HANDWRITTEN AUTHORSHIP ATTRIBUTION WITH IMAGE RECOGNITION AND NATURAL LANGUAGE PROCESSING

Brionna Slaughter
Rose-Hulman Institute of Technology
Email: slaughb@rose-hulman.edu

ABSTRACT

Being able to attribute authorship of a short, handwritten note could expand the possibilities of using other information besides textual content to achieve a high accuracy on an authorship attribution problem. Generally, high accuracies are achieved by using a large dataset, but this project seeks to utilize other information instead of relying on more data. When looking at the potential impact of this authorship attribution software, it is important to note that in modern times, handwritten notes are often concise, thus this project focuses on maintaining accuracy for a small amount of short notes for each author. For the neural networks trained in this project, a Recurrent Neural Network was used as our main tool for processing the text, and a Convolutional Neural network utilizing ImageNet for transfer learning was used to analyze the corresponding handwriting. In order to test the results of our program, a variation of the IAM Online Handwriting Database was used, resulting in a final average accuracy of 74.17%.

1. INTRODUCTION

We describe and evaluate a system designed to classify the author of a note when not only the length of the note is short, but the amount of notes available for that author are limited. We developed a system that evaluates authorship based on the textual information from the content of the note as well as the visual information from the handwriting itself. To accomplish this, we use image recognition and natural language processing techniques. Some of these techniques include N-grams, style markers, frequency analysis, feature extraction, and neural network classification of features and contents.

Solving authorship attribution of a handwritten note is valuable in settling disputes that range from academic dishonesty to legal matters. For instance, when high school students take the SAT and ACT standardized tests, they must complete a handwritten essay. There have been cases in which a student was able to submit someone else's work. Our system could be used for the handwritten portions of these exams, to detect if there were sufficient changes in handwriting or style to justify an inquiry. At the very least, the knowledge of such a system would make it more difficult for a person to submit plagiarized work. Most authorship attribution systems focus either on the content of the text [need citations] or for handwritten text, features of the handwriting style. Our system combines both of those approaches to take maximum advantage of the information contained in both. An additional benefit of the combined approach is that this increases cases in which our system can be successfully applied.

Identifying the author of a piece of writing can be a difficult problem to solve due to the fact that people may have similar writing and handwriting styles. Additionally, the risk of a high false positive and negative rate is especially high for short notes due to the fact that there is overall less information to extract.

When examining research that has been done on the subject of authorship attribution, the most common approaches investigated semantic and syntactic clues left by an author. These systems resulted in fairly high accuracies by using only the text itself. In several instances, N-grams, syntactical style, grammatical structure, and combinations thereof were used as style markers for authorship attribution with highly accurate results [1][7][8]. Schwartz et al. [8] combined character N-grams, word N-grams, and flexible patterns, achieving the highest accuracy.

To extract visual information from the handwriting of the author, we utilize work on forgery detection and signature verification. In one study, geometric feature extraction for images of signatures resulted in sufficient data for a neural network to be used for the classification, and it was able to reach a 77.1% accuracy on its final test set [3]. Similarly, in another study, features such as the ratios between letter and word sizes, slant feature, and detection of taller letters were used to reasonably determine the forgery of a signature on a bank note [6]. While we use the same features as these previous studies in the extraction process of our system, our system differs in the way the information from the extracted features is then utilized. To explain further, in forgery detection,

systems perform verification steps such as determining if the taller letters match with the assumed spelling of a person's name [6]. Our work, on the other hand, will be focused entirely on the way each tall letter was written. We compare how a certain letter from one person was written as compared to the same letter written by other people. As a result, although this feature extraction is similar to that of forgery detection or signature verification, its overall functionality serves a different purpose.

While authorship attribution is an explored topic, one of the key elements to these systems' accuracies is their large dataset. In particular, when comparing the results from tests performed with texts that have a very short length compared to those that were of a larger size, the accuracy of authorship attribution was noticeably lower. In one study, the authors attempted to attribute different online posted messages with 140 characters or less, which resulted in a 72.4% accuracy with a 500 message dataset with 50 authors [7]. Although this was indeed a small dataset, we will show that we can maintain reasonable accuracy while having more authors with fewer messages per author within the dataset.

2. ARCHITECTURE

Our system uses a combination of semantic, i.e. contents based evaluation, and image recognition techniques commonly found in forgery detection. At this point, our system is modular and executes each task individually, with no interaction, as shown in Figure 1. The result of the processing will be combined by weighing the outputs of these four processing pipelines.

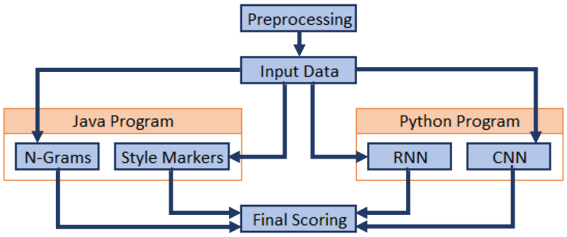


Fig. 1. Broad overview of the architectural setup

Our system makes use of two separate neural networks: a Recurrent Neural Network (RNN) and a Convolutional Neural Network (CNN). The RNN is used for the content based authorship attribution. RNNs are a form of neural network that allow the network to model a temporal sequence. In order to do this, RNNs are created from a form of feedforward network. Their overall structure forms nodes such that it could be modelled as a directed graph. They are commonly used in speech and language recognition due to their ability to process variable length sequences of inputs. On the other hand, the CNN is used to identify features of the handwriting style. Convolutional Neural Nets (CNNs) are a type of deep neural network, and they are commonly used for analyzing images. The overall architecture of a CNN consists of an input layer, output layer, and hidden layers. The hidden layers are a combination of rectified linear unit (ReLU), pooling, fully connected, and normalization layers. The ReLU layers found in a CNN make use of an activation function that only does comparisons, additions, and multiplication. Pooling layers are used to condense the information in the hidden layers down so that the end result can be a single output layer. The fully connected layers found at the end of the CNN are generally responsible for learning the proper weights. Furthermore, these fully connected layers are able to be retrained without retraining the whole CNN in a process called transfer learning.

2.1. Semantic Evaluation

2.1.1. Textual Data Preprocessing

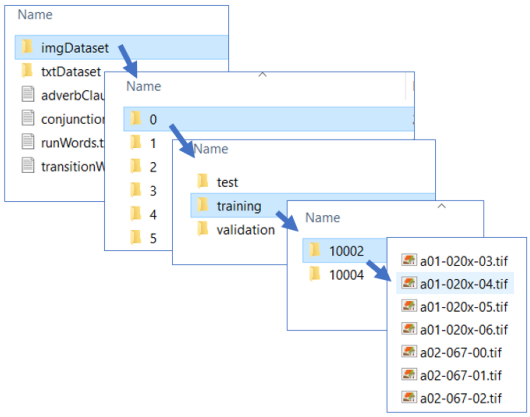


Fig. 2. Visual representation of file structure for datasets

In order to repurpose the IAM Online Handwriting Database to function as a data set for an authorship attribution classification problem, the dataset had to first be re-ordered by author. The original structure of the IAM Online Handwriting Database filed each note under the lexographical location of the note's contents in the LOB Corpus. Additionally, the ASCII transcriptions provided by the database did not include the handwritten author's identification. Instead, this information was available in a separate XML file [9]. Since the core of this project is authorship attribution, it was necessary to re-order the notes such that they were ordered by the author instead of by lexographic location. To remedy this inconsistency, we created a script separate from the main project that cross matched the author identification number with each

individual note name. Then, the script separated the ASCII notes into folders, each with a training, validation, and testing dataset consisting of two authors that were manually matched up to be an author-forger pair. These pairs can be referenced in Appendix A-1. The system, within these separate datasets, then labelled the author and forger notes per the information available in the XML file. The basic file structure followed for this project is shown in Fig. 2.

Furthermore, not all of the authors within the dataset had an equal number of notes. We were aiming to have an average of under 10 notes available for each author, however, some authors only had 2 or 3 notes each. These authors were removed from the dataset since they did not have enough data to be paired with any other author. A total of 9 different authors were removed from the dataset for this reason, resulting in a dataset of 212 authors instead of the former 221.

2.1.2. N-grams

N-grams are a commonly used authorship attribution technique. However, since the notes found in our dataset are specifically short notes, we decided to utilize not only a word-based N-gram with a length of two, but also one with a length of three. These are also known as bigrams and trigrams, respectively. In order to capture the bigrams and trigrams, the program stored the N-gram as a key in a Hash Map. For every instance of a particular N-gram, the following word in the sentence was added to the N-gram's associated words. The associated words each have a frequency value associated with them, which is the amount of times that particular N-gram appeared for that author. So, if the associated word is already in the Hash Map, then the program will simply increment the frequency by one. One example of this is shown in Fig. 3. below, in which the bigrams and trigrams can be seen for the sentence "I like to code."

Sentences: "I like to code."	
Bigrams:	**Trigrams:**
[I, like] => [(to, 1)]	[I, like, to] => [(code,1)]
[like, to]=> [(code, 1)]	

Fig. 3. Visual representation of the storage of bigrams and trigrams

After the bigrams and trigrams are stored, the most common word to follow each N-gram becomes clear for each potential author. Therefore, we are able to compare the N-gram results for each unclassified note to the categorical distributions of previous texts written by the author. In turn, this allows us to accurately evaluate the probability of whether or not a given person wrote the handwritten note based on the text itself. In order to do this, we decided to compute the Jaccard Similarity Coefficient for the note in question compared to each author. The equation used for this is shown in Fig. 4. Once the coefficients are calculated for each author, the top five highest scores are passed along to final scoring.

$$M = Number\ of\ correctly\ matching\ Ngrams$$
$$I = Number\ of\ matching\ keys\ that\ do\ not\ match\ associated\ words$$
$$Result = \frac{M}{M + I}$$

Fig. 4. Variation of the Jaccard Similarity Coefficient equation used for N-grams

2.1.3. Style markers

Style markers are things such as an author's use of punctuation and their word choice. The grammatical and syntactical style that a person uses when writing typically does not vary much. Our system records style markers from the author's prior written work.

0	Used a period
1	Used a semicolon
2	Used a forward slash
3	Used a backward slash
4	Used a parentheses
5	Used brackets
6	Used curly brackets
7	Used an exclamation point
8	Used a question mark
9	Used an ampersand
10	Used a comma for an adverb clause
11	Used a comma for a conjunction
12	Used a comma for a transition
13	Used a comma for a series
14	Used a comma for unidentified reason
15	Used an apostrophe
16	Used an @ sign
17	Used a pound sign
18	Used a dollar sign
19	Used a percentage sign

Fig. 5. The vector created when identifying style markers

Grammar tokens, such as commas, can be used in a variety of ways. However, most people tend to use them in only a few ways that fit with their style of writing. Our system records how many of each type of punctuation are used, and then evaluates its basic function within a given sentence. For instance,

commas are marked as used for introductory clauses, serial items, sentence conjunction, etc. All of the different grammatical style markers that we chose to identify are listed above in Fig. 5.

While the vector of covered style markers does not necessarily address every particular writing style, it does allow for the differentiation between comma usage, which allows for the software to recognize which usages the author is the most comfortable with using. In order to differentiate between these usages, the words at the beginning and end of each clause were analyzed to see which usage was probablistically the most likely. For example, when looking at the sentence, "I rode my bike down the street, and then I fell," one can see that a conjunction is used. In order to classify this in the software, the program would look at the words "I", "street", "and", and "fell" in order to determine the use of this comma. In this case, the word "and" is directly succeeding the comma and is a conjunction, which makes it highly likely that the comma is functioning as a conjunction. Our system is primarily able to identify how the words surrounding the comma function by utilizing the Stanford Parser and by comparing the words to common transition words, conjunctions, and introductory phrases. Furthermore, the program also analyzes how far between words the commas appear. If the sentence, "I ate an apple, banana, and strawberry," appeared, then the program would recognize that there is only one word between the two commas, which means that the comma is likely used in a series, even though the Stanford Parser would mark the "and" that is directly following the second comma as a conjunction word. With these special deliniations for the comma, it allows us to capture more of the nuances in each author's style.

Finally, once the authors have all been analyzed for their style markers, the note in question will be compared to the normalized array in order to compute a variation of the Jaccard Similarity Coefficient. The equation used to compute this value is shown in Fig. 6. Then, in a similar fashion to the N-grams, the program reports the top five matches.

$$S = Number\ of\ correctly\ matching\ style\ marker\ ratios\ within\ a\ \pm 0.1\ range$$

$$N = Number\ of\ style\ marker\ ratios\ that\ do\ not\ match\ within\ \pm 0.1\ and\ are\ not\ 0$$

$$Result = \frac{S}{N + S}$$

Fig. 6. Variation of the Jaccard Similarity Coefficient equation used for Style Markers

2.1.4. Recurrent Neural Network Classification

Unlike the N-gram classification and the Style Marker classification portions of the project, we decided to train the RNN only on two authors. The idea behind this was that the note in question would either be real or forged. Therefore, it made sense for the classification problem for the RNN to be simplified between "this author truly wrote this note" and "this note was forged". In order to accomplish this, we utilized an Tensorflow for Python in order to create a RNN. With this framework, each of the individual layers could be fully customized, along with the number of epochs. Furthermore, this framework provided tools that could create a vocabulary from the data, tokenize the text, and prepare the data for entry into the RNN. Therefore, the only thing that was needed for the RNN to function was for the text files containing the notes to be read into the program. Once these files were read in and tokenized, the text passed through 2 dense convolutional layers, a maxpooling layer, and 2 additional dense layers for the RNN. The exact structure of the RNN is shown in Fig. 7.

Layer (type)	Output Shape	Param #
input_61 (InputLayer)	[(None, None)]	0
embedding_58 (Embedding)	(None, None, 128)	2560000
dropout_119 (Dropout)	(None, None, 128)	0
conv1d_116 (Conv1D)	(None, None, 128)	114816
conv1d_117 (Conv1D)	(None, None, 128)	114816
global_max_pooling1d_58 (Glo	(None, 128)	0
dense_65 (Dense)	(None, 128)	16512
dropout_120 (Dropout)	(None, 128)	0
predictions (Dense)	(None, 1)	129

Total params: 2,806,273
Trainable params: 2,806,273
Non-trainable params: 0

Fig. 7. Structure of the RNN used for semantic analysis

In order to optimize the hyperparameters, the RNN was ran for 1000 epochs, with the training and validation test sets being tested after each epoch. Upon the discovery that the accuracy of the test set severely depleted with this many epochs, we concluded that the RNN had drastically overfit the data. So, to optimize the amount of epochs, we subtracted 100 epochs for each run and tested to see if the data was overfit. However, this still was resulting in too high of an accuracy. We concluded that due to the very limited amount of the text files for each author, the amount of epochs in which the RNN was run would be much smaller. With this information, we chose to train our RNN each time with 20 epochs and a batch size of 32, saving the neural network that performed the best on

the validation dataset. Then, the test set was run on the saved neural network in order to obtain the predicted labels, and by extent, the accuracies.

2.2. Image Recognition

2.2.1. Preprocessing

As previously mentioned in the preprocessing section for the semantic evaluation, the IAM Online Handwriting Database is consistently ordered such that each note is found under its lexographical location [9]. Likewise to the semantic side, the line image dataset needed to be re-ordered by author. A similar script was created to modify the dataset for line images, in which each line image was cross matched with the author identification number in the XML file. However, since there are multiple line images per note, the IAM Online Handwriting Database chose to differentiate the multiple pictures by adding an extra character following the base file name. For example, a file named 'exampleFile_1.png' would become "exampleFile_1a.png" and "exampleFile_1b.png" in the line image database. In order to fix this issue, a substring was taken of the file name to compare to the XML details, but when the image was copied over to the final dataset, the original file name was preserved. By doing this for each author, all of the line images were able to be sorted into folders labelled with each author's unique identification number.

2.2.2. Convolutional Neural Network Classification

Similar to the RNN for the semantic evaluation, the CNN was written such that it took in two separate authors instead of all of the authors. The CNN was also created in Python using the Tensorflow extension. This allowed us to easily utilize transfer learning, which we hypothesized would be more effective than training from scratch since we had a small dataset. In order to implement transfer learning, we removed the final three layers from ImageNet, and retrained it to fit handwriting instead.

To train the CNN properly, we had a dataset with 13,110 images of 1,910 handwritten notes. This is clearly a lot more images than there were text files, and that is due to the images of the handwritten notes being split into multiple images. This way, each image only contained a few words as opposed to the whole paragraph. This was preferable because it allowed for the neural network to focus on spacing between letters, words, and also on the overall slant of the writing. Additionally, having more images of the same note allowed for the system to have more data to pass into the CNN. Splitting the images by line allowed for each author to have approximately 60 images instead of only 8 or 9, which, in turn, allowed for the neural network to achieve a greater accuracy.

In the same way as the RNN, the hyperparameters were optimized manually, starting with 1000 epochs. For the CNN, the number of epochs at which the neural network stopped overfitting was 500 epochs. The system saved off the network with the highest accuracy for the validation set at this point, which was then used for testing.

2.3. Final Scoring

2.3.1. Final Weighting

The final crucial component of our system was the weighting function. Throughout our system, each component independently reports its scores without talking to any other component. As a result, the weighting function is the final element that determines which of the components the system listens to. After performing numerous experiments, it was discovered that the neural networks performed both the best and the worst of all the components. In other words, the RNN and the CNN were less consistent than the N-gram and style marker analysis. In order to mitigate this variability, the final score of these components were weighted less than the other components. The weighting score equation that was used is shown in Fig. 8.

$$n = Ngram\ accuracy$$
$$s = Style\ Marker\ accuracy$$
$$r = RNN\ accuracy$$
$$c = CNN\ accuracy$$

$$Accuracy_{Final} = (n * 0.30) + (s * 0.30) + (r * 0.20) + (c * 0.20)$$

Fig. 8. Average weight equation

2.3.2. Interpreting the Final Score

Based on the notion that two human beings may have similar handwriting styles and lingual patterns, especially in the case where there is an expert forgery, this system will be built for the scenario where the true author is indistinguishable from a different author. In order to handle this edge case, the scoring for this program will not simply return a binary answer of "yes" or "no" in regards to whether or not a particular person likely wrote it. Instead, the system will return a score between -100 and 100, where the negative numbers indicate a "no", and the positive numbers indicate a "yes". A final score that is further away from zero indicates that the system is more confident about its answer.

3. RESULTS

3.1. Results

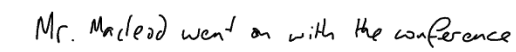
The data set that we used for this project had a total of 1,910 notes written by 221 different authors, which equals approximately 8 or 9 notes per author. As was previously mentioned, all of the notes and author information was extracted from the IAM Online Handwriting Database [9]. For the semantic evaluation, this database was modified as described in section 2.1.1. The image evaluation, on the other hand, modified the dataset as detailed in section 2.2.1.

Accuracy			
	Minimum	**Maximum**	**Average**
N-grams	60.59%	78.92%	73.46%
Style Markers	70.35%	84.93%	79.55%
RNN	54.55%	90.91%	71.44%
CNN	54.81%	87.47%	69.92%
FINAL	**63.87%**	**81.16%**	**74.17%**

Fig. 9. The average accuracies of each component and the total accuracy for the test set

For the final results, the authors that were compared were chosen manually from the dataset. Each author in the dataset was compared to a different author that was manually chosen with similar handwriting. For instance, in Fig. 10., one can see that the two authors had a fairly similar slant and spacing in their handwriting. As a result, author 10004 was matched as the "forger" for "author" 10002. The full list of matched authors and forgers can be found in Appendix A-1. Once each author was run against an author with a similar handwriting, all of the accuracies of each component were averaged and are reported in Fig. 9. While the average accuracies for each of the components seem to be in an acceptable range, it is important to note that the reliability of the components was a major weakness for this project. For instance, the minimum accuracy score for the RNN was 54.55% and the maximum accuracy score was 90.91%. This sort of difference is unacceptable for a component that is relied upon for the main decision making abilities of the classification system. In order to remedy this issue, we decided to weigh the results given by the RNN and the CNN much lower than the N-grams and style markers, which had both been proven to have more consistent scoring.

(a)



(b)

Fig. 10. A handwritten note image from two separate authors (a) 10002 (b) 10004

In testing, we discovered that author 10002 and author 10004 were a difficult case for the system. In Fig. 11., one can see that the scores for these two writings in particular were quite low compared to the overall averages. The accuracies for every other author pair can be found in Appendix A-2, for reference. This was the most unexpected result that we saw from the system, since these two handwritings were not as similar as some of the others present in the dataset. We believe that some of this error may be attributed to the textual content that these particular notes contained. For these authors specifically, the majority of the notes contained more than one proper noun, and each proper noun was rarely repeated between different short notes. By having these proper nouns in the short notes, the probability of the system matching N-grams or being able to pick up on patterns with the RNN become much slimmer.

Accuracy – 10002 vs. 10004	
N-grams	60.59%
Style Markers	74.73%
RNN	59.47%
CNN	56.87%
FINAL	**63.87%**

Fig. 11. The relevant accuracies when classifying notes between authors 10002 and 10004

4. DISCUSSION

4.1. General Discussion

In order to keep this project within a reasonable scope, we decided to limit the classification problem to a closed domain. In other words, we decided that any note that was tested in the system needed to belong to an author within the dataset. In an ideal world, the program would be able to supply the information that the true author is not in the dataset if given a note outside the system, however, this scope was much too large. Instead, in order to narrow the scope, this

program was made to identify whether or not a note was written by a particular person instead of on an open domain.

Overall, this handwriting authorship attribution software was a success within its scope. In our final test set, we were able to correctly classify 74.17% of notes, which met our expectation of greater than or equal to a 72.4% accuracy. Although not every single classification received a high confidence score by the system, with the constraints on the dataset size, the program was a success.

It is important to note that the individual components that this system was comprised of were not able to achieve the individual accuracies found in literature. Thus, the overall score may be easily improved if each component was able to be made in the exact likeness of these other systems as was mentioned in the introduction of this paper.

4.2. Semantic Analysis Performance

Unlike the image recognition side of the software, the semantic analysis has three separate contributing components. When looking at these components individually, there is some question that should be mentioned as to whether or not the RNN is truly adding anything to the software on top of the N-grams. An RNN is known for its ability to handle temporal data, and it functions similarly to an N-gram, just on a broader scale. However, the focus of this project was on short notes specifically, and not all of the notes are complete, grammatically correct sentences. While this may be the case, we decided to keep the information from the RNN in the project in order to explore how it might provide a broader look at the note as a whole. However, it was revealed that the bigrams and trigrams generally maintained a similar accuracy to the RNN while remaining more predictable.The higher accuracies present for the RNN in some cases may have been due to the RNN focusing completely on the two notes in question instead of comparing to all authors in the dataset. The potential correlation between the N-gram analysis and the RNN performance is shown below in Fig. 12. While the two separate lines are not perfectly correlated, one does notice that the moments in which the N-grams performed at its lowest usually lined up with when the RNN also performed poorly.

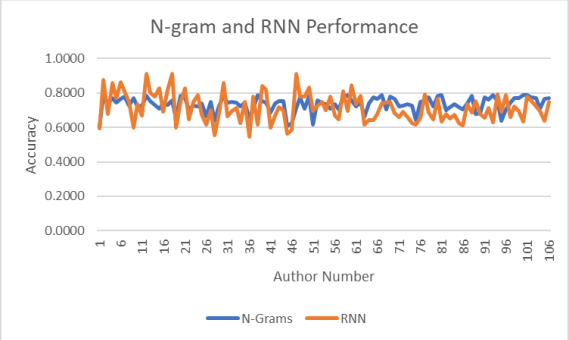


Fig. 12. A graphical representation of the accuracy of the N-gram analysis vs. the RNN for each author

While it is debatable whether or not the RNN was necessary for this system, the style markers, on the other hand, resulted in very high accuracies that added valuable information. As seen in Fig. 9, the overall average accuracy of the style markers tended to be higher, yet it also remained the most consistent component of the system with a differential of ±14%. This, in some ways, is not surprising, as the notes are quite short. The lack of word content restricts the ability for most traditional semantic analyses to work on the short notes. Style markers, on the other hand, are independent of the length of the note, and will commonly be used in familiar ways by an author. IN turn, this allows them to be a great comparison.

4.3. Image Analysis Performance

Examples of the handwritten note images from our training set that were misclassified are shown in Fig. 13. When looking at these, it may seem as if it would be easy for the program to determine the difference between the authors. However, the computer has to learn to ignore many of the features that humans would automatically eliminate. For instance, the system had to be trained to ignore color since all of the handwritten images were purely black and white. After this, the system then has to decide which features do, in fact, matter. In a CNN, the system is only able to look at a few pixels at a time within the ReLU layers, and then they are pooled together in the maxpool layers, thus preventing the machine from looking at the whole picture in the same way a human being would. Due to this, the computer is unable to easily achieve the same accuracy as a human being.

(a)

(b)

Fig. 13. Images incorrectly classified by the CNN
(a) 10039 (b) 10040

Despite these misclassification errors, the program was still able to accurately attribute the handwriting of a note for a large number of difficult images. Some examples of successfully classified images are shown in Fig. 14.

(a)

(b)

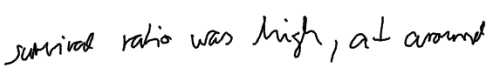


Fig. 14. Images successfully classified by the CNN
(a) 10042 (b) 10043

It may have been easier for the system to classify these images as apposed to the ones found in Fig. 13 due to the overall size of the writing. Since the CNN is able to more easily recognize overarching patterns instead of patterns within individual letters, the fact that writer 10042 generally wrote much larger than 10043 may have allowed the system to establish the difference between the forgery and the regular handwriting, even though the letters themselves were fairly similar. Despite not being able to classify each picture correctly each time, our CNN was able to provide useful information for the system as a whole.

5. CONCLUSION

In conclusion, the incorrectly classified notes are likely a result of the system needing to remain generalized. With the extremely small dataset, the risk of overfitting the data is much higher. Although each component of our system allows for more aspects of the data to be extracted, each component still functions individually within the system. Thus, the restriction of the small data directly continues to inhibit the ability for any individual accuracy to be high.

One of the most noteable barriers that we faced when developing this software was the lack of a dataset dedicated to handwritten authorship attribution. Since no previous work had been done on this problem, no dataset existed that would better represent the greater population of this problem. Due to time restrictions, a new dataset could not be created for this purpose. To rectify the issue, the IAM Online Handwriting database was modified to fit our purposes [9]. However, by modifying a dataset that was created for another problem, the handwritten portion and the textual semantics of the authors did not correlate in the same way that they would if a dataset were to be dedicated to this issue. Because of this, we believe that creating a dataset that is devoted to this particular problem would be one of the easiest ways to improve the accuracy of the system. While it may take a long time to develop a dataset like this, it would be greatly beneficial for the program as an accurate representation of the public.

Another improvement that could be made to the system is the comparison of particular letters. It would be difficult to develop a software that could recognize the key letters in a person's writing, however, how each individual author writes a letter is typically very distinct. By narrowing the scope down to the comparison of individual letters, the forged writing would have a difficult time overcoming the neural network. When the neural networks are looking at words at a time, they generally focus on things such as the slant of the letters or the spacing. But if the individual letters were to also be analyzed, the thickness of the pen strokes and the shakiness of the author's hand would also be known to the system. So, while this implementation would be difficult, it would add valuable information to the image recognition portion of this project, thus bringing up the overall accuracy.

In the long-term, this project could be expanded to become an open domain problem. As of right now, the program needs to know to train on both authors as a basic classification problems. However, if the software was able to be expanded to tell if the author is not in the dataset at all, then the uses for this type of software drastically increases. If it were to be successful on an open domain, other projects in the field of artificial intelligence would be able to better explore the idea of using multiple forms of the same information. This would drastically reduce the size of the dataset for any classification problem, and thus could make a significant difference in storage and computational runtime.

6. REFERENCES

[1] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[2] J. Patchala and R. Bhatnagar, "Authorship Attribution By Consensus Among Multiple Features," *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2766–2777, Aug. 2018.

[3] K. Huang and H. Yan, "Off-line signature verification based on geometric feature extraction and neural network classification," *Pattern Recognition*, vol. 30, no. 1, pp. 9–17, 1997.

[4] M. Kestemont, "Function Words in Authorship Attribution. From Black Magic to Theory?," *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, 2014.

[5] M. Koppel, J. Schler, S. Argamon, "Authorship attribution in the wild". *Language Resources and Evaluation*, vol. 45, no. 1, pp. 83-94. 2011

[6] Nagel and Rosenfeld, "Computer Detection of Freehand Forgeries," IEEE Transactions on Computers, vol. C-26, no. 9, pp. 895–905, 1977.

[7] P. Shrestha, S. Sierra, F. Gonzalez, M. Montes, P. Rosso, and T. Solorio, "Convolutional Neural Networks for Authorship Attribution of Short Texts," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017.

[8] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, "Authorship attribution of micromessages", *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1880–1891, October, 2013.

[9] U. Marti and H. Bunke. "The IAM-database: An English Sentence Database for Off-line Handwriting Recognition." *Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46,* 2002.

7. APPENDIX A

A-1: Mappings of Author to Forger

Group #	Writer	Forger
0	10002	10004
1	10005	10007
2	10006	10009
3	10008	10010
4	10011	10017
5	10012	10018
6	10014	10015
7	10019	10020
8	10021	10026
9	10022	10023
10	10024	10027
11	10028	10029
12	10030	10031
13	10032	10038
14	10033	10035
15	10034	10036
16	10039	10040
17	10041	10044
18	10042	10043
19	10045	10048
20	10046	10047
21	10050	10053
22	10051	10055
23	10052	10058
24	10056	10060
25	10057	10064
26	10059	10061
27	10062	10063
28	10065	10066
29	10067	10068
30	10069	10075
31	10070	10073
32	10071	10072
33	10074	10077
34	10075	10076
35	10078	10079
36	10080	10084
37	10081	10083
38	10085	10090
39	10086	10088
40	10087	10089
41	10090	10091
42	10092	10093
43	10095	10096
44	10097	10098
45	10099	10101
46	10100	10105
47	10102	10106
48	10103	10104
49	10107	10108
50	10109	10110
51	10112	10115
52	10113	10114
53	10116	10119
54	10117	10118
55	10120	10124
56	10121	10123
57	10122	10126
58	10125	10131
59	10127	10128
60	10129	10133
61	10130	10132
62	10134	10135
63	10136	10137
64	10138	10139
65	10140	10142
66	10143	10144
67	10145	10147
68	10146	10154
69	10148	10151
70	10149	10157
71	10152	10153
72	10154	10156
73	10155	10159
74	10156	10160
75	10157	10158
76	10161	10162
77	10163	10167
78	10164	10170
79	10165	10166
80	10168	10169
81	10171	10180
82	10172	10177
83	10173	10176
84	10174	10175
85	10178	10181
86	10179	10182
87	10183	10184
88	10185	10187
89	10186	10194
90	10188	10192
91	10189	10193
92	10190	10195
93	10191	10197
94	10196	10198
95	10199	10200
96	10201	10203
97	10202	10204
98	10205	10207
99	10206	10209
100	10208	10210
101	10211	10216
102	10212	10214
103	10213	10215
104	10217	10218
105	10220	10221

A-2: Averaged Results

Group #	N-Grams	Style Markers	RNN	CNN	Weighted Avg.
0	0.6059	0.7473	0.5947	0.5687	0.6387
1	0.7872	0.8088	0.8750	0.6185	0.7775
2	0.7398	0.7363	0.6774	0.6590	0.7101
3	0.7695	0.7883	0.8591	0.8158	0.8023
4	0.7438	0.7851	0.7692	0.7857	0.7697
5	0.7642	0.8392	0.8636	0.7069	0.7951
6	0.7773	0.7881	0.8095	0.8131	0.7941
7	0.7257	0.8296	0.7500	0.7320	0.7630
8	0.7703	0.7985	0.6000	0.8044	0.7515
9	0.7222	0.7478	0.7368	0.5518	0.6987
10	0.7255	0.7261	0.6667	0.6360	0.6960
11	0.7827	0.8228	0.9091	0.6551	0.7945
12	0.7538	0.8076	0.8000	0.6001	0.7485
13	0.7291	0.8296	0.7778	0.5905	0.7413
14	0.7088	0.8176	0.8286	0.5681	0.7373
15	0.7390	0.8333	0.6923	0.6174	0.7336
16	0.7321	0.7432	0.8182	0.7611	0.7584
17	0.7574	0.7495	0.9091	0.7523	0.7843
18	0.6510	0.7912	0.6000	0.5541	0.6635
19	0.7819	0.8128	0.7500	0.5942	0.7473
20	0.7686	0.8331	0.8286	0.5708	0.7604
21	0.7036	0.8350	0.6455	0.6175	0.7142
22	0.7274	0.7857	0.7500	0.5481	0.7136
23	0.7225	0.8246	0.7857	0.5825	0.7378
24	0.7382	0.7823	0.6714	0.6175	0.7139
25	0.6641	0.7329	0.6154	0.5583	0.6538
26	0.7466	0.8095	0.7000	0.5754	0.7219
27	0.6390	0.7999	0.5556	0.7958	0.7020
28	0.7260	0.8171	0.6855	0.8065	0.7613
29	0.7814	0.8493	0.8571	0.7548	0.8116
30	0.7419	0.7960	0.6655	0.6876	0.7320
31	0.7495	0.8385	0.6932	0.7348	0.7620
32	0.7459	0.8247	0.7143	0.8391	0.7818
33	0.7232	0.7932	0.6250	0.5999	0.6999
34	0.7438	0.8367	0.7500	0.6392	0.7520
35	0.6497	0.8421	0.5455	0.5982	0.6763
36	0.7088	0.8086	0.7778	0.5901	0.7288
37	0.7879	0.7979	0.6154	0.6023	0.7193
38	0.7507	0.8412	0.8386	0.7410	0.7935
39	0.7439	0.8121	0.8182	0.8745	0.8053

Group #	N-Grams	Style Markers	RNN	CNN	Weighted Avg.
40	0.6872	0.8414	0.6000	0.8617	0.7509
41	0.7390	0.7801	0.6667	0.6879	0.7266
42	0.7530	0.8294	0.7182	0.6786	0.7541
43	0.7539	0.8192	0.7000	0.7915	0.7702
44	0.6059	0.8003	0.5615	0.5971	0.6536
45	0.6247	0.8458	0.5833	0.7338	0.7046
46	0.7149	0.7802	0.9091	0.6569	0.7617
47	0.7738	0.7532	0.7778	0.6802	0.7497
48	0.7089	0.8437	0.7778	0.8550	0.7923
49	0.7825	0.7849	0.8333	0.7749	0.7919
50	0.6170	0.7333	0.6942	0.6041	0.6648
51	0.7569	0.8191	0.7273	0.7835	0.7750
52	0.7414	0.7906	0.7500	0.7290	0.7554
53	0.7331	0.8100	0.7000	0.8025	0.7634
54	0.7098	0.8075	0.7778	0.6640	0.7435
55	0.7337	0.7198	0.6667	0.7232	0.7140
56	0.7054	0.7474	0.6478	0.5895	0.6833
57	0.7735	0.7106	0.8099	0.8419	0.7756
58	0.7879	0.8230	0.6943	0.6852	0.7592
59	0.7569	0.8015	0.8433	0.7322	0.7826
60	0.7228	0.7425	0.7339	0.6628	0.7190
61	0.7487	0.8252	0.7853	0.7124	0.7717
62	0.6579	0.7391	0.6141	0.8595	0.7138
63	0.7398	0.7472	0.6438	0.6103	0.6969
64	0.7744	0.8245	0.6406	0.8574	0.7793
65	0.7677	0.7814	0.6757	0.8180	0.7635
66	0.7877	0.8159	0.7394	0.7043	0.7698
67	0.7032	0.8389	0.7438	0.8398	0.7794
68	0.7791	0.7987	0.7475	0.7812	0.7791
69	0.7643	0.8166	0.6806	0.6339	0.7372
70	0.7227	0.8039	0.6610	0.7610	0.7424
71	0.7261	0.8029	0.6907	0.6350	0.7238
72	0.7329	0.8185	0.6599	0.7819	0.7538
73	0.7239	0.8000	0.6265	0.6792	0.7183
74	0.6425	0.7813	0.6171	0.5933	0.6692
75	0.7490	0.8248	0.6577	0.5598	0.7156
76	0.7579	0.7847	0.7913	0.8535	0.7917
77	0.7749	0.7886	0.6864	0.7142	0.7492
78	0.7204	0.8173	0.6457	0.6961	0.7297
79	0.7848	0.8217	0.7677	0.7366	0.7828
80	0.7876	0.8190	0.6320	0.6323	0.7348
81	0.7007	0.8398	0.6790	0.6458	0.7271

Group #	N-Grams	Style Markers	RNN	CNN	Weighted Avg.
82	0.7165	0.8015	0.6491	0.5693	0.6991
83	0.7331	0.8112	0.6739	0.7305	0.7442
84	0.7165	0.8106	0.6226	0.6361	0.7099
85	0.7054	0.7887	0.6125	0.8325	0.7372
86	0.7408	0.7987	0.7341	0.5812	0.7249
87	0.7821	0.7806	0.6887	0.6276	0.7321
88	0.6760	0.7942	0.7545	0.6279	0.7175
89	0.6802	0.8024	0.6764	0.5594	0.6919
90	0.7742	0.7488	0.6552	0.8367	0.7553
91	0.7595	0.7854	0.7151	0.6686	0.7402
92	0.7868	0.7189	0.6312	0.7151	0.7210
93	0.7507	0.7831	0.7941	0.7189	0.7627
94	0.6382	0.8218	0.6904	0.7400	0.7241
95	0.7029	0.7326	0.7877	0.8352	0.7553
96	0.7416	0.7904	0.6621	0.8747	0.7670
97	0.7719	0.7035	0.7235	0.8203	0.7514
98	0.7716	0.8037	0.6953	0.8661	0.7849
99	0.7860	0.7430	0.6356	0.7814	0.7421
100	0.7892	0.7867	0.7769	0.6164	0.7514
101	0.7728	0.7390	0.7545	0.5885	0.7222
102	0.7700	0.7813	0.7228	0.7368	0.7573
103	0.7089	0.8397	0.6937	0.7863	0.7606
104	0.7655	0.8290	0.6395	0.8136	0.7690
105	0.7693	0.7940	0.7484	0.5911	0.7369