Rectangular Filter Inception Layers for Authorship Identification

Thomas Henningson, Zubair Qazi, Lucca Psaila
Computer Science Department
University of California, Riverside
`<thenn002, zqazi004, lpsai001>@ucr.edu`

Abstract

Forensic handwriting analysis is a prevalent technique used to help identify the perpetrators of crimes where handwritten documents are relevant evidence. Human individuals painstakingly go over handwritten works by hand, attempting to find and recognize certain handwriting patterns unique to each suspect. They then use these patterns to identify the author of an unknown work. This paper presents a novel convolutional neural network architecture to solve this problem. Other works have used models such as singular value decomposition, long-short term memory, and gated recurrent units to solve this problem, but very few have used a deep learning convolution neural network model on image data. An experimental evaluation shows the potential of out proposed architecture to be useful in solving authorship identification.

1 Introduction

For nearly a century, forensic handwriting analysis has been used as evidence in criminal trials [1]. In some cases, it was the key piece of evidence to convict the perpetrator of a crime, such as in the 1932 case, the kidnapping of Charles Lindbergh [1]. Over the years, handwriting analysis experts have been honing their techniques to help put away vicious criminals such as Robert Durst and the BTK Killer [2]. What if there was a way to help these hard-working analysts bring these killers and kidnappers to justice?

With rapid advances in computer processing power, memory, and storage, AI technology has been making a name for itself in the past few years. In fact, AI is already being used in the court room for risk assessment [3]. It would be no surprise if in the next few years, AI becomes increasingly sophisticated enough to be used as admissible evidence in court. We hope to present a start to an idea as to how AI could be used successfully in criminal trials.

In this paper, we propose a novel artificial neural network that utilizes deep learning techniques in order to predict whether a particular piece of handwriting matches to a specific author. The neural network is essentially performing the same type of work as forensic handwriting analysts. The network uses the previous works of some number of authors to learn patterns and handwriting ticks unique to each author, and then attempts to identify those same patterns in the unidentified writing. Our network has over 78 thousand parameters which are trained in order to identify specific features of handwriting. We utilize rectangular convolution filters to extract the features, and we structure our network similarly to GoogleLeNet [4].

The rest of the paper is structured as follows: Section 2 is an overview of related work while Section 3 defines the methodology of both our network and our experiments. Section 4 contains the experimental evaluation of our neural network and Section 5 concludes the paper. Section 6 lists the individual contribution of each team member. All relevant code can be found on the GitHub.

2 Related works

Authorship identification has been a prominent area of research for many years now, and recently has been actively pursued as a classification task in machine learning. Many supervised methods have been applied to this task, including multi-nomial logistic regression, random forest, support vector machines (SVM), transformers (BERT), etc [5]. Since many of these models are relatively simple in nature, it is necessary to perform feature extraction on the data. Feature extraction on text data is its own area of research as well, and the merit of many techniques, such as Bag of Words, Latent Semantic Analysis, and Singular Value Decomposition are being actively researched [5].

Since the focus of this project is on deep learning based methods, we focused mostly on searching for other approaches which employed deep learning methods for the purpose of authorship identification. However, we found that there has been very little published research on applying deep learning methods for this task, and most of the compiled work online consists of unpublished papers, student projects, or journals. These projects utilize Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) models for the task [6], and are not based on image data. There is even less research on applying deep learning methods for authorship identification based on image data. So, our task and our attempt at solving it are both novel.

The relevant deep learning papers that we found generally worked with text data directly, rather than through images as this project has aimed for. This may be due to the complexity of the task; regardless, we attempted to search for authorship identification using image data, and did not find any reasonable results beyond blog posts and articles. Since there were not many results to be found, we referred to other papers tackling image classification tasks [7].

Image classification is a common task typically tackled via deep learning. Since our task is an image classification task, albeit a very complex image classification task, we referred to some other papers implementing deep learning models for said task. The most common model utilized for image classification is the Convolutional Neural Network (CNN) [8]. The model is able to perform well on image-based datasets, and has many variations with varying complexity and depth. The structure of the model is heavily influenced by the structure of GoogleLeNet and the use of inception blocks [4], which have already been proven to perform well on image classification tasks.

3 Methodology

Our methodology begins with the research we did to find an appropriate data set to run our network on. In particular we were interested in a large data set of high quality images of English handwriting, that came labeled with the author. With the problem we were trying to solve, we were also interested in a data set that had many authors. Since classifying handwriting as being from a single writer is extremely difficult, we were looking to be able to classify it as coming from a certain group of authors.

We found exactly what we were looking for in the IAM Handwriting Database [9]. This data set contains 1,539 pages of scanned text from 657 unique authors. "The database contains forms of unconstrained handwritten text, which were scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels." [9] Figure 1 shows some examples of the images within the dataset.

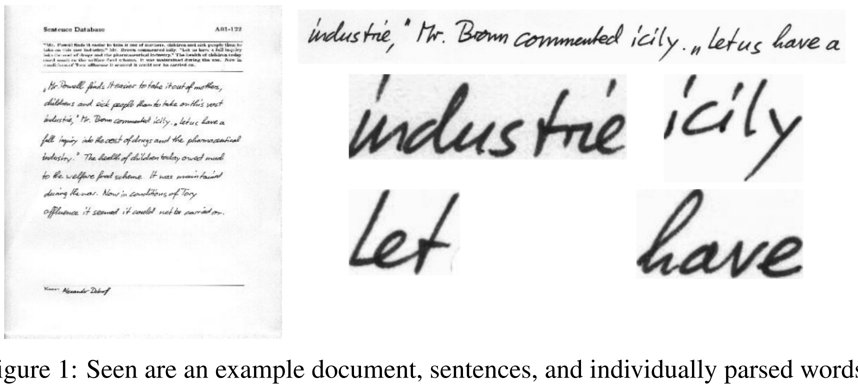

industrie," Mr. Brown commented icily. " Let us have a
industrie
icily
let
have

Figure 1: Seen are an example document, sentences, and individually parsed words

Through these images, the database divides the text into several subsections by extracting information with an automatic segmentation scheme that was manually verified for quality and accuracy. The subsections of images they offer are isolated sentences, text lines, and words. The choice of which subsection to use was one of the first decisions we had to make when developing our project, and we ended up choosing the isolated words. The main reason is that we needed some uniformity in our images. With sentences, you can have an extremely varied lengths. With a word, you are using a slightly more standard length and width for an image size. Lastly was our computing power. We had tested our network on many different sample sizes, such as 256x256 images, but the dimension of each image has an extremely large impact on loading the database into memory. Thus, the image samples that were the smallest were the easiest to load and perform tests on. We eventually settled on a dataset of 192x192 pixels.

We cleaned this dataset by first reducing the number of authors from 657 to 100, taking the 100 authors with the most examples. This gave us a large enough set of images to work with, while reducing the complexity of the task. We settled on 120 images from each author, since this was the amount of images the last author had that were under 192x192 pixels. This gave us a dataset of 12,000 images.

We padded all of the images so that the input dimensions in our network were uniform at 192x192 pixels. We randomly split the database into 10,000 training images and 2,000 test images. We would also later invert the images to fix a compatibility issue, but we will addressed in the Experiments section. This is the final image dataset we used for all the Experiments. We had also done some testing with a 10-author dataset, but this dataset turned out to be too small to train the network effectively, so we reverted back to the 100-author dataset.

Now, we will begin to describe the neural network architecture we implemented. Our network architecture consists of two unique features: Rectangular convolution filters and Inception layers. Inception layers were a concept introduced by GoogleNet[4]. They consisted of multiple convolution layers of different sizes that are then concatenated together to restore the original input dimensions Each convolution utilizes padding to keep the height and width dimensions, but it decreases the channel dimension. However, the concatenation of all channels restores to original channel dimension.

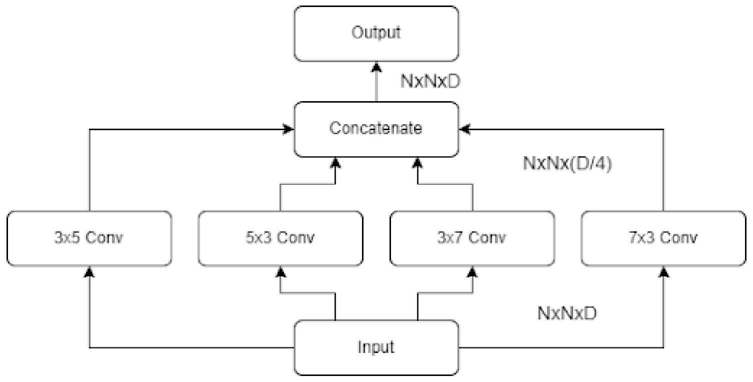


Figure 2: Architecture of an Inception Block

Figure 2 shows the inception block architecture from our network. Since we used 4 convolution layers, each one had one-fourth the original number of channels, so that when they were concatenated it restored the dimension. In some versions of the network (not shown) implemented residual connections, where the input was added to the output after the concatenation, and ReLU activation was performed again.

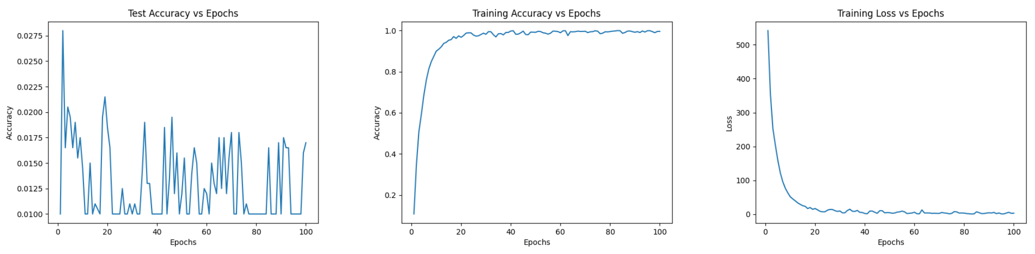
We used 4 rectangular filters in the Inception block: 3x5, 3x7, 5x3, and 7x3. The motivation behind this was relatively simple; most handwriting is characterized by vertical and horizontal strokes. To help the network capture the differences between styles of writing, it is important to capture these features. We believed this process would be easier with rectangular filters.

The rest of the network is quite simple. We performed an initial 3x3 convolution on the input to increase the number of channels from 1 to 16. Therefore, the input to the first inception layer was 192x192x16. Then, we had 5 Inception layers, which consisted of two Inception blocks back to back, and another 3x3 convolution that would half the height and width dimensions, while doubling the channel dimension. Therefore, the outputs to each layer are as follows: 96x96x32, 48x48x64, 24x24x128, 12x12x256, and 6x6x512. After the 5 layers, there was an average pooling layer which reduced the dimensions to 1x1x512. Finally, there was one fully connected layer that would output the 100 class predictions.

In the following section, Experiments, we trained this network architecture (with a few tweaks that will be explicitly mentioned) for 100 epochs with mini-batches of size 64. We used a constant learning rate of 0.001, an Adam optimizer[10], and the Cross Entropy loss function. All networks were trained in this manner using CUDA on an NVIDIA GeForce RTX 2070 graphics card.

4 Experiments

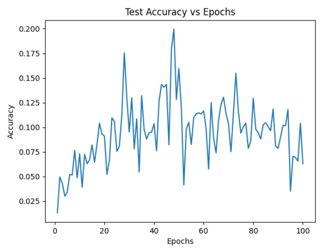
Before we show the results of the experiments, we quickly wanted to address an issue we discovered after the poster presentations. During said presentations, we had shown results that showed our network was no better than random guessing. We have discovered why that is the case. Our original image files came as one channel grey scale images of black ink text on a white background. For a floating point grey scale, white is assigned 1, and black is 0. However, when PyTorch implements a convolution layer, any padding added is automatically assigned a value of 0. Essentially, this means that a black border was put around each image, which was on white paper. Since the important features of handwriting would be most apparent in the black text part of the image, this is most likely what caused our issues with our network being unable to predict authors correctly. All we did to fix the issue was to simply invert the scale on each image, essentially making them white text on black paper, and now the black border has no significant meaning to the network.



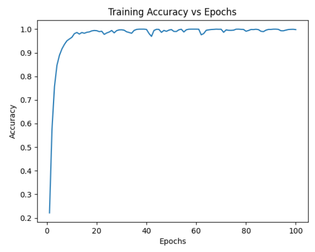
(a) Test accuracy (b) Training accuracy (c) Training loss

Figure 3: Results of training the network for 100 epochs at a 0.001 learning rate (No data transformations, No residual connections).

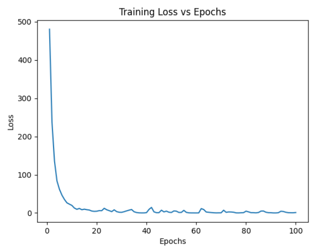
Figure 3 shows the results after implementing the inversion on the images. While these results are not the best, they told us that the network is indeed learning, albeit not very well. What we can see is that the test accuracy never dipped *below* random guessing (which is 1% for classifying 100 authors). Upon inspecting the training accuracy, and seeing that it came to near 100% we came to the conclusion that our model may be over-fitting. Since the test accuracy was not following the same trend, we decided to try to implement residual connections between the inception blocks. We had originally tested residual connections before we discovered the inversion issue, but they did not perform well. This time, we got different results.



(a) Test accuracy



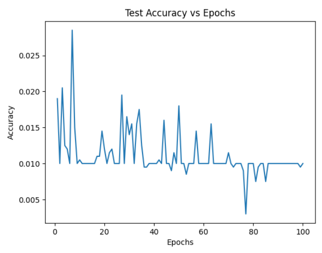
(b) Training accuracy



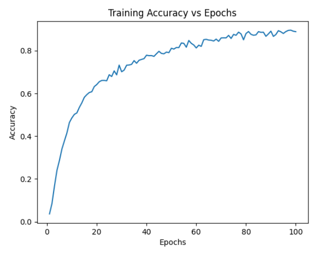
(c) Training loss

Figure 4: Results of training the network for 100 epochs at a 0.001 learning rate (No data transformations, with residual connections).

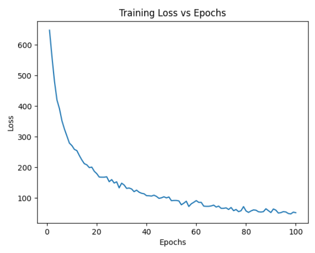
Figure 4 shows the results after implementing residual connections. These results were a huge breakthrough, reaching an accuracy of 20% at epoch 44 and ending with an accuracy of 5.7%. This means our model is better than random guessing by about a factor of 5-20. It also suggests that our model may be too complicated for this problem, as the residual connections were able to improve the test accuracy significantly. However, it still seemed to be the case that our model was over-fitting, due to the fact that training accuracy was still nearly 100% by the end of the training, while test accuracy remained relatively low. Therefore, we decided to try data transformations, while keeping the residual connections.



(a) Test accuracy



(b) Training accuracy

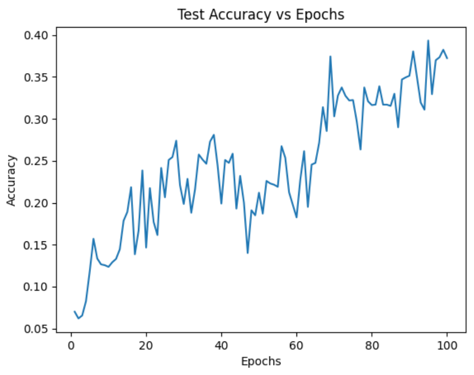


(c) Training loss

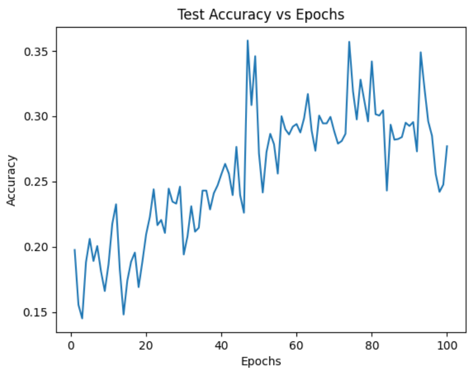
Figure 5: Results of training the network for 100 epochs at a 0.001 learning rate (with data transformations and residual connections).

Figure 5 shows the results after implementing data transformations. We implemented the following transforms, on a per batch basis: a random 192x192 crop, random translations on the page, random scale changes between 0.5x and 2x, a 50% chance to flip horizontally and vertically, and a random perspective change up to 10%. These results were not very beneficial for the network. The data transformations did help to reduce the convergence rate of the training accuracy, but severely hindered the networks ability to predict unseen data. The training ended with a test accuracy of 1%, which is no better than random in this case. We also checked a less intensive version of data transformations, in which we only applied the standard augmentation method of transformation [11]. Those results were very similar to those in Figure 3. Therefore, we decided to abandon data transformations for future tests.

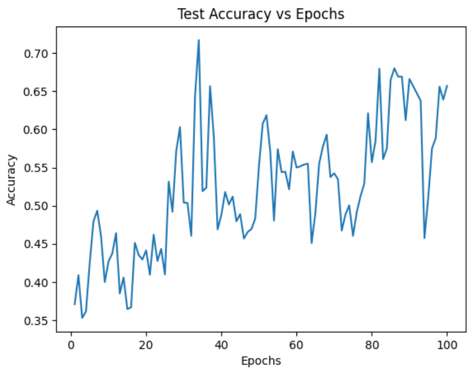
Instead, we focused on what else the network could be learning. It seems to be the case that many people may have similar handwriting, or in some sense have the same *style* of handwriting. It could be possible that the neural network does not have enough predictive power to distinguish the individual authors that have the same styles. Therefore, we decided to look at various Top-K accuracy's, which may indicate that the network has trouble predicting just one author when two authors have very similar handwriting.



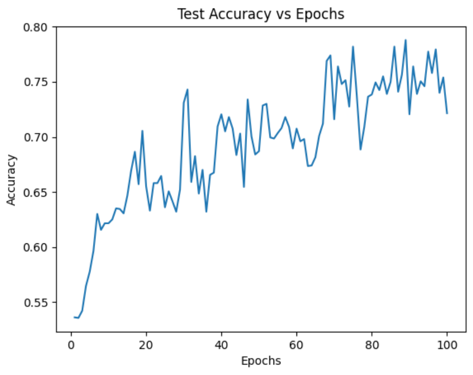
(a) Top-5 test accuracy



(b) Top-10 test accuracy



(c) Top-25 test accuracy



(d) Top-50 test accuracy

Figure 6: Top-K accuracy's .

Figure 6 shows the test accuracy for various choices of Top-K. The most significant graph is 4a, the Top-5 accuracy. This is because we are able to gain substantial improvements in accuracy just by looking at the top 5 authors over just the top author. By the end of training 100 epochs, the author of the test data being the first predicted author is only 5.7%, however, the chance it is among the first 5 predicted authors jumps all the way 37.3%. This discovery reinforces our idea that the network may be isolating certain styles of handwriting, and is unable to distinguish between authors of very similar handwriting as well as it could the style. The graph also shows an upward trend as the epochs progress, which indicates that perhaps with more epochs and more training data, we could see even more improvements in the Top-5 accuracy. Interestingly, the Top-10 accuracy performed about as well as the Top-5 accuracy, which suggests that if the author is among the top 10 predicted authors, it is almost certainly also among the top 5 authors as well.

The next most significant jump occurs in the Top-25 accuracy. We found a final accuracy of 66.1%. However, having to look through a quarter of the potential authors to achieve this accuracy is not nearly as impressive as only having to look through 5, even if the overall accuracy is higher. Similarly to the Top-10 accuracy, the Top-50 graph mirrors the Top-25 graph, although with slight improvements, ending at a test accuracy of 72.3%. Again, this suggests that the actual author being among the Top-50 authors almost certainly means they are also among the Top-25 authors, so Top-50 does not offer any major significance, or at the most, just less significance than Top-25.

5 Conclusion

We presented a novel deep learning CNN architecture to solve the forensic authorship identification problem. No other deep learning network to our knowledge has attempted to use image data of handwriting to solve this problem, opting instead to use GRU and LSTM models. This network, although not accurate enough to be particularly useful, has shown the potential to learn this problem and accurately predict authors handwriting from their previous works. Most significantly, after only 100 epochs of training the network was able to identify the correct author within the top 5 authors for over one-third of testing examples. Given more time, we could most likely improve the network by tweaking the following attributes:

- Increase the number training epochs.
- Increase the number of training and test examples.
- Fine-tune hyper-parameters such as the learning rate, activation function, and loss function.
- Provide handwritten letters as input, over handwritten words (or perhaps both).
- Decrease/increase the number of authors the network distinguishes.
- Add more fully connected layers to the end of the network.
- Adjust the filter sizes in the inception block.
- Reduce the number of inception blocks/layers.
- Increase the number of inception blocks, if reducing them yields no significant results.
- Change the order of the inception blocks among the 3x3 convolutions.

6 Contribution of Each Member

We ensured that each team member had an equal level of contribution to the final project. The projects main areas of focus were: data pre-processing, image augmentation, and model architecture construction and training. Each team member had a hand in each area so that we were not limited in the experience we gained, but each team member was responsible for seeing the completion of certain areas through. The contributions, specifically, are as follows:

Thomas H.
Thomas was primarily responsible for the architecture of the CNN model and network-related PyTorch code.

Zubair Q.
Zubair was primarily responsible for image augmentations and training/testing-related PyTorch code, and secondarily responsible for the architecture of the CNN model.

Lucca P.
Lucca was primarily responsible for the pre-processing and loading of the data, including writing processing-related python code and dataloading-related PyTorch code.

Each member had a primary objective to complete, but in actuality each member contributed equally to each part of the project, since each part required deliberation and unanimous consent to be able to proceed. For example, each member's input was necessary to decide on image augmentations to apply to the data, as well as the correct parameters for various layers of the model's architecture. As a result, each team member had an equal contribution overall to the final deliverable.

References

[1] Stephan Lewandowsky. Who kidnapped charles lindbergh, jr? forensic handwriting analysis and expertise, 2022. URL `https://featuredcontent.psychonomic.org/who-kidnapped-charles-lindbergh-jr-forensic-handwriting-analysis-and-expertise/`.

[2] High-profile cases cracked with handwriting analysis, 2022. URL `https://www.drexdoclab.com/high-profile-cases-cracked-with-handwriting-analysis/`.

[3] Caleb Watney. It's time for our justice system to embrace artificial intelligence, 2022. URL `https://www.brookings.edu/blog/techtank/2017/07/20/its-time-for-our-justice-system-to-embrace-artificial-intelligence/`.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. URL `https://arxiv.org/abs/1409.4842`.

[5] Noura Khalid Alhuqail. Author identification based on nlp. *European Journal of Computer Science and Information Technology*, 9(1):1–26, 2022. URL `https://www.eajournals.org/wp-content/uploads/Author-Identification-Based-on-NLP.pdf`.

[6] Chen Qian, Tianchang He, and Rao Zhang. Deep learning based authorship identification. 2022. URL `https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760185.pdf`.

[7] Nadia Jmour, Sehla Zayen, and Afef Abdelkrim. Convolutional neural networks for image classification. In *2018 International Conference on Advanced Systems and Electric Technologies (*IC_ASET*)*, 2018.

[8] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, 2017.

[9] Urs-Viktor Marti and H. Bunke. The iam-database: An english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5: 39–46, 11 2002. doi: 10.1007/s100320200071.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `https://arxiv.org/abs/1412.6980`.

[11] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 2019. doi: 10.1186/s40537-019-0197-0.