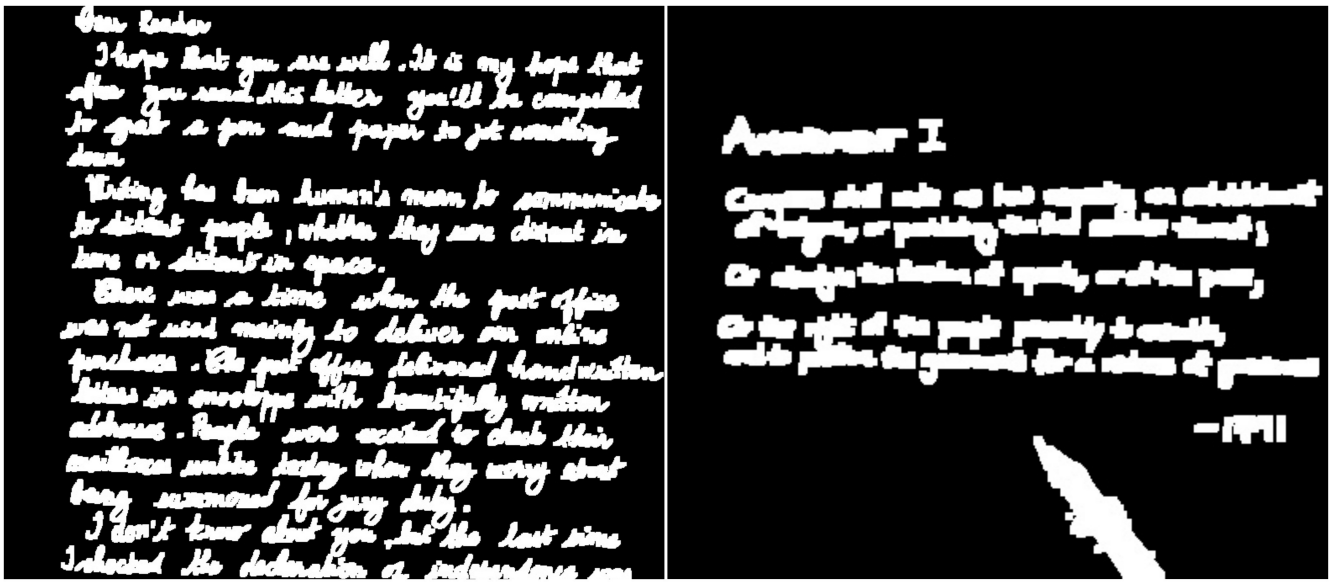
Thresholding: Each document is converted to a binary image using thresholding. We have used binary inverse thresholding, which results in the image having a black background, with the text written in

white.



Thresholding of image

Dilation: The binary document image is dilated with a kernel size (5, 5), to merge the letters of one word together. The optimal kernel size for each document varies with image and letter spacings of a word. A smaller kernel size of (3, 3) was found to not have the desired effect of clubbing the letters of one word into a blob, whereas a larger kernel size of (7, 7) clubbed two different words as a single word.



Dilated binary images

We have tried two different approaches to get the segments from this dilated binary document image.

Approach 1:

This makes use of the watershed algorithm. Any greytone image can be considered as a topographic surface. Region edges correspond to high watersheds and low-gradient region interiors correspond to catchment basins. Catchment basins of the topographic surface are homogeneous in the sense that all pixels belonging to the same catchment basin are connected. Such catchment basins then represent the regions of the segmented image.

The idea of the algorithm to distinguish and detect these catchment basins and watershed lines by flooding the surface from this minima and preventing merging of waters coming from different basins.

Markers are defined to prevent over segmentation due to noise. The markers control flooding in an efficient manner (restricting flooding to start from these points alone) so as to reduce over-segmentation.

Our objective is to segment the words out of the document and is essentially similar to separating objects close to each other. The idea is thus to create borders or watershed lines as far as possible from centre of adjacent objects. Distance transform is applied to transform the image to a grayscale image in which the intensities of pixels for any word represent the distance of that pixel to the closest boundary from it.

To use the watershed algorithm, we define markers in the image. Given the markers, we get a distance map for the pixels using the Euclidean Distance Transform, which is used by the watershed algorithm to find the segments in the image.

A bounding box for each segment is obtained, and these segmented words can be used to determine the

word distribution of each document.

On trying this with multiple images, it is observed that this approach gives a large number of smaller segments. This means that it sometimes segments one word, which should be considered as a single segment, into multiple segments. This would be a drawback since we need the entire word to classify it into some word class. Future work would include trying out more changes in the hyper parameters to check if it can give better results.

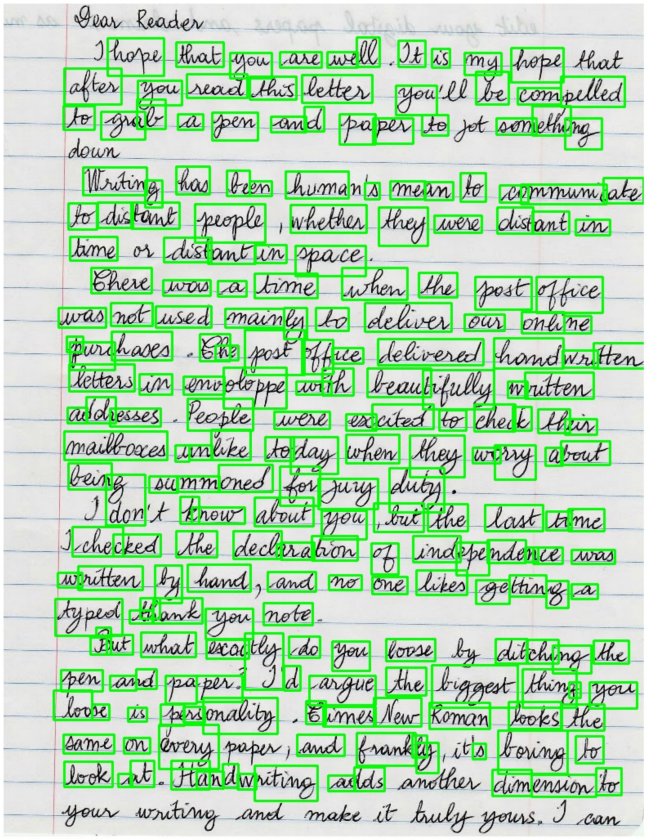
Approach 2:

This makes use of the connected components concept. Connected components gives clusters of pixels with same value, and since we have a binary dilated image, this works well. 4-pixel connectivity would group all pixels that contact each other on either of their four faces, while 8-pixel would group pixels that are connected along any face or corner. This gives us the word segments from the image which can be used later.

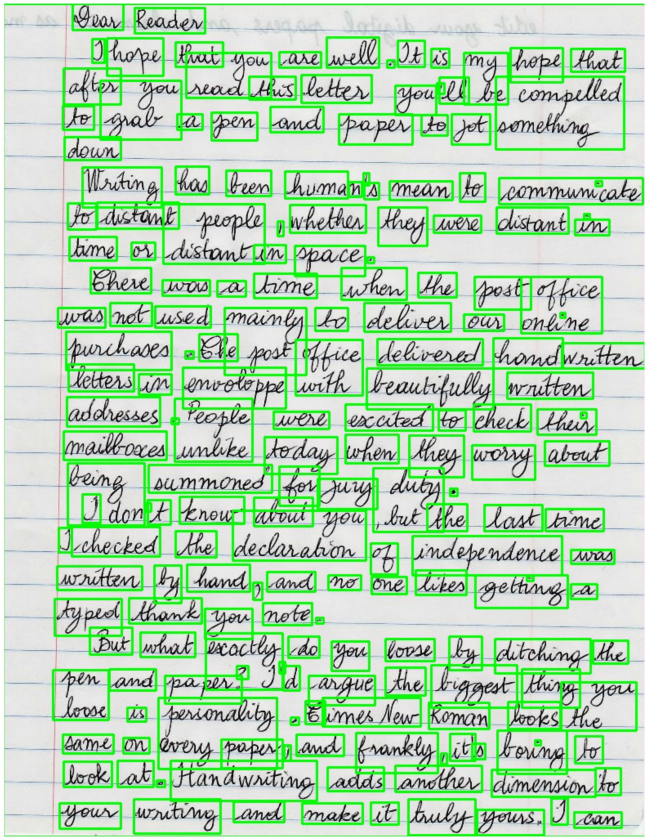
This approach sometimes considers more than one word as a connected component, even if there is just a slight overlap between adjacent dilated word blobs. This can be reduced if dilation is reduced, but that would result in improper segmentation of other words.

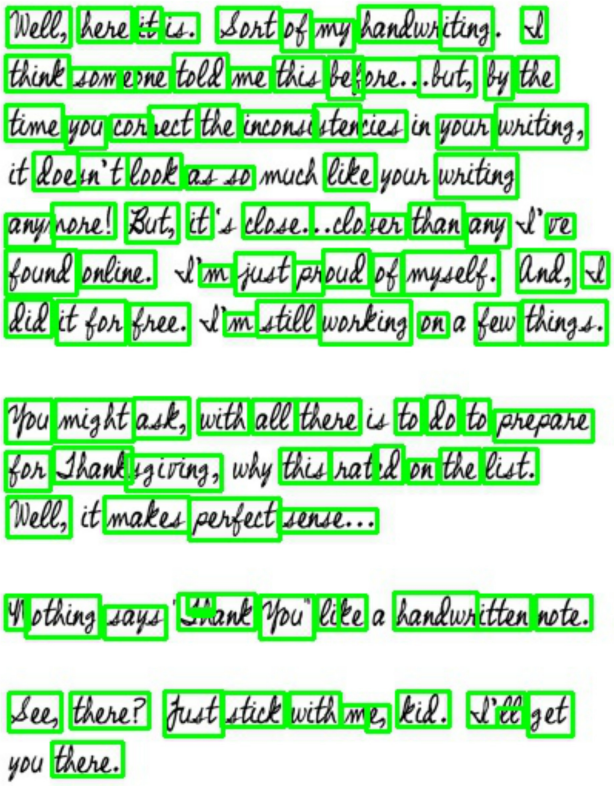
This approach seems to be **better** as it gives a better word segmentation than the 1st approach, because it takes the entire word as one segment, as opposed to segmenting a word into multiple segments, which happens when we use the previous procedure. So we are using this approach for segmentation.

Approach 1



Approach 2





Well, here it is. Sort of my handwriting. I think someone told me this before...but, by the time you correct the inconsistencies in your writing, it doesn't look as so much like your writing anymore! But, it's close...closer than any I've found online. I'm just proud of myself. And, I did it for free. I'm still working on a few things.

You might ask, with all there is to do to prepare for Thanksgiving, why this rated on the list. Well, it makes perfect sense...

Nothing says "Thank You" like a handwritten note.

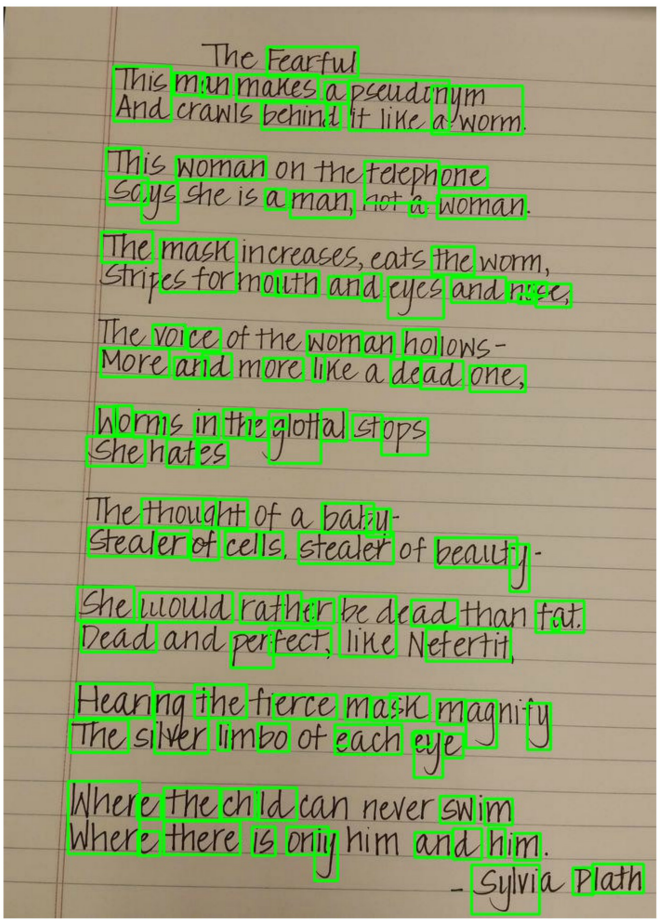
See, there? Just stick with me, kid. I'll get you there.

Well, here it is. Sort of my handwriting. I think someone told me this before...but, by the time you correct the inconsistencies in your writing, it doesn't look as so much like your writing anymore! But, it's close...closer than any I've found online. I'm just proud of myself. And, I did it for free. I'm still working on a few things.

You might ask, with all there is to do to prepare for Thanksgiving, why this rated on the list. Well, it makes perfect sense...

Nothing says "Thank You" like a handwritten note.

See, there? Just stick with me, kid. I'll get you there.



The Fearful

This man makes a pseudonym
And crawls behind it like a worm.

This woman on the telephone
Says she is a man, not a woman.

The mask increases, eats the worm,
Stripes for mouth and eyes and nose,

The voice of the woman hollows-
More and more like a dead one,

Worms in the glottal stops
She hates

The thought of a baby-
Stealer of cells, stealer of beauty-

She would rather be dead than fat.
Dead and perfect, like Nefertiti,

Hearing the fierce mask magnify
The silver limbo of each eye

Where the child can never swim
Where there is only him and him.

- Sylvia Plath

The Fearful

This man makes a pseudonym
And crawls behind it like a worm.

This woman on the telephone
Says she is a man, not a woman.

The mask increases, eats the worm,
Stripes for mouth and eyes and nose,

The voice of the woman hollows-
More and more like a dead one,

Worms in the glottal stops
She hates

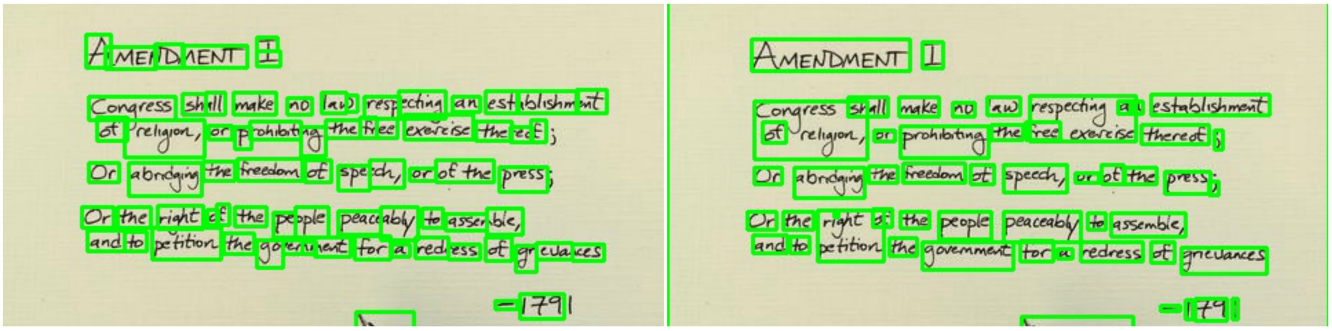
The thought of a baby-
Stealer of cells, stealer of beauty-

She would rather be dead than fat.
Dead and perfect, like Nefertiti,

Hearing the fierce mask magnify
The silver limbo of each eye

Where the child can never swim
Where there is only him and him.

- Sylvia Plath



Module 2] CNN Training:

We are trying to model a word classification task using the CNN. Number of classes corresponds to the number of words in the dataset. Given an input word image, the CNN should be able to classify it into one of the word classes, where each class corresponds to one word from the dictionary, i.e. the dataset.

We performed a 75 % - 25 % training-testing split for checking the accuracy of the different trained CNNs.

Multiple CNN architectures with increasing number of target classes were tried. Following is the summary of these architectures:

Architecture 1:

This CNN had two Convolutional layers with 20 and 50 square filters, both with dimensions 5. This means we are learning 20 and 50 feature maps respectively. This is next connected to a fully connected layer with 64 neurons. The last layer uses a fully connected (FC) layer with dimension equal to number of classes, and is further connected to the softmax layer to compute the class specific probabilities. Rectified linear units are used as the non-linear activation units after each weight layer and 2 × 2 max pooling is applied after the convolutional layers with a stride of two and padding is done to preserve the spatial dimensionality. Max pooling is used to reduce the dimensions of the feature map, it outputs the maximum activation in the 2×2 input region.

For 5 word classes with total 500 images, this architecture gave a **very low accuracy of 32%.**

```
375/375 [==============================] - 112s - loss: 1.5589 - acc: 0.3440
Epoch 18/20
375/375 [==============================] - 117s - loss: 1.5478 - acc: 0.3253
Epoch 19/20
375/375 [==============================] - 83s - loss: 1.5427 - acc: 0.3893
Epoch 20/20
375/375 [==============================] - 81s - loss: 1.5385 - acc: 0.2987
[INFO] evaluating...
125/125 [==============================] - 10s
[INFO] accuracy: 32.00%
```

Accuracy of 5 classes with 500 images

Architecture 2:

This CNN had 5 Convolutional layers with 64, 128, 256, 512 and 512 square filters of size 5, 5, 3, 3, 3. Next it is connected to two FC layers, one with 64 neurons and next with neurons equal to number of classes with a softmax layer. Rectified linear units are used as the non-linear activation units after each weight layer and 2 × 2 max pooling is applied after the convolutional layers with a stride of two.

For 5 word classes with total 500 images, this architecture gave a **very low accuracy of 13.60%.**

```
375/375 [==============================] - 360s - loss: 1.5840 - acc: 0.3973
Epoch 47/50
375/375 [==============================] - 360s - loss: 1.5848 - acc: 0.3040
Epoch 48/50
375/375 [==============================] - 360s - loss: 1.5837 - acc: 0.2507
Epoch 49/50
375/375 [==============================] - 359s - loss: 1.5839 - acc: 0.2453
Epoch 50/50
375/375 [==============================] - 360s - loss: 1.5833 - acc: 0.3493
[INFO] evaluating...
125/125 [==============================] - 21s
[INFO] accuracy: 13.60%
```

Accuracy of 5 classes with 500 images

Architecture 3:

Three Convolutional layers with 32, 32, 64 square filters of size 3 each. This is connected to a FC layer with 2048 neurons, which is then connected to another FC layer with neurons equal to number of classes. It is further connected to sigmoid layer to compute class specific probabilities. As in previous architectures, Relu activation units are used with similar max pooling layers.

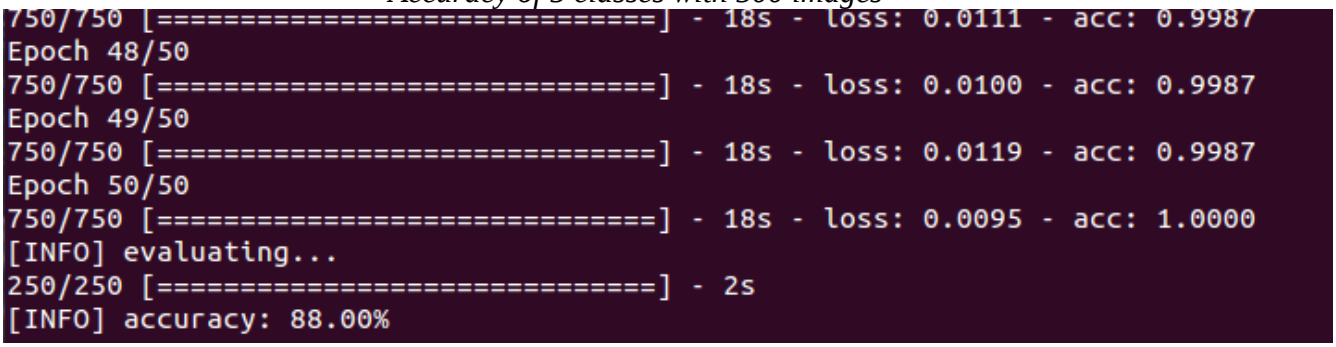
This architecture gave the **highest accuracy** of all the architectures:

Number of word classes	Total number of images	Accuracy
5	500	87.20%
10	1000	88%
100	10000	86.88%
500	50000	82.66%

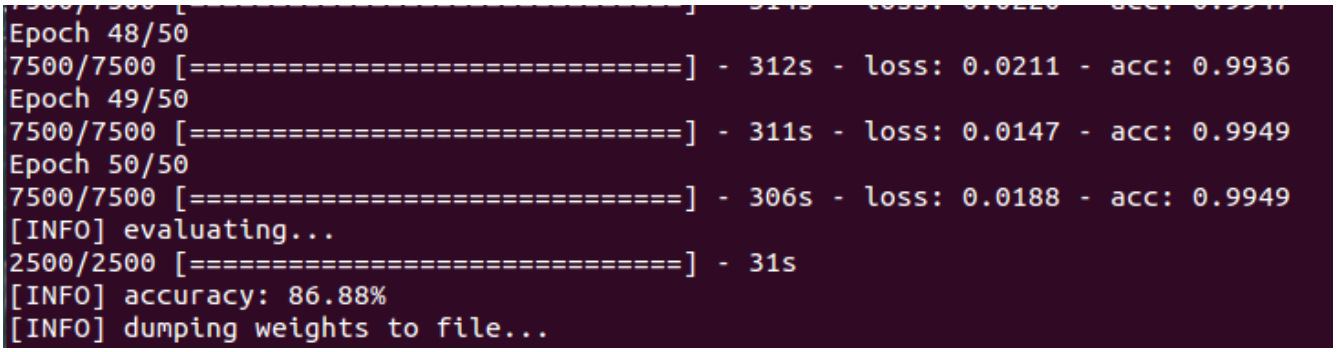
Thus we have decided to use this architecture.

```
375/375 [==============================] - 7s - loss: 0.8544 - acc: 0.6773
Epoch 48/50
375/375 [==============================] - 7s - loss: 0.8723 - acc: 0.6347
Epoch 49/50
375/375 [==============================] - 7s - loss: 0.6213 - acc: 0.8533
Epoch 50/50
375/375 [==============================] - 7s - loss: 0.5425 - acc: 0.8507
[INFO] evaluating...
125/125 [==============================] - 0s
[INFO] accuracy: 87.20%
```

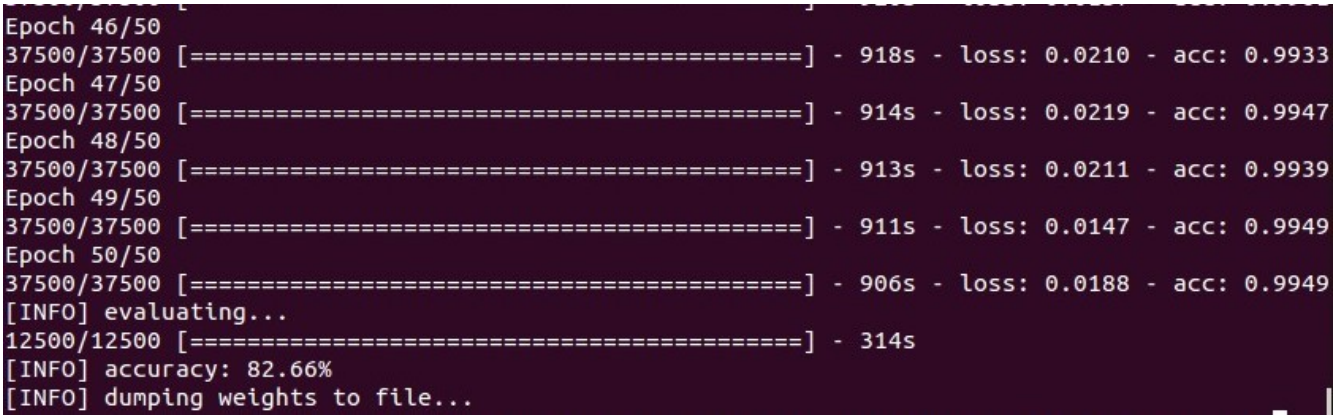
Accuracy of 5 classes with 500 images



Accuracy of 10 classes with 1000 images



Accuracy of 100 classes with 10000 images



Accuracy of 500 classes with 50000 images

Using different traditional features and classification methods:

Apart from using CNN for classification, we have tried 2 different methods of classification: BOW using SIFT features, LBP features.

I Bag of Visual Words using SIFT features:

We have used the BOW approach for classification. In this, we have considered each word to be made up of “SIFT words”. A vocabulary of visual words is created from the training set. This vocabulary is

used to represent each word image as a histogram of these visual words.

We have used SVM and K Nearest Neighbors with various hyper parameter configurations for the classification task. They are trained using the histogram feature vector generated for all word images.

For implementation purpose, we have used the classes provided by Opencv:

BowKMeansTrainer: Class used to generate BOVW vocabulary using K-means. Takes SIFT descriptors of keypoints of images as input.

BowImgDescriptorExtractor: Class used to compute image descriptor using provided bag of visual words vocabulary.

Procedure:
1. Vocabulary Creation:

i. Extract SIFT features from images.

ii. Add these SIFT descriptors to instance of *BowKMeansTrainer()*.

iii. Cluster these descriptors to form k visual words using K-means. Use *cluster()* method of *BowKMeansTrainer*.

iv. These cluster centers will represent k visual words of the vocabulary. This vocabulary is written to disk to be used later (using python pickle). We don't need to generate vocabulary every time we run the program.

2. Training

i. For each image in training set, create a histogram of words from the vocabulary by assigning the closest cluster center to the features. This is done by using an instance of *BowImgDescriptorExtractor()* which has its vocabulary set to the generated vocabulary from the previous step. Thus each image is represented using the vocabulary words histogram.

ii. Train (i.e remember) k-Nearest Neighbours by using these descriptors of images. Alternatively, train an SVM.

Effect of hyper parameters:

1] Number of "SIFT word clusters" in BOW vocabulary:

We tried using different number of clusters in K-Means, to get varied length image descriptors. Tested numbers included 30, 50, 70, 100, 700. Out of these, 50 gave reasonable accuracy compared to 30 and 70. But significant increase in accuracy was found by setting K=700, and pairing this along with the SVM's final chosen parameters.

2] K-NN:

K-NN was not effective in classifying the word images. For 10 word classes, K= 5 gave an accuracy of

25%, but as the number of classes was increased to 100, it reduced to 5%. With K=5 and 10 classes, accuracy was 22% and K=7, it was 24%. Thus, the value of K did not significantly affect classification accuracy

```
(array([[ 13505.]], dtype=float32), array([[ 72962.,  73784.,  13505.]], dtype=float32))
Actual: 72770 Predicted:[ 13505.]
(array([[ 72824.]], dtype=float32), array([[ 72824.,  72824.,  11500.]], dtype=float32))
Actual: 72824 Predicted:[ 72824.]
62
Accuracy:0.225454545455
```

Accuracy with K = 3

```
(array([[ 11150.]], dtype=float32), array([[ 72962.,  73784.,  13505.,  11150.,
 13695.]], dtype=float32))
Actual: 72770 Predicted:[ 11150.]
(array([[ 15024.]], dtype=float32), array([[ 72824.,  72824.,  11500.,  15024.,
 15024.]], dtype=float32))
Actual: 72824 Predicted:[ 15024.]
Accuracy:0.254545454545
```

Accuracy with K = 5

```
(array([[ 11150.]], dtype=float32), array([[ 73784.,  73918.,  72962.,  11150.,  13505.,  11150.,  73784.]], dtype=float32))
Actual: 72824 Predicted:[ 11150.]
(array([[ 72770.]], dtype=float32), array([[ 72770.,  72770.,  11150.,  72770.,  13695.,  72824.,  15024.]], dtype=float32))
Actual: 72770 Predicted:[ 72770.]
66
(array([[ 11150.]], dtype=float32), array([[ 72962.,  73784.,  13505.,  11150.,  13695.,  72824.,  73918.]], dtype=float32))
Actual: 72770 Predicted:[ 11150.]
(array([[ 11500.]], dtype=float32), array([[ 72824.,  72824.,  11500.,  15024.,  15024.,  11500.,  72770.]], dtype=float32))
Actual: 72824 Predicted:[ 11500.]
Accuracy:0.24
```

Accuracy with K = 7

3] SVM:

We tried linear SVM with no extra parameters set to train 100 word classes. The model was tested by performing a 75%-25% train - test split. This gave a very low accuracy of 15%.

To improve the classification accuracy, different values for the hyper parameters were tried. By setting Kernel to SVM_RBF, C to 1000, Type to SVM_C_SVC and P to 0.001, a drastic increase in accuracy was obtained, leading to 58%.

```
(0.0, array([[ 11500.]], dtype=float32))
Actual: 72770 Predicted:11500.0
(0.0, array([[ 72770.]], dtype=float32))
Actual: 72770 Predicted:72770.0
159
(0.0, array([[ 72824.]], dtype=float32))
Actual: 72824 Predicted:72824.0
160
Accuracy:0.581818181818
```

Accuracy with BOW, SVM and Number of word clusters = 700

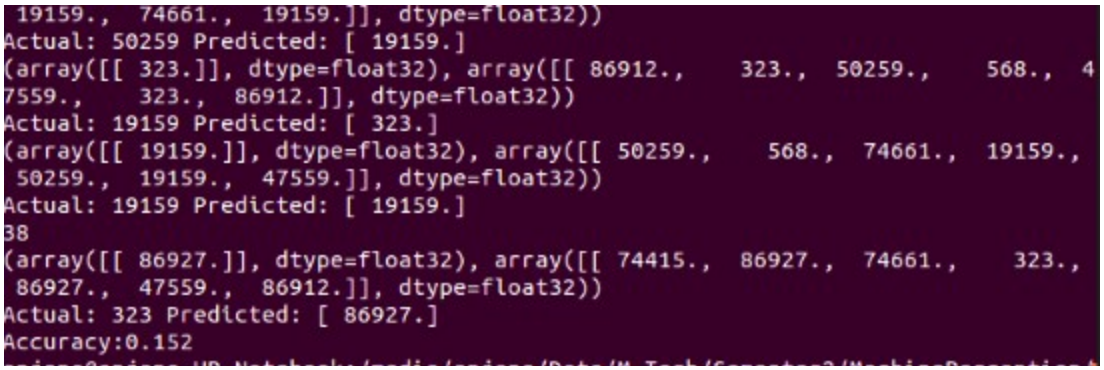
II LBP features:

LBP stands for local binary pattern. The training process was also performed by extracting LBP patterns from the word images. The LBP operator labels the pixels of an image by thresholding neighborhood of each pixel (the radius and number of neighbors of which are hyper parameters) with the center value and considering the results as a binary number.

The vocabulary is initially built by extracting LBP features of each of the training images and performing K-means clustering over the feature descriptors. The resulting clusters form the visual words and together constitute the vocabulary.

Further each of the training image is represented as the histogram of these visual words and also labeled with the corresponding class label. The histograms are now the features representative of the images. Training was performed using both SVM with RBF kernel and KNN with a k value of 7.

The model was evaluated by splitting 25% of the images to test set. The accuracy was found to be 26% for 5 classes(words) and 15.2% for 10 classes. LBP was thus decided to be unsuitable for the purpose, as it focuses more on the texture histograms, which may not differ much for handwritten words.



Accuracy with LBP features

As we can see, CNN classification accuracy is much better than any of the other approaches that were tested.

Module 3] Similarity Score Calculation:

The first two modules of the project were construction of the classifier and the image segmentation. Both the modules need to be integrated to perform the actual document matching.

In this module, we are comparing the word distributions of given documents. For this, we represent each document as a histogram of constituent words. This is formed by classifying the document's word segments into classes and getting a histogram over these classes.

For this purpose, the documents are first passed through the segmentation procedure. This results in two distinct list of constituent words corresponding to each of the documents.

Now both the lists of words are passed through the trained word classifier to classify and label the words to their corresponding classes. A special class 0 is defined to dump the junk words like stop words which do not serve any purpose in similarity determination. This step results in the formation of the document representation, i.e., the word histogram.

Finally to determine similarity, the number of common labels across the two histograms of classified words (non-zero class label) is determined. We take the minimum count between the 2 histograms for every matched label. This ensures that multiple occurrences of same word are also taken into consideration. This is the match count. Similarity score is computed as:

$$\textit{Match count} = \sum_{i=1}^{labels} min(D1(i), D2(i)) \quad \textit{where D1, D2 are the document histograms}$$

$$\textit{Score} = (\textit{Match count} / \textit{length(D1)} + \textit{length(D2)}) * 2$$

The score is multiplied by 2 to set its range in [0 , 1].

As we are considering only the histogram of words, this method takes care of the spatial distribution of words. This means, even paraphrasing of the same lines will result in a high similarity score.

Eg. of similarity score here

Future work and scope:

- Better techniques for word segmentation can be formed, which take care of the slant lines and skew in document images. This will result in a better classification accuracy.
- The segmentation technique that we have used is a little image dependent when it comes to the kernel size for dilation used. For images with highly spaced letters of a word, a bigger kernel size is required, and for images with words with very low letter spacing, a smaller kernel suffices. This can be incorporated by using some techniques to analyze such spacings.
- The scope can be increased by including more number of word classes in the training.
- Another way of estimating the similarity score will give better accuracy. As mentioned in the reference paper, l_2 distance between l_2 normalized CNN descriptors can be used along with nearest neighbor search using KD trees to avoid exhaustive searching.