Siamese Convolutional Neural Networks for Authorship Verification

William Du
Stanford University
450 Serra Mall, Stanford, CA 94305
willadu@stanford.edu

Michael Fang
Stanford University
450 Serra Mall, Stanford, CA 94305
mjfang@stanford.edu

Margaret Shen
Stanford University
450 Serra Mall, Stanford, CA 94305
marshen@stanford.edu

Abstract

Determining handwriting authorship of a written text has practical significance in the realm of forensics, signature verification, and literary history. While there have been studies in signature verification and handwriting classification, a vast literature review reveals that very little work has been done in handwriting verification. Recent advances in convolutional architectures, particularly those involving facial verification, suggest that the task can be tackled effectively. In this study, we build a Siamese convolutional neural network to determine whether two pieces of handwriting are written by the same author. We examine questions such as whether long pieces of handwriting must be present to achieve good results, how many samples are needed, what features are important, and how different architectures perform on this task. We explore different convolutional architectures like VGG, GoogLeNet and ResNet, to determine which architecture produces the best encoding of each sample. We note that our best performing single model, TinyResNet, achieves a 92.08% accuracy on the held out test set.

1. Introduction

Determining the authorship of a written text has practical significance in the realm of forensics, signature verification, and literary history [3]. In manuscript analysis, for instance, historians frequently ask questions regarding the number of authors for a text, whether an anonymous work can be confidently attributed to a historical figure, and what time period a text might be from. These kinds of analyses are all based upon comparisons between different writing samples [1]. Techniques in the field have remained largely subjective, however, making the transition to automatic tools difficult.

In addition, handwriting analysis is an established area of study in forensics, but there has not yet been any formal experiments measuring the accuracy of such analysis. As a result, the field is surrounded by much skepticism because of how subjective the process is (compared to, say, DNA testing) [5]. In addition, forensic handwriting analysis is time-intensive and requires two years of training for a person to obtain proper qualifications. The primary objective of this project is to develop an automatic, high-accuracy system which can determine if any two writing samples are written by the same person. In addition, our system should be able to handle authors it has never encountered before.

2. Background and Related Work

Our objective fits well with the Siamese CNN neural network architecture, which was first developed in 1993 to tackle the signature verification problem. [3] This type of architecture takes in two inputs and outputs a distance metric for the inputs. Bromley et al. was able to detect 95% of genuine signatures using this architecture. However, note that the signature verification problem expects a pair of inputs to be very similar to each other to be considered a match. This setup would not be effective for the problem we are trying to tackle, because our system should be agnostic to the actual text in a writing sample.

Other researchers have focused more closely on the authorship identification problem. A study in 2015 by Xing et al. reported an accuracy of 97% in classifying English writing samples for 657 authors. [11] They used the same dataset we will be using in this paper, the IAM Handwriting Database, and a 4-layer CNN. This study gave us confidence that we can achieve high accuracies on authorship problems using the IAM dataset. In a very recent research study from 2016, Yang et al. was able to achieve a 95% accuracy in classifying the authors for Chinese text samples, and a 98% accuracy in classifying the authors for English text samples. [12] They used a combination of 5 convolutional layers on samples preprocessed to show their path signatures (i.e. the order and direction of strokes which were made). While this accuracy is extremely high, the model is an ill fit for many practical applications. First, for historical manuscript analysis or forensics, it is impossible to collect information on the pen movements that created the sample. Second, a supervised classification model would not suffice because it only works on a limited set of pre-defined author classes. Our test inputs will include authors which the training set has never seen, making this a problem that is beyond the scope of simple classification. So far, there have been published studies using Siamese networks for general handwriting verification.

3. Method

3.1. Data

The Institut fur Informatik und angewandte Mathematik (IAM) Handwriting Database, published in 2002, is composed of 1539 pages of scanned handwritten text written by 657 different writers.[6] These pages are also segmented into 6685 isolated and labeled sentences, 13353 text lines, and 115320 words. The images are grayscale PNGs and thus only have one channel with values ranging from 0 (black) to 255 (white).

To construct the pairwise dataset used to train our model, we separate writers used in our training, validation, and test sets. We do so in order to verify that the network is trained to be agnostic of specific author handwriting styles. For a positive match, we randomly sample a writer and then randomly sample two different lines written by that writer. For a negative mismatch, we sample two different writers and then randomly sample a line from each. We construct our datasets to have equal numbers of positive and negative examples.

3.2. Data Preprocessing

For this study, we used images from the segmented words and lines. Pages were not used due to their resolution and subsequent issues with memory. Figure 1 details the data preprocessing pipeline visually. Images were first padded to standardize the size of all segmentations. Images were padded on the top and bottom equally and were padded on the right side to mimic actual English text.

Many of the images had noise due to poor scanning and/or poor optical character recognition highlighting. To fix this, images were thresholded to convert all of the gray patches in the images to white pixel values.

Finally, to improve efficiency in training our networks, images were cropped and then downsampled. Words were padded and cropped to a length of 500 pixels and downsampled by 50%. Lines were padded and cropped to a length of 1000 pixels and downsampled by 25%. Visual examination of the downsampled images revealed that text was still indeed legible and all characteristics of the handwriting were still preserved.

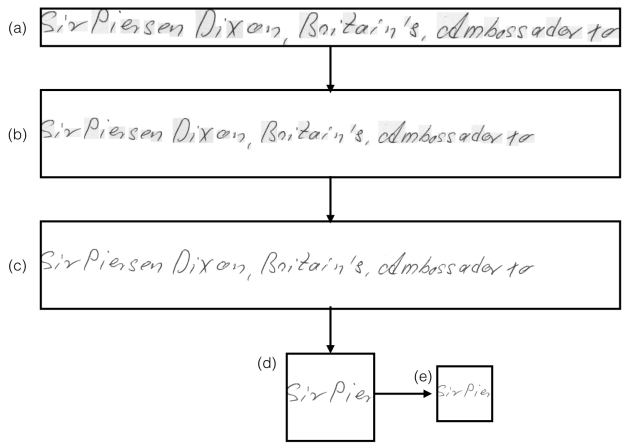


Figure 1. A visual example of data and preprocessing. (a) a line of writing from the dataset. (b) the line which has been padded to max height and width of the dataset. (c) removal of scanning artifacts. (d) cropping for efficiency. (e) scaling for efficiency.

3.3. Architecture

3.3.1 Siamese Architecture

In a Siamese network described in Figure 2, two inputs are taken and evaluated for a score. The two inputs are fed into Siamese convolutional networks that translate each image into latent encoding space. In this study, we vary these convolutional networks to determine which convolutional architecture produces the best encoding of handwriting. These latent encodings are then concatenated and used to produce class probabilities, or in this case the probabilities of whether or not two inputs are duplicates.

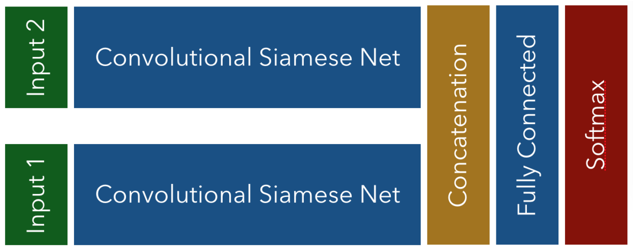


Figure 2. Overall architecture of the Siamese neural network. The convolutional Siamese net is the portion of the network that is varied to produce different encodings of each input.

The convolutional neural network is trained such that each of the Siamese networks share weights, and thus each twin of the network outputs an encoding of an image using the same filters as the other. These networks are trained from scratch.

Although Siamese networks can be trained utilizing a distance function to maximize the distance between mismatches and minimize the distance between matches, this model requires search over multiple distance functions as well as different thresholds for distances. Furthermore, even if we were able to achieve good results with a distance measure, we remain unsure what the "natural" distance measure for the image space is. Early trials optimizing over the $L2$ Euclidean distance did not perform well on this dataset. To address this, we chose to train our network by outputting a softmax over the two classes, author match and author mismatch, thus allowing the model to learn the representation and distance function that best separates the classes.

The output of the Siamese networks for each image is encoded in a n-dimensional vector, where n is 200 in our baseline. Call the output of Siamese net 1 with image A, a, and the output of Siamese net 2, with image B, b. We perform a concatenation of these vectors, the square of their differences, and the Hadamard product of each vector. This results in the following vector of dimension $4n$.

$$v = [\ a \quad b \quad (a-b)^2 \quad a \odot b\ ] \tag{1}$$

This concatenation was used by [2] for natural language inference and then modified for a Quora duplicate question detector [9]. We feed this vector to a fully connected layer with 2 hidden units to output class scores. Finally, using these scores, we train our model with softmax cross-entropy loss.

3.3.2 Baseline Architecture

Figure 3 details the overall architecture used in a branch of our Siamese network. We designed our baseline intentionally to have few layers and large kernels, as a point of comparison to the later architectures which are deep with small kernels. The first convolutional layer uses 32 filters of size 10 x 10. The second convolutional layer uses 64 filters of size 8 x 8. The final convolutional layer uses 64 filters of size 4 x 4. Fully connected layer 1 uses 400 hidden units and fully connected layer 2 uses 200 hidden units. Both are followed by a Relu non-linearity. We apply l2 regularization and dropout with a probability of 0.5.

3.3.3 Advanced Architectures

Three other architectures that were trained to encode images were VGG13[8], GoogLeNet[10], and Residual Networks (ResNet) [4]. These networks are all recent and well-performing networks designed for the ImageNet challenge [7]. Each network employs a novel architecture of convolutional neural networks.

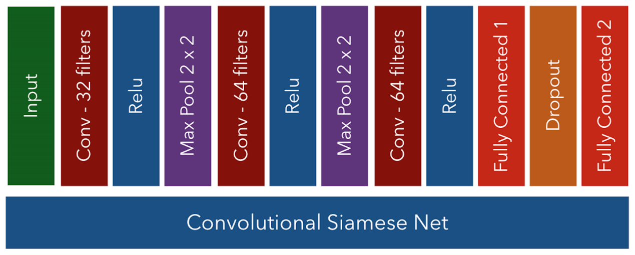


Figure 3. Baseline network architecture used to evaluate words and lines

VGG-13 is a modification of the well known VGG-16 network created by Oxford's Visual Geometry Group[8]. We use VGG-13 instead of VGG-16 due to the memory constraints. Although VGG performs well on the ImageNet challenge, we found that the network was hard to train and ultimately performed poorly as the encoder.

In this study, we trained the 22 layer GoogLeNet to encode our images from scratch. [10] The main advance that GoogLeNet provides over VGGNet is the introduction of efficient Inception modules. These modules concatenate many convolutions with different filter sizes together. Overall, these modules allow the model to only consist of 5 million parameters compared to over 138 million parameters for VGGNet.

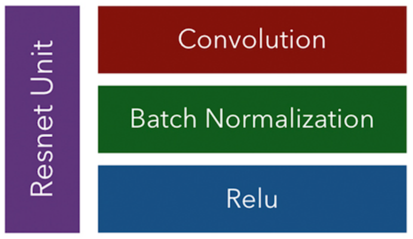


Figure 4. A unit of ResNet consists of a convolution layer followed by a batch normalization step and then the Relu activation function.

Our best performing model was the ResNet. The ResNet architecture distinguished itself from previous architectures by being able to train increasingly deep networks without degradation of performance. This is achieved by simply adding periodic identity shortcut connections to the network. We follow the architecture mentioned in section 4.2 of [4] [1]. A single ResNet unit is described in 4, and the architecture is summarized in 4. The architecture begins with a single ResNet unit with 16 filters with stride 1 and followed by a stack of $2n$ ResNet units with filter size 16, $2n$ units of filter size 32 with stride 2, and $2n$ units of size 64 with stride 2, where n is a pre-specified hyperparameter. Finally, there is a fully-connected layer which brings the output to 10 dimensions, for a total of 6n + 2 layers. Al-

[1] Code adapted from https://github.com/ry/tensorflow-resnet

though ResNet was designed for deeper networks, our preliminary experiments did not show a significant improvement with deeper configurations, so for the sake of speed we kept $n = 1$, yielding an 8 layer net. We dub this $n = 1$ network TinyResNet.

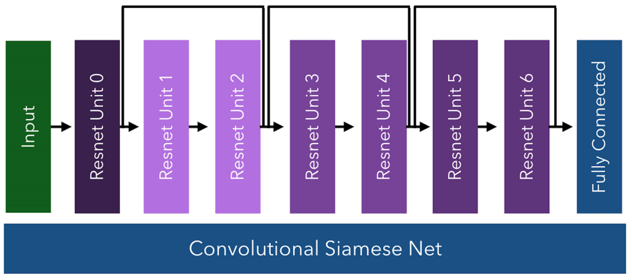


Figure 5. Overall architecture of our ResNet model.

3.3.4 Ensembling

Using the best performing model architecture, we built a framework to ensemble multiple models together. In our case, we train 5 TinyResNet models on 10,000 datapoints with random initializations. The softmax classifier for each model outputs probabilities: t for the probability that the authors of each sample are a match and f for the probability that the authors are a mismatch. We note that $t + f = 1$. Thus for each sample in the validation and test set, we output 5 f and 5 t probabilities. We select each final probability over all 5 different models, a, such that the difference between the t and f is maximized.

$$argmax_a |t_a - f_a| \tag{2}$$

Intuitively, we treat the softmax output as confidence scores to determine whether or not the handwritten pieces are the same, and select the classification that maximizes the confidence in the score.

4. Experiments

To begin, we wanted to determine how much data within each sample was necessary to produce decent results. Ideally, we would have been able to compare single words against lines of words against pages of words, but ultimately, we were limited by the amount of memory available. Table 1 details the performance of using different inputs, single words versus lines in the baseline model. Overall we note that lines, even those that have been slightly trimmed, performed better than single words on the baseline model. Lines achieved a 72.40% accuracy compared to words which merely achieved a 60.25% accuracy. This insight leads us to believe that not enough signal and information is carried in a single word for the model to make a strong and accurate prediction. Thus, in the future, when more memory is available, it would be interesting to explore pages as input to the network. Because of this experiment, we continued training our models solely using lines as the input feature.

Model	Data	N Train	N Val	Val Accuracy
Baseline	Words	10000	1000	60.25%
Baseline	Lines	10000	1000	72.40%

Table 1. Baseline architecture results comparing words with lines

Next, having determined that lines are the ideal input to the model, we test different architectures to determine which model is most successful. Table 2 demonstrates each network's performance as an encoder is measured by its accuracy. The baseline model and VGG13 perform significantly worse than GoogLeNet and ResNet. Ultimately, we believe that the baseline model was not complex enough to represent the handwritting. We also note that VGG was difficult to train from scratch using the data provided. We attribute that to the extremely deep nature of VGGNet paired with the extremely high number of parameters in the model.

GoogLeNet performs similarly to ResNet, achieving 88.18% compared to 91.97%. As stated earlier, we did not notice a significant difference in accuracy when extending the Resnet to be deeper.

Model	N Train	N Val	Val Accuracy
Baseline	10000	1000	72.40%
VGG13	10000	1000	60.25%
GoogLeNet	10000	1000	88.18%
TinyResNet	10000	1000	91.97%
Resnet3	10000	1000	89.3%%

Table 2. Single model architecture results using lines as input

Having trained our best model at 10,000 training examples, we examine the effect of training size on the accuracy of the classifier in table 3. We namely do this in order to better understand the trade-offs between accuracy and training data size. In our final model, the ensembled model, we wanted to ensure that we were using a training set size good enough to produce very strong results, yet would not take significantly long to train. Interestingly, we note that even TinyResNet with just 500 samples for training outperforms our baseline model with 10000 data-points, indicating the important of architecture. As the number of data-points increased, we noticed an increase in validation accuracy as expected. We note that there was a trend of diminishing returns on accuracy based on the amount of data provided. Ideally, we would be able to explore larger datasets, however, we were not able to due to memory constraints.

Finally, having determined our best model TinyResNet with $[a, b, (a-b)^2, a \odot b]$ as the concatenation, we trained multiple models using this configuration from scratch and

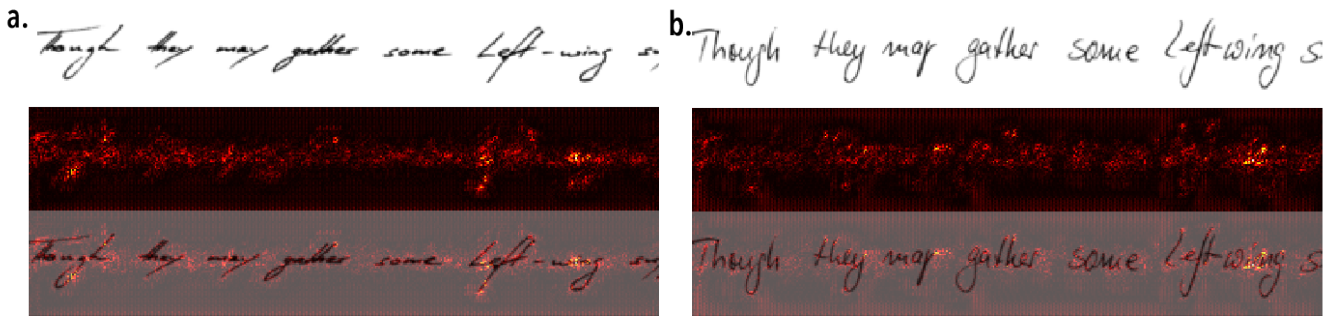


Figure 6. Saliency maps for a sample of text written by two different people.

Model	N Train	N Val	Val Acc	Test Acc
TinyResNet	500	1000	79.27%	-
TinyResNet	1000	1000	82.29%	-
TinyResNet	5000	1000	85.42%	-
TinyResNet	10000	1000	91.97%	-
TinyResNet	25000	1000	93.75%	92.08%

Table 3. TinyResNet results varying the training size

with random initialization. The ensemble of 5 of these models, trained on 10000 samples, ensembled as previously described, produced a validation accuracy similar to the best TinyResNet model trained with 10000 samples. The test accuracy also was similar to the validation accuracy. We note that in this case ensembling did not lead to a significant increase in accuracy against the best single model used in the ensemble, but overall it increased the accuracy against the many other single models used in its own construction. And in fact, using a larger dataset on the single model produced slightly better results.

Model	N Test	N Val	Val Acc	Test Acc
TinyResNet	1000	1000	89.79%	-
TinyResNet	1000	1000	88.22%	-
TinyResNet	1000	1000	86.04%	-
TinyResNet	1000	1000	88.44%	-
TinyResNet	1000	1000	91.97%	-
Ensemble	1000	1000	91.46%	91.875%

Table 4. Ensembling multiple runs of the best performing model. Each single model was trained on the same dataset (n=10000) and was tested on the same validation and test sets.

We believe that the ensembled models ultimately did not perform significantly better than our best single model because each of the models are trained on the same dataset, and ultimately, each models' weights converge in a similar optima. Because of this, all of the classifications outputted by each single model are ultimately extremely similar. Thus, ensembling similar models does not lead to a significant performance boost.

4.1. Saliency Maps

In order to analyze how our model captures writing style, we analyze the saliency maps of a line of text written by two different individuals, shown in Figure 6. Unsurprisingly, the maps are active in areas corresponding to where the text is, but many further factors seem to be at play. We offer some interpretations below.

In the left example, we see importance placed on the portions of text that go below the line of the text, particularly the bottom portion of the first 'g' and 'f'. These also seem to be the most noticeably unique letters in the text (notice that the smaller, second 'g' garners almost no attention). Letters that were tightly packed didn't seem to gather much attention ('Th' in 'Though', as well as 'gather'). In the right example, we see an example of activations in negative space, after the 'y' in 'may'. The model may have expected a continuation of the letter (e.g. into a 'p'). Furthermore, more attention seems to be paid at the tips of the letters in the right example, suggesting that perhaps the way in which people end (or begin) their strokes may be indicative of style. In both cases, particular attention seem to be paid on the 'in' in 'wing'. Perhaps this is notable because both writers manage to keep each of the letters distinct, rather than slurring them together. [2]

4.2. Filter Visualization

In order to understand the network and the features it extracts, we visualize the filters produced by the model. Thus, we examined the 16 filters produced the TinyResNet in the first convolutional layer in figure 7. Although these filters are somewhat noisy, in some we see general curvatures emerging. We note that these filters were extracted from a well-trained model which had very good validation performance as well, thus the noisy pattern is not a indicative of the training time or the regularization strength of the network. Finally, although these filters do not look as clean as the ones generally produced by networks such as AlexNet on regular pictures, this may indicate that the filters/weights

[2] Our saliency map code was adapted from CS231N homework assignment 3.

required for handwriting verification are significantly different than those of general image classification tasks like ImageNet.

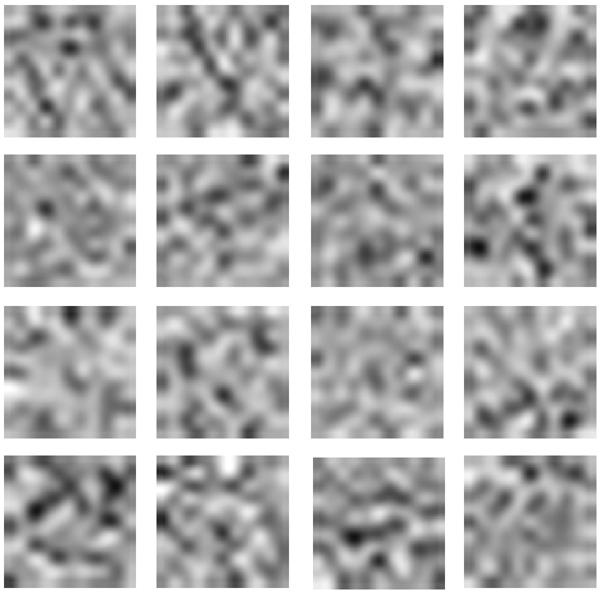
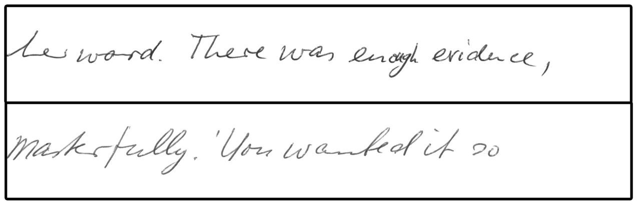


Figure 7. Visualization of the filters from the first convolutional layer of the TinyResNet network

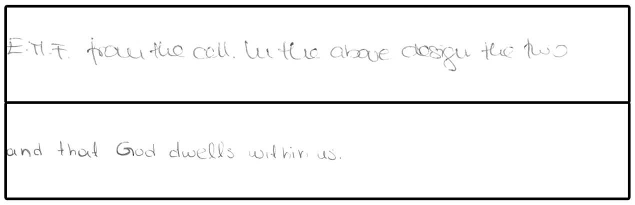
4.3. Error Analysis

4.3.1 Qualitative


he word. There was enough evidence,

masterfully. 'You wanted it so

Figure 8. Representative example of a false negative.


E.T.F. from the call. In the above design the two

and that God dwells within us.

Figure 9. Representative example of a false positive.

Figures 8 and 10 show representative examples of a false negative classification and a false positive classification, respectively, by the TinyResNet model. In the false negative case, the bottom example contains handwriting that is noticeably more sloppy than the first. The *an* in *wanted* is reduced to a squiggle, and the first character of the last word *so* is illegible. This perhaps suggests that the writer was getting tired or feeling rushed, which caused their writing to deteriorate in style. If the convolutional filters in the model were sensitive to kerning (spacing between letters), then it is logical that the two samples would have been attributed to different authors.

In the false positive case, it is difficult to tell at first glance that the two samples are actually by different authors. They share many stylistic similarities that the convolutional layers would have been able to detect: both have curvy, loopy letters (see the *g*s and *n*s of the top sample and the *l*s of the bottom sample), both are written lightly (see the false negative examples as a comparison), and both take up approximately the same height in the text sample.

4.3.2 Quantitative

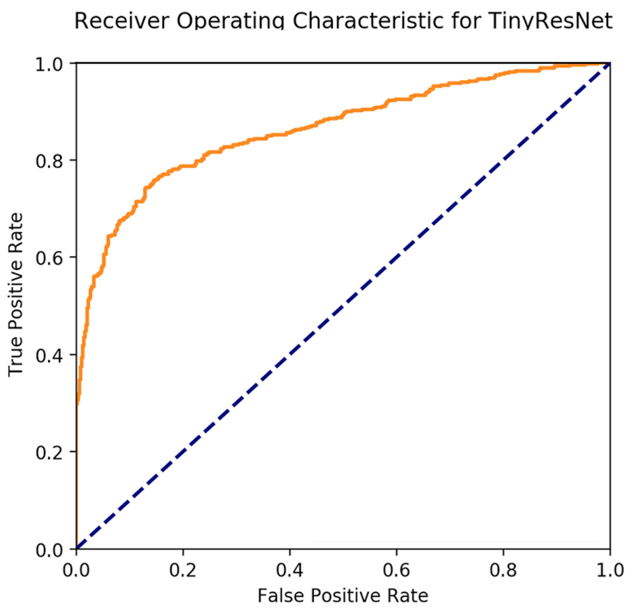


Figure 10. ROC Curve diagrams the relationship between specificity and sensitivity of our model. We have slightly lower sensitivity than specificity.

Our model attempts to assign a large distance between dissimilar examples and a small distance between similar examples. By thresholding this score to 0 or 1 using softmax, we can obtain the accuracy and confusion matrix of our model. Furthermore, this distance metric allows us to plot an ROC curve on our test set, since it represents how confident the model is about the prediction. One metric we wanted to evaluate is the false positive versus false negative rate. We found that in our best model, the number of false positives was almost twice the number of false negatives in the test set. This means that if our model makes a mistake, it is more likely because it is labeling two samples by different authors as a match. Given this, it would be interesting to see how well the model can detect forgeries. It is also important to keep in mind that the raw values for false positives and false negatives are very small relative to the true positives and true negatives. Thus, our sensitivity (92.9%) and specificity (89.2%) are actually quite close. Whether a pair of inputs is by the same author or by different authors should not drastically affect the predictive capability of our model.

5. Conclusion

In this study, we examined Siamese convolutional neural network architectures to verify authorship of handwritten text. We first examined the length of text required to make an accurate prediction and discovered that lines of words performed much better than single words. Semantically, this allowed us to determine how much is writing is necessary to create an accurate writer verification system. In this case, each line contains from 5 to 10 words on average. In the future, the next logical step will be to expand use the complete images of pages rather than single lines. This has been dependent on available memory on our Google Cloud machine.

Next, we examined different architectures to determine which architecture provided the most informative encoding that was able to differentiate between authors. Beyond our baseline model, we explore VGGNet, GoogLeNet and different sizes of ResNet. The best performing model was a shortened version of ResNet which we coined TinyResNet. A single TinyResNet model trained on 25,000 samples was able to achieve 92.08% accuracy on a held out test set. Ensembling of 5 TinyResNet models trained on 10,000 datapoints each did not significantly improve the model. Thus, in the future, it would be interesting to continue training on even larger datasets not limited by memory restrictions.

Visualizing the saliency maps for the Siamese network trained with TinyResNet shows that the networks unsurprisingly examines the handwritten text in each image. Strokes that go below or above the line of text typically have high importance. We note that the end and beginning of strokes typically have high importance as well.

Visualizing the filters for the network ultimately produces somewhat noisy filters despite the network being well-trained. These filters do show some semblance of curvature. However, perhaps most importantly, these filters which are trained from scratch perform extremely well on the given task. These filters look extremely different from those typically trained on image classification tasks like ImageNet. This could indicate that the type of filter necessary to achieve good results on images of handwriting are different than those for image classification. This means that pretrained weights would theoretically perform poorly on this task. An interesting exploration in the future would be to use pretrained weights from ImageNet on this task in order to validate our hypothesis.

Future improvements of this model could incorporate the word embeddings for each of the words included in the image as well as mathematical representations of vectorized handwriting.

Finally, one interesting tangential extension of our project would be to apply Generative Adversarial Nets (GANs) to construct counterfeit pieces of writing. The goal of this sub-problem would be to train a network that would be able to generate handwriting similar to a given input. Ideally, the goal will be: given a piece of handwriting, can we generate a counterfeit handwritten piece? However, we first will need to confirm that a GAN on handwriting generates handwriting with a proper alphabet rather than random lines.

Thus, in this paper, we have shown that using Siamese convolutional neural networks are able to perform well on the handwriting verification task. This network could have large impacts on forensic analysis, historical text verification, and signature verification.

References

[1] Centre for the study of the renaissance. *LIMA: Handwriting: Forgeries*, Jul 2005.

[2] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts. A fast unified model for parsing and sentence understanding. *CoRR*, abs/1603.06021, 2016.

[3] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *In NIPS Proc*, 1994.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] A. Liptak. Prosecutors hope new study of handwriting analysis will silence skeptics. *The New York Times*, May 2002.

[6] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[9] S. Sy, Y. Homma, and C. Ye. Detecting duplicate questions with deep learning. 2016.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.

Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[11] Y. Tang and X. Wu. Text-independent writer identification via cnn features and joint bayesian. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[12] W. Yang, L. Jin, and M. Liu. Deepwriterid: An end-to-end online text-independent writer identification system. *IEEE Intelligent Systems*, 31(2):4553, 2016.