MATCHA FONT: HELPING IMPROVE HANDWRITING FOR READABILITY

Eshan Sankar
Student# 1009928367
eshan.sankar@mail.utoronto.ca

Philip Wu
Student# 1008911224
phillip.wu@mail.utoronto.ca

Samuel Ho
Student# 1010146450
samueldy.ho@mail.utoronto.ca

Joseph Yeh
Student# 1009824950
joseph.yeh@mail.utoronto.ca

1 INTRODUCTION

The purpose of this project is to create a deep learning model that matches handwriting to its most similar font, to help writers improve the legibility of their handwriting and subsequently help readers better understand handwritten text. There is ongoing research into easy-to-read fonts, to help those with learning disabilities who have difficulty with tracking a sentence. Even when the most readable font is found, there will remain situations where handwriting is necessary. Therefore, a model to help users hone their handwriting to match a particular font would be beneficial. Deep learning is suitable for this problem because the solution must recognize vague patterns that humans cannot easily identify, and the model's confidence could act as a numerical "similarity percentage" to guide users.

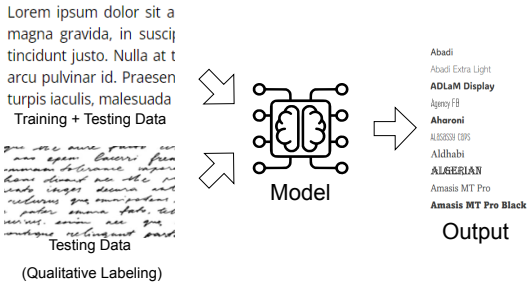

Figure 1: Our font recognition model plan.

We plan to first train the the model on text automatically generated with python, and then test our model with both python generated text, and handwritten text. Currently, most of artificial intelligence out there are using manual processing which is hard coded in, and the few machine models within the area are simply bare-bone CNNs with pooling attached to fully connected layers. We plan to improve on these models by including present-day neural network architecture: strided convolutions, normalization, dropout layers, and 1 by 1 convolutions.

—-Total Pages: 9

2 BACKGROUND AND RELATED WORK

Handwriting is an essential part of daily life, especially in the education field. However, especially based on personal experiences, the handwriting of educators differs greatly, and lessons require both educators and students to write very quick, such that it can be difficult to follow what is being written if it is not in a form that is easily absorbed. This issue is further exacerbated for students

with dysgraphia, a difficulty reading handwriting (ClevelandClinic, 2022). As such, the search for the most easily followed font has begun. Several models have been created that either convert handwriting into typed text or identify the font in which a text is typed. However, we propose a unique solution which combines these two ideas: we plan to train our model to identify the font in which a typed text is typed and ultimately use the model to identify the fonts most similar to a handwritten text based on a numerical accuracy and visualization (to help users amend and refine their handwriting to match a particular font). This would improve accessibility for students and overall improve the convenience and quality of lectures.

One of the earliest font recognition machine learning models by Lee & William J.B. (1990) dates back to the 1990s. Their Model H-H1 and Model H-H2 used cascading memory matrices to distinguish the characters of six different fonts with an accuracy rate of 100%.

Nowadays, many online websites such as Fontspring, WhatTheFont etc., are able to find fonts by searching through its database, and comparing the font used. This, however requires a lot of manual pre-processing, and it does not work if the text is rotated or blurry (Wang et al., 2015).

In more recent years, researchers working with Adobe.inc constructed a model called DeepFont (Wang et al., 2015). DeepFont is trained on the AdobeVFR Dataset which contains 2383 Font Categories with 1000 training images per font, and it achieved an accuracy of over 80%. Deepfont is a domain adapted Convolutional Neural Network (CNN) with 5 convolutional layers and 3 fully connected layers, and its learning is based on model compression.

Another interesting model created by Garfinkiel (No date) was a neural network with two outputs: font and character. The errors of these two outputs were then plotted in two dimensions using PCA to determine the similarities of fonts.

Additionally, Handwriting recognition and verification have seen significant progress which is displayed in the paper titled "Handwriting identification and verification using artificial intelligence-assisted textural features" (Zhao & Li, 2023). The study uses a CNN approach to recognize the writer of a signature with the help of a Spatial Variation-dependent Verification (SVV) scheme using textural features (TF). This is relevant to our project as it demonstrates how written characters can be traced to a specific writer and for MATCHA FONT it would apply to various fonts.

Lastly, Recurrent Neural Networks (RNNs) were also experimented with to improve the accuracy of the CNNs in text recognition (Sumathy et al., 2023). Researchers found that the combination of RNNs and CNNs increased the accuracy of recognizing handwritten texts.

3 Data Processing

We automated dataset generation process by using a Python script using the image generation library Pillow (and supervising the results to ensure the validity of the data). There exist hundreds of thousands of fonts on the internet but we started with only 249 of them which were found preinstalled in the Windows operation system. We removed fonts which were basic variants of other fonts (e.g. "ariblk" was just a bolded version of "arial", so the former was removed), and performed visual inspection to remove other extremely similar/seemingly identical fonts, and fonts which could not be displayed (e.g. "wingdings"), bringing the total number of classes down to 42. This also balances our dataset; if very similar/almost identical fonts are present in the data, that particular type of font is then more prevalent.

Since various words of different lengths and spelling are required, the text we have chosen for font image generation was George Orwell's book, 1984, for a good representation of the English language. Random strings of varying lengths from 3 characters to 100 characters from the book are pasted on each image. To imitate people's handwriting styles, we would rotate each character by a random amount in the range of $[-10°C, 10°C]$. Furthermore, we have set the line spacing and font size for our fonts to be 20 pixels and 15 respectively. The generated images are seen in Figure 2.

(a) Consola

(b) Inkfree

Figure 2: Example images from font training dataset.

For each font, we made 4000 images of size 224×224 pixels, and randomly split the dataset as 60% for training, 20% for validation, and 20% for testing, which helped balance our dataset. This dataset is used to train our model on patterns and features exhibited by perfect fonts, trying to group words of the same font into clusters (clustering the dataset).

After this initial dataset, we further applied our model to a new testing dataset. We have included images of handwritten text with no label to observe how the model classifies the closest font of a handwritten text sample. Each of us created 5 handwriting samples to create a dataset of 20. We cropped the images to 224×224 pixels before passing it to the data-loader. These handwritten texts are not going to be perfect matches for text classification and we want the model to be able to determine a probability of match based on the deformation and other similar fonts. Hence, these handwritten images will only be used to test the model after the initial dataset of typed font is used to train the model (as we do not want to teach the model to perfectly classify deformed fonts). This allows the model to still give feedback to the user on when they can improve their handwriting. Although there is uncertainty associated with using two different datasets to test the model, the results of the tests will be analyzed visually and qualitatively to make observations about this novel testing method. This demonstrates a form of semi-supervised learning.

4 Baseline Model

For the baseline model, we have used a standard 3-layer CNN that feeds into a single-layer fully connected network. The architecture of this model can be seen in Figure 2.

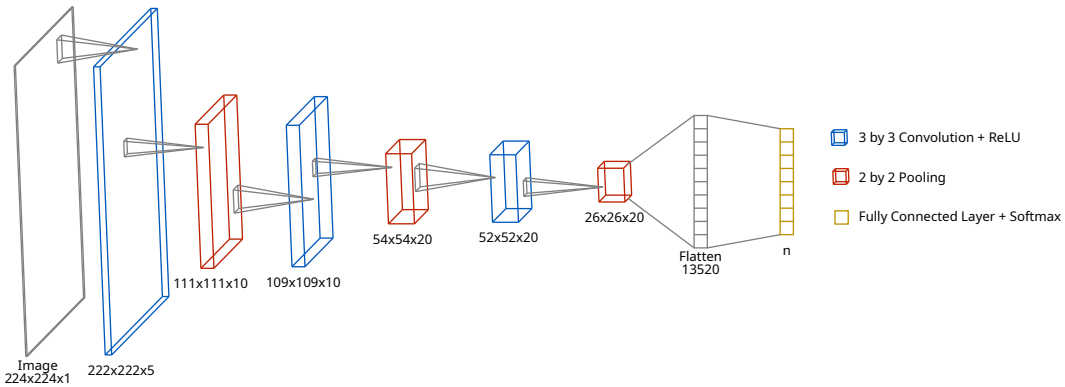

Figure 3: Diagram of the baseline model.

This model was trained using our dataset of 4000 images each of 42 different fonts, using the Adam optimizer and the cross-entropy loss function. We set the batch size to be 32 and set the learning rate at 0.005. We used 15 epochs to ensure convergence. Its results are discussed in section 6: results.

5 ARCHITECTURE

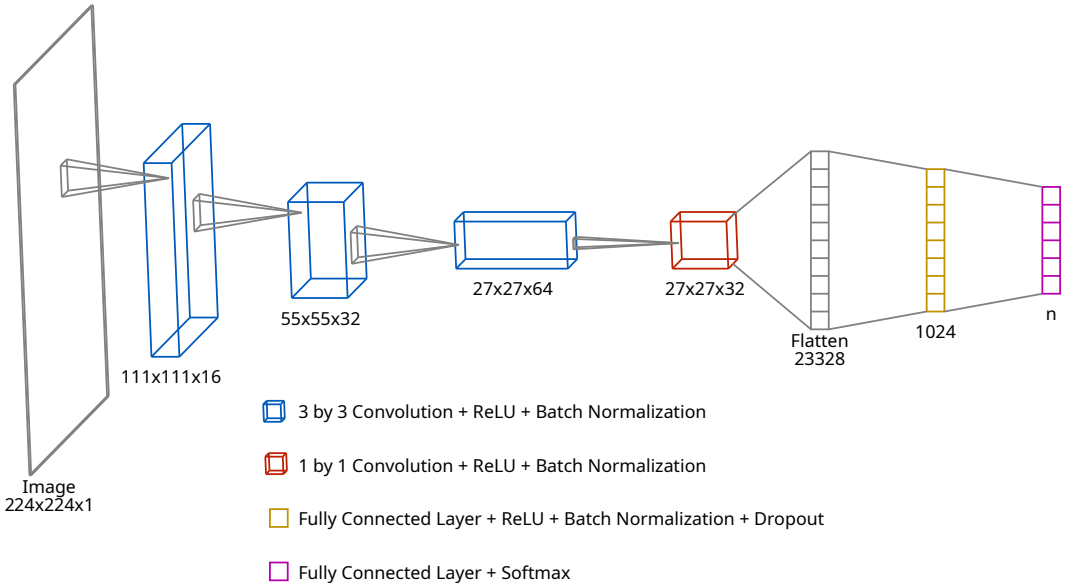

Figure 4: Diagram of current primary model. n represents the number of font classes which is 42.

Our primary model consists of 2 sections: A convolutional neural network to extract spatial features that can be used for the classification of these fonts and a fully connected classifier with 42 outputs matching the 42 fonts.

5.1 CONVOLUTIONAL NEURAL NETWORK

The CNN takes the $1 \times 224 \times 224$ images and processes it to extract spatial features. It consists of 4 convolutional layers. The first 3 layers have a ReLU activation function, stride of 2 to reduce spatial dimensions (as opposed to pooling) (Quercus, 2024a), and a 3×3 kernel size, using 16, 32, and 64 filters respectively. The fourth convolutional layer has a kernel size of 1×1 reduces the channels from 64 to 32. This dimension reduction ensures that only relevant information is retained into the next layer (Quercus, 2024b), and reduces the number of connections needed in the fully-connected layers, improving both feature recognition and efficiency (Sakthi, 2020). Each layer is followed by a batch normalization layer to ensure that no one pixel is paid too much attention. The result of the CNN is a feature map of size $32 \times 27 \times 27$, which provides adequate dimensions for embedding the most essential features.

5.2 FULLY CONNECTED NETWORK

A fully connected network with 1 hidden layer then classifies the CNN's outputted feature map to each of the 42 possible fonts. The hidden layer has 1024 neurons to preserve the most important features, a 50% dropout is used here to prevent overfitting (dropout is only present here as it would not have been as effective in the CNN) (Hinton et al., 2012), and the output layer has 42 neurons. The hidden layer uses ReLU activation, and the output layer uses the softmax activation function as required for multi-class classification tasks. Since this is a multi-class classification problem, cross-entropy with logits loss is used. The Adam optimizer with weight decay is used for a simple yet effective implementation.

6 RESULTS

This section includes the Quantitative and Qualitative Final results of our primary model, as well as an evaluation of the model on new data (testing data).

After testing the model with completely untouched labeled testing data consisting of 800 images of each font, the final accuracy of the primary model was 89.21%, compared to our baseline which performed at an accuracy of 87.86%. The 89.21% accuracy from our primary model is an impressive accomplishment due to the vast number of classes there are and our limited hardware.

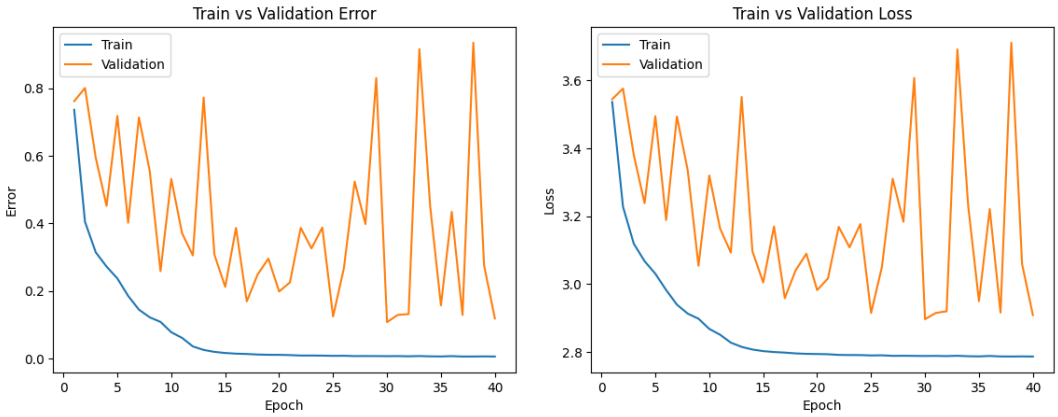

Figure 5: The training and validation error/loss graph for the primary model.

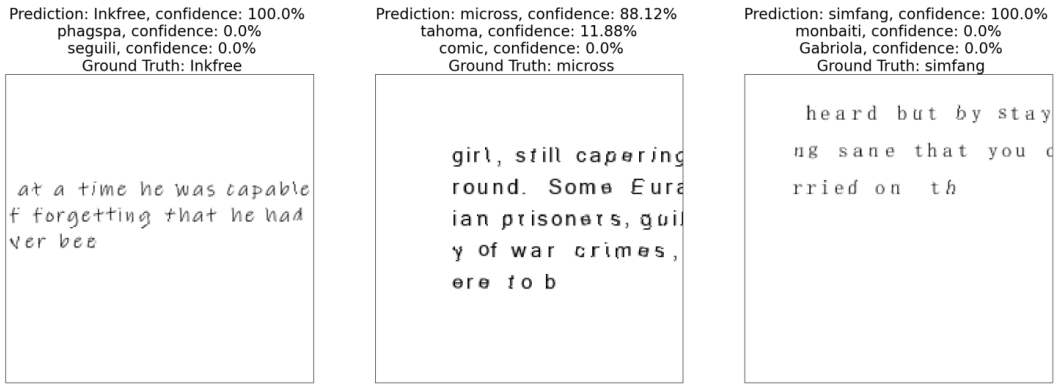

Figure 6: As shown above, our model is extremely confident with completely new, and unseen testing samples.

To ensure that the training, validation, and testing datasets were unique, we randomized the dataset indices at the beginning of our notebook, saved the indices, and used those same indices for creating all datasets. Hyperparameters were not altered after or during the evaluation of the testing dataset.

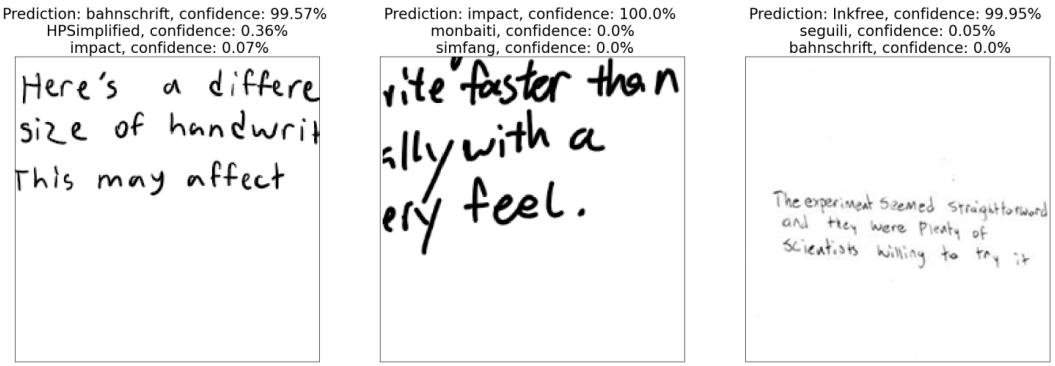

Figure 7: Our model is also extremely confident when predicting the closest font matching the handwritten samples.

The implications of this are discussed in the subsection: 7.2.

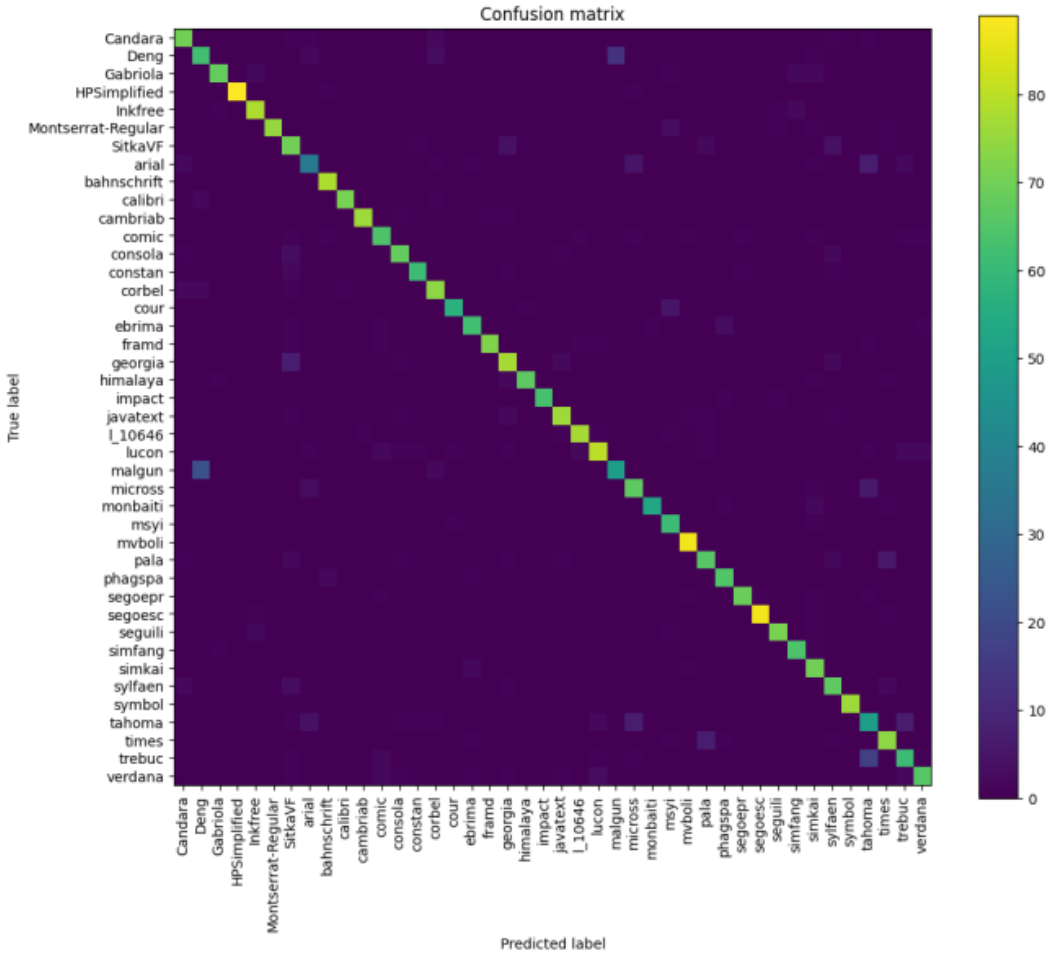

Figure 8: Confusion matrix for the primary model. The colour bar represents the percent of samples predicted within the testing dataset. Overall, the primary model performs pretty consistently since there is not one specific font which is being left out, and predicted inaccurately. The diagonal line shown in green is a good qualitative indicator of high accuracy.

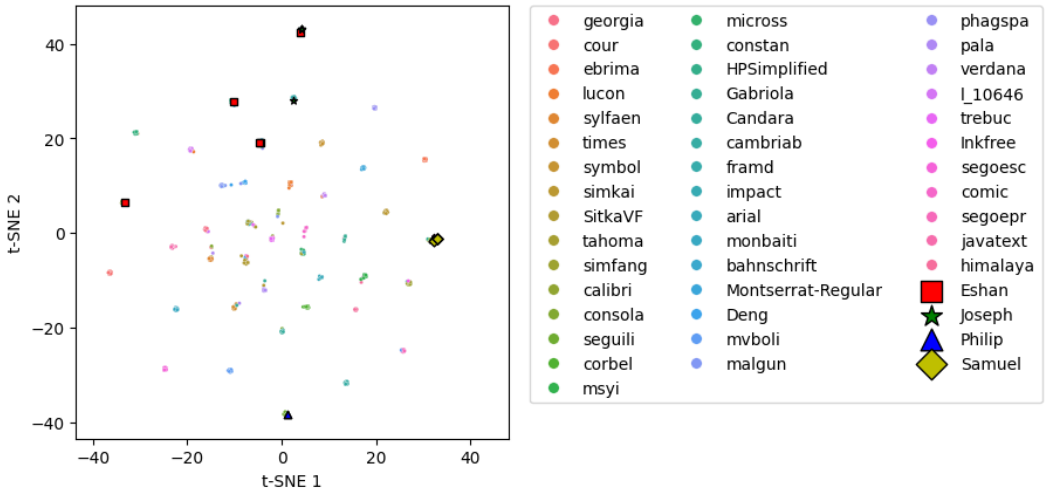

Figure 9: t-SNE visualiations of the model output. 20 samples of each font were plotted on a 2D plane using t-SNE. Furthermore, 5 samples of each of our handwritten fonts were plotted on the graph as well. There is evident clustering by our model, and we can easily visualize which fonts/handwriting styles are similar to each other based on how close they are.

The clustering shown above is important for the model's output of confidence scores relating to handwritten data. This allows us to classify handwritten data without the model learning to treat them as examples of perfect fonts.

Altogether, we believe that our model exceeded its expectations in terms of classifying fonts, beating all previously mentioned models in the background, section:2, in terms of accuracy when analyzing perfect fonts, the same input previously mentioned models take in. Our model also accomplishes something new that has not been done before: comparing handwriting to existing fonts. This novel method of creating a learning tool for instructors to better their handwriting is a step towards more legible handwriting in classrooms.

7 Discussion

7.1 Model Performance

The model's performance, indicated by the t-SNE and confusion matrix shows that the final model we created can successfully cluster similar fonts and differentiate between unique styles. The diagonal distribution in the confusion matrix indicates each font is being predicted correctly, though there is some confusion among similar-looking fonts. When applied to handwritten images, the model displays very strong confidence in matching them with a font. This suggests its practicality in aiding people to refine their handwriting into specific fonts. Despite including 42 font classes, it is possible for the model to classify handwriting to the closest of our chosen fonts but not closest to all possible fonts. This showcases a limitation in our hardware as this would require much more computational power and memory. The final model ultimately had an accuracy of 89.21%. Considering the sizeable number of font classes and hardware constraints, the result is surprising.

7.2 Handwritting Output Implications

The confidence in identifying closely matching fonts in handwriting as shown in figure 7 may indicate the model views any legible handwriting as similar to legible fonts; however, this also means that what is defined as legible to students may change. For example, some perfect fonts may be awkward for individuals to read, thus causing a match to that font to be unwanted. Another interesting note is that zoomed-in photos of handwriting tend to be classified as bolded fonts, no matter the style of writing.

7.3 Training and Architectural Insights

We have learned the importance of dataset size to the performance of the model. We started at 100 images per font and each time we increased this by a magnitude of 10, the accuracy significantly improved with the same model. This suggests that further increasing the dataset size and classes could enhance the model's generalization and accuracy. Another thing we had to alter in the dataset from our initial plan was getting rid of fonts that were too similar for even the human eye to interpret as different. Since there are hundreds of fonts available, it is inevitable that some are not unique, and are instead repeated under different names. This created clear lines of confusion in the confusion matrix, which on closer examination revealed this flaw. This does not devalue our model for our application as we want to match handwriting to reproducible fonts to the human eye for legibility. If fonts were too similar for the human eye to differ, there is no point in the writer creating text to try to match one or the other.

We also learned that sometimes a simpler model is better at a task. At first, we tried using an autoencoder to denoise the input images before classification, which gave a decent accuracy similar to our baseline model. However, when we replaced the autoencoder with more CNN layers, we found the added CNN layers had learned to denoise the images as well and extract features congruently. This made applying tools such as batch normalization more efficient and easier.

8 Ethical Considerations

This model is intended to improve accessibility and teaching efficacy, but there are ethical concerns regarding writing individuality, bias introduced by the handwritten samples in the dataset, and potential uses for committing fraud.

8.1 Model may Repress Individuality

This model can train educators to write more legibly, but this may cause their writing styles to become more similar to each other. Some may see this as an attack on individuality or human identity, or an attempt to make humanity more similar to machinery. It could also cause a desensitization to cultural differences in handwriting. This would cause prejudice in educational settings, looking down on those with different handwriting, and may hinder inclusivity efforts. Thus, the limitation of the model to be used only in professional development settings and as a tool instead of as a grading scheme must be emphasized.

8.2 Potentially Biased Dataset

Since we are English speakers from an English-speaking culture, our custom handwritten dataset represents the writing of that culture. Since we are using this dataset in the development of our model, the model may inherently be biased towards writers from English-speaking cultures and may be prejudiced in favour of Anglo-centric cultures. As such, users of this model must be informed of our methods, so they are aware of its limitations and potential dangers of purely relying on this tool for professional development.

8.3 Matching Handwriting for Fraud

The probabilities outputted by the model can not only give feedback on how similar handwriting is to a font but also how similar it is to another person's handwriting. Therefore, the training data could be modified to include the handwriting of other people, enabling the model to train malicious individuals to commit fraud and forge other people's handwriting and signatures. As such, the model should primarily be shared with educational institutions. The usage of the model could also include a waiver that acknowledges the responsibilities that come with its usage. Although this does not completely safeguard against this risk, it is one positive step towards ensuring its proper use.

9 DIFFICULTY

Though our initial model planned and trialed in our Progress Report consisted of an autoencoder, CNN, and Fully Connected network, we found a CNN on its own was able to accomplish the task our initial autoencoder was trained to do. Because of this, we optimized our CNN as much as possible through the previously mentioned batch normalization, 1×1 convolutions, and very deliberate dimensions. The development process was especially difficult, as we implemented and trained many preliminary models with different architectures before settling here. For example, we attempted to use an autoencoder to create an embedding layer but later found an intermediary CNN with 32 channels of size 27×27 to be a better representation of the fonts and could be used as an embedding layer. We also attempted to use different types of skip connections, such as concatenating or adding our autoencoder's embedding layer to the output of a separate CNN. However, we found these skip connections to instead increase our validation error and cause the model to overlearn on the training data. As a final CNN optimization idea, we drew inspiration from Sunkara et al.: (Sunkara & Luo, 2022): thinking that our "strided convolution and pooling [were skipping]" important characteristics (p. 2), we tried to use a method that would not "cause non-discriminative loss of information" (p. 6), by using space-to-depth layers.

Ultimately, our best-performing model used none of these additions. Our final model was able to "denoise" text images that were rotated at different angles using only the first layers of the CNN. We were surprised to learn that this was possible, since in APS360, we were only introduced to denoising with autoencoders. This was indeed a difficult task, especially given that fonts have minute features that sometimes even humans cannot detect and classify, and they end up becoming even more indistinguishable with rotation. However, our results show that this is also a testament to the power of deep learning.

10 PROJECT LINKS

Here are the relevant links for accessing the code of our project.

10.1 GITHUB REPOSITORY LINK

`https://github.com/EshanSankar/Matcha-Font-APS360-Project`

REFERENCES

ClevelandClinic. Dysgraphia, Jun 2022. URL https://my.clevelandclinic.org/health/diseases/23294-dysgraphia.

Edy Garfinkiel. *Font Similarities with Artificial Neural Networks*. No date. URL https://www.dgp.toronto.edu/~egarfink/Final_report.pdf.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. URL https://arxiv.org/abs/1207.0580.

Ming-Chih Lee and Oldham William J.B. Font recognition by a neural network. *International Journal of Man-Machine Studies*, 33(1):41–61, 1990. ISSN 0020-7373. doi: https://doi.org/10.1016/S0020-7373(05)80114-2. URL https://www.sciencedirect.com/science/article/pii/S0020737305801142.

APS360 Quercus. Week 4 convolutional neural networks - part i, 2024a.

APS360 Quercus. Week 5 convolutional neural networks - part ii, 2024b.

Raj Sakthi. Talented mr. 1x1: Comprehensive look at 1x1 convolution in deep learning. *Analytics Vidhya*, Jan 2020. URL https://medium.com/analytics-vidhya/talented-mr-1x1-comprehensive-look-at-1x1-convolution-in-deep-learning-f6b35582557

R. Sumathy, S. Narayana Swami, T. Pavan Kumar, V. Lakshmi Narasimha, and B. Premalatha. Handwriting text recognition using cnn and rnn. In *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pp. 766–771, 2023. doi: 10.1109/ICAAIC56838.2023.10140449.

Raja Sunkara and Tie Luo. No more strided convolutions or pooling: A new cnn building block for low-resolution images and small objects, 2022. URL https://arxiv.org/abs/2208.03641.

Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S. Huang. Deepfont: Identify your font from an image. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pp. 451–459, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334594. doi: 10.1145/2733373.2806219. URL https://doi.org/10.1145/2733373.2806219.

Heng Zhao and Huihui Li. Handwriting identification and verification using artificial intelligence-assisted textural features. *Scientific Reports*, 13(1), Dec 2023. doi: 10.1038/s41598-023-48789-9.