Handwritten Text Recognition using Deep Learning

Batuhan Balci, Dan Saadati, Dan Shiferaw

Objective

Our project seeks to classify handwritten words. Our initial approach was classifying the handwritten words directly. After exploring this approach, we moved to segmenting individual characters and reconstructing the word based on the character classification.

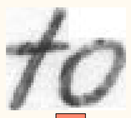
Background

Recognizing handwritten text has historically proven to be a difficult problem. The United States Postal Service was the first to attempt OCR (Object Character Recognition) in 1982 to classify addresses. In the 1990s and 2000s, this problem was approached using HMMs, which required massive, manual, and error-prone feature engineering. More recently, deep learning, in the form of RNNs and CNNs, has been used for handwritten recognition and has experienced great success due to automated feature representation based off of raw data. Language models further refined classifications by estimating probabilities of candidate character/word sequences.

Data

- For character segmentation from source text images, we used the Tesseract English language training data with over 450,000 lines of text spanning about 4500 fonts
- For direct word classification, we use the IAM dataset which provides handwritten text with over 110,000 labeled words contributed from 650 writers.

Preprocessing



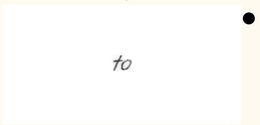


Figure 1: The input word is taken and is padded with white pixels so the images are consistent in size.

- Rotating some word images slightly(before the padding step below) as some words might not have been written straightly
- Padding zeros around the word images to make all the images the same size
- Subtracting the dataset mean from the images

Figure 2: Images are tilted and rotated slightly in order to account for text not written completely straight.

Methodology

Word Classification

- We first construct a vocabulary based on randomly selecting 50 words with at least k occurrences in our dataset.
- We train our word classifier with multiple CNN architectures: RESNET-18, RESNET-34, and VGG-16
- We use cross entropy softmax objective function with Adam optimization for the training step.

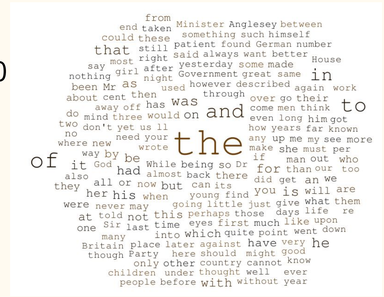


Figure 3: A cloud visualization of the words that can contribute to the word vocabulary.

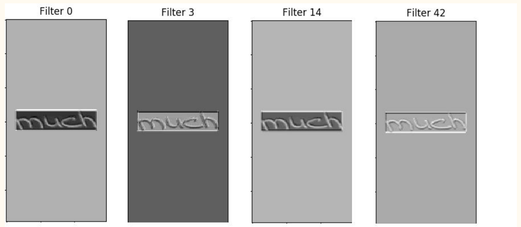


Figure 4: The activations of the first convolutional layer in the ResNet architecture for various filters when applied on one of the words in our dataset: 'much'

Character Classification

Same underlying architecture. Changes included:

- Character-segmented input images
- Character-level vocab for final softmax output layer
- Different parametrization for model after training

Segmentation

- Fine-tuned the Tesseract LSTM-CNN segmentation model originally trained on internal English language dataset on IAM Handwriting dataset to output segmented character images from an input word/line/form image file
- Segmented character images were then fed into character-level classifier

Figure 5: Sample word image, character classification, confidence values, and bounding box values.

```
'M';    conf: 70.89; BoundingBox: 0,2,63,48;
'O';    conf: 79.18; BoundingBox: 61,4,117,45;
'V';    conf: 91.01; BoundingBox: 124,2,161,43;
'E';    conf: 75.47; BoundingBox: 169,1,213,42;
```

Results & Analysis

Word Classification Results(with Low Epochs due to Time and Budget Constraints)

	Training Accuracy	Validation Accuracy	Speed (Relative to RESNET-18)
VGG-16	28%.	22%	3.98
RESNET-18	31%	23%	1.00
RESNET-34	35%	27%	1.67
Char-Level Classification	38%	33%	1.8

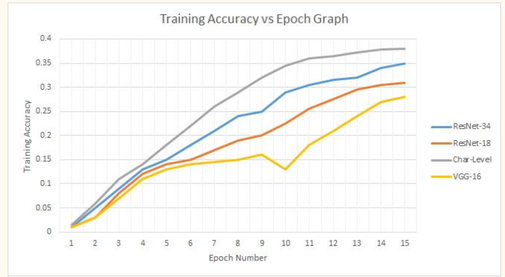


Figure 6: Training accuracies over 15 epochs of the various approaches we have implemented.

- The curves on our loss graph show that we have room to increase our training accuracy by increasing the number of epochs.
- The loss graph shows the majority of improvement occurs after around 3 epochs.

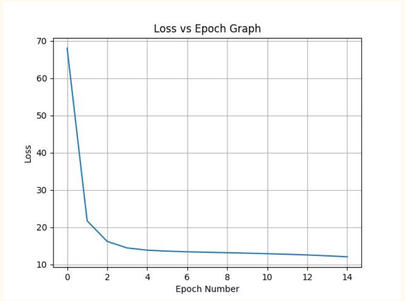


Figure 7:(Cross Entropy) Loss over 15 epochs of the ResNet-18 Model

Conclusion/Future Work

We have found that classifying words directly is a very challenging problem because of the extensive vocabulary size. Segmenting characters and reconstructing words does better, but is limited by having any character be misclassified. We plan to extend our character-level word classifier by support with it with a language model that can calculate how likely it is that that word was correctly classified. This will allow us to substitute characters and search for a more probable classification.